PM

Sistema Automático de Medição do Nível de Água em Ribeiras do Funchal Através de Videovigilância

PROJETO DE MESTRADO

João André Santos Brás

MESTRADO EM ENGENHARIA ELETROTÉCNICA - TELECOMUNICAÇÕES

UNIVERSIDADE da MADEIRA

A Nossa Universidade

www.uma.pt

setembro | 2022

Sistema Automático de Medição do Nível de Água em Ribeiras do Funchal Através de Videovigilância

PROJETO DE MESTRADO

João André Santos Brás

MESTRADO EM ENGENHARIA ELETROTÉCNICA - TELECOMUNICAÇÕES

ORIENTAÇÃO

Joaquim Amândio Rodrigues Azevedo

UNIVERSIDADE da MADEIRA

Faculdade de Ciências Exatas e da Engenharia

Sistema automático de medição do nível de água em ribeiras do Funchal através de videovigilância

João André Santos Brás nº 2032216

Dissertação submetida à Universidade da Madeira para a obtenção do Grau de Mestre em Engenharia Eletrotécnica-Telecomunicações

Orientador: Professor Doutor Joaquim Amândio Rodrigues Azevedo

setembro de 2022

Resumo

A Ilha da Madeira ao longos dos anos tem sido fustigada pela ocorrência de aluviões com consequências por vezes prejudiciais para a população. Devido a esta situação foi possível concluir que era necessário um sistema de monitorização do nível de água nas ribeiras para monitorização de situações de risco de aluviões.

Para a monitorização do nível de água em rios, ribeiras ou cursos de água urbanos é considerado cada vez mais procedimentos de processamento de imagem. A maioria dos sistemas utiliza medidores de água, que são réguas que auxiliam a medição do nível de água. No entanto, estes sistemas podem falhar, quando são aplicados a cursos de água urbanos, devido à ondulação da água, detritos existentes na superfície da água, a traços de chuva ou outros efeitos adversos capturados nas imagens que podem provocar resultados erróneos. A importância de considerar todos estes efeitos é que estes geralmente estão associados à variação do nível de água com a ocorrência de precipitação.

O sistema desenvolvido neste projeto contém uma câmara infravermelha posicionada num prédio a apontar para a ribeira de Santa Luzia, no concelho do Funchal. Esta câmara, pelas suas características, permite operar durante o dia e durante a noite. A zona de captura da imagem é maior do que sistemas existentes para poder minimizar os efeitos que tendem a obstruir a linha de água.

Inicialmente foi desenvolvido um sistema de referência para que fosse possível converter os pixéis numa unidade de medição, neste caso, em centímetros. De seguida procedeu-se a implementação do algoritmo de forma a obter o nível de água. Primeiro aplicaram-se certas técnicas de forma a melhor a qualidade da imagem. Depois efetuou-se a compensação do movimento da câmara através da deteção de uma das bordas superiores da ribeira para compensar a rotação da câmara e através da deteção de padrões para compensar a translação da câmara. De seguida foi obtida a região de interesse e analisou-se a mesma em diferentes condições climatéricas por forma a obter a linha de água para cada situação. Neste processo foi efetuada uma média temporal e espacial de imagens para obter um melhor perfil de escala de cinza e de gradiente de cada imagem. Por fim foi obtido o nível de água através do sistema de referência.

A exatidão teve uma variação entre 0,7 cm e 1,6 cm durante o dia e entre 2,7 cm e 3,3 cm durante a noite. A exatidão média obtida através das diferentes condições climatéricas foi de 1,1 cm durante o dia e 2,9 cm durante a noite. Uma das soluções para melhorar a exatidão da deteção do nível de água durante a noite é através da implementação de iluminação artificial orientada para a região de interesse.

A técnica desenvolvida permite obter uma boa exatidão para diferentes condições climatéricas, combinando informações de várias posições de deteção da linha de água para evitar problemas na zona de deteção. De referir que deste trabalho resultou um artigo publicado em revista.

Palavras-chave: aluvião; monitorização; cursos de água; medição do nível de água; processamento de imagem.

Abstract

Madeira Island over the years has been fustigate by the occurrence of flooding with sometimes harmful consequences for the population. Due to this situation, it was possible to conclude that a system to monitor the water level in the streams was necessary to control alluvium.

For water level monitoring in rivers, streams or urban streams, image processing procedures are increasingly being considered. Most systems use staff gauges, which are rulers that assist in measuring the water level. However, these systems can fail when applied to urban waterways due to water ripples, debris on the water surface, traces of rain, or other adverse effects captured in the images that can cause erroneous results. The importance of considering all these effects is that they are generally associated with the variation of the water level with the occurrence of precipitation.

The developed system in this project contains an infrared camera positioned on a building to the Santa Luzia stream, in the municipality of Funchal. This camera, due to its characteristics, can operate during day and night. The image capture zone is larger than existing systems to minimize the effects that tend to obstruct the water line.

Initially, a reference system was developed so that it was possible to convert the pixels into a measurement unit, in this case centimeters. Then we proceeded to implement the algorithm to obtain the water level. First, some techniques were performed to improve the image quality. Then the camera movement was compensated by detecting one of the upper edges of the stream to compensate for camera rotation and by detecting patterns to compensate for camera translation. Then the region of interest was obtained and analyzed in different weather conditions to obtain the waterline for each situation. In this process, temporal and spatial averaging of images was performed to obtain a better grayscale and gradient profile for each image. Finally, the water level was obtained through the reference system.

The accuracy ranged from 0.7 cm to 1.6 cm during the day and from 2.7 cm to 3.3 cm at night. The average accuracy obtained across the different weather conditions was 1.1 cm during the day and 2.9 cm at night. One of the solutions to improve the accuracy of water level detection at night is through the implementation of artificial lighting oriented to the region of interest.

The developed technique allows to obtain good accuracy for different weather conditions by combining information from several waterline detection positions to avoid problems in the detection zone. If should be noted that this work resulted in an article published in a journal.

Keywords: alluvium; monitoring; streams; water level measurement; image processing.

Agradecimentos

Agradeço primeiramente ao meu orientador, Professor Doutor Amândio Azevedo, pela disponibilidade e paciência demonstrada ao longo do projeto, e pela maneira clara e objetiva de esclarecer as dúvidas que foram aparecendo.

Agradeço aos vários docentes da Universidade da Madeira que fizeram parte do meu percurso académico.

Agradeço ao Engenheiro Filipe Santos por toda ajuda e sabedoria fornecida ao longo do meu percurso académico.

Agradeço aos meus colegas Márcio Oliveira, Francisco Sardinha, David Gonçalves, João Rodrigues, Alexandre Branco, Raquel Barros e Lisandro Marote, com quem tive a oportunidade, ao longo da minha licenciatura e mestrado, de partilhar bons momentos e por toda a ajuda prestada.

Agradeço à minha namorada Filipa por toda a ajuda prestada ao longo deste projeto e por nunca me deixar desistir nos momentos mais complicados na minha vida académica.

Agradeço aos meus pais pelo apoio incondicional que sempre me habituaram e por nunca terem duvidado das minhas competências.

Finalmente agradeço a todos aqueles que com as suas críticas e ideias ajudaram a melhorar este projeto.

Índice

Índice de figuras

Índice de tabelas

Lista de acrónimos

CLAHE	*Contrast Limited Adaptative Histogram Equalization*
CMOS	*Complementary Metal Oxide Semiconductor*
FTP	*File Transfer Protocol*
GCP	*Ground Control Points*
GSM-GPRS	*Global System for Mobile Comunications – General Packet Radio Service*
IDE	*Integrated Development Enviroment*
LED	*Light-Emitting Diode*
MMD	*Maximum Mean Difference*
NAS	*Network-Attached Storage*
OpenCV	*Open Source Computer Vision Library*
OSF	*Order-Statistic Filtering*
ROI	*Region of Interest*
VPN	*Virtual Private Network*

1. Introdução

Neste capítulo são apresentados a motivação para a realização deste trabalho, os objetivos a alcançar com o mesmo e a estrutura deste relatório.

1.1. Motivação

A ilha da Madeira nos últimos anos tem sido vítima de aluviões que tem causado diversos danos materiais e até perdas humanas. A precipitação repentina dá origem a grandes quantidades de água e material sólido nas ribeiras, podendo levar ao acumular de material a jusante e transbordo das paredes das mesmas. Com isso foi necessário efetuar monitorização das ribeiras a fim de prevenir eventuais situações críticas para as populações.

Com a evolução da tecnologia em todo o mundo foram surgindo dispositivos que permitem, através de diversos métodos, detetar o nível de água em rios, ribeiras e outros cursos de água. Os dipositivos mais populares utilizados para obter o nível de água são os sensores de pressão e os ultrassónicos [1]. O primeiro permite obter a quantidade de água por unidade de área, sendo que necessita de estar submerso no local de medição. Apesar da fácil utilização é um equipamento que é frágil, danificando-se muito facilmente o que se torna complicado a sua instalação em ribeiras na Madeira devido a quantidade de detritos que normalmente são arrastados no curso de água. O segundo equipamento emite e recebe ondas ultrassónicas e a partir das mesmas é possível obter o nível de água. Em comparação com o primeiro, não necessita de estar em contacto com a água, mas para que a onda reflita, é necessário que não existam detritos na água, algo que também é difícil nas ribeiras da Madeira.

Com a criação dos satélites, estes também começaram a ser utilizados em diversas áreas, entre as quais a medição do nível de água através de câmara incorporadas nos satélites. Estes permitem uma monitorização do caudal de qualquer curso de água bem definido. Apesar disto, este tipo de sistema tem um custo muito avultado e para recolher informação é necessário que as condições climatéricas sejam favoráveis [2].

Um outro sistema que permite obter o nível de caudal é os sistemas baseados em imagem. Estes sistemas basicamente baseiam-se na captura de imagens e a partir das mesmas efetuar processamento de forma a obter o nível de água. Dentro desta área já existem certos sistemas com algoritmos que permitem obter o nível de água em cursos de água, sendo que os mesmos apresentam vantagens e desvantagens, quer seja no erro cometido na deteção ou o custo associado ao sistema.

A motivação deste trabalho é a criação de um sistema de baixo custo, baseado numa câmara e num Raspberry Pi que permita a captura de imagens e armazenamento das mesmas e efetuar um algoritmo que permita detetar e monitorizar o nível de água em ribeiras.

1.2. Objetivos

O principal objetivo deste trabalho foi a implementação de um sistema de baixo custo que permita monitorizar o nível de água em ribeiras, sendo esta situação bastante útil para situações em que haja um aumento considerável do caudal da ribeira. O sistema implementado é composto por duas componentes: uma componente de *hardware*, necessária à captura e envio de imagens para um servidor, e uma componente de *software*, que permite determinar o nível de água. O foco deste trabalho foi nesta segunda componente devido ao facto que a componente de *hardware* ter sido usada de trabalhos anteriores e a mesma já estava em funcionamento de forma a ter informação para testar o bom funcionamento do algoritmo implementado.

Assim sendo, foram estabelecidos os seguintes objetivos específicos:

- Analisar o estado da arte referente às abordagens utilizadas para medição do nível de água;
- Implementação de um sistema de baixo custo para a monitorização do nível de água em ribeiras;
- Implementação de um algoritmo que permita a obtenção do nível de água na ribeira;
- Testar o sistema completo na ribeira de Santa Luzia, no Funchal;
- Análise dos resultados do parâmetro de interesse.

1.3. Estrutura do relatório

Este relatório está dividido em cinco capítulos e inclui anexos. Neste primeiro capítulo é realizada a apresentação da motivação e objetivos deste trabalho.

No segundo capítulo é apresentado o estado da arte, que inclui um breve enquadramento histórico em relação às aluviões ocorridos na ilha da Madeira, bem como várias das técnicas utilizadas para efetuar a medição do nível de água. Posteriormente são apresentados vários trabalhos que usam sistemas baseados em imagens e é apresentada uma técnica que permite converter a dimensão de um objeto no plano de uma imagem para o plano real. Por fim são apresentadas técnicas de processamento de imagem que serão usadas ao longo do trabalho.

O desenvolvimento do projeto é apresentado no terceiro capítulo. Neste são descritos os requisitos do sistema a implementar e apresentação do sistema de medição. Também neste capítulo é apresentada a técnica utilizada para a calibração da câmara, neste caso, a técnica que permite determinar um ponto no plano do objeto a partir de um ponto no plano da imagem. Por fim é apresentado o todo o desenvolvimento do algoritmo que permite efetuar a deteção da altura do caudal da ribeira.

No quarto capítulo é apresentada a análise dos resultados. Numa fase inicial é efetuada uma análise de forma a perceber a diferença na precisão entre utilizar ou não a média temporal de imagens. De seguida faz-se a análise de situações de ligeira e forte precipitação na zona urbana com aumento de caudal. Posteriormente procedeu-se à análise de resultados referentes a situação do “período de sombra” e por fim é analisado os resultados para um longo período de observação.

O quinto capítulo é dedicado às conclusões e também são apresentadas sugestões de trabalhos futuros.

2. Estado da arte

Neste capítulo são apresentados o enquadramento histórico, as técnicas utilizadas para medição do nível de caudal em ribeiras, os trabalhos existentes na área dos sistemas baseados em imagens de videovigilância de riscos ambientais e as técnicas de processamento de imagem para determinação de bordas em imagens.

2.1. Enquadramento histórico

A Ilha da Madeira encontra-se situada no arquipélago da Madeira, fazendo parte da Macaronésia, localizada nas coordenadas 32° 42' N 17° O. Esta ilha apresenta um clima regular durante o ano inteiro, sendo que ao longo dos anos começaram a ocorrer certos períodos de forte precipitação originando assim grandes aluviões, provocando diversos danos materiais e até perdas humanas.

No dia 9 de outubro de 1803, ocorreu uma das maiores aluviões na história da ilha, sendo que os concelhos do Funchal, Machico, Santa Cruz, Ribeira Brava e Calheta foram os mais afetados [3]. Relatos nessa altura indicaram que tinham caído chuvas algo intermitentes nos dez/doze dias que precederam o dia 9 de outubro. No dia 8 por volta das 8 horas começou a ocorrência de precipitação, intensificando por volta das 20 horas, fazendo com que as ribeiras de João Gomes e de Santa Luzia não suportassem a enorme quantidade de água e material rochoso, acabando por transbordarem provocando uma enorme destruição. A ribeira de João Gomes foi a mais danificada por ter sido destruída em 3 pontos, criando correntes de água, aumentando as inundações. Certos prédios em volta da ribeira de Santa Luzia ficaram muito danificados. O número de mortes indicado na altura foram de cerca de 1000. A falta de encanamentos das ribeiras foi a principal causa dos estragos que aconteceram pela aluvião.

A aluvião de 1815 teve consequências muito semelhantes a 1803. O concelho do Funchal foi novamente o mais afetado, com as ribeiras a voltarem a transbordar e a inundar toda a cidade. O caudal intenso da ribeira de São João destruiu as pontes existentes ao longo da bacia hidrográfica, deixando um rasto de destruição. Os paredões da ribeira de Santa Luzia também não suportaram a quantidade de água excessiva, danificando casas e partes das paredes da ribeira [3].

Em 1993, no Funchal, o udómetro localizado no Observatório Meteorológico registou uma precipitação de 88,9 mm, entre as 9 horas do dia 28 e as 9 horas do dia 29 de outubro. Ocorreu precipitação ao longo de todo o dia 28, sendo que entre as 21 horas e as 3 horas foi onde ocorreu maior precipitação. Num período de 6 horas ocorreu uma precipitação de 66,4 litros/m^2, dos quais 29,8 foram entre as 2 e 3 horas. Com isto as ribeiras de São João, Santa Luzia, João Gomes e Socorridos transbordaram ao longo do percurso de água. Como consequência a zona baixa da cidade ficou completamente inundada de água, lama e material rochoso. Vários prédios, embarcações e estradas danificadas e a escassez de água potável foram as consequências desta aluvião, causando 200 desalojados, 30 feridos e 9 vítimas mortais [4].

Ocorreram novamente cheias em 1997, devido à precipitação intensa do dia 19 de outubro para o dia 20, chegando a precipitação a 349,9 mm, entre as 9 horas do dia 19 e as 9 horas do dia 20. As ribeiras de Santa Luzia, de São João, dos Socorridos, de

João Gomes e de Machico, na zona Sul, e as ribeiras do Juncal e Metade (Santana), na zona norte, tiveram como consequência o aumento do caudal. A ribeira de Santa Luzia teve as consequências mais graves, com a destruição da ponte velha dos Socorridos e com a queda da ponte de acesso à Estação de Tratamentos de Água dos Tornos. Como nas anteriores aluviões, os troços finais das três ribeiras principais do Funchal (ribeira de Santa Luzia, João Gomes e São João) continham imenso material rochoso [4].

A aluvião ocorrida em 20 de fevereiro de 2010 foi considerada uma das maiores catástrofes ocorridas no arquipélago desde que há registos, 207 anos após a aluvião de 1803. Este evento ocorreu na sequência de um prolongado período de precipitação entre outubro de 2019 e fevereiro de 2010, com precipitação acumulada, nos vários postos de monitorização da ilha (por parte do LREC – Laboratório Regional de Engenharia Civil), superior a 1000 mm. No mês de fevereiro registaram-se valores significativos de precipitação diária, por vezes superior a 100 mm. Com isto resultaram inundações e derrocadas em toda ilha, sendo que a parte sul da ilha abrange as áreas mais afetadas, nomeadamente os concelhos do Funchal, Ribeira Brava, Câmara de Lobos e Santa Cruz. Grande parte das inundações ocorreram na sequência do elevado caudal e transporte de material sólido ao longo das bacias hidrográficas, como é possível observar na figura 2.1.

Figura 2.1 - Ribeira de João Gomes afetada pela aluvião de 2010 [5].

Desta aluvião resultou em 48 perdas humanas, 250 feridos, 600 desalojados, 500 carros danificados e 800 habitações destruídas. Os prejuízos estimados foram de 1080 milhões de euros [6] [7].

2.2. Técnicas para medição do nível de caudal

Neste capítulo serão apresentadas as técnicas mais utilizadas para medição da altura do caudal, sendo estas: sensores de pressão, sensores ultrassónicos, sistemas baseados em satélite e sistemas baseados em imagem.

2.2.1. Sensores de pressão

Os sensores de pressão podem ser utilizados para medir o nível da água através da força por unidade de área, sendo esta representada por um produto de massa e aceleração da gravidade da água, obtendo como resultado a quantidade de água por

unidade de área. Através da quantidade de água por unidade de área pode-se chegar ao valor para o nível da água [1].

Na figura 2.2 é apresentado um exemplo de um sensor de pressão utilizado para medir o nível da água. Para que este tipo de sensor possa medir o nível de água, o mesmo tem de estar ligeiramente submerso para efetuar as medições.

Figura 2.2 - Sensor de pressão [8].

Estes tipos de sensores apresentam certas vantagens, tais como: fácil utilização, um baixo consumo de energia e permitem fornecer informação “quase” em tempo real. Apesar destas vantagens, estes sensores necessitam de ser calibrados frequentemente ou até mesmo substituídos devido à pressão contínua da água. Outra situação que é necessário ter em atenção é que caso estes sensores estejam submersos a uma grande profundidade poderão ficar danificados [1] [9].

2.2.2. Sensores ultrassónicos

Os sensores ultrassónicos emitem e recebem um impulso de ondas ultrassónicas. Estes sensores têm uma distância mínima de medição do nível de água, chamada de distância de supressão (geralmente 30 – 45 cm). Se o nível de água subir até a distância de supressão, o sensor não será capaz de medir o nível de água de forma correta. No entanto, o sensor deve ser colocado o mais próximo possível do nível de água para obter a máxima precisão. Também deve ser instalado perpendicularmente à água para que o impulso reflita diretamente de volta ao sensor. O sensor deve estar longe o suficiente das paredes para evitar que o impulso atinja as mesmas. A distância mínima do sensor a qualquer parede é obtida através da multiplicação da distância vertical da face do sensor ao nível de água e a tangente (ângulo de feixe) [9]. Na figura 2.3 é apresentado como é feito a instalação do sensor e o ângulo de feixe do sensor.

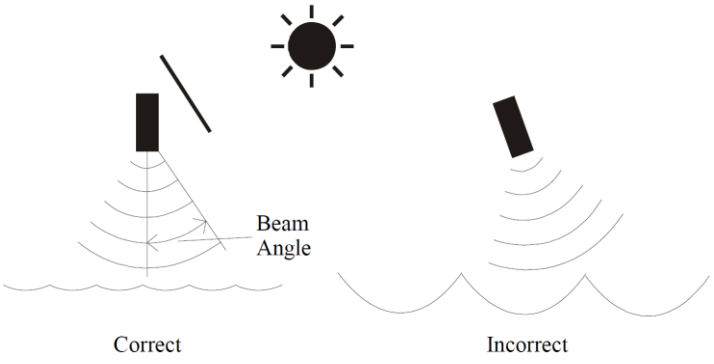


Figura 2.3 - Instalação de um sensor ultrassónico [9].

Após a emissão e receção do impulso por parte do sensor é medido o intervalo de tempo entre o sinal transmitido e recebido que, através de um conversor eletrónico, é convertido no valor da distância, utilizando a velocidade do som no ar [1] [9]. Na figura 2.4 é apresentado um exemplo de um sensor ultrassónico.

Figura 2.4 - Sensor ultrassónico [10].

O sensor ultrassónico é um sensor de fácil instalação, não necessita de estar em contacto com a água, o que diminui o desgaste do mesmo e também permite uma fácil calibração. Apesar disto, apresenta diversas desvantagens, visto que, este sensor apresenta grandes erros quando a água está turbulenta [9]. As flutuações da temperatura e do ar também afetam o seu desempenho e necessita de ser alinhado com precisão. Outra desvantagem é que o sinal emitido pelo sensor pode encontrar um detrito na zona da superfície da água e acabar por refletir para o sensor a partir da superfície do detrito e não da água [1] [9].

2.2.3. Sistemas baseados em satélite

Nos sistemas baseados em satélite são utilizados, tal como o nome indica, satélites com câmaras integradas para capturar imagens através dos pontos da superfície da água onde as margens são bem definidas e a partir dos resultados das mesmas são realizadas as medições através de várias metodologias [2].

Estes sistemas tem a vantagem de captar imagens de rios, bacias hidrográficas e lagos de grandes dimensões, onde as margens são bem definidas. Apesar disto, estes sistemas não fornecem resolução espacial ou temporal suficiente, especialmente para curtos cursos de água. A presença de vegetação e de nuvens podem prejudicar a medição e a recolha de dados, tal como as condições atmosféricas. Outros fatores que prejudicam as medições são os sedimentos e as margens não definidas nos cursos de água. Uma outra desvantagem deste sistema é o custo elevado do mesmo [2] [11].

2.2.4. Sistemas baseados em imagens

Nos sistemas baseados em imagens são normalmente utilizadas câmaras que capturam imagens da zona da bacia hidrográfica que está a ser avaliada e posteriormente são utilizadas técnicas de análise nas imagens capturadas para estimar o nível da água. As câmaras poderão ser fixas ou acopladas a veículos aéreos não tripulados que têm a capacidade de aquisição de imagens para monitorização do nível da água [12] [13].

Este tipo de sistemas tem a vantagem de ser de baixo custo, de fácil instalação ao lado de um rio ou próximo de casas, de adquirir imagens em tempo real e é de fácil manutenção. Apesar disto, visto que este sistema poderá adquirir centenas ou milhares

de imagens, é necessário que o sistema tenha um armazenamento em disco capaz de alocar todas as imagens adquiridas. Outra desvantagem que este sistema apresenta é que a câmara poderá ficar danificada devido a grandes fenómenos atmosféricos, como chuva ou trovoada [12] [13].

2.3. Sistemas existentes baseados em imagens

Os sistemas baseados em imagens são essenciais para a monitorização de caudais. Nas secções seguintes serão apresentados vários exemplos de trabalhos desenvolvidos, envolvendo sistemas baseados em imagens, em diferentes locais.

2.3.1. Deteção melhorada do nível de água por processamento de imagem

Um dos trabalhos na área dos sistemas baseados em imagens foi a implementação de um sistema de medição do nível de água em um canal urbano aberto em Singapura [14]. Os equipamentos utilizados neste sistema foram uma câmara com um sensor de cor de 3 megapixéis e um sensor de preto e branco de 1 megapixel, um *router* e uma NAS (*Network-Attached Storage*) para armazenamento das imagens. Nesta situação não foi necessário adicionar iluminação artificial durante a noite, devido à baixa sensibilidade do sensor preto e branco e também devido à existência de iluminação pública próxima do sistema instalado. Na figura 2.5 estão apresentados o sistema instalado e a zona de medição.

Figura 2.5 - Instalação do sistema no canal pluvial [14].

O sistema implementado é autónomo, visto que contém um modem GSM-GPRS (*Global System for Mobile Comunications – General Packet Radio Service*), uma bateria de 12 V que serve para alimentar o sistema, com capacidade de permitir armazenar dados durante 14 dias. Os dados são enviados em intervalos de 5-6 minutos quando o sistema está em funcionamento normal, mas quando o nível da água está acima do normal o sistema passa a enviar os dados de 1 em 1 minuto. Os dados são transmitidos para o centro de dados onde os mesmos são processados. O sistema tem a capacidade de capturar imagens durante o dia, a noite e na ocorrência de chuvas tropicais fortes.

O processamento das imagens para a medição do nível da água foi dividido em duas fases, um algoritmo para deteção do nível da água e a estimativa da profundidade da água. Na figura 2.6 são apresentadas as imagens mais relevantes correspondentes ao processamento realizado nas imagens captadas. O algoritmo para deteção do nível da água tem os seguintes passos [14]:

1. A imagem captada (figura 2.6 a)) é transformada na visão frontal não distorcida (figura 2.6 b));
2. É selecionada a região de interesse (ROI – *Region Of Interest*) a partir da imagem transformada anteriormente (figura 2.6 c));
3. É aplicado um algoritmo de deteção de bordas para obtenção de bordas na região de interesse (figura 2.6 d));
4. É aplicada a transformada de Hough para encontrar a linha reta mais longa na imagem que irá indicar a linha de água, como é apresentado na figura 2.6 e) (linha a vermelho);

A segunda fase foi baseada na medição de campo efetuada na região de interesse, sendo que os quatro círculos apresentados na figura 2.6 f) foram utilizados como referência para calibrar e calcular a profundidade da água [14].

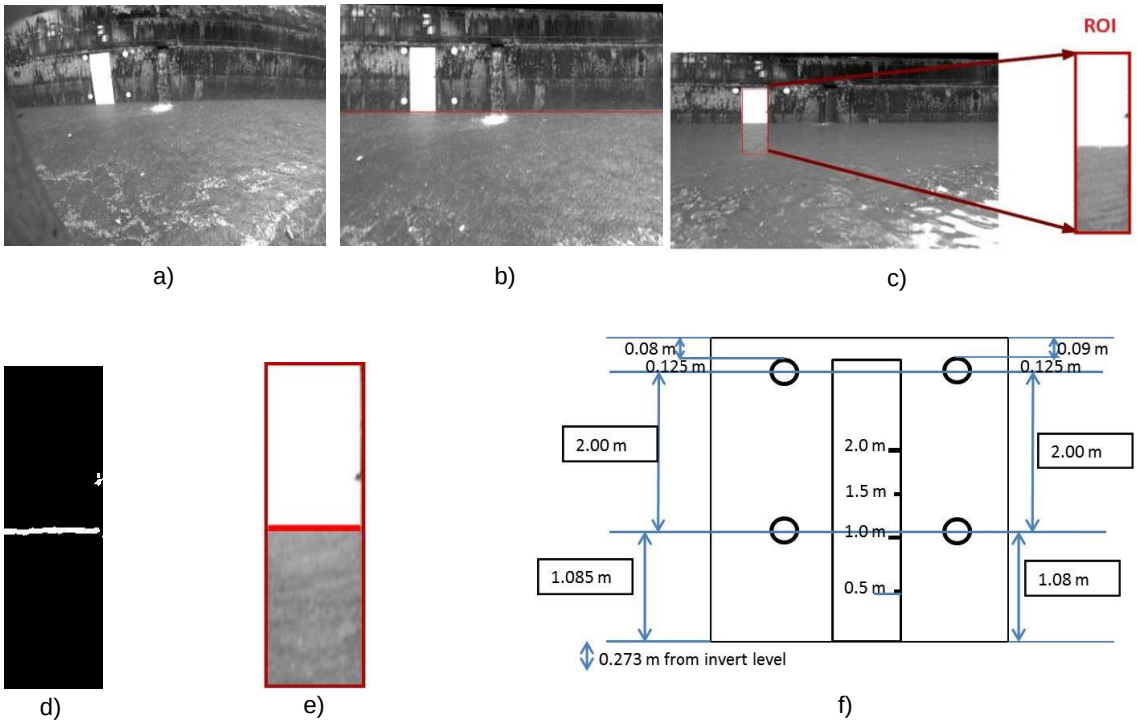


Figura 2.6 – Processamento nas imagens capturadas: a) imagem original; b) imagem transformada com visão frontal não distorcida; c) deteção da ROI; d) deteção da borda; e) deteção do nível da água; f) medição do nível da água [14].

O sistema começou o seu funcionamento em abril de 2011 e a análise de resultados envolveu os eventos nas águas pluviais até dezembro de 2011. Também foram efetuadas comparações dos resultados da medição do nível da água com outros dois sistemas implementados no mesmo local, um sensor radar e um sensor medidor de pressão [14].

Para comparar os resultados dos sistemas referidos anteriormente (câmara, sensor radar e sensor de pressão) foi utilizado um sistema de referência baseado na imagem original captada pela câmara, obtida num determinado instante. Após a orientação da imagem, foi efetuada a contagem dos pixéis na direção vertical dentro da região de interesse, visto que cada pixel equivale a 1 cm, de acordo com a orientação efetuada [14]. A profundidade real foi definida por:

$$profundidade = (nº\ de\ pixéis \times 1\ cm) + offset \qquad (2.1)$$

Na figura 2.7 é apresentado um gráfico comparativo dos três sistemas da medição do nível de água no período de 26 a 27 de abril de 2011.

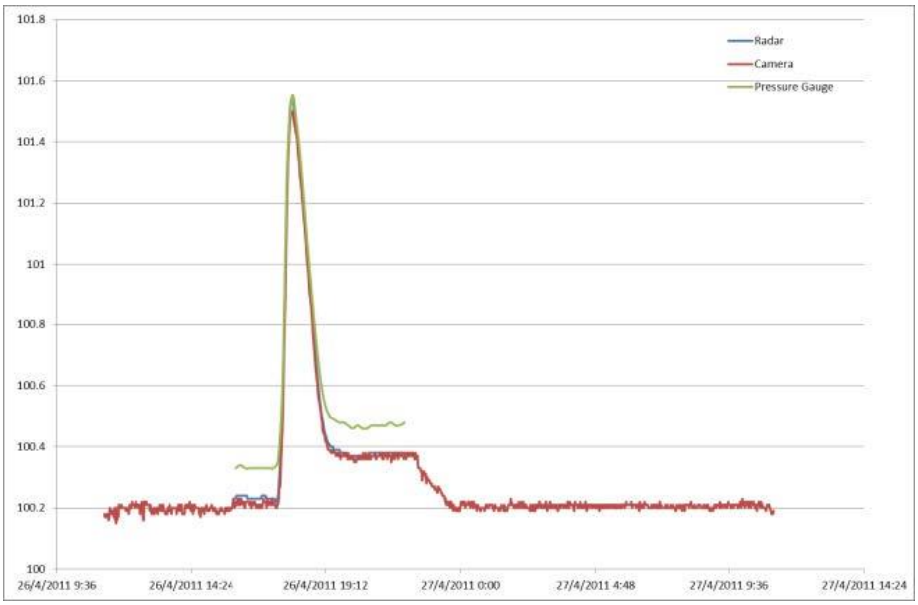


Figura 2.7 - Gráfico de comparação da medição do nível de água [14].

Comparando o sistema implementado com o sistema de sensor radar, este primeiro apresentou uma melhor exatidão do que o sensor radar, com 1,1 % e 1,5 %, respetivamente, em relação a medição real do nível da água. Também apresentou uma melhor exatidão na medição do nível da água em comparação ao sensor medidor de pressão, com 1,1 % e 3,9 %, respetivamente [14].

Apesar dos resultados, este sistema apresenta diversas desvantagens, visto que o mesmo não foi testado com certas condições climáticas, como nevoeiro, neve ou granizo. Também é necessário melhorar o processamento das imagens para que seja realizado em tempo real, tornando o sistema disponível em estações remotas onde a ligação de dados e a fonte de alimentação são limitadas.

2.3.2. Medição do nível da água com dois sistemas baseados em imagem

Zhang et al. [15] apresentam um trabalho onde foram desenvolvidos dois sistemas baseados em imagem para a medição do nível da água em dois locais distintos da China.

O primeiro sistema foi localizado no rio Jurong, em Nanjing, na estação hidrológica Qianhancun. Esta estação hidrológica consiste na monitorização do nível da água e do processo de inundação, visto que a mesma está localizada numa zona montanhosa onde ocorrem inundações na época de chuvas (maio a setembro).

Na figura 2.8 está apresentado o primeiro sistema de medição.

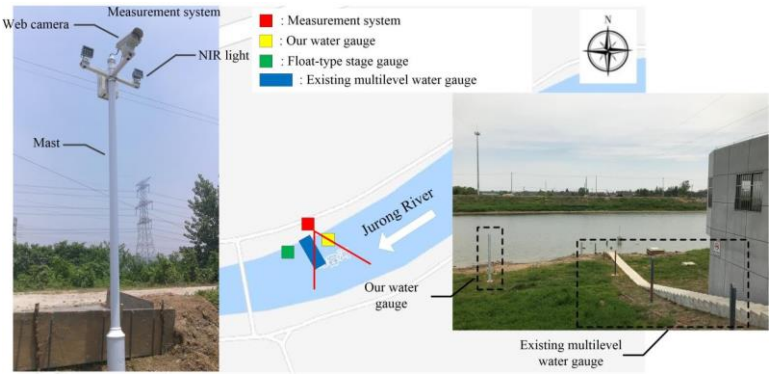


Figura 2.8 - Instalação do primeiro sistema de medição baseado em imagens no rio Jurong [15].

O sistema apresentado na figura 2.8 está implementado na margem norte do rio e contém como equipamentos um indicador do nível de água do tipo flutuador, cinco medidores de água (*staff gauge*), uma câmara de 4 megapixéis, um filtro ótico passa-banda montado na lente da câmara para obter uma imagem quase infravermelha e um *router* 4G. A câmara tem uma inclinação de 12,9º e está posicionada a sul direcionada para o medidor de água. O *router* é utilizado para o utilizador aceder ao sistema de forma remota através de uma VPN (*Virtual Private Network*).

O segundo sistema está localizado na província de Zhejiang, no riacho Yanglou, que contém muito cascalho. Na época de chuva a profundidade da água no leito do riacho pode subir até 3 metros [15]. Na figura 2.9 está apresentado o segundo sistema de medição.

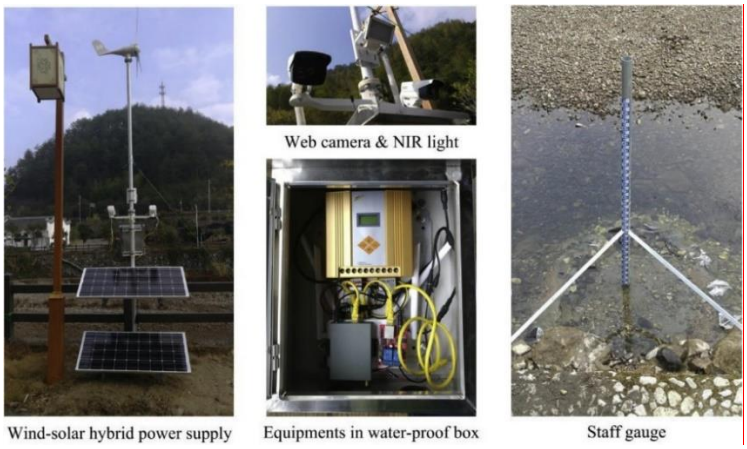


Figura 2.9 - Instalação do segundo sistema de medição baseado em imagens no riacho Yanglou [15].

O sistema da figura 2.9 está implementado na margem sul do rio e contém como equipamentos um medidor de água, uma câmara com sensor CMOS (*Complementary Metal Oxide Semiconductor*) de 2 megapixéis, uma lente de 8 mm, uma luz LED (*Light-Emitting Diode*) e uma fonte de alimentação híbrida eólica-solar [15].

O processo inicial para medição do nível da água começou com a obtenção da região de interesse. Para isso foi captada uma imagem através da câmara e foi aplicado um processo de correção da distorção da imagem. Na figura 2.10 são apresentados os vários passos descritos anteriormente até chegar a imagem com a região de interesse.

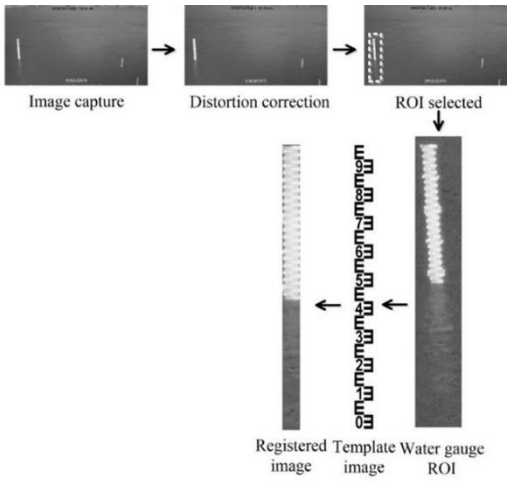


Figura 2.10 - Passos efetuados até a obtenção da região de interesse [15].

Após a obtenção da região de interesse é efetuado um processo de forma a obter a posição da linha de água. Primeiramente a região de interesse foi convertida na escala de cinza e depois foi usado o método de deteção de bordas. Posteriormente foi efetuada a extração de características da imagem, sendo determinada a média da escala de cinza na dimensão horizontal ao longo da dimensão vertical da região de interesse. Esta extração é realizada na imagem em escala de cinza e também na imagem resultante da aplicação do método de deteção de bordas.

O passo seguinte foi a obtenção do posicionamento grosseiro da linha de água, sendo que para isso foi calculado o gradiente na dimensão vertical entre duas pequenas regiões de interesse adjacentes. Estas pequenas regiões de interesse tem uma dimensão vertical T. O valor máximo do gradiente é o que indica a posição, de forma grosseira, da linha de água.

O valor de gradiente obtido anteriormente foi comparado com um determinado limite de forma a verificar se está perante uma condição desfavorável. Se sim, foi assumido que as coordenadas (posição vertical) da linha de água são zero. Caso contrário foi efetuado o posicionamento fino da linha de água.

Este processo de posicionamento fino é parecido ao efetuado no posicionamento grosseiro, sendo que o que diferencia é que o deslocamento com que é calculado o gradiente. No posicionamento grosseiro o deslocamento é de T pixéis, enquanto no posicionamento fino o deslocamento é de um pixel. O valor máximo do gradiente obtido no posicionamento fino é o que indica a posição de linha de água.

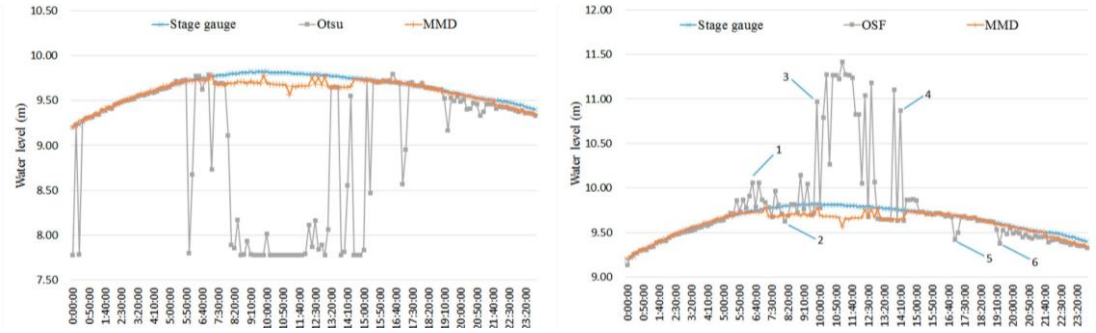
Por fim foram determinadas as coordenadas (posição vertical) da posição da linha de água (l), em pixéis, sendo que é dado pela seguinte expressão:

$$l = k_1' + T - 1 \tag{2.2}$$

em que k_1' é a posição obtida através do posicionamento preciso e T corresponde a dimensão vertical da região de interesse.

Este método referido anteriormente assume que a linha de água é geralmente localizada na posição aonde ocorre a maior diferença entre duas regiões de interesse adjacentes numa imagem de escala de cinza ou numa imagem resultante de um método de deteção de bordas [15]. Todo este processo (posicionamento grosseiro e fino) de deteção da linha de água é chamado de MMD (*Maxinum Mean Difference*).

Para comparação de resultados foi utilizado o indicador de nível de água (*stage gauge*) como referência. Para o primeiro sistema foi analisado a precisão do método MMD sob diferentes condições de iluminação, tendo os resultados obtidos comparados também com outros métodos de binarização, Otsu e *Order-Statistic Filtering* (OSF), que permitem também obter a partir dos mesmos a linha de água [15]. Na figura 2.11 estão apresentados os resultados da medição do nível de água no dia 18 de agosto de 2018.



a) b)

Figura 2.11 - Resultados da medição do nível da água para o primeiro sistema: a) comparação dos métodos MMD e Otsu; b) comparação dos métodos MMD e OSF [15].

Observando a figura 2.11 verifica-se que os métodos Otsu e OSF apresentam grandes erros entre 07:10h até às 15:50h devido à alteração das condições de iluminação que ocorreram. Entre 07:10h até às 14:40h e 21:50h até às 23:50h os resultados do método MMD foram ligeiramente inferiores, com uma exatidão de 12 cm e 5 cm, respetivamente. Isto tem a ver com a zona de medição do nível de água, em que nestes períodos existia ervas daninhas enroladas ao redor do medidor de água, resultando em erros de medição. Assim esses resultados foram descartados da análise efetuada [15].

Também devido à vibração da câmara causada pelo vento, o erro na captação da imagem foi de ±5 cm, que foi reduzido para ±2 cm através da introdução de um filtro mediano para redução de ruído. Estes erros estão relacionados com a conversão do nível real da água, mas que podem ser corrigidos com uma estratégia de deteção do movimento da câmara e ajuste dos parâmetros de orientação da câmara [15].

Depois de descartar os erros devido às ervas daninhas, foi obtido a precisão do método MMD para seis condições diferentes de iluminação, estando na figura 2.11 b) representado o período de cada uma dessas condições. Assim sendo, tem-se que: (1) luz difusa, com uma precisão de 1,1 cm; (2) projeção de sombra, com uma exatidão de 11 cm; (3) brilho da água, com uma exatidão de 3,8 cm; (4) luz do sol direta, com uma exatidão de 10,6 cm; (5) luz solar lateral, com uma exatidão de 0,4 cm e (6) iluminação artificial, com uma exatidão de 1,2 cm. Apesar de a exatidão seja alta em certas situações, no geral a exatidão obtida para o método MMD durante o período de análise foi de 1,18 cm [15].

No segundo sistema, no riacho Yanglou, os resultados do método MMD foram comparados com o medidor de água e com o método OSF. Estes resultados foram obtidos durante 7 dias, desde 4 até 10 de março de 2018, com diferentes condições climatéricas, desde céu nublado, leve e moderada precipitação e dia ensolarado [15].

Após efetuar uma análise aos resultados a exatidão obtida pelo método MMD foi de 0,87 cm e pelo método OSF foi de 1,16 cm. O método MMD obteve uma melhor exatidão devido ao facto que a exatidão durante dias ensolarados (1 cm) ser melhor do que o

método OSF (2,9 cm), o que provoca a ligeira diferença de exatidão apresentada anteriormente [15].

Apesar de apresentar melhores resultados quando comparado com o método OSF, apresenta certas desvantagens, visto que não é possível analisar se, caso exista a ocorrência de detritos em volta do medidor de água, como o sistema se irá comportar. Outra desvantagem verificada é a estabilidade da câmara havendo movimento da mesma devido ao vento e consequentemente são obtidos resultados erróneos do nível da água.

2.3.3. Deteção automática do nível de água que considera o movimento da câmara

Lin et al. [16] apresentam um trabalho onde foi desenvolvido um sistema baseado em imagens com o objetivo de monitorizar o nível de água em instalações como rios e reservatórios. O caso de estudo ocorreu em uma área urbana propensa a inundações rápidas e com problemas de segurança rodoviária devido à alta densidade populacional.

O sistema é composto por uma câmara que capta grupos de 10 imagens, a uma cadência de uma por segundo, a cada meia hora, sendo estas depois usadas para processamento da obtenção do nível de água.

Na figura 2.12 são apresentados o sistema e o local aonde o mesmo foi instalado.

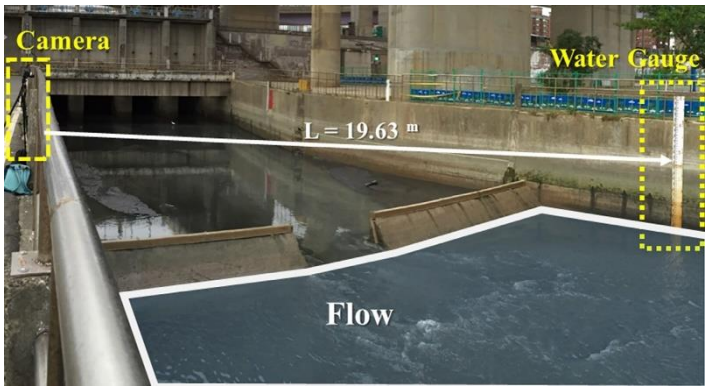


Figura 2.12 - Sistema implementado com uma única câmara para deteção do nível de água [16].

A abordagem para a deteção do nível de água proposto em [16] consistiu em dois processos: identificar a linha de água na região de interesse (medidor de água apresentado na figura 2.12) e obter o nível de água com base em princípios fotogramétricos (ou equações colineares).

Ao instalar a câmara foram efetuadas várias inicializações de forma a estabelecer a relação entre o sistema de coordenadas do plano real e o do plano da imagem. Este processo inclui três etapas: identificar pontos de calibração, medir as coordenadas dos pontos de calibração e determinar a orientação inicial da câmara. Por forma a poder obter o nível de água foram utilizados pelo menos dez pontos de calibração.

Antes de efetuar o processo de obtenção de linha de água e posteriormente do nível de água foi efetuada uma média de uma série de imagens (10). Isto permitiu reduzir o ruído do ambiente e também os movimentos da câmara. De seguida foi obtida a região

de interesse na imagem média e é aplicada a transformada de Hough para identificar a posição da linha de água.

Através da posição da linha de água e dos pontos de calibração foi obtido o nível de água por meio de equações colineares. Para comparação de resultados foram utilizados como referência os valores observados manualmente através do medidor de água.

Na figura 2.13 são apresentados os resultados do nível de água obtidos pelo sistema descrito, os valores de referência e a diferença entre estes.

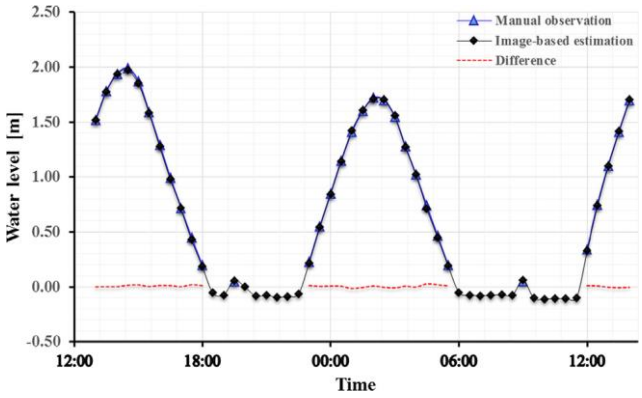


Figura 2.13 - Resultados obtidos para um determinado período entre o sistema desenvolvido e os valores de referência [16].

Através dos resultados obtidos na figura 2.13 e ao longo do período de estudo foi verificado que o sistema proposto apresenta uma exatidão de 1 cm [16].

Em relação a este sistema verifica-se que apresenta a vantagem de resolver as dificuldades na deteção do nível de água em situações em que possa ocorrer movimento da câmara e para imagens desfocadas.

Apesar disto, apresenta várias deficiências, visto que este método requer que a linha de água esteja relativamente parada, porque foi projetada para rios e reservatórios, logo, para outros locais onde exista ondulação ou até detritos na água o sistema pode não conseguir obter o nível de água. Por fim, este sistema usa como região de interesse um local onde tem colocado um medidor de água, o que permite uma melhor deteção, mas o mesmo não foi testado com a presença de detritos, o que pode dificultar a obtenção do nível.

2.3.4. Medições do nível de água efetuada por *drones*

Ridolfi et al. [17] desenvolveram um trabalho que permite a deteção do nível de água em um lago artificial através de um *drone* e uma câmara. Sistemas com apenas uma câmara fixa numa determinada superfície pode ser afetada por variações de iluminação, movimento da câmara devido ao vento ou condensação da lente. Os *drones* possibilitam a monitorização não evasiva de rios e lagos artificiais e são uma ferramenta eficiente em ambientes de difícil acesso.

O método apresentado foi testado em um lago perto de uma barragem, podendo ser aplicado a qualquer outra zona, incluindo rios, áreas de inundação, zonas costeiras e em estuários. O caso de estudo para realização dos testes foi o lago Ridracoli, um

lago artificial gerado pela barragem na região da Emilia Romagna, região central da Itália, tal como é apresentado na figura 2.14.

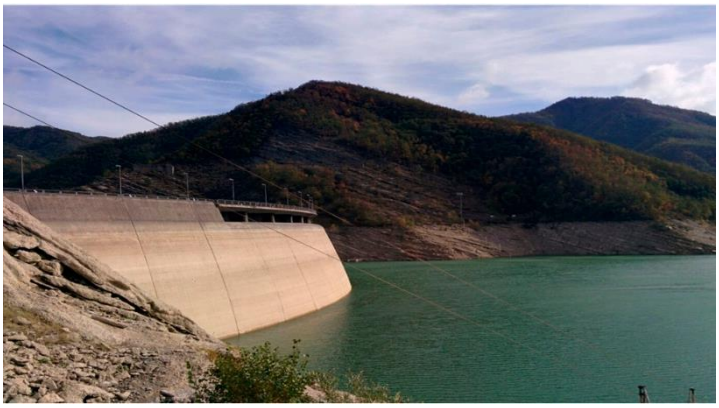


Figura 2.14 - Lago Ridracoli na região da Emilia Romagna [17].

Para efetuar a medição do nível de água foi necessário obter pontos de controlo de solo (GCP – *Ground Control Points*) na face da barragem. Os GCP foram utilizados para proceder à calibração das imagens e também como pontos de referência para obter os valores do nível de água. Foram usados quatro GCP, sendo efetuada a aquisição das coordenadas (nas três dimensões) dos mesmos através de um topógrafo. As imagens foram capturadas a partir do *drone* a uma distância de 15 metros da barragem. Após isto foi efetuada a calibração da imagem para eliminar distorções introduzidas pela câmara.

O nível de água foi estimado em quatro locais diferentes para testar a fiabilidade do método. O processo para determinar o nível de água começou com a categorização dos pixéis da imagem em quatro classes diferentes: água, face do betão e as partes pretas e brancas do marcador, sendo isto feito através de uma *tool kit* de um projeto *open-source* chamado de *Interactive Learning and Segmentation*. De seguida foi aplicado o método Canny por forma a encontrar bordas e, assim, obter a posição da linha de água.

Na imagem 2.15 são apresentados todos os passos que são realizados até obter a posição da linha de água.

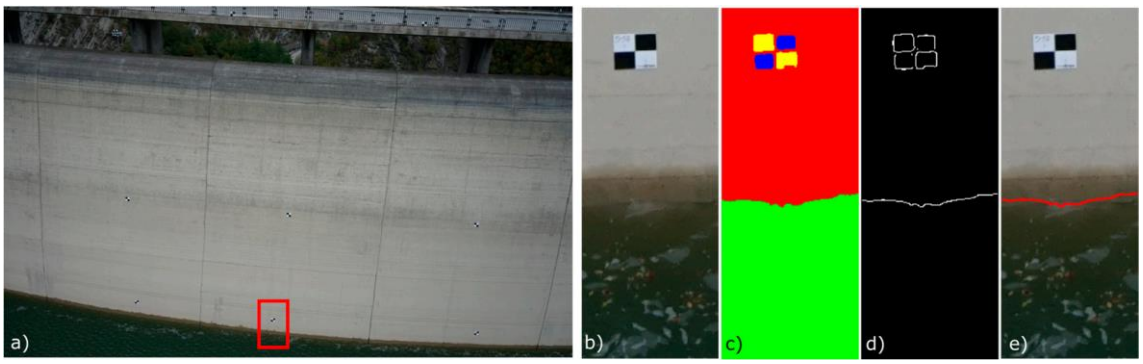


Figura 2.15 - Processo de obtenção da linha de água: a) imagem capturada pela câmara; b) região de interesse; c) imagem após categorização dos pixéis; d) imagem após aplicação do método Canny; e) imagem com a posição da linha de água [17].

A posição da linha de água foi convertida em coordenadas utilizando o centro de cada um dos quatro GCP para saber o nível de água no lago. Este nível de água obtido

foi comparado com o valor de referência. O valor de referência foi obtido a partir de um dinamômetro, que mede o peso da coluna de água e retorna o nível de água correspondente do lago. Na imagem 2.15 e) verifica-se que a linha apresenta uma ligeira oscilação, sendo isto devido à velocidade do vento existente no momento do voo do *drone*, o que nesta situação permitiu também testar este método em condições meteorológicas adversas.

Para comparação de resultados foi determinado o erro absoluto médio para os quatro GCP. Verificou-se que a exatidão se situa entre 1,2 cm e 5,1 cm, que corresponde ao segundo e quarto GCP, respetivamente. Em relação ao quarto GCP, este teve pior exatidão devido ao facto que a perspetiva com que a foto é capturada afeta as medições do nível de água.

Em relação a este sistema verifica-se que este permite obter o nível de água com uma boa exatidão, sendo que a maior vantagem que este sistema apresenta é que não apresenta impacto ambiental, visto utilizar um *drone* para efetuar a captura das imagens.

Apesar disto, verifica-se certas deficiências neste sistema, sendo que uma delas é reportada pelos autores do mesmo, que é o facto que a ondulação da água pode afetar substancialmente as medições, podendo causar erros maiores que o esperado. Outra deficiência tem a ver com a perspetiva com que a imagem foi obtida, pois o sistema não está preparado para uma variação de perspetiva, causando assim erros na medição. Por fim este tipo de sistemas pode ter um custo elevado devido a utilização de *drones*, sendo que os mesmos atualmente são caros.

2.3.5. Medição do nível de água a montante de um riacho

Leduc et al. [18] apresentam um sistema baseado em imagens em que utiliza uma câmara com um sistema de *time-lapse* para medir o nível de água utilizando classificação de imagens. Neste sistema também foram efetuadas medições da largura do caudal do riacho, só que não será descrito, visto que o intuito do trabalho desenvolvido será medição do nível do caudal. O local de estudo foi um riacho a aproximadamente 100 metros a jusante da saída do curto lago da Geleira Dome, no Parque Nacional Jasper, Alberta, Canadá. Na figura 2.16 são apresentadas a localização do sistema no riacho e uma das imagens capturadas pela câmara.

a) b)

Figura 2.16 - a) Localização do sistema no riacho; b) imagem capturada pela câmara [18].

O sistema contém uma câmara presa a um poste que foi enterrado no solo rochoso, estando esta a apontar para a face quase vertical de uma grande pedra na margem oposta ao riacho. As imagens foram capturadas de 15 em 15 minutos, entre as 6:00h e as 22:00h. Também foram instaladas dois medidores de água e um transdutor de pressão que regista de 15 em 15 minutos o nível de água para comparar com os resultados obtidos através do sistema implementado. Os dois medidores contêm duas funções: calibrar a câmara e servir como regiões de interesse para obtenção do nível de água.

Após captação das imagens procedeu-se à uma classificação automática das mesmas. As imagens capturadas durante episódios de neve ou chuva forte, aquando da luz do sol bate diretamente na câmara ou durante a noite são removidas. Durante a deteção do nível de água uma sombra interferiu na deteção, sendo que nestas situações é utilizado um processo de deteção diferente.

Foi verificado que a estimativa da linha de água apresentou melhores resultados quando utilizada a superfície da pedra.

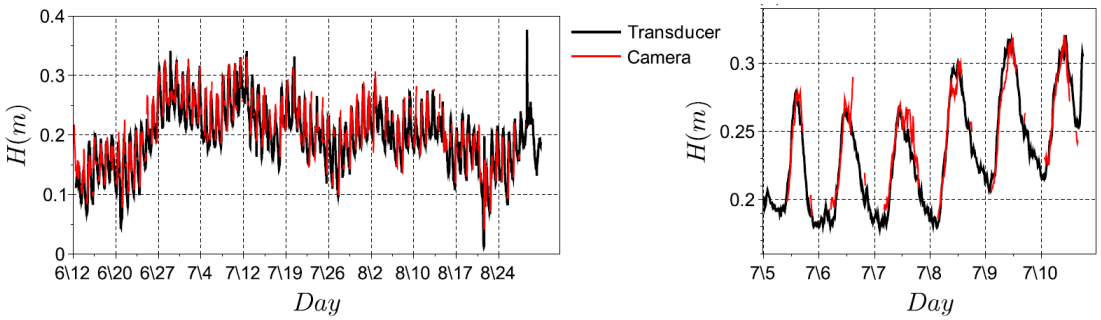
A transição da água correspondeu a um ponto de inflexão no perfil de escala de cinza e um pico local no gradiente do perfil de escala de cinza. O ponto de inflexão foi detetado utilizando duas condições. Primeiro foi utilizado o perfil do gradiente para escolher os valores de gradiente mais altos. A segunda condição foi baseada em um limiar de tonalidade de cinza, de modo que apenas os valores mais baixos no perfil de escala de cinza sejam considerados, visto que valores mais altos representam a rocha ficando mais escura quando esta está molhada. Usando estas duas condições combinadas a posição da linha de água pode ser detetada automaticamente. Em imagens com o problema de sombra referido anteriormente o segundo maior valor de gradiente foi considerado em vez do primeiro.

Dado que a superfície da pedra é quase vertical e aproximadamente perpendicular ao eixo da lente da câmara, é assumido uma relação linear entre as hastes e a posição da linha de água em pixéis. Assim sendo, o nível de água (H_m) pode ser obtido através da seguinte expressão:

$$H_m = a \times d_{pixel} + b \tag{2.3}$$

em que d_{pixel} corresponde à posição da linha de água em pixéis. O parâmetro a que corresponde à inclinação é dado pela relação milímetros-pixel ($mm.px^{-1}$), sendo que esta é constante em todas as imagens. Utilizando uma régua graduada na superfície da pedra foi obtido o valor de $a = 5{,}5 \times 10^{-3}\ mm.px^{-1}$. O parâmetro b foi obtido através de uma parte dos dados obtidos pelo transdutor de pressão dando o valor de -1,25 m [18].

Na figura 2.17 são apresentados os resultados obtidos no verão de 2015.



a) b)

Figura 2.17 - Resultados no verão de 2015: a) de meados de junho até final de julho; b) de 5 a 10 de julho [18].

Analisando os resultados apresentados na figura 2.17, verifica-se que para um longo período (figura 2.17 a)) os valores obtidos pelo sistema apresentam ligeiras variações em comparação com os valores do transdutor de pressão, sendo que a exatidão do sistema nos resultados obtidos é cerca de 3 cm. Relativamente aos resultados apresentados na figura 2.17 b) verifica-se quando existe um grande caudal de água os resultados obtidos pelo sistema tendem a ser ligeiramente superiores aos do transdutor, sendo que acontece o inverso quando existe um caudal baixo [18].

Como desvantagens, este sistema elimina muitas imagens dos seus testes, muitas delas (exemplo de chuva forte) podem indicar um aumento considerável do nível de água que seria importante de analisar em situações de inundação.

2.4. Técnica de conversão da dimensão do objeto para uma unidade de medição

Existem técnicas que permitem extrair informações geométricas e construir modelos tridimensionais a partir de imagens não calibradas de um determinado ambiente em apenas uma perspetiva. A técnica que será apresentada neste ponto é a chamada de *single-view metrology* [19]. Esta técnica apresenta-se através da medição de duas situações: comprimentos de segmentos em superfícies planas e distâncias entre pontos em relação a superfícies planas. Em muitos casos estes dois tipos de medições são suficientes para uma reconstrução tridimensional parcial ou completa do objeto ou ambiente observado [19].

Esta técnica foi projetada para trabalhar com uma estrutura não calibrada, ou seja, não há necessidade de que a posição da câmara ou os parâmetros internos da mesma sejam conhecidos ou calculados. Por outro lado, as restrições do ambiente, como ortogonalidade e paralelismo de estruturas, são exploradas, tornando a técnica especialmente adequada para ambientes contendo estrutura feitas por humanos, como elementos arquitetónicos ou padrões geométricos [19].

O modelo da câmara utilizado é apresentado na figura 2.18. Dada uma imagem de uma superfície plana, os pontos no plano da imagem podem ser mapeados em pontos correspondentes no plano real por meio de uma transformação chamada de homografia [19].

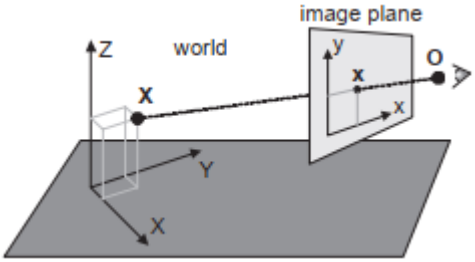


Figura 2.18 - Modelo da câmara [19].

Na figura 2.18, o ponto X no plano tridimensional é visto no plano da imagem no ponto x. As coordenadas euclidianas X, Y, Z e x, y, z são utilizadas para os sistemas de referência do plano real e o plano da imagem, respetivamente. O ponto O é o centro de projeção, ou seja, o centro ótico da câmara [19].

Os pontos do plano da imagem são mapeados nos pontos correspondentes no plano real através da seguinte expressão:

$$X = Hx \tag{2.4}$$

onde x é um ponto no plano da imagem, X é um ponto correspondente no plano real e H é uma matriz 3×3 que representa a transformação homográfica [19].

Portanto, conhecendo a matriz homográfica, qualquer ponto da imagem pode ser mapeado para o plano real e as distâncias entre pontos podem ser extraídas através do seguinte processo [19]:

1. Dada uma imagem de uma superfície plana, estima-se a matriz homográfica H;
2. Repetir:
    a. Selecionar dois pontos x_1 e x_2 no plano da imagem;
    b. Projetar cada ponto do plano da imagem para o plano real através da equação (2.4) para obter os dois pontos X_1 e X_2 no plano real;
    c. Calcular a distância euclidiana $d(X_1; X_2)$

O único problema é estimar a matriz homográfica H. No caso de câmaras não calibradas, a estimativa precisa da homografia entre o plano da imagem e o plano real pode ser conseguida diretamente a partir de um conjunto de pontos ou linhas c onhecidas que fazem correspondência entre os dois planos [19].

Considerando um plano xy (ou seja, $z = 1$ e $Z = 1$), através da equação (2.4) a correspondência de pontos entre o plano da imagem e o plano real pode ser representada através da seguinte expressão:

$$\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.5}$$

Desenvolvendo a expressão (2.5) tem-se um conjunto de três equações, sendo estas:

$$\begin{cases} X = h_{11}x + h_{12}y + h_{13} \\ Y = h_{21}x + h_{22}y + h_{23} \\ h_{31}x + h_{32}y + h_{33} = 1 \end{cases} \tag{2.6}$$

Multiplicando ambos os membros da primeira equação com a última equação e a segunda com a última da expressão (2.6), tem-se duas equações com os elementos da matriz H, sendo estas:

$$h_{31}xX + h_{32}yX + h_{33}X = h_{11}x + h_{12}y + h_{13} \quad (2.7)$$

$$h_{31}xY + h_{32}yY + h_{33}Y = h_{21}x + h_{22}y + h_{23} \quad (2.8)$$

Para n correspondências de pontos obtém-se um sistema de $2n$ equações. Se $n = 4$, como apresentado na figura 2.19, então obtém-se uma solução exata. Caso contrário, se $n > 4$, a matriz é sobre determinada, e a sua estimativa é feita por um esquema de minimização adequado [19]. A utilização de um valor $n > 4$ permite minimizar os erros cometidos na determinação das distâncias no plano real para obtenção de H.

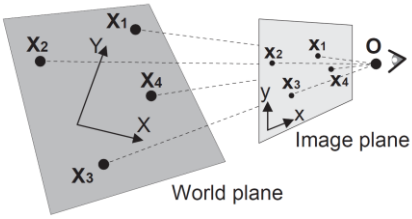


Figura 2.19 - Plano real e plano da imagem com quatro pontos correspondentes necessários para determinar a matriz homográfica entre os dois planos [19].

2.5. OpenCV

O OpenCV (*Open Source Computer Vision Library*) é uma biblioteca *open source* de *software* de visão computacional e *machine learning*. Esta biblioteca foi construída para fornecer uma infraestrutura comum para aplicações de visão computacional e acelerar o uso da perceção da máquina nos produtos comerciais. Sendo um produto licenciado isto torna mais fácil para as empresas utilizar e modificar o código [20].

Esta biblioteca possui mais de 2500 algoritmos otimizados, que incluem um conjunto de algoritmos de visão computacional e *machine learning* clássicos e de última geração. Estes algoritmos podem ser usados para: detetar e reconhecer rostos, identificar objetos, classificar ações humanas em vídeos, rastrear os movimentos da câmara, rastrear objetos em movimento, extrair modelos 3D de objetos, unir imagens para produzir uma imagem de alta resolução, encontrar imagens semelhantes em um banco de dados de imagens, remover olhos vermelhos de imagens tiradas com flash, seguir os movimentos dos olhos, reconhecer cenários e estabelecer marcadores para sobrepô-los com realidade aumentada, etc. A biblioteca é amplamente utilizada em empresas, grupos de pesquisa e por órgãos governamentais [20].Possui interfaces C++, C, Python, Java e Matlab e suporta Windows, Linux, Android, iOS e Mac OS. O OpenCV é escrito nativamente em C++ para aproveitar o processamento multi-core [20].

Para instalar esta biblioteca em Windows é necessário abrir a linha de comandos e digitar o seguinte: *pip install opencv-python*, para poder utilizar esta biblioteca em Python. Para importar a mesma em Python é necessário digitar o seguinte: *import cv2* [21].

2.6. Processamento de imagem para determinação de bordas

Nesta secção serão apresentados vários métodos de processamento de imagem que permitem detetar e representar bordas em imagens.

2.6.1. Deteção de bordas

Existem vários métodos que permitem evidenciar bordas de uma determinada imagem, sendo que os mais conhecidos são: Canny, Sobel e Prewitt.

O algoritmo de deteção de bordas Canny foi proposto por John Canny em 1986, visando satisfazer os três critérios gerais de deteção de bordas [22] [23]:

- Baixa taxa de erro: significa que este método apenas deteta bordas existentes;
- Boa localização: a diferença entre os pixéis reais da borda e os pixéis da borda detetados deve ser minimizada;
- Resposta mínima: a borda detetada deve ser marcada apenas uma vez, não muitas vezes.

As etapas de deteção de bordas através deste método são as seguintes:

1. Remoção de qualquer ruído da imagem através de um filtro. Normalmente é utilizado um filtro gaussiano, visto que o ruído gaussiano é o ruído mais comum em sistemas de imagem.
2. Encontrar o gradiente da imagem:
    a. Os gradientes ao longo das direções x e y são calculados utilizando máscaras de convolução, sendo definidos por:
    $$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{2.9}$$
    b. A amplitude do gradiente e a direção da borda são calculadas por:
    $$G = \sqrt{G_x^2 + G_y^2} \qquad \theta = \tan^{-1}\left(\frac{G_y}{G_x}\right) \tag{2.10}$$
3. Executar a supressão não máxima. Esta etapa serve para evitar bordas falsas e, portanto, é feita a supressão de qualquer valor de pixel que não seja considerado como fazendo parte de uma borda.
4. Executar o limiar de histerese. Nesta etapa final são usados dois limites (superior e inferior), permitindo assim ter estas três situações seguintes:
    a. Se o valor do gradiente de um pixel for maior do que o limite superior, o pixel é considerado um pixel de borda;
    b. Se o valor do gradiente de um pixel for menor que o limite inferior, o pixel será rejeitado;
    c. Se o valor do gradiente do pixel estiver entre os limites inferior e superior o pixel será aceite apenas se estiver conectado a um pixel que está acima do limite superior.

Sobel é outro método de deteção de bordas que realiza uma medição do gradiente espacial bidimensional das imagens, o que transfere uma matriz de pixéis bidimensional para um conjunto de dados estatisticamente não correlacionados para eliminar dados repetidos e, portanto, reduzir a quantidade de dados que é necessário para representar uma imagem digital [24].

Este método utiliza um par de máscaras de convolução, sendo que uma estima o gradiente no eixo x e outra estima o gradiente no eixo y. É um método de alta sensibilidade ao ruído em imagens permitindo assim destacar eficientemente as bordas [24].

As máscaras de convolução de Sobel são as seguintes [24]:

$$G_x = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \qquad G_y = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{2.11}$$

Em cada pixel há um par de números S_1 e S_2 que correspondem à saída da máscara (linha e coluna, subsequentemente). Esses números são utilizados para calcular duas matrizes: a amplitude da borda e a orientação, aplicando as seguintes equações [24]:

$$Edge\ magnitude = \sqrt{S_1^2 + S_2^2} \tag{2.12}$$

$$Edge\ direction = \tan^{-1}\left(\frac{S_1}{S_2}\right) \tag{2.13}$$

O método de deteção de bordas Prewitt é um método de diferenciação discreto, sendo que este calcula a aproximação da função de amplitude do gradiente. O resultado deste método é o vetor gradiente correspondente ou a normal desse vetor [25].

Este método é computacionalmente mais rápido para a deteção de bordas, sendo que é apropriado apenas para imagens bem contrastadas. A diferença entre o método Prewitt e o Sobel é a resposta espacial [25].

A aproximação calculada por este método é aplicada às derivadas da função de amplitude. Isto irá resultar em bordas onde o gradiente da função de amplitude tem um valor máximo. Este método deteta dois tipos de bordas: bordas horizontais e verticais. As máscaras de convolução para a técnica de deteção de bordas devem ter as seguintes propriedades [25]:

- A máscara deve conter sinais opostos;
- A soma da máscara deve ser igual a zero.

As máscaras de convolução mais usadas neste método são as seguintes [25]:

$$G_x = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \qquad G_y = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \tag{2.14}$$

2.6.2. Transformada de Hough

A transformada de Hough é uma técnica que foi inventada em 1962 por Paul Hough para permitir extrair qualquer forma/borda de uma imagem [26] [27]. Neste trabalho a transformada de Hough será utilizada para determinar uma linha na borda superior da ribeira.

Esta transformada pode ser descrita como uma função de mapeamento que converte um ponto no espaço da imagem (espaço xy) em uma linha ou uma sinusoide no espaço Hough [26] [27].

O espaço da imagem é o espaço de todos os pixéis pertencentes a uma imagem de onde se quer extrair um determinado recurso. Neste espaço uma linha ou aresta é definida pela seguinte equação [26] [27]:

$$y = ax + b \tag{2.15}$$

onde a representa a inclinação da linha e b é o valor aonde a linha interseta o eixo y.

Na equação (2.15) os valores de posição $(x, y) \in \mathbb{R}^2$ são definidos e bem conhecidos, sendo que as restantes incógnitas não são conhecidas (a e b) O objetivo de usar a transformada de Hough é criar um espaço aonde essas duas incógnitas possam ser determinadas [26] [27].

O espaço Hough é um espaço híbrido paramétrico, visto que está ligado a imagem, mas não tem realidade física em comparação com a imagem. Este espaço é composto por dois parâmetros calculados usando a base do espaço bidimensional (θ, ρ).É possível expressar as coordenadas (x, y) da imagem na base apresentada anteriormente através das seguintes equações [26] [27]:

$$\begin{cases} x = \rho\cos(\theta) \\ y = \rho\sin(\theta) \end{cases} \quad com \quad \begin{cases} \rho = \sqrt{x^2 + y^2} \\ \theta = \tan^{-1}\left(\frac{x}{y}\right) \end{cases}, \quad (x, y) \in \mathbb{R}^{+2} \tag{2.16}$$

Na imagem 2.20 é apresentada uma ilustração de várias sinusoides no plano Hough (θ, ρ) associadas a cada um dos pontos no plano da imagem (plano xy).

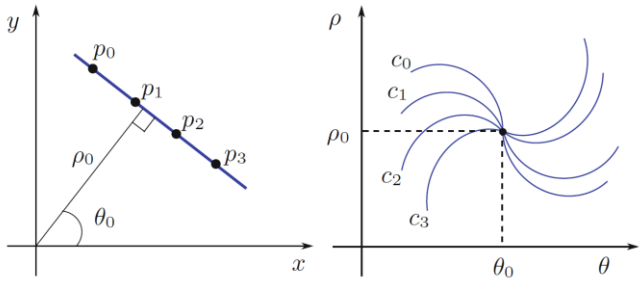


Figura 2.20 - Transformação de pontos colineares no plano xy para o plano Hough (θ, ρ) [28].

Assim, é possível associar a cada ponto do espaço da imagem, que pertença a uma aresta ou linha, uma onda sinusoidal no espaço de Hough usando a seguinte transformação [29]:

$$\rho = x\cos(\theta) + y\sin(\theta) \tag{2.17}$$

em que ρ é a distância perpendicular da origem até à linha e θ é o ângulo formado por esta linha e pelo eixo dos xx, como mostra a figura 2.21. Esta direção varia dependendo de como se representa o sistema de coordenadas (referência). Esta representação é a usada no OpenCV [29].

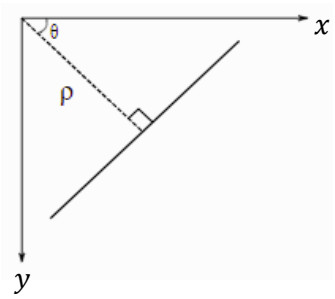


Figura 2.21 - Representação usada no OpenCV para a obtenção dos parâmetros da transformada de Hough [30].

3. Desenvolvimento do sistema

Neste capítulo serão apresentados, primeiramente, os requisitos do sistema que irão permitir, a partir dos mesmos, efetuar o desenvolvimento do trabalho. De seguida será apresentado o desenvolvimento do sistema, começando por descrever o sistema de medição e depois o algoritmo, que permite obter a altura do caudal da ribeira.

3.1. Requisitos do algoritmo

Com este trabalho pretende-se desenvolver um algoritmo de monitorização automático de deteção de riscos ambientais, neste caso concreto, um algoritmo de medição do nível do caudal através da aquisição de imagens por um sistema.

Este sistema efetua a aquisição de imagens, a uma determinada cadência, e envia as mesmas via *Wi-Fi* para um servidor situado na Universidade da Madeira.

O algoritmo a ser desenvolvido tem certos requisitos, tais como:

- Ser desenvolvido na linguagem de programação Python;
- Permitir a leitura das imagens provenientes do sistema;
- Permitir efetuar uma média temporal das imagens de forma a melhorar a qualidade das mesmas;
- As imagens deverão ser compensadas em rotação e translação de modo que a região de interesse de obtenção da linha de água esteja bem definida;
- Permitir detetar a linha de água independentemente das condições climatéricas;
- Armazenar, num ficheiro de texto, o valor do nível de caudal numa unidade de medição, neste caso, centímetros;
- Na ocorrência de erros o algoritmo deverá continuar a sua execução notificando a origem dos mesmos.

3.2. Sistema de medição

O sistema de aquisição de imagem utiliza um Raspberry Pi 3 modelo B e uma câmara Pi NoIR V1 [31]. Esta câmara de infravermelhos permite operar durante o dia e a noite em diferentes condições de luminosidade.

A câmara foi instalada no teto de uma varanda de um andar, num prédio em frente à ribeira de Santa Luzia, situada no concelho do Funchal. Esta forma de instalação da câmara tem várias vantagens, tais como:

- Como a câmara fica por baixo de uma varanda, ela está protegida da chuva, o que evita pingos de chuva na lente da câmara;
- Não foi necessário instalar um mastro para suportar a câmara, minimizando o impacto ambiental da solução;
- O sistema utiliza a rede *Wi-Fi* da casa onde está instalada a câmara, evitando a instalação de um sistema de comunicação dedicado;
- O sistema pode ser alimentado a partir da rede elétrica do edifício ou a partir de uma fonte de energia renovável.

Na figura 3.1 é apresentado o sistema de aquisição de imagem instalado próximo da ribeira de Santa Luzia, sendo esta ribeira um dos três principais cursos de água do Funchal com um elevado risco de inundação.

Figura 3.1 - Sistema de aquisição de imagem instalado na ribeira de Santa Luzia.

Neste trabalho foi utilizado um painel fotovoltaico de 80 W, instalado no terraço do edifício, e foi utilizada uma bateria de 100 Ah @ 12 V como sistema de armazenamento. Para efetuar a conversão da tensão da bateria (12 V) para 5 V para poder alimentar o Raspberry Pi foi utilizado um conversor DC-DC, neste caso, o conversor Pololu S18V20F5 [32]. Na figura 3.2 é apresentada a arquitetura do sistema implementado.

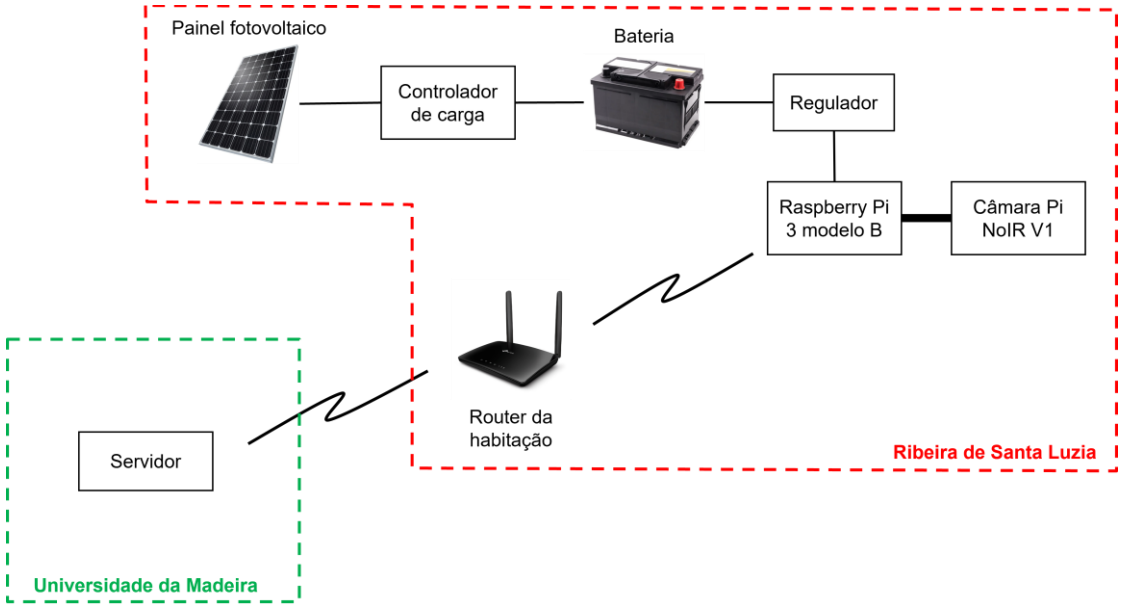


Figura 3.2 - Arquitetura do sistema implementado.

Em relação à arquitetura apresentada na figura 3.2, a informação que foi enviada para o servidor é uma imagem de uma das paredes da ribeira, sendo a mesma enviada pelo protocolo FTP (*File Transfer Protocol*).

É de referir que o sistema anteriormente apresentado foi instalado no âmbito de um projeto para monitorização de riscos ambientais (Vimetri – MAC), estando o mesmo já em funcionamento no decurso deste trabalho, por forma a dispor de informação suficiente para a testagem do algoritmo implementado de deteção da altura do nível do caudal.

3.3. Calibração da câmara

Relativamente à câmara utilizada neste trabalho, a Pi NoIR V1, os principais parâmetros especificados pelo fabricante são os seguintes [33]:

- Resolução do sensor: 2592×1944 pixéis;
- Tamanho do pixel: $1{,}4\ \mu m \times 1{,}4\ \mu m$;
- Comprimento focal: $(3{,}6 \pm 0{,}01)\ mm$.

Na figura 3.3 é apresentado um esquema em que permite efetuar uma correspondência entre um objeto visualizado no plano da imagem e no plano real.

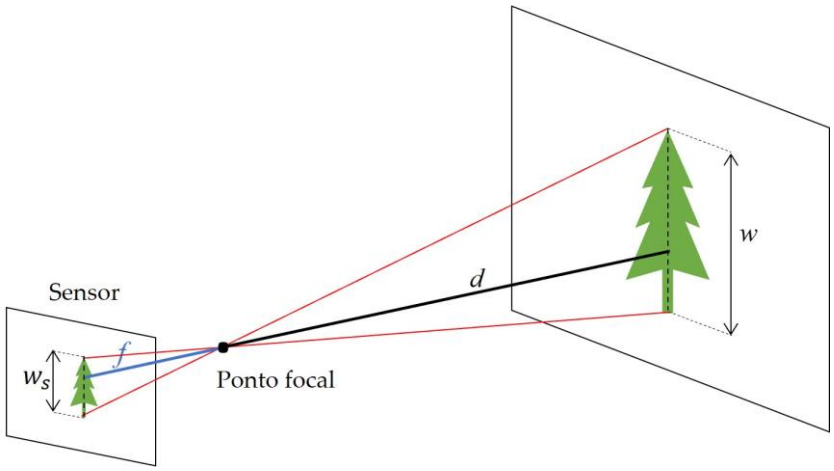


Figura 3.3 - Correspondência entre um objeto visualizado no plano da imagem e no plano real.

Utilizando o esquema da figura 3.3 e por considerações geométricas, a dimensão de um objeto num plano paralelo ao plano da câmara é dada pela seguinte expressão:

$$w = \frac{P \times T \times d}{f} = \frac{w_s \times d}{f} \tag{3.1}$$

em que P é a dimensão do objeto em pixéis, T é o tamanho do pixel em milímetros, d é a distância entre a lente da câmara e o plano do objeto em metros, w é a dimensão do objeto em metros, w_s é a dimensão do objeto na imagem em milímetros e f é o comprimento focal em milímetros.

Utilizando a equação (3.1) e as distâncias conhecidas efetuou-se uma experiência utilizando uma câmara NoIR para confirmar o comprimento focal. Assim sendo, obteve-se um valor para o comprimento focal muito próximo do valor fornecido pelo fabricante.

Para definir a região de interesse utilizada para medir a altura do caudal da ribeira foi necessário determinar certos parâmetros locais. Na figura 3.4 é apresentada uma imagem obtida pela câmara com uma resolução de 1280×720.

Figura 3.4 - Imagem retirada pela câmara.

Na câmara, foi necessário efetuar a alteração da proporção da tela de forma a obter esta resolução, que corresponde a um corte de 75% do tamanho total do sensor na dimensão vertical. A nível do processamento de dados, a imagem captada deve incluir toda a região vertical da parede e uma parte correspondente à zona de água. A partir de experiências iniciais verificou-se que a parede da ribeira tem nove metros de altura e uma inclinação de $5{,}8°$ em relação à vertical. Como a ribeira encontra-se entre duas estradas rodoviárias e por razões de privacidade, foi efetuado um corte de 15% no topo da imagem para que não ocorra a captação de veículos.

De seguida as imagens foram convertidas para a resolução de 1280×720 pixéis. A região de interesse utilizada nas medições da altura do caudal da ribeira é apresentada na figura 3.4 por um retângulo vermelho. A figura também inclui padrões utilizados neste trabalho, definidos pelos retângulos laranja e azul. A região de interesse inclui toda a dimensão vertical da face de betão. Pelas medições efetuadas foi obtido que a zona de betão tem dois metros de altura.

O acesso ao fluxo de água na ribeira para efetuar medições apresentou ser uma tarefa difícil. Assim sendo, a distância entre a câmara e a parede da ribeira foi determinada através de um método indireto. O sistema de referência usado nos cálculos é apresentado na figura 3.5.

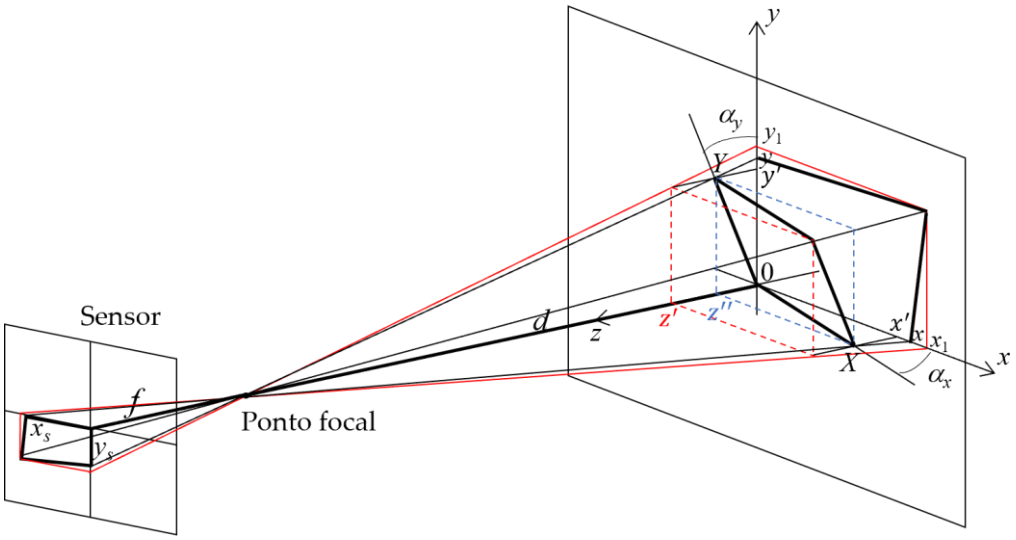


Figura 3.5 - Sistema de referência usado para relacionar um ponto no plano da imagem com um ponto no plano do objeto.

O plano XY é o plano do objeto e o xy é o plano da imagem. O objetivo era a obtenção da distância d, o ângulo horizontal α_x e o ângulo vertical α_y para caracterizar o plano da parede da ribeira. Estes ângulos foram obtidos nos planos xz e yz, respetivamente. O desenvolvimento deste novo método revelou-se necessário, visto que o objetivo era simplificar a parametrização necessária para obter o nível do caudal, dado pelos parâmetros referidos anteriormente.

As distâncias horizontais e verticais foram obtidas a partir da equação (3.1), dando:

$$x = x_s \frac{d}{f} \qquad y = y_s \frac{d}{f} \tag{3.2}$$

com

$$x_s = P_x \times T \times \frac{S_H}{R_H} \qquad y_s = P_y \times T \times C_1 \times C_2 \times \frac{S_V}{R_V} \tag{3.3}$$

em que S_H e S_V são o tamanho do sensor na horizontal (2592 pixéis) e na vertical (1944 pixéis), respetivamente, R_H e R_V são a resolução da imagem na horizontal (1280 pixéis) e na vertical (720 pixéis), respetivamente, C_1 é o corte da imagem devido à alteração da proporção da tela (0,75) e C_2 é o segundo corte da imagem (0,85). Utilizando a representação geométrica da figura 3.4 e considerando que $x'/(d - z'') = x/d$, $x' = X\cos(\alpha_x)$ e $z'' = X\sin(\alpha_x)$, a distância X pode ser obtida a partir de x usando a seguinte expressão:

$$X = \frac{xd}{d\cos(\alpha_x) + x\sin(\alpha_x)} \tag{3.4}$$

Da mesma forma, considerando que $y'/(d - z'') = y/d$, $y' = Y\cos(\alpha_y)$ e $z'' = Y\sin(\alpha_y)$, a distância Y é obtida pela seguinte expressão:

$$Y = \frac{yd}{d\cos(\alpha_y) + y\sin(\alpha_y)} \tag{3.5}$$

Para fins de calibração com distâncias reais na região de interesse, uma régua graduada de dez metros de comprimento foi colocada verticalmente na parede em diferentes posições horizontais, tal como é apresentado na figura 3.6. Estas medições também permitiram fornecer valores para avaliar o erro cometido pela técnica proposta para medição da altura do caudal da ribeira. A partir de duas distâncias obtidas na imagem em direções opostas em torno da origem ($P_{y_a} = 230$ pixéis e $P_{y_b} = -149$ pixéis), foram definidos dois valores verticais y_{sa} e y_{sb} utilizando a equação (3.3). As distâncias reais correspondentes Y_a (3,8 m) e Y_b (-2,8 m) foram obtidas a partir da régua graduada. Utilizando a equação (3.5) com estas duas distâncias, os parâmetros d e α_y foram determinados pela resolução de um sistema de duas equações. Os resultados obtidos foram $d = 23{,}68\ m$ e $\alpha_y = 27{,}2°$. Substituindo d na equação (3.4), α_x foi obtido a partir dos valores conhecidos de x (474) e X (8,5 m), dando o valor de 7,4°.

Figura 3.6 - Régua graduada colocada verticalmente na parede da ribeira.

Por forma a relacionar um ponto no plano xy com um ponto no plano XY, a equação do plano XY foi derivada usando a forma geral $ax + by + cz + d = 0$. As constantes a, b, c e d foram obtidas utilizando três pontos no plano. Os pontos foram $(0,0,0)$, $(0, y', z'')$ e $(x', 0, z'')$. Substituindo cada um destes pontos na forma geral obtém-se um sistema de três equações, em que:

$$\begin{cases} d = 0 \\ b = -c\tan(\alpha_y) \\ a = -c\tan(\alpha_x) \end{cases}$$

Após obter a, b e d volta-se a substituir os mesmos na forma geral, resultando na seguinte equação do plano no sistema de referência xyz:

$$z = \tan(\alpha_x)\, x + \tan(\alpha_y)\, y \qquad (3.6)$$

Utilizando a representação geométrica da figura 3.4 e considerando que $x'/(d - z') = x_1/d$, em que $z' = \tan(\alpha_x)\, x' + \tan(\alpha_y)\, y'$, $x' = X\cos(\alpha_x)$ e $y' = Y\cos(\alpha_y)$, para um ponto X no plano do objeto, um ponto x_1 no plano da imagem é dado por:

$$x_1 = \frac{Xd\cos(\alpha_x)}{d - X\sin(\alpha_x) - Y\sin(\alpha_y)} \tag{3.7}$$

Da mesma forma, considerando que $y'/(d - z') = y_1/d$, em que $z' = \tan(\alpha_x)\,x' + \tan(\alpha_y)\,y'$, $x' = X\cos(\alpha_x)$ e $y' = Y\cos(\alpha_y)$, para um ponto Y no plano do objeto, um ponto y_1 no plano da imagem é dado por:

$$y_1 = \frac{Yd\cos(\alpha_y)}{d - X\sin(\alpha_x) - Y\sin(\alpha_y)} \tag{3.8}$$

Resolvendo estas duas últimas equações em ordem a X e Y, um ponto no plano do objeto pode ser determinado a partir de um ponto no plano da imagem.

3.4. Compensação da imagem

Nesta secção serão descritos os passos efetuados por forma a permitir obter uma determinada zona de interesse que permita, posteriormente, medir o nível de água da ribeira. No fluxograma da figura 3.7 são apresentados os vários processos efetuados. A compensação da imagem consiste em compensar a mesma em termos de rotação e translação, visto que o sistema de aquisição, devido ao vento ou a manutenção do equipamento, poderá provocar ligeiras alterações do posicionamento da câmara, logo foi necessário compensar essas oscilações. Esta compensação também permite conseguir um sistema de referência de forma a obter uma região de interesse que possibilita adquirir, posteriormente, a altura do caudal da ribeira.

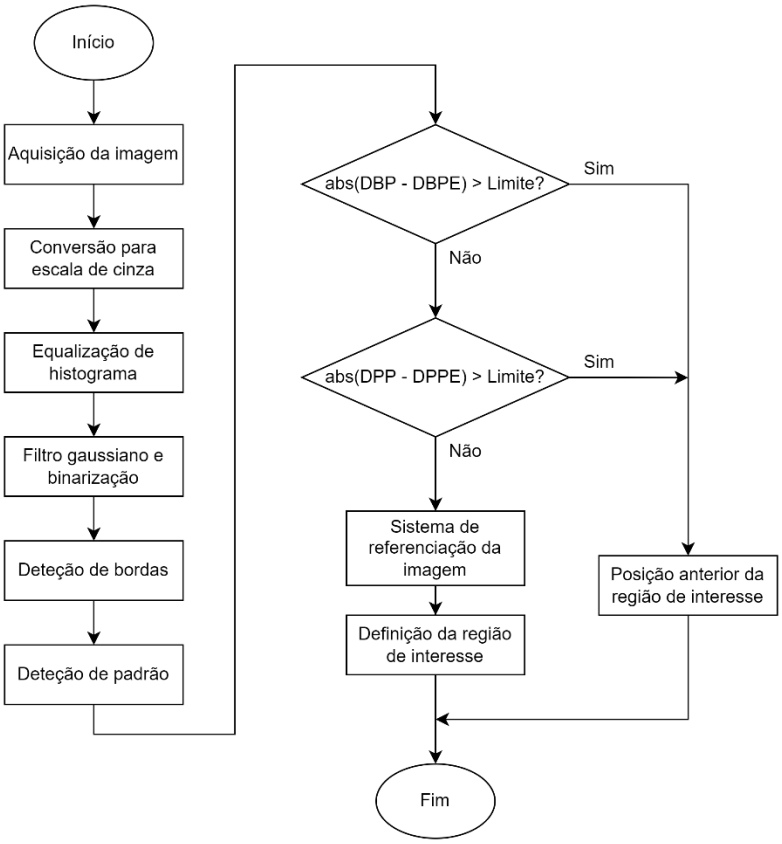


Figura 3.7 - Fluxograma do processo da calibração da imagem e de obtenção da região de interesse.

Primeiramente foi efetuada a aquisição da imagem através do sistema montado próximo da ribeira de interesse. De seguida foi efetuada a conversão da imagem para escala de cinza, visto que a imagem a cores não acrescenta informação relevante relativamente à linha de água, trabalhando assim com apenas um esquema de cores. Após isto, foi aplicada a equalização de histograma para permitir equilibrar a imagem em termos de contraste e para realçar certas zonas da imagem. Um filtro gaussiano permitiu eliminar o ruído gaussiano da imagem e a binarização permitiu realçar as bordas, para posteriormente aplicar um método de deteção de bordas, por forma a detetar uma das bordas superiores da ribeira para que seja possível efetuar uma compensação da imagem em termos de rotação. A deteção de padrão permitiu efetuar uma compensação na imagem em termos de translação.

Na maioria dos casos, o algoritmo deteta as bordas superiores da ribeira e a posição correta do padrão. No entanto, para imagens com condições de iluminação muito difíceis ou com existência de muita chuva, pode ocorrer uma deteção incorreta de um desses parâmetros ou de ambos. Nesse caso, o erro cometido por uma configuração da região de interesse incorreta pode resultar num erro muito elevado na deteção do nível de água da ribeira. Para evitar essa situação, foi determinada a distância entre a borda e a posição do padrão principal (DBP) e comparada com a distância esperada (DBPE) e também foi determinada a distância entre a posição do padrão principal e a posição de um padrão secundário (DPP) e comparada com a distância esperada (DPPE). Esta última comparação serviu para verificar se a inclinação da borda detetada estava dentro de um determinado limite, sendo que esta foi apenas efetuada durante o dia. Se uma destas diferenças ultrapassar um determinado limite, a região de interesse é obtida através dos parâmetros da última imagem em que foi efetuado uma compensação bem-sucedida. Como o algoritmo não sabe qual foi a borda detetada, a primeira comparação apresentada no fluxograma da figura 3.7 é realizada para dois valores do parâmetro limite e para dois valores de DBPE.

Para o desenvolvimento da aplicação foi utilizado o *software* Python 3.8.5. A nível do IDE (*Integrated Development Environment*) foi utlizado o Visual Studio Code [34] [35]. Ao longo de toda a aplicação foram utilizadas diversas funções que já estão documentadas através da biblioteca OpenCV e que permitiram facilitar certos dos processos efetuados [36].

3.4.1. Equalização de histograma

Após a conversão da imagem para a escala de cinza fez-se a equalização de histograma. A equalização de histograma é um dos métodos usados para melhorar o contraste de uma imagem devido à sua elevada eficiência e simplicidade. Isto é conseguido através da normalização da distribuição da intensidade da imagem usando a função de transformação cumulativa, de modo que a imagem resultante possa ter uma distribuição uniforme em termos de intensidade [37].

Foram considerados dois tipos de equalização de histograma neste trabalho, a equalização de histograma global e a equalização de histograma adaptativa limitada por contraste (CLAHE – *Constrast Limited Adaptative Histogram Equalization*).

No método de equalização de histograma global, o histograma da imagem em escala de cinza é usado para calcular a função de transformação do histograma. Como resultado, a gama dinâmica do histograma da imagem é achatada e estendida e o

contraste geral é melhorado [37]. Na figura 3.8 é apresentado como é efetuada a equalização de histograma global.

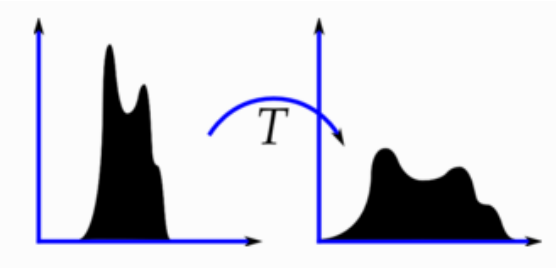


Figura 3.8 - Equalização de histograma global [38].

Em *python* a função que permite efetuar a equalização de histograma global é a seguinte:

image_equ = cv.equalizeHist(image_gray).

O parâmetro de entrada da função é a imagem (imagem de canal único de 8 *bits*) em escala de cinza que se pretende efetuar a equalização de histograma (*image_gray*). Tem-se como saída a imagem equalizada do mesmo tipo e tamanho que a imagem de entrada (*image_equ*) [39].

A equalização de histograma adaptativa limitada por contraste (CLAHE) difere da equalização de histograma global na sua limitação de contraste. Neste caso, o procedimento de limitação de contraste deve ser aplicado para cada região da qual uma função de transformação é derivada [40]. Este método foi desenvolvido para prevenir a amplificação do ruído que a equalização de histograma global pode gerar. Isto é conseguido limitando o aumento do contraste da equalização de histograma [40].

A CLAHE limita a amplificação através do corte do histograma em um valor predefinido antes de calcular a função de distribuição acumulada. O valor em que o histograma é cortado, chamado de *clip limit*, depende da normalização do histograma e, portanto, do tamanho da região utilizada [40]. A parte do histograma que excede o *clip limit* não é descartado, visto que é redistribuído igualmente ao longo do histograma. Na figura 3.9 é apresentado como é efetuada essa redistribuição. Esta redistribuição irá empurrar certos valores para além do *clip limit* (região sombreada a verde), resultando num *clip limit* efetivo maior do que o limite predefinido. Se esta situação for indesejável, o procedimento de redistribuição pode ser repetido recursivamente até que o excesso seja desprezável [40].

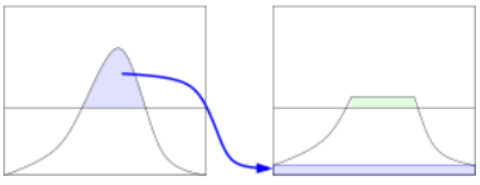


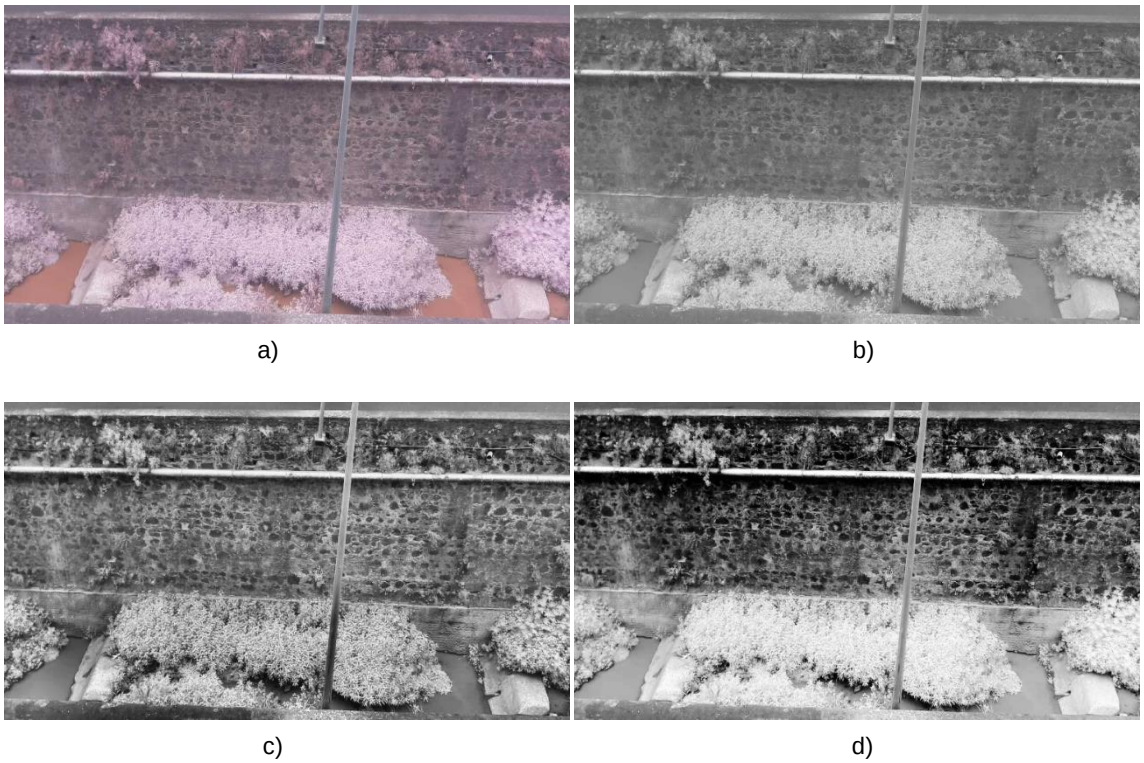
Figura 3.9 - Redistribuição do histograma [40].

Em *python* as funções que permitem efetuar a CLAHE podem ser utilizadas da seguinte forma:

clahe = cv.createCLAHE(clipLimit = 2.0, tileGridSize = (8,8));
image_clahe = clahe.apply(image_gray).

A primeira função permite instanciar o algoritmo CLAHE. Tem como parâmetros o *clip limit*, referido anteriormente, e o *tileGridSize* que permite dividir a imagem de entrada em blocos *M*×*N* e, em seguida, aplica a equalização de histograma a cada bloco. A segunda função permite aplicar a CLAHE a uma determinada imagem. Os parâmetros da primeira função são os tipicamente utilizados por defeito [39].

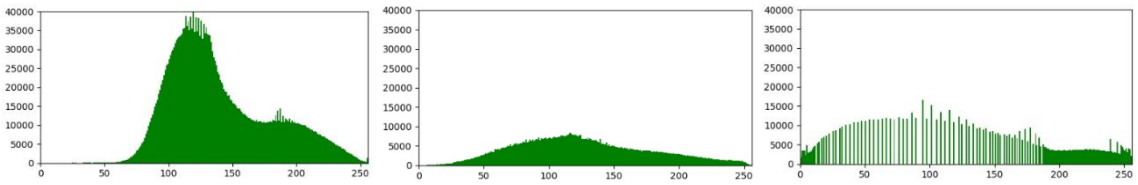
Para fazer uma comparação entre os dois métodos, estes foram aplicados às imagens obtidas na zona da ribeira. As figuras 3.10 a 3.13 são os resultados em duas situações, para uma imagem obtida durante o dia e para uma imagem obtida durante a noite. Na figura 3.10 tem-se o primeiro caso, apresentando-se a imagem original (figura 3.10 a)), a imagem em escala de cinza (figura 3.10 b)), a imagem obtida pelo método CLAHE (figura 3.10 c)) e a imagem obtida pela equalização global (figura 3.10 d)). Na figura 3.11 representam-se os histogramas respetivos. Observando os resultados obtidos pelos dois métodos de equalização, verifica-se que ambos os casos permitem melhorar o contraste, não existindo diferenças de detalhe relevantes entre as duas imagens.



a) b)

c) d)

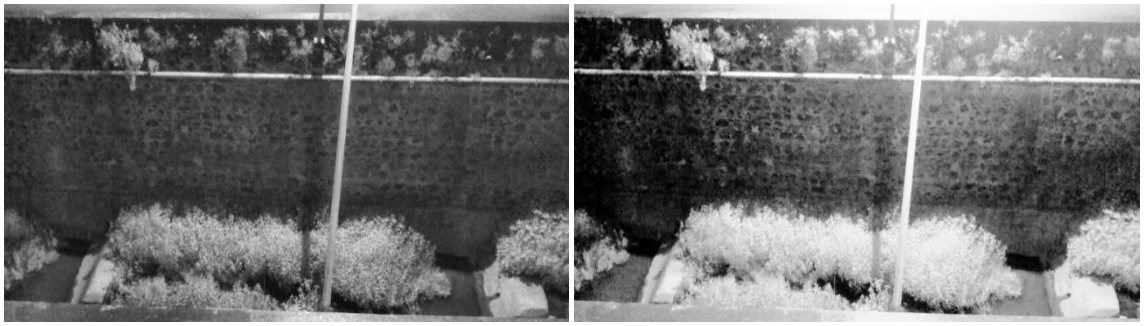
Figura 3.10 - Imagem durante o dia: a) original; b) escala de cinza; c) método CLAHE; d) método equalização global.



a) b) c)

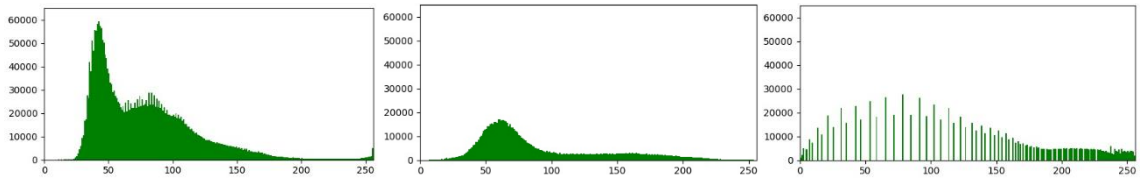
Figura 3.11 – Histograma da imagem durante o dia: a) escala de cinza; b) método CLAHE; c) método equalização global.

a) b)



c) d)

Figura 3.12 - Imagem durante a noite: a) original; b) escala de cinza; c) método CLAHE; d) método equalização global.



a) b) c)

Figura 3.13 - Histograma da imagem durante a noite: a) escala de cinza; b) método CLAHE; c) método equalização global.

Em relação aos resultados da figura 3.12 (imagem durante a noite) verifica-se que existem várias diferenças, visto que a equalização de histograma CLAHE faz um melhor balanceamento do contraste na imagem em geral, notando-se uma maior perceção dos pormenores da imagem em relação à obtida pela equalização global.

Apesar de não se evidenciar diferenças entre os dois métodos durante o dia, durante a noite já foi percetível, pelo que se optou pelo método CLAHE para a equalização de histograma em imagens.

3.4.2. Compensação do movimento

Neste ponto irá ser apresentado como se efetuou a compensação do movimento da câmara, neste caso, em termos de rotação e translação. Este movimento da câmara pode dever-se ao vento forte ou reposicionamento da câmara.

3.4.2.1. Compensação da rotação

Nesta secção serão descritos os passos efetuados para a compensação da imagem em termos de rotação, devido as ligeiras alterações do posicionamento da câmara do sistema de medição. No fluxograma da figura 3.14 são apresentados os vários processos efetuados.

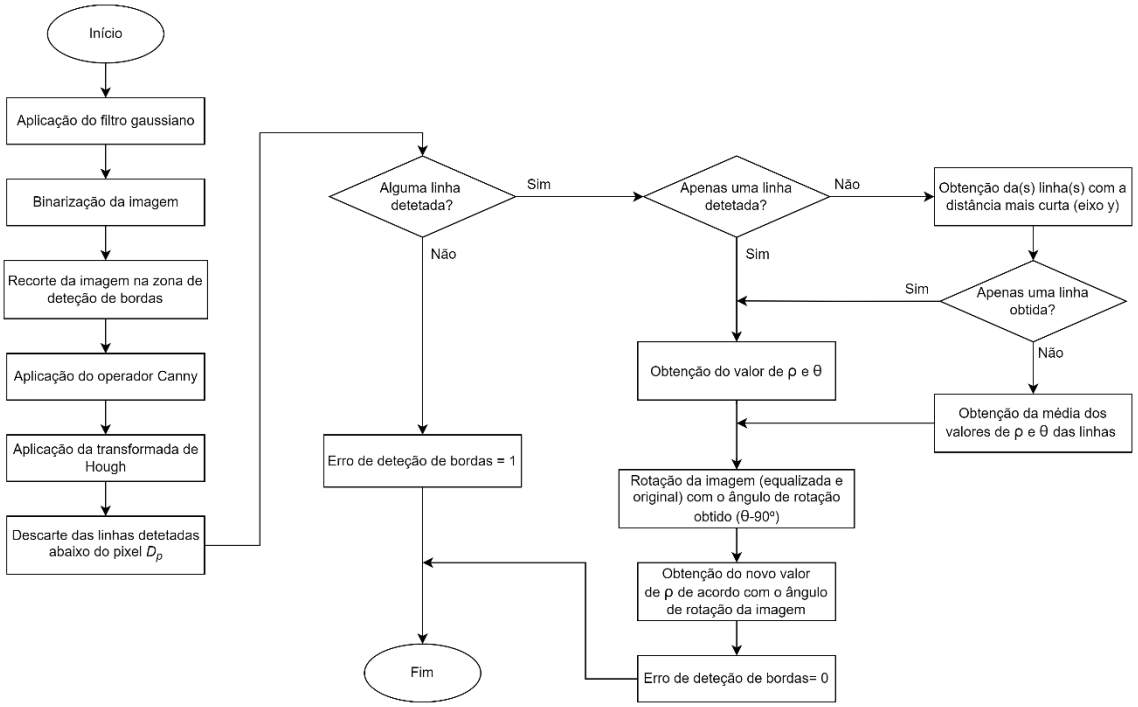


Figura 3.14 - Fluxograma do processo de compensação da imagem em termos de rotação.

Primeiro aplicou-se um filtro gaussiano à imagem depois de equalizada de forma a reduzir algum ruído da mesma. De seguida foi aplicada a binarização da imagem de forma a permitir realçar bordas de interesse. Com a imagem binária recortou-se uma zona da imagem que contempla as duas bordas superiores das ribeiras. Com essa zona da imagem aplicou-se o operador Canny e a transformada de Hough de forma a detetar uma das bordas superiores da ribeira, sendo que para isso descartou-se linhas detetadas abaixo do pixel D_p. No caso de ter sido detetado alguma linha/borda foi necessário saber quantas foram detetadas. Se apenas é detetada uma linha tem-se o valor de ρ e θ obtido pela transformada. Caso seja detetado mais do que uma linha foi necessário avaliar qual a que tem a distância, a meio da zona da imagem recortada, mais curta, no eixo vertical (y). Se existir mais do que uma linha com a mesma distância é efetuada a média dos valores de ρ e θ obtidos. Após isto é efetuado a rotação da imagem com o ângulo de rotação obtido. De seguida foi necessário obter o novo valor de ρ, visto que, dependendo do ângulo de rotação da imagem, este valor tende a ser diferente. Por fim foi enviada a informação numa variável caso ocorra erro na deteção de uma das bordas da ribeira. Nos próximos subcapítulos serão apresentados de uma forma mais concreta os passos efetuados nesta parte de compensação da rotação.

3.4.2.1.1. Filtro gaussiano

A filtragem no contexto de processamento de imagem é utilizada para reduzir o ruído e suavizar a imagem (também chamada de desfoque). A filtragem gaussiana consiste na convolução da matriz da imagem com uma matriz de pequenas dimensões conhecida por máscara ou *kernel*, que é preenchida utilizando a função de distribuição gaussiana bidimensional definida por [41]:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{3.9}$$

em que (x,y) é a posição de cada pixel da máscara e σ o desvio padrão. O desvio padrão controla a variação em torno de um valor médio da distribuição gaussiana, que determina a extensão do efeito de suavização em torno de um pixel. Em geral, quanto maior é a dimensão da máscara maior é o efeito de filtragem. No entanto, à medida que se afasta do valor médio da distribuição gaussiana os valores diminuem e torna-se desprezáveis, pelo que normalmente se utilizam máscaras de pequena dimensão [41].

Em *python* a função que permite aplicar um filtro gaussiano a uma determinada imagem é a seguinte:

image_gaussian = cv.GaussianBlur(image_gray, (3,3), 0).

Esta função tem como parâmetros a imagem que se pretende aplicar o filtro (*image_gray*), o tamanho da máscara (*(3,3)*) e o desvio padrão (*0*). Definindo o valor de 0 para o desvio padrão está-se a instruir à biblioteca OpenCV a calcular automaticamente o desvio padrão com base no tamanho da máscara [42].

Aplicando este tipo de filtro a uma imagem equalizada obtém-se o seguinte resultado apresentado na figura 3.15.

a)

b)

Figura 3.15 - Imagem durante o dia: a) equalizada (método CLAHE); b) após a aplicação do filtro gaussiano.

Observando a figura 3.15, apesar das diferenças existentes não serem muito nítidas, este tipo de filtro permite a redução do ruído existente na imagem, suavizando as bordas existentes na imagem, neste caso, as bordas superiores da ribeira.

3.4.2.1.2. Binarização

A binarização consiste na passagem para uma imagem binária. Este processo foi apenas aplicado para efetuar a deteção de uma das bordas superiores da ribeira.

Esta é a representação de imagem mais simples (vulgarmente conhecida por imagem a preto e branco) em que só tem dois valores possíveis para os seus pixéis (0 – preto, 255 – branco). A utilização deste tipo de imagens é útil no contexto do processamento de imagem, pois permite destacar melhor as características dos objetos e bordas e esconder certos detalhes não relevantes [43].

Existem duas categorias de binarização: binarização global e binarização adaptativa [43]. Na binarização global é escolhido um limiar a partir do qual os pixéis na escala de cinza são transformados em 0 ou 255. Se o valor de um pixel em escala de cinza for superior ao limiar, então o pixel passa a ser representado por 255 (branco), caso contrário passa a ser representado por 0 (preto) [43]. Esta conversão pode ser representada matematicamente por:

$$dst(x,y) = \begin{cases} 255 & if \quad src(x,y) > thresh \\ 0 & otherwise \end{cases} \tag{3.10}$$

em que $dst(x,y)$ representa um pixel binário na posição (x,y) da imagem, $src(x,y)$ representa um pixel em escala de cinza na posição (x,y) e $thresh$ representa o limiar utilizado (supondo que se está a considerar a representação dos pixéis por 8 *bits*) [44].

Em *python* a função que permite efetuar a binarização global é a seguinte:

ret, binary_image = cv.threshold(image_gray, 127, 255, cv.THRESH_BINARY).

A função tem como parâmetros a imagem que se pretende efetuar a binarização (*image_gray*), o limiar (*127*), o valor máximo que o pixel deve tomar (*255*) se a condição for satisfeita e o tipo de binarização a aplicar (*cv.THRESH_BINARY*), sendo que a escolhida efetua a mesma conversão que foi apresentada na expressão (3.10). Tem-se como saídas o valor do limiar utilizado (*ret*) e a imagem binária (*binary_image*). Os parâmetros apresentados são os tipicamente utilizados [44].

Quanto à binarização adaptativa, o conceito base é o mesmo, com a diferença da binarização ser aplicada a vários segmentos da imagem, utilizando valores limiares baseados na intensidade dos pixéis adjacentes. A vantagem desta técnica é a possibilidade de realizar uma melhor binarização em imagens com condições de iluminação variáveis [43].

Em *python* a função que permite efetuar a binarização adaptativa é a seguinte:

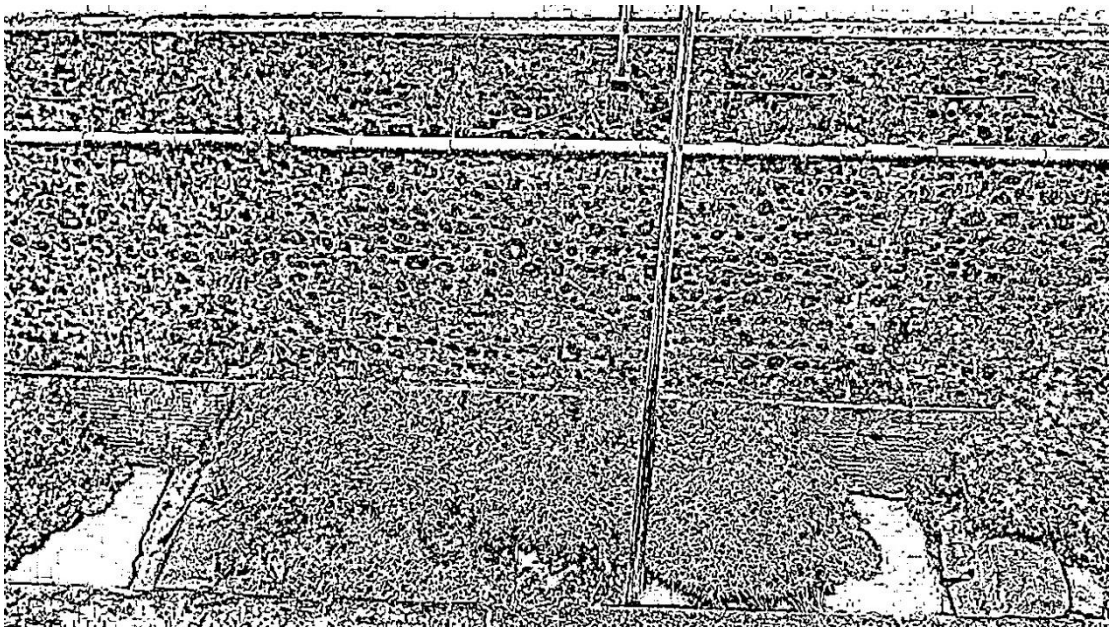
binary_image = cv.adaptiveThreshold(image_gray, 255,
cv.ADAPTIVE_THRESH_GAUSSIAN_C, cv.THRESH_BINARY, 33, 2).

Esta função tem como parâmetros a imagem que se pretende efetuar a binarização (*image_gray*), o valor máximo que o pixel deve tomar (*255*) se a condição da expressão (3.10) for satisfeita, o algoritmo a ser usado para obter o limiar a utilizar (*cv.ADAPTIVE_THRESH_GAUSSIAN_C*), o tipo de binarização a aplicar (*cv.THRESH_BINARY*), o tamanho da região que é usada para calcular o limiar a utilizar (*33*) e a constante subtraída da soma ponderada (*2*) [44].

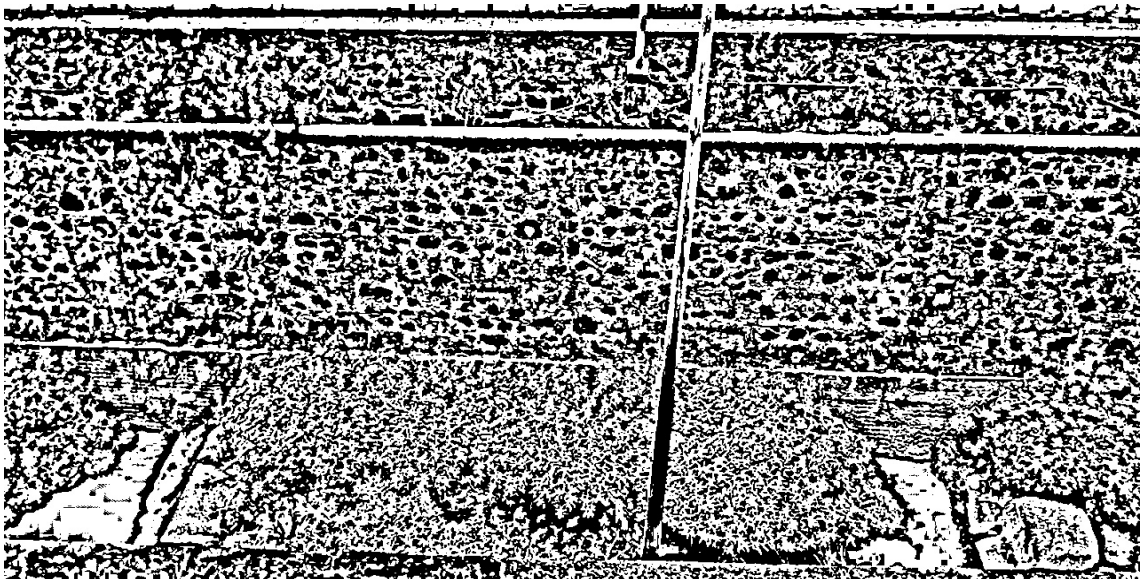
O limiar obtido no algoritmo apresentado na função (*cv.ADAPTIVE_THRESH_GAUSSIAN_C*) é uma soma ponderada (correlação cruzada com uma janela gaussiana) do tamanho da região menos a constante (neste caso *2*) [44].

Em relação ao parâmetro correspondente ao tamanho da região, este foi alterado de 11 (típico) para 33 por forma a permitir evidenciar mais as bordas superiores da ribeira em relação a outras bordas existentes.

Na figura 3.16 é apresentada para uma imagem durante o dia a diferença que existe utilizando os dois valores referidos anteriormente.



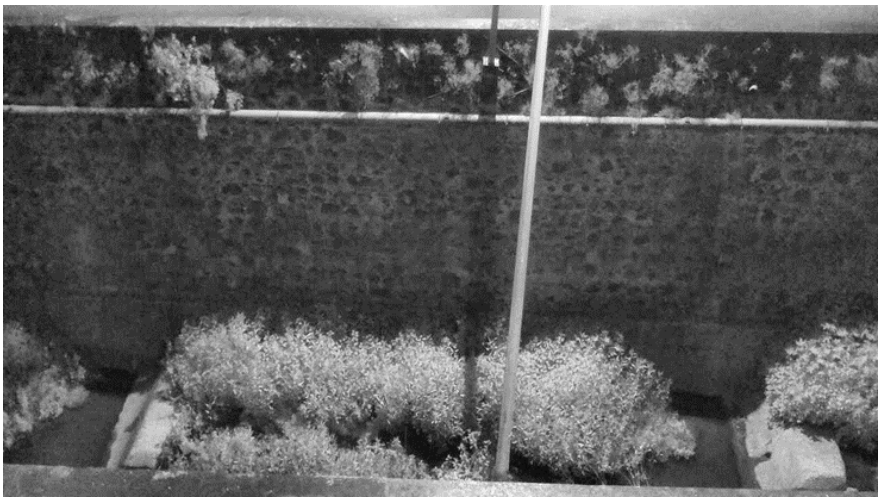
a)



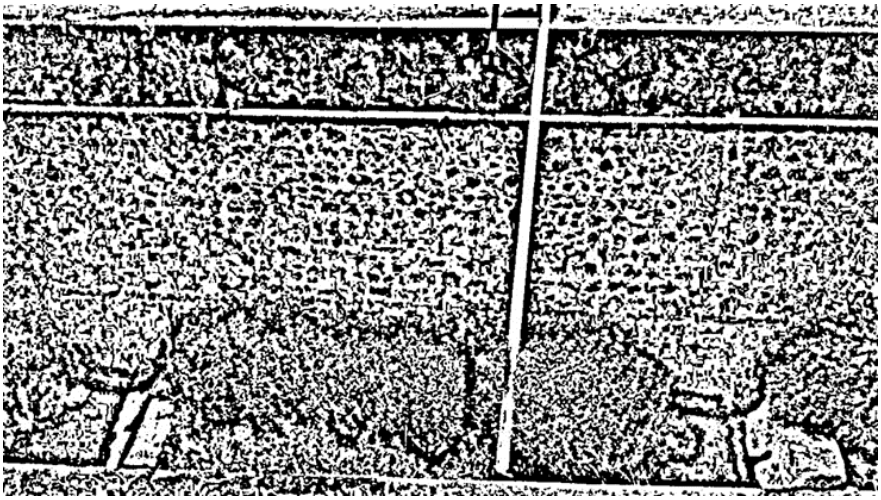
b)

Figura 3.16 - Imagem binarizada com um tamanho da região: a) 11; b) 33.

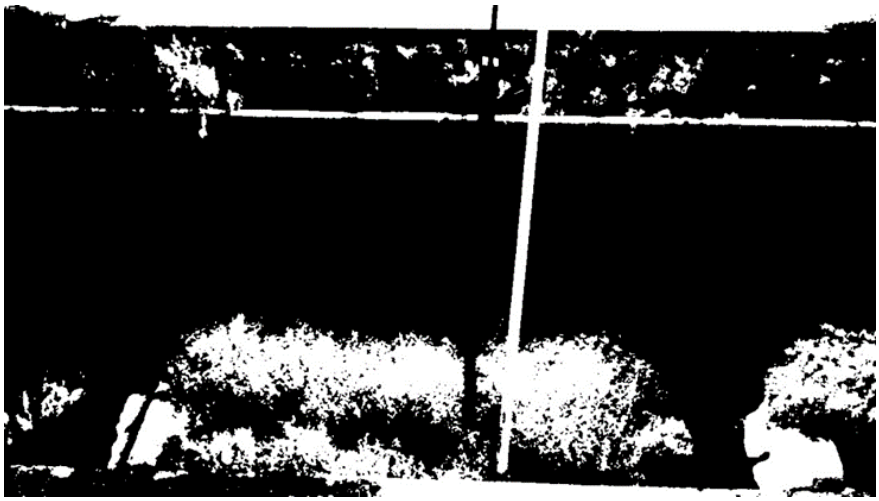
Na figura 3.17 a 3.19 são apresentados outros exemplos de imagens durante a noite, transição noite-dia e dia com os dois tipos de binarização apresentados anteriormente.



a)



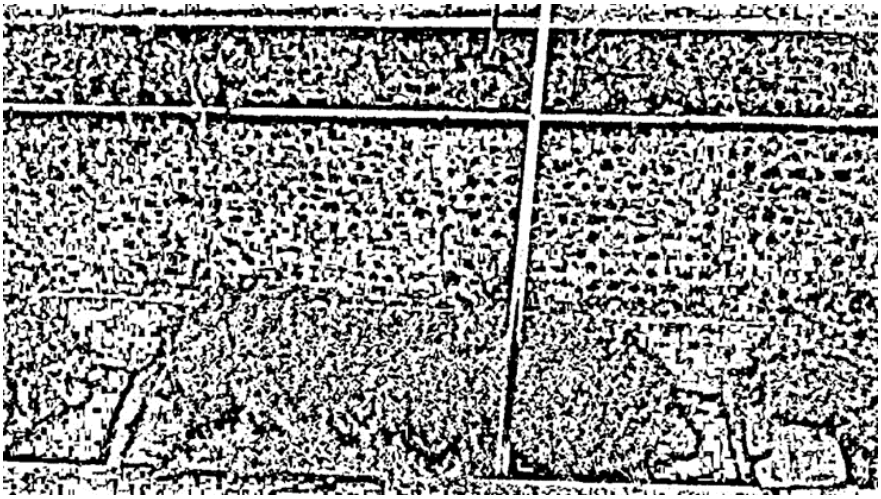
b)



c)

Figura 3.17 – Imagem durante a noite: a) equalizada; b) binária adaptativa; c) binária global

a)

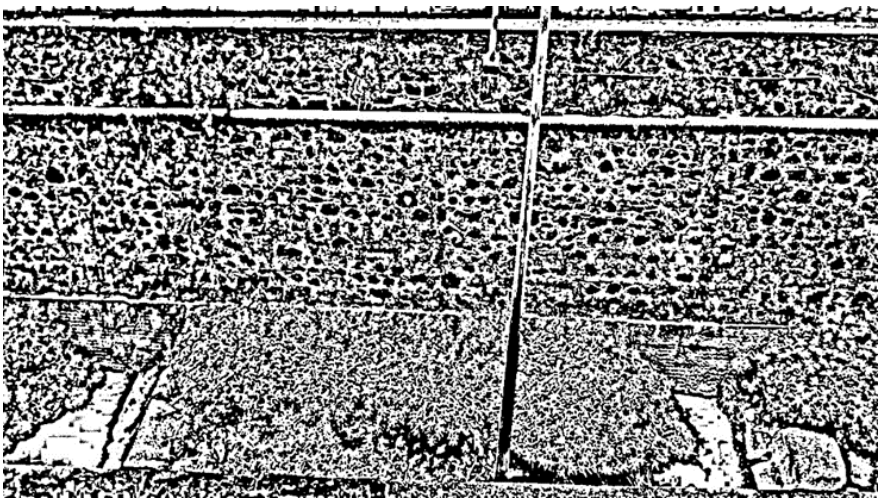


b)

c)

Figura 3.18 – Imagem durante a transição noite-dia: a) equalizada; b) binária adaptativa; c) binária global

a)



b)

c)

Figura 3.19 – Imagem durante o dia: a) equalizada; b) binária adaptativa; c) binária global

Nas imagens durante a noite (figura 3.17) verifica-se que para a binarização global os extremos na zona das bordas apresentam bastante ruído, não evidenciando as

bordas. Na binarização adaptativa, nos extremos da imagem já é possível visualizar as bordas superiores da ribeira.

Nas imagens durante a transição noite-dia (figura 3.18) verifica-se que se perde a zona onde se localiza as bordas superiores da ribeira com a binarização global. Esta situação advém do facto que, como a imagem durante a transição é relativamente escura, os pixéis na zona da borda superior tendem a ter o valor de 0 (preto). Na binarização adaptativa verifica-se que permite evidenciar bem a borda superior da ribeira.

Relativamente às imagens durante o dia (figura 3.19), verifica-se que ambos os métodos de binarização podem evidenciar as bordas superiores da ribeira.

Considerando os três tipos de imagens (dia, noite e transição noite-dia) o método que permite evidenciar mais as bordas é o de binarização adaptativa, sendo que será este o adotado no algoritmo.

3.4.2.1.3. Deteção de bordas

Para poder efetuar uma correta deteção de uma das bordas superiores da ribeira foi necessário utilizar um método de deteção de bordas que permita evidenciar uma das bordas de interesse. Neste caso, os três métodos mais conhecidos de deteção de bordas são: Canny, Sobel e Prewitt.

Em *python* a função que permite aplicar o método de deteção de bordas Canny é a seguinte:

edges_canny = cv.Canny(cropped_image, 100, 200).

Nesta função tem-se como parâmetros a imagem que se pretende aplicar o método (*cropped_image*), o limite de histerese inferior (*100*) e o limite de histerese superior (*200*). Estes dois últimos parâmetros são os tipicamente utilizados [45].

Relativamente ao método de deteção de bordas Sobel, as funções em *python* são as seguintes:

edges_sobelx = cv.Sobel(cropped_image, cv.CV_8U, 1, 0, ksize = 3);

edges_sobely = cv.Sobel(cropped_image, cv.CV_8U, 0, 1, ksize = 3);

edges_sobel = edges_sobelx + edges_sobely.

A primeira função permite identificar as bordas horizontais e a segunda função permite identificar as bordas verticais. Apenas a primeira função foi aplicada na situação de interesse, visto que se pretende detetar as bordas superiores da ribeira.

Nas primeiras duas funções tem-se como parâmetros a imagem que se pretende aplicar o método (*cropped_image*) e a profundidade da imagem de destino (*cv.CV_8U* - sinal inteiro de 1 *byte*, ou seja, um pixel pode ter valores de 0 a 255). O terceiro parâmetro é a ordem da derivada x (gradiente) e o quarto parâmetro é a ordem da derivada y (gradiente). Ao calcular *edges_sobelx*, define-se a derivada x como 1 e a derivada y como 0, e ao calcular *edges_sobely* o caso será invertido. O último parâmetro é o tamanho da máscara (*kernel*) Sobel, que tem o valor de 3 [42].

No caso do método de deteção de bordas Prewitt tem-se as seguintes funções:

kernel_x = np.array([[1,1,1],[0,0,0],[-1,-1,-1]]);

kernel_y = np.array([[-1,0,1],[-1,0,1],[-1,0,1]]);

edges_prewitt = cv.filter2D(cropped_image, -1, kernel_x+kernel_y).

As primeiras duas funções permitem a criação das máscaras de convolução enquanto a última função permite aplicar o método Prewitt. Mais uma vez, tal como foi referido no método Sobel, das duas primeiras funções, apenas a primeira foi aplicada, visto que permite realçar as bordas horizontais. Na última função tem-se como parâmetros a imagem que se pretende aplicar o método (*cropped_image*), a profundidade da imagem de destino (*-1* – a imagem de destino terá a mesma profundidade da imagem de entrada) e a máscara de convolução (*kernel_x*) [42].

Antes de aplicar os métodos de deteção de bordas foi necessário efetuar o recorte da imagem na zona de interesse, neste caso, na zona em que permite posteriormente efetuar a deteção das bordas superiores da ribeira. Assim sendo, considerando que a imagem tem uma dimensão de 1280×720 pixéis foi efetuado o recorte da imagem da seguinte forma:

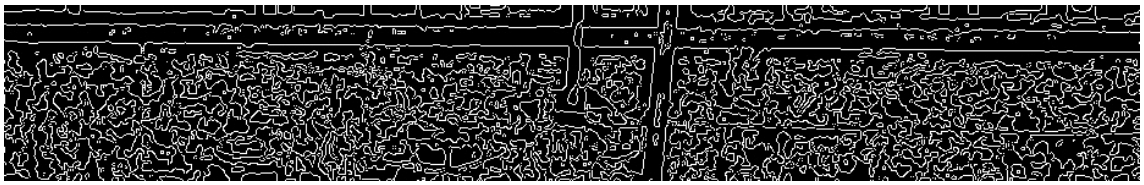
- Eixo x: desde 250 até 1180;
- Eixo y: desde 0 até 144;

A zona de interesse fica com as seguintes dimensões: 930×144 pixéis. Em relação ao eixo x, descartaram-se os primeiros 250 pixéis e os últimos 100 pixéis, visto que, ao observar a imagem binária na dimensão original, estas zonas apresentavam mais ruído na zona das bordas.

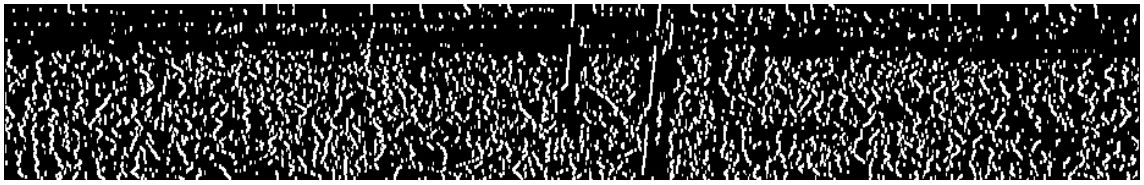
Na figura 3.20 é apresentada a imagem original com um retângulo a indicar a zona que será utilizada para a deteção de bordas. É importante referir que a imagem apresentada na figura 3.20 é a cores apenas para que seja mais percetível a zona de interesse que será usada.

Figura 3.20 - Imagem durante o dia que indica a zona utilizada para efetuar a deteção de bordas.

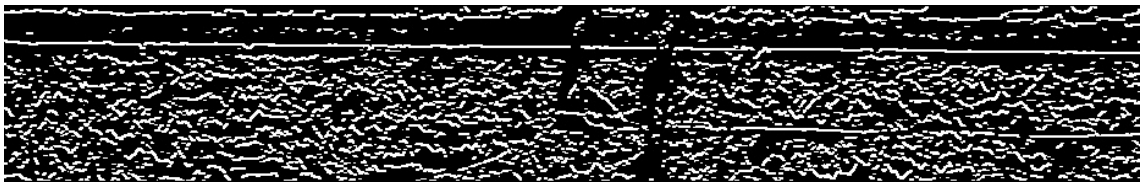
Foram aplicados os três métodos de deteção de bordas às imagens binárias, obtendo-se os resultados apresentados na figura 3.21.



a)



b)



c)

Figura 3.21 - Imagem durante o dia (zona de interesse): a) método Canny; b) método Sobel; c) método Prewitt.

Observando os resultados obtidos nos testes iniciais verificou-se que o método Sobel, mesmo para imagens nítidas durante o dia, não permitiu evidenciar as bordas da ribeira, algo que não acontece para os outros dois métodos (Canny e Prewitt). Assim sendo, para os testes posteriores apenas foram utilizados o método Canny e Prewitt.

Utilizando os dois métodos restantes (Canny e Prewitt) aplicou-se a transformada de Hough que permitiu obter parâmetros que possibilitaram extrair uma linha nas zonas aonde existem bordas. Em *python* a função que permite efetuar a transformada de Hough é a seguinte:

lines = cv.HoughLines(edges_canny, 1, np.pi/360, points).

Esta função tem como parâmetros a imagem de saída de um dos métodos de deteção de bordas (*edges_canny*), a resolução do parâmetro ρ em pixéis (*1* pixel), a resolução do parâmetro θ em radianos (*np.pi/360*) e o número mínimo de interseções das sinusoides no plano Hough (*points*) para “detetar” uma linha. Tem-se como saída um *array* com os valores de θ (em radianos) e ρ (em pixéis) correspondente a cada linha detetada [30].

Em relação ao parâmetro da resolução de θ, o tipicamente utilizado é $\pi/180$ radianos. No entanto, efetuando testes preliminares verificou-se que esta resolução era pequena para o que se pretendia detetar (bordas superiores da ribeira), tendo-se aumentado para $\pi/360$ radianos.

Antes de efetuar a aplicação da transformada de Hough foi necessário, para os dois métodos de deteção de bordas utilizados (Canny e Prewitt), obter o número de

interseções mais adequado. Para isso foram utilizados quatro grupos de imagens: dia normal, difíceis de detetar durante a noite, durante períodos de chuva e durante períodos de muito sol na zona da borda superior da ribeira. Estes grupos permitiram cobrir a maioria das situações existentes ao longo do ano, sendo que no total foram utilizadas 40 imagens.

Para obter o número de interseções foi efetuado um ciclo desde 0 até 930 (largura da imagem utilizada), em passos de 5, e analisaram-se, para cada imagem, os resultados obtidos. Para cada imagem registou-se o valor a partir do qual se detetava linhas numa das bordas de interesse (bordas superiores da ribeira), sem a existência de linhas acima destas, e o valor a partir do qual deixava de existir linhas nas bordas de interesse. Os primeiros 10 pixéis do eixo y ($D_p = 10$) foram descartados, visto que, em certas imagens, foi possível detetar as linhas a tracejado da estrada. Esta situação de descartar estes primeiros pixéis não provoca nenhum problema na deteção das bordas de interesse, desde que a imagem esteja bem enquadrada.

Nos gráficos das figuras 3.22 e 3.23 são apresentados os resultados obtidos, para cada um dos métodos de deteção de bordas, os valores (mínimo e máximo) que permitem detetar a borda de interesse.

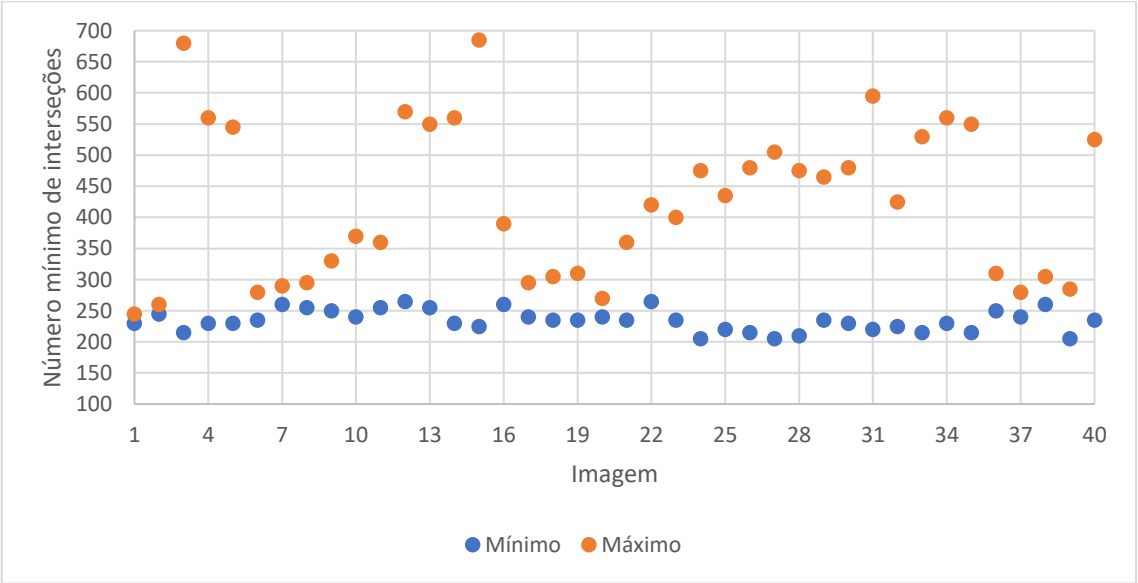


Figura 3.22 - Resultados obtidos para o método Canny.

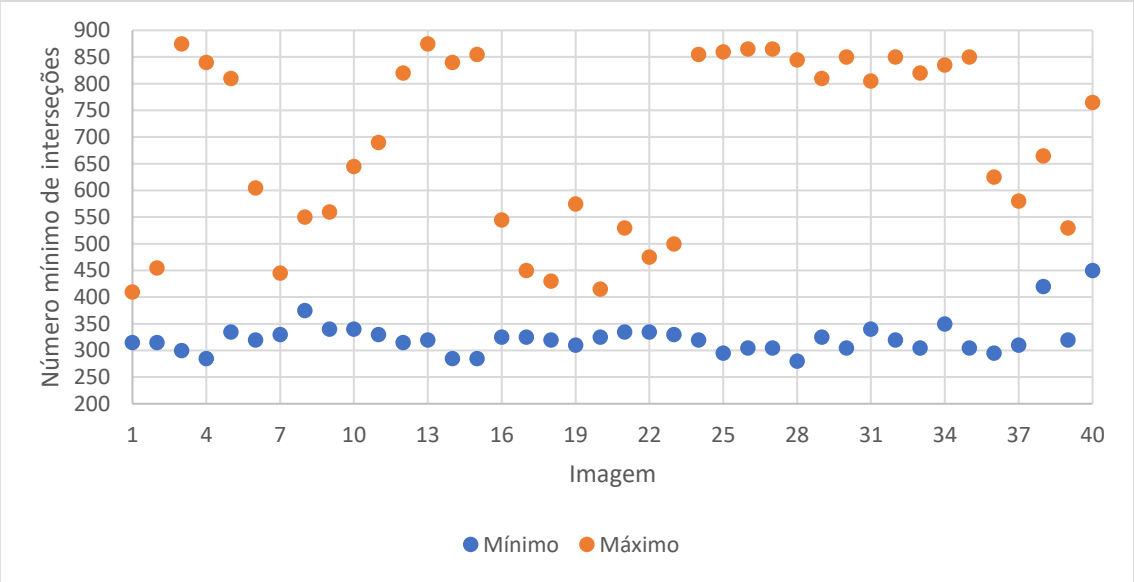


Figura 3.23 - Resultados obtidos para o método Prewitt.

Através dos resultados obtidos nos gráficos das figuras 3.22 e 3.23 foi escolhido um valor para o número mínimo de interseções que permita detetar as bordas de interesse na maioria dos casos. Em relação ao método Canny tem-se uma janela de escolha do valor desde 260 a 280, enquanto no método Prewitt tem-se de 380 a 400. Assim sendo, para o método Canny foi escolhido o valor de 270 e para o método Prewitt foi escolhido o valor de 390.

Após isto aplicou-se novamente a transformada de Hough já com os valores escolhidos anteriormente, e analisaram-se, das 40 imagens, quantas é que cada método conseguia detetar pelo menos uma das bordas. Para isso foi necessário efetuar um algoritmo que permitisse, caso haja mais do que uma linha na imagem utilizada, apenas fique a linha de interesse, neste caso, em uma das bordas superiores da ribeira.

Caso seja detetado mais do que uma linha, é calculada primeiramente a distância (na dimensão vertical) desde o início da imagem até às várias linhas detetadas, sendo que esta distância é retirada a meio da imagem de interesse. De seguida são analisados os valores obtidos e o valor mais baixo é o considerado para a deteção da borda, ficando apenas a linha (valor ρ e de θ) que corresponde a essa distância. Caso se tenha dois ou mais distâncias iguais, efetua-se a média das linhas, ou seja, efetua-se a média dos valores de ρ e de θ.

Para efeitos de análise das bordas detetadas por cada um dos métodos, fez-se a aplicação dos algoritmos às várias imagens utilizadas. Ambos os métodos permitem efetuar a deteção de pelo menos uma das bordas superiores da ribeira. Na maioria das referências nesta área referem o método Canny como tendo o melhor desempenho [46]. Assim sendo, será este o utilizado no processo de medição do nível da água.

Tendo a borda da ribeira como referência, pode-se efetuar a compensação da imagem em termos de rotação. Para isso utilizou-se a seguinte função em *python*:

rotated_image = ndimage.rotate(image_gray, angle_rotated, reshape = True).

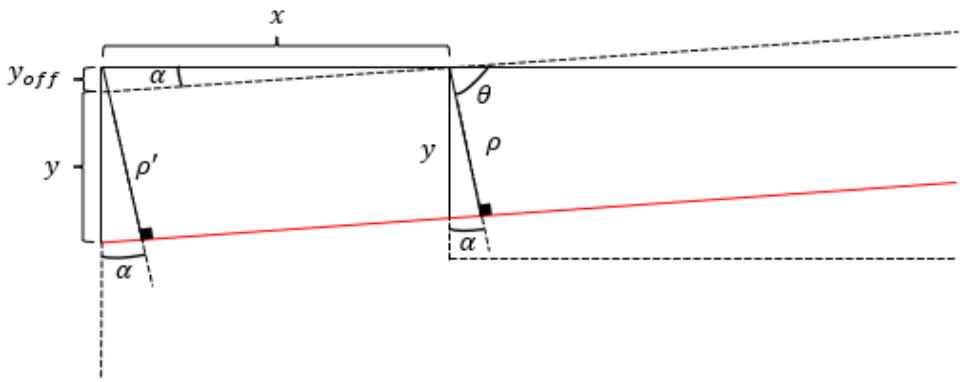
Tem-se como parâmetros a imagem que se pretende efetuar a rotação (*image_gray*) e o ângulo de rotação em graus (*angle_rotated*). O último parâmetro é o *reshape*, que quando é *True* (usado como padrão), a imagem de saída é adaptada para

que a imagem de entrada seja contida completamente na imagem de saída [47]. O ângulo utilizado na função anterior é o valor de θ (convertido para graus) menos 90º, correspondendo assim ao valor que a imagem necessita de ser rodada para que a borda fique na horizontal. Assim sendo, se a linha tiver um declive negativo ($\theta > 90°$), o resultado do ângulo de rotação será positivo, caso contrário, o ângulo de rotação será negativo.

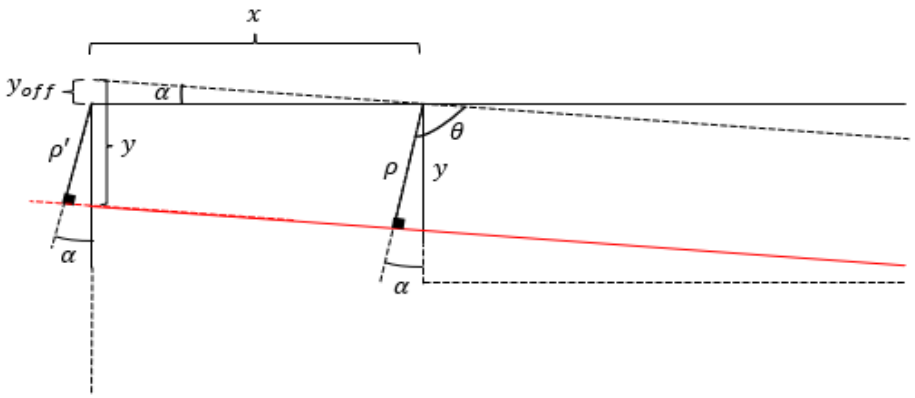
Na figura 3.24 é apresentado um exemplo de como uma imagem durante o dia fica depois de efetuar a compensação da rotação.

Figura 3.24 - Imagem compensada em termos de rotação.

Após efetuar a rotação da imagem foi necessário efetuar a obtenção do novo valor de ρ. Para isso foi necessário primeiramente obter o valor de ρ no início da imagem total (1280×720), visto que o valor que é obtido corresponde ao início da imagem utilizada para a deteção de bordas (930×144). Os sistemas usados para efetuar os cálculos são apresentados na figura 3.25, para a situação de $\theta < 90°$ e $\theta > 90°$, respetivamente, em que a linha vermelha corresponde a borda detetada.



a)



b)

Figura 3.25 - Sistema de referência utilizado para o cálculo no novo valor de ρ: a) $\theta < 90°$; b) $\theta > 90°$.

Através dos esquemas apresentados na figura 3.25, valor de ρ' pode ser determinado, para as duas situações de θ, através das seguintes expressões:

$$\rho' = \begin{cases} (y + y_{off}) \times \cos(\alpha) = \left(\frac{\rho}{\cos(\alpha)} + x \times \tan(\alpha)\right) \times \cos(\alpha) \quad , \ \theta < 90° \\ (y - y_{off}) \times \cos(\alpha) = \left(\frac{\rho}{\cos(\alpha)} - x \times \tan(\alpha)\right) \times \cos(\alpha) \quad , \ \theta > 90° \end{cases} \tag{3.11}$$

em que x corresponde ao valor desde o início da imagem total (1280×720) até o início da zona na imagem usada para deteção de bordas na dimensão horizontal, sendo que neste caso $x = 250$ pixéis. O parâmetro α corresponde, para ambas as situações, ao ângulo desde ρ até à vertical, sendo que este é obtido através do valor de θ.

Ao efetuar a rotação da imagem, para o caso em que $\theta < 90°$, a imagem rodada não interfere com o valor de ρ'. Assim sendo, para esta situação tem-se que o valor de ρ depois da imagem ser rodada é $\rho'' = \rho'$. Já para o caso em que $\theta > 90°$, a imagem rodada provoca um *offset* fazendo com que este tenha de ser adicionado ao valor obtido pela equação (3.11). Assim sendo, para esta situação tem-se que:

$$\rho'' = \rho' + offset = \rho' + w \times \cos(\alpha) \tag{3.12}$$

em que *w* corresponde à largura da imagem original (1280 pixéis).

A função que efetua todo o processo de compensação da rotação da imagem é a seguinte:

rotated_image_clahe, rotated_original_image, rotated_rho, rotated_theta, error_detection_edge = image_rotation(image, image_clahe, w, h, line_points, offset_initial_x_edge_detection, offset_final_x_edge_detection, distance_y_edge_detection, num_discarded_pixels).

Nesta função tem-se como parâmetros a imagem original (*image*), a imagem equalizada (*image_clahe*), a largura e altura da imagem (*w* e *h*, respetivamente), o número mínimo de interseções das sinusoides no plano Hough (*line_points*) para "detetar" uma linha, os *offsets*, no eixo horizontal, aonde será efetuado o recorte da imagem para deteção de bordas, neste caso, 250 pixéis (*offset_initial_x_edge_detection*) e 100 pixéis (1280-100=1180 pixéis) (*offset_final_x_edge_detection*), a distância (*distance_y_edge_detection*), no eixo vertical, de recorte da imagem para deteção de bordas, neste caso, 144 pixéis e o

número de pixéis (*num_discarded_pixels*) abaixo do qual as linhas detetadas serão descartadas, neste caso o valor é de 10 pixéis. Em relação às saídas tem-se a imagem equalizada rodada (*rotated_image_clahe*), a imagem original rodada (*rotated_original_image*), o valor de ρ e de θ da borda detetada (*rotated_rho* e *rotated_theta*, respetivamente) e o erro de deteção da borda (*error_detection_edge*).

3.4.2.2. Compensação da translação

Após a deteção da borda de interesse e a compensação da rotação procedeu-se à compensação da translação. Para isso foi utilizado um método de correspondência de padrão que permitiu a deteção de dois padrões na imagem. Um dos padrões foi usado como referência para questões de referenciação da imagem e o outro padrão foi usado para verificar se a inclinação da borda detetada pelo método de deteção de bordas estava dentro de um determinado limite. Este segundo padrão foi apenas utilizado para imagens durante o dia, visto que para imagens durante a noite a flutuação da inclinação da borda detetada é quase nula, visto que a iluminação pública permite iluminar e evidenciar as bordas superiores da ribeira.

A correspondência de padrão é um método de pesquisa e localização de uma imagem padrão (*template*), em uma imagem maior. Isto é efetuado deslizando o padrão sobre a imagem de entrada (maior que o padrão) e comparando-as em todas as posições [48].

O resultado do método de correspondência de padrão é uma imagem em tons de cinza, onde cada pixel mostra quanto dos arredores desse pixel corresponde ao padrão [48].

Existem 3 métodos de correspondência diferentes no OpenCV e para cada um deles existe uma versão normalizada. As versões normalizadas melhoram a correspondência por diferentes condições de iluminação [48].

Os métodos de correspondência do OpenCV estão apresentados de seguida [48]:

- **Método de correspondência de diferença quadrática:** as diferenças quadradas entre o padrão e a imagem são combinadas e como resultado tem-se 0 para uma combinação perfeita e um valor elevado para uma combinação má.
- **Método de correspondência de diferença quadrática normalizada:** neste método uma combinação perfeita tem como resultado o valor 0 e uma combinação má tem como resultado o valor 1.
- **Método de correspondência de correlação cruzada:** a combinação é feita por uma multiplicação entre o padrão e cada zona da imagem e como resultado um valor elevado para uma boa combinação e o valor 0 para uma má combinação.
- **Método de correspondência de correlação cruzada normalizado:** uma incompatibilidade completa com este método dá como resultado o valor 0, mas uma correspondência perfeita dá como resultado o valor 1.
- **Método de correspondência de coeficiente de correlação:** o resultado deste método será um valor elevado positivo para uma boa combinação e um grande valor negativo para uma má combinação.

- **Método de correspondência de coeficiente de correlação normalizado:** uma combinação perfeita dá como resultado o valor 1 e uma má combinação dá como resultado o valor -1.

Em *python* a função que permite aplicar o método de correspondência de modelo é a seguinte:

result = cv.matchTemplate(image_gray, template, method).

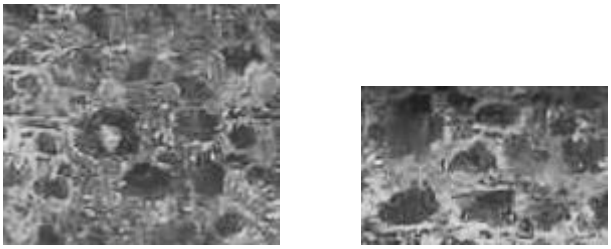
Esta função tem como parâmetros a imagem onde a pesquisa está sendo executada (*image_gray*), devendo ser uma imagem de canal único (8 *bits*), o padrão pesquisado (*template*), sendo que não deve ser maior do que a imagem de origem, e o método de correspondência utilizado. Tem como saída o mapa de resultados de comparação, ou seja, se a imagem de origem tem uma dimensão de $W \times H$ e o padrão tem uma dimensão de $w \times h$, então o resultado terá uma dimensão de $(W - w + 1) \times (H - h + 1)$ [49].

Após aplicar a função anterior, utilizou-se uma função que permitiu encontrar as coordenadas onde se localiza o padrão na imagem de origem. Essa função em *python* é a seguinte:

min_val, max_val, min_loc, max_loc = cv.minMaxLoc(result).

A função tem como parâmetro o resultado da função *cv.matchTemplate*. Em relação às saídas, tem-se o valor mínimo (*min_val*) e máximo (*max_val*) do resultado de correspondência, as coordenadas (*min_loc*) para o valor mínimo do resultado da correspondência e as coordenadas (*max_loc*) para o valor máximo do resultado de correspondência. Este último é o parâmetro de interesse, por serem as coordenadas onde se atinge a maior coincidência do padrão, que corresponde ao ponto superior esquerdo onde o padrão foi detetado [50].

Na figura 3.26 são apresentados os padrões utilizados, sendo que o da figura 3.26 a) (padrão principal) foi usado para questões de referenciação da imagem e o da figura 3.26 b) (padrão secundário) foi usado para verificação da inclinação da borda detetada. Ambos os padrões foram obtidos através de uma imagem nítida, captada durante o dia. O padrão principal e secundário tem as dimensões de 138×120 pixéis e 123×79 pixéis, respetivamente.



a) b)

Figura 3.26 - Padrões utilizados: a) principal; b) secundário.

Na figura 3.24, apresentada na secção 3.4.2.1.3, é apresentado um exemplo que como é efetuada a deteção dos padrões na imagem.

Os vários métodos de correspondência foram aplicados a um grande número de imagens. Através dos resultados obtidos verificou-se que o método que apresenta maior

percentagem de sucesso na deteção do padrão é o método de correspondência de coeficiente de correlação normalizado, sendo que será este o utilizado neste trabalho.

A função que efetua a compensação da translação, ou seja, a deteção dos padrões, é a seguinte:

x_first_temp, y_first_temp, x_second_temp, y_second_temp = translation_compensation(rotated_image_clahe, first_template, second_template, image_type).

Nesta função tem-se como parâmetros a imagem rodada equalizada (*rotated_image_clahe*), o padrão principal (*first_template*), o padrão secundário (*second_template*) e o tipo de imagem (*image_type*), ou seja, se foi capturada durante o dia ou a noite. Tem como saídas as coordenadas do ponto superior esquerdo (*x_first_temp* e *y_first_temp*) do padrão principal na imagem global e as coordenadas do ponto superior esquerdo (*x_second_temp* e *y_second_temp*) do padrão secundário.

3.4.3. Obtenção da região de interesse

Após efetuar a compensação da imagem procedeu-se à obtenção de uma região que permitiu realizar as medições para obtenção do nível da água. No fluxograma da figura 3.27 são descritos os vários processos efetuados até a obtenção da região de interesse.

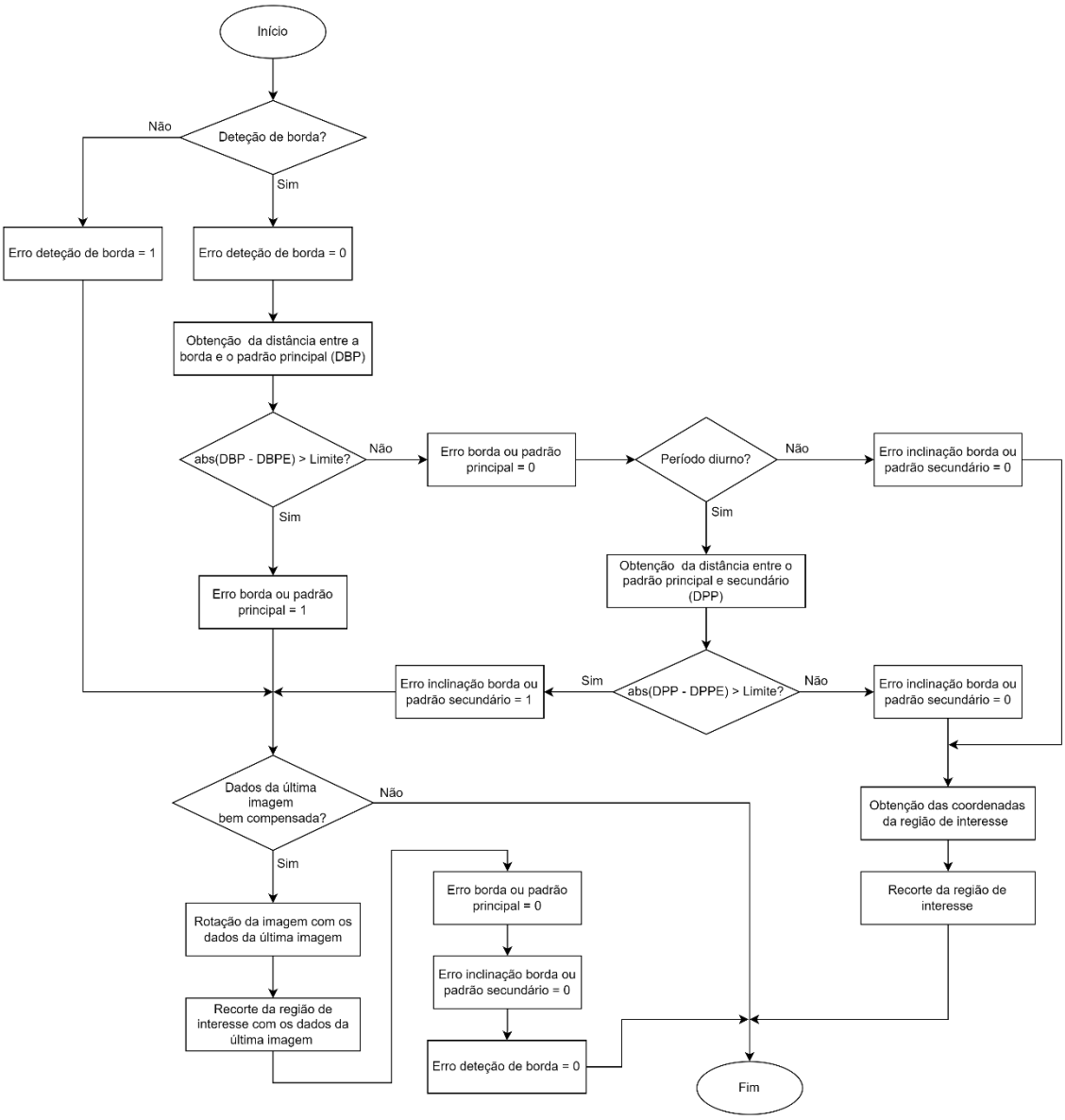


Figura 3.27 - Fluxograma com os passos efetuados até a obtenção de região de interesse.

Primeiro no fluxograma da figura 3.27 é verificado se foi detetada alguma borda superior da ribeira e caso tenha sido detetada é obtido a distância, na dimensão vertical, entre a borda detetada e o padrão principal (DBP). Essa distância é comparada com uma referência (DBPE) e caso esteja acima de um determinado limite é indicado um erro na deteção da borda ou no padrão principal. Caso esteja dentro do limite é verificado se a imagem capturada é durante o dia, visto que a verificação do padrão secundário é apenas efetuada durante o dia. Se a imagem é capturada durante o dia é obtido a distância, na dimensão vertical, entre o padrão principal e secundário (DPP) e é comparado com uma referência (DPPE) e se esta diferença esteja acima de um determinado limite é indicado erro na deteção do padrão secundário ou na inclinação da borda detetada. Se a diferença estiver dentro do limite é efetuada a obtenção das coordenadas da região de interesse, através do padrão principal, e o recorte da mesma. Caso ocorra algum dos erros indicados anteriormente existe ainda uma forma de poder obter a região de interesse através dos dados da última imagem bem compensada.

De forma a perceber melhor as distâncias obtidas e a obtenção da região de interesse na figura 3.28 é apresentada uma imagem que ilustra essas situações.

Figura 3.28 - Distâncias obtidas para realização de verificações e os *offsets* utilizados para obter a região de interesse.

Primeiramente foi avaliado o valor correspondente à distância (na dimensão vertical) entre o ponto superior esquerdo do padrão principal e a borda detetada para se obter um valor de referência (DPB_{sup} e DPB_{inf} da figura 3.28). É de notar que não é possível saber qual das bordas foi detetada, devendo existir dois valores de referência, um para a borda inferior e outro para a borda superior. Desta forma, foi efetuada uma análise para 8 dias distintos e foram obtidas estas distâncias, sendo que na análise foi considerado a utilização de dias que apresentassem diferentes inclinações das bordas da ribeira.

Na tabela 3.1 estão apresentados os resultados obtidos, tendo sido descartados os *outliers*.

Tabela 3.1 - Resultados obtidos para as distâncias entre o padrão principal e cada uma das bordas.

Dia	**Distância (pixel)**			
	Borda inferior		**Borda superior**	
	Min.	**Máx.**	**Min.**	**Máx.**
03/04/2020	185	186	200	201
17/04/2020	185	186	200	201
01/12/2020	186	187	201	202
25/12/2020	185	187	-	-
20/02/2021	186	187	-	-
27/03/2021	186	188	-	-
02/06/2021	-	-	199	202
03/06/2021	-	-	199	201

Através dos resultados da tabela 3.1 efetuou-se uma média, tendo-se obtido os valores utilizados como referência de 186,5 pixéis aquando da deteção da borda inferior e 200,5 pixéis aquando da deteção da borda superior. O valor limite considerado no fluxograma da figura 3.26 (e figura 3.7) é de ± 2 pixéis.

Após isto foi avaliado o valor correspondente à distância (na dimensão vertical) entre o ponto superior esquerdo do padrão principal e o ponto superior esquerdo do padrão secundário (DPP da figura 3.28), aplicando já os valores de referência e de variação da distância entre padrão principal e a borda detetada. É importante referir que esta distância apenas é feita durante o dia, visto que a deteção do padrão secundário é realizada para esse período. Foram utilizadas imagens desde às 09:00h até às 18:00h, a uma cadência de 2 minutos, de 6 dias distintos por forma a efetuar a análise da distância, sendo que na análise foi considerado a utilização de dias que apresentassem diferentes inclinações das bordas da ribeira.

No gráfico da figura 3.29 é apresentado os resultados obtidos, sendo que nesta situação também são apresentados os *outliers*.

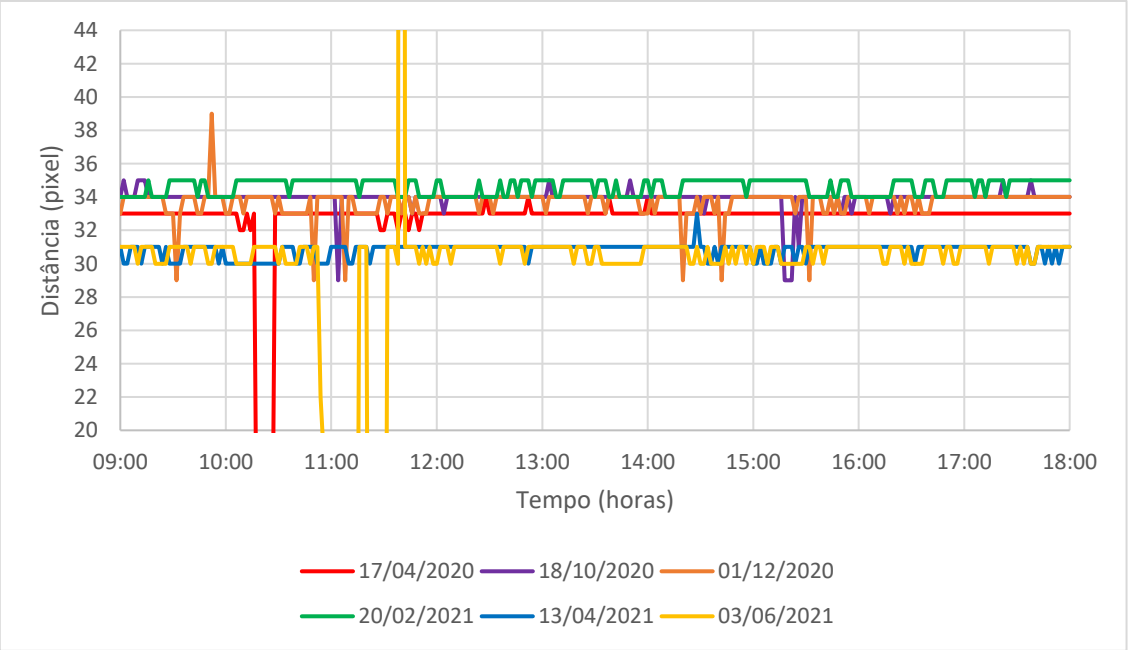


Figura 3.29 - Resultados obtidos para a distância entre o padrão principal e secundário.

Em relação aos resultados apresentados na figura 3.29, no dia 17 de abril de 2020 ocorreram erros desde às 10:18h até às 10:26h devido a deteção do padrão secundário, visto que neste período havia a incidência de sol na parede da ribeira e estava a ocorrer sombra no local do padrão secundário, sendo difícil a sua deteção. Retirando estes erros, a distância obtida variou entre 32 e 34 pixéis. No dia 18 de outubro de 2020 ocorreram erros em certos períodos devido à inclinação da borda detetada não estar correta, fazendo com que o valor da distância seja mais baixo do que os restantes. Neste dia ocorreu precipitação urbana que também pode ter influenciado a existência destes erros. Retirando estes erros, a distância obtida variou entre 33 e 35 pixéis. O mesmo problema ocorre no dia 1 de dezembro de 2020, com erros na inclinação da borda detetada, sendo que, retirando os erros, a distância obtida variou entre 33 e 34 pixéis. No dia 20 de fevereiro de 2021 não ocorreram erros nem no padrão nem na inclinação, sendo que a distância obtida variou entre 34 e 35 pixéis. No dia 13 de abril de 2021 também não ocorreu erros, logo, a distância obtida variou entre 30 e 33 pixéis. No dia 3

de junho de 2021 ocorreram erros nos períodos de 10:54h e 11:14h, de 11:22h e 11:30 e às 11:40 devido a deteção do padrão secundário, ocorrendo o mesmo problema do que no dia 17 de abril de 2020. Retirando estes erros, a distância obtida variou entre 30 e 31 pixéis.

Efetuando uma análise mais criteriosa nos resultados apresentados na figura 3.29 verificou-se que, para os 6 dias, os valores desta distância encontram-se no intervalo de 30 a 35 pixéis. Efetuando uma média, o valor utilizado como referência para esta distância foi de 32,5 pixéis. O valor limite apresentado no fluxograma da figura 3.27 (e figura 3.6) é de ± 3 pixéis.

Caso a distância entre o padrão principal e a borda detetada e entre o padrão principal e secundário estejam dentro dos limites apresentados anteriormente procede-se à obtenção da região de interesse. Para isso utilizaram-se as coordenadas obtidas através da deteção do padrão principal, neste caso, o ponto superior esquerdo do padrão, como referência e a partir do qual se irá efetuar um *offset* por forma a obter a região de interesse. O *offset* efetuado foi de 420 pixéis na dimensão horizontal ($offset_x$ da figura 3.27) e 200 pixéis na dimensão vertical ($offset_y$ da figura 3.28). A região de interesse tem as dimensões de 100×250 pixéis. Esta área permite garantir que seja possível observar a evolução do caudal da ribeira. A região de interesse deve incluir a zona da água e uma face da parede onde será medido o nível da água. Procurou-se um local onde se tenha, a maior parte do tempo, uma zona desobstruída de vegetação.

Na figura 3.28 foi apresentada a zona onde são detetados o padrão principal (laranja) e a região de interesse (vermelho). Na figura 3.30 é apresentado o recorte da região de interesse.

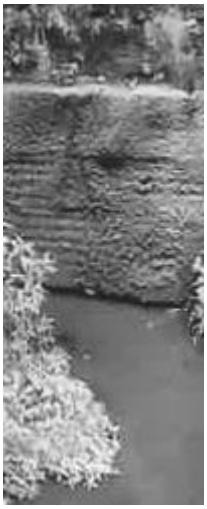


Figura 3.30 - Região de interesse para a deteção da altura do caudal da ribeira.

Caso estas distâncias não estejam dentro dos limites apresentados, a região de interesse é obtida através dos parâmetros da última imagem em que foi efetuado uma compensação bem-sucedida. Para isso é necessário guardar o parâmetro θ (em radianos), que permite efetuar a rotação da imagem, e as coordenadas da posição da região de interesse na imagem.

A função implementada que permite realizar as verificações necessárias e obtenção da região de interesse nas várias situações é a seguinte:

roi, original_roi, rotated_image_clahe_bright, roi_bright, coord_roi_top_left, coord_roi_bottom_right, error_detection_edge, error_edge_template, error_templates, good_verification = verification_edge_template(image, rot_comp_original_image, image_clahe, rotated_image_clahe, image_clahe_bright, rotated_rho, coord_x_ftemp, coord_y_ftemp, coord_x_stemp, coord_y_stemp, error_detection_edge, difference_edge_inf_temp, difference_edge_sup_temp, difference_templates, delta_diff_edge_temp, delta_diff_templates, image_type, sun_period, offset_x_temp, offset_y_temp, height_roi, width_roi, rotated_theta, previous_theta, coord_roi_top_left_prev, coord_roi_bottom_right_prev).

Nesta função tem-se como parâmetros a imagem original (*image*), a imagem original rodada (*rot_comp_original_image*), a imagem equalizada (*image_clahe*), a imagem equalizada rodada (*rotated_image_clahe*), a imagem equalizada com aumento de brilho (*image_clahe_bright*), o novo valor de ρ depois da imagem ter sido rodada (*rotated_rho*), as coordenadas do ponto superior esquerdo do padrão principal e secundário (*coord_x_ftemp*, *coord_y_ftemp* e *coord_x_stemp*, *coord_y_stemp*, respetivamente), o erro de deteção de bordas (*error_detection_edge*), os valores de referência da distância entre o padrão principal e a borda inferior e superior (*difference_edge_inf_temp* e *difference_edge_sup_temp*, respetivamente), o valor de referência da distância entre o padrão principal e secundário (*difference_templates*), o valor limite de variação da distância entre o padrão principal e uma das bordas detetadas (*delta_diff_edge_temp*), o valor limite de variação da distância entre o padrão principal e secundário (*delta_diff_templates*), o tipo de imagem captada (*image_type*), ou seja, se a imagem é captada durante o dia ou durante a noite, a variável que indica se a imagem está dentro do "período de sombra" (*sun_period*), os *offsets* a realizar para obter as coordenadas do ponto superior esquerdo da região de interesse (*offset_x_temp* e *offset_y_temp*, respetivamente), a altura e largura da região de interesse (*height_roi* e *width_roi*, respetivamente), o ângulo de rotação da imagem atual (*rotated_theta*), o ângulo de rotação da imagem anterior que tenha sido bem compensada (*previous_theta*), as coordenadas do ponto superior esquerdo e inferior direito da região de interesse da imagem anterior que tenha sido bem compensada (*coord_roi_top_left_prev* e *coord_roi_bottom_right_prev*, respetivamente). Tem como saídas a região de interesse da imagem equalizada (*roi*), a região de interesse da imagem original (*original_roi*), a imagem equalizada com aumento de brilho rodada (*rotated_image_clahe_bright*), a região de interesse da imagem equalizada com aumento de brilho (*roi_bright*), as coordenadas do ponto superior esquerdo e inferior direito da região de interesse da imagem atual (*coord_roi_top_left* e *coord_roi_bottom_right*, respetivamente), o erro de deteção de bordas (*error_detection_edge*), o erro de deteção de borda e de padrão principal (*error_edge_template*), erro de deteção do padrão secundário ou inclinação da borda (*error_templates*) e por fim a variável que indica se foi efetuado uma boa compensação da imagem ou não (*good_verification*).

3.5. Obtenção da altura do caudal da ribeira

Nesta parte do trabalho procedeu-se à obtenção da altura do caudal da ribeira. Este procedimento vem após a definição da região de interesse em torno de uma zona que permite extrair a linha de água e posteriormente a altura do caudal da ribeira.

3.5.1. Perfil de escala de cinza e gradiente

Como referido anteriormente, a região de interesse foi definida através de um ponto de referência, que são as coordenadas do ponto superior esquerdo do padrão principal apresentado na figura 3.26 a). Este procedimento permite obter um sistema de referenciação da imagem, tal como é apresentado na figura 3.31. Uma linha vertical no plano do objeto é vista em perspetiva no plano da imagem. A linha dentro da região de interesse é usada para detetar a posição da linha de água, definida pelo ponto (P_x, P_y) (ou (P'_x, P'_y) considerando o eixo $x'y'$).

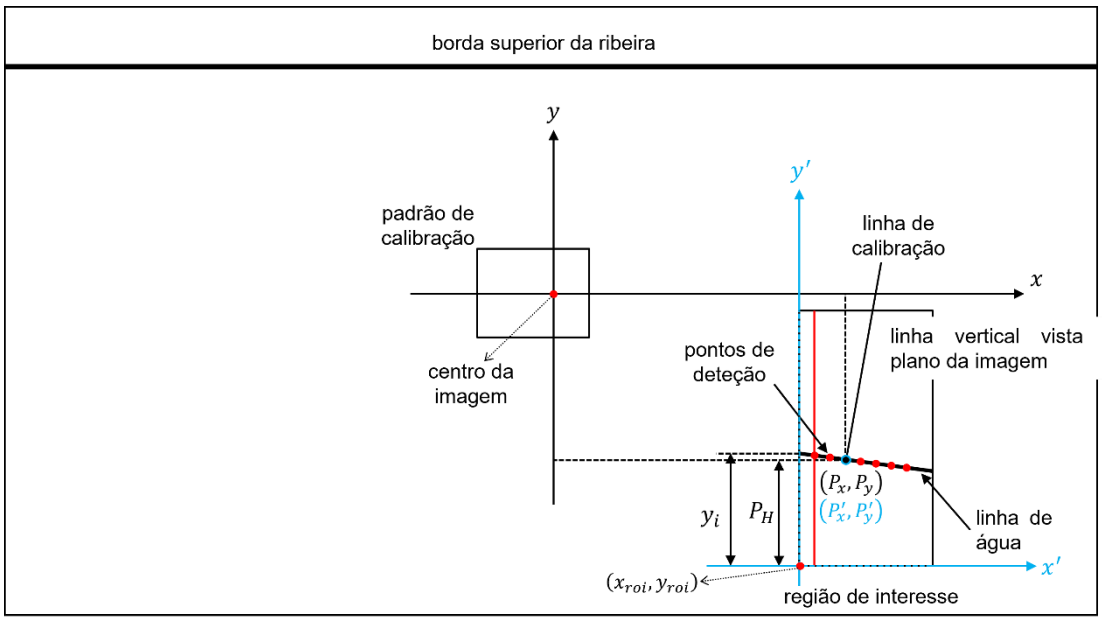
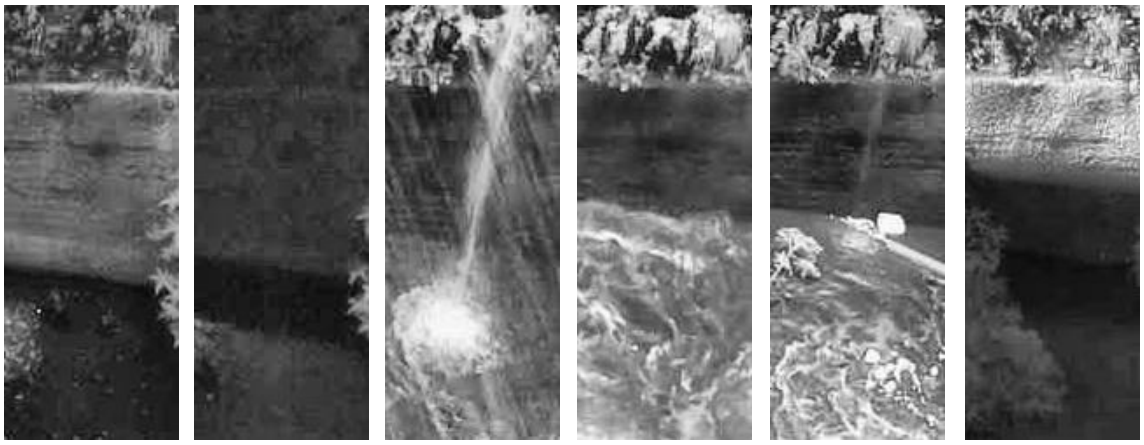


Figura 3.31 - Esquema do plano da imagem.

Antes de proceder à explicação de todo o processo, foram analisadas as diferentes imagens obtidas para a região de interesse que poderão existir ao longo de todo o ano, estando representadas na figura 3.32. Na maioria dos casos, o nível de água não se altera, resultando em imagens como a apresentada na figura 3.32 a). A figura 3.32 b) mostra uma imagem típica obtida durante a noite. Períodos de precipitação podem afetar a qualidade da imagem, tal como é apresentado na figura 3.32 c). A figura 3.32 d) é uma situação que ocorre durante períodos de precipitação, com o nível de água a se alterar e, como consequência, ocorre oscilações na água. Uma outra situação é a existência de detritos na superfície da água, conforme mostrado na figura 3.32 e). A figura 3.32 f) apresenta um exemplo em que ocorre um efeito de sombra criada pelo sol dentro da região de interesse.



a) b) c) d) e) f)

Figura 3.32 – Diferentes conteúdos da região de interesse: a) durante o dia; b) durante a noite; c) situações de chuva; d) situações de flutuação da água; e) situações com detritos na água; f) situações de sombra do sol.

Analisando os vários tipos de resultados dentro da região de interesse da figura 3.32 verifica-se que os métodos de deteção de bordas apresentados na secção 3.4.2.1.3 não são adequados para a obtenção da altura do caudal devido às várias especificidades existentes na região de interesse.

Por forma a minimizar vários dos efeitos observados nas imagens da figura 3.32 foram captadas T imagens com uma diferença de três segundos entre elas e foi obtida a média das imagens. Esta imagem média é convertida para escala de cinza e é aplicada a equalização de histograma para destacar a linha de água e efetuar todo o processo de compensação (rotação e translação).

A função, em *python*, que foi implementada de forma a poder efetuar a média temporal das imagens é a seguinte:

time_average(initial_directory, images_directory, temp_directory, average_directory, num_images).

Nesta função tem-se como parâmetros o diretório principal (*initial_directory*) que contém todos os ficheiros deste algoritmo, o diretório (*images_directory*) que contém as imagens a utilizar para a média, o diretório temporário (*temp_directory*) para guardar o número de imagens (*num_images*) que foram utilizadas para efetuar a média e o diretório (*average_directory*) para guardar a imagem resultante da média temporal.

A ideia inicial para a obtenção do nível do caudal foi a utilização de uma determinada zona de pixéis (normalmente a meio da imagem), ou seja, utilizar apenas um ponto de deteção. Foram obtidos gráficos de uma média móvel da escala de cinza e do gradiente para avaliar a posição da linha de água. Para isso foi definido um sistema de referência para a região de interesse por forma a suportar o procedimento de deteção da linha de água. Assim sendo, (x', y') representa um ponto na região de interesse com a origem no pixel inferior esquerdo (figura 3.30). No plano da imagem, esta origem é representada pelo ponto (1,1), enquanto na aplicação (*python*), como a região de interesse é composta por uma matriz de valores de escala de cinza, a origem é representada pelo ponto (0,0). Posto isto, por forma a que a referência na aplicação corresponda ao do plano da imagem, no algoritmo é necessário somar 1 ao resultado obtido para a posição da linha de água.

A primeira função criada permitiu efetuar uma média móvel com os valores de escala de cinza de uma imagem (esta também na escala de cinza). A expressão matemática que explica como é efetuado a média móvel dos valores de escala de cinza é dada por:

$$grayscale\left(x' + \frac{N_x - 1}{2}, y' + \frac{N_y - 1}{2}\right) = \frac{1}{N_x \times N_y} \sum_{n=0}^{N_y - 1} \sum_{m=0}^{N_x - 1} pixel(x' + m, y' + n) \quad (3.13)$$

$$com\ x' = x + p \quad e \quad 1 \leq y' \leq h - N_y$$

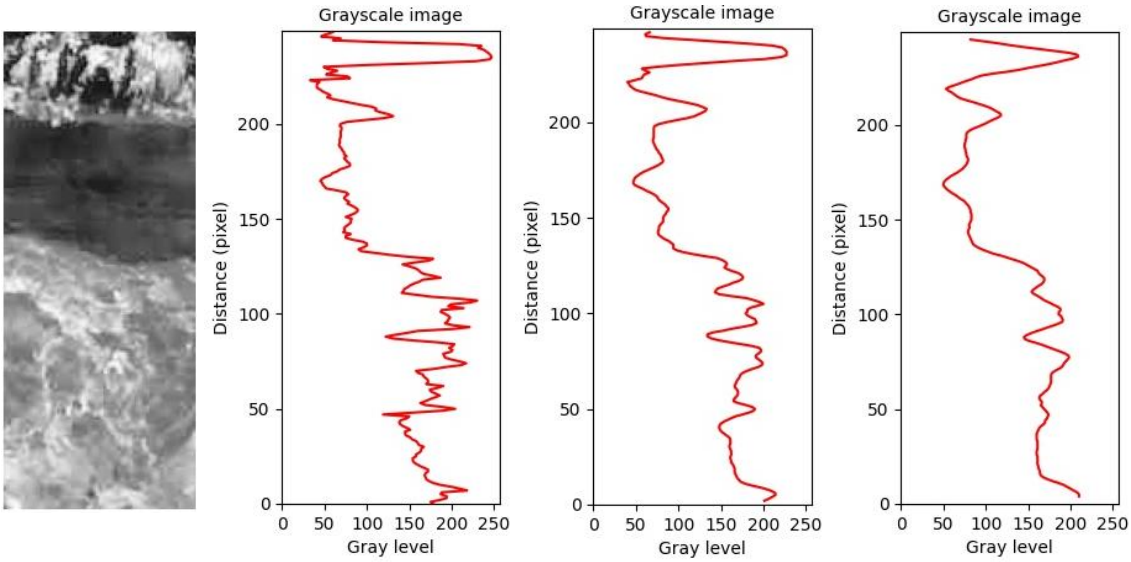
em que N_x é o número de pixéis usado no eixo x, N_y é o número de pixéis usados no eixo y, p é o passo efetuado (em pixéis) no eixo x, ou seja, a zona da imagem onde é retirada os valores de escala de cinza e h é o número de pixéis na dimensão vertical da imagem. No algoritmo efetuado tem-se que $N_x = N_y$.

A função que permitiu efetuar a média móvel dos valores de escala de cinza é a seguinte:

array_grayscale, axis_y = average_grayscale(image, height, img_zone, pixels).

Esta função tem como parâmetros a imagem onde serão retirados os valores de escala de cinza (*image*), a altura da imagem utilizada (*height*), a coluna da imagem (valor em pixéis) a partir da qual serão retirados os valores em escala de cinza (*img_zone*) e o valor em pixéis usado para retirar um bloco (*pixels*×*pixels*) de valores em escala de cinza para efetuar a média. Esta função tem como saídas o *array* com os valores da média móvel (*array_grayscale*) e o *array* com os pixéis (posição na dimensão vertical) correspondentes a cada valor da média móvel (*axis_y*).

O valor do parâmetro *pixels* utilizado foi 5, visto que permite ter um compromisso de redução do ruído provocado pelas irregularidades existentes na parede da ribeira ou dos detritos que possam existir na água sem perder a informação na zona da linha de água. Na figura 3.33 são apresentados para uma determinada imagem os resultados dos gráficos para os valores do parâmetro *pixels* de 1 (sem média), 5 e 9, tendo estes sido retirados na coluna 49 (50 na imagem), ou seja, na zona central da região de interesse. Observando a imagem verifica-se que para 9 já se perde alguma informação referente à linha de água e sem média (1) existe muito ruído no gráfico, podendo efetuar uma deteção errada da posição da mesma. Assim sendo, o valor de 5 é o que se mais se ajusta a esta situação.



a) b) c) d)

Figura 3.33 - Valor do parâmetro *pixels*: a) região de interesse; b) sem média; c) 5; d) 9.

A segunda função implementada permitiu calcular o gradiente a partir da média móvel dos valores de escala de cinza. Esta função percorre o *array* com os valores de escala de cinza e a cada dois valores consecutivos efetua a subtração destes, obtendo assim o gradiente. A expressão matemática que permite a obtenção do gradiente através dos valores da média móvel de escala de cinza é:

$$gradient(x', y') = grayscale(x', y' + 1) - grayscale(x', y') \qquad (3.14)$$

A função que permitiu efetuar o gradiente é a seguinte: *array_gradient = gradient(array_grayscale)*, que tem como parâmetro o *array* com os valores da média móvel da escala de cinza (*array_grayscale*) e como saída o *array* com os valores do gradiente (*array_gradient*).

Após a implementação das funções efetuaram-se testes preliminares e verificou-se que o *array* com os valores do gradiente, neste caso o valor máximo absoluto do gradiente (dentro de uma zona de pesquisa), permitiu extrair informação necessária para indicar a posição da linha de água. Mas, como se estava a usar apenas uma zona da imagem (um ponto de deteção), neste caso a zona central, a existência de detritos na superfície da água, os vestígios de chuva capturado pela câmara, a ondulação da água ou outros efeitos, podem criar um gradiente máximo absoluto numa posição que não corresponde à linha de água.

Na figura 3.34 é apresentado um exemplo que evidencia o que foi descrito anteriormente, ou seja, uma má deteção da linha de água devido à existência de detritos na superfície de água.

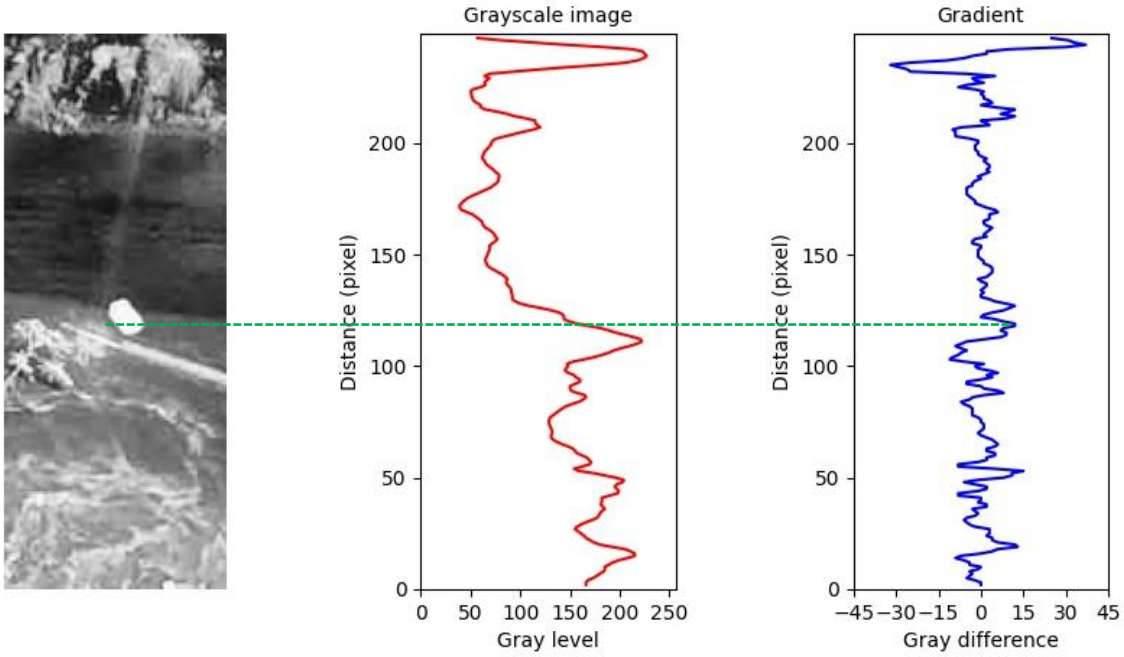


Figura 3.34 - Deteção da linha de água para uma imagem com detritos na água usando um ponto de deteção.

Assim sendo, optou-se por, em vez de apenas retirar informação na zona central da imagem, utilizar toda a imagem (neste caso, desde a posição 0 até ao 90 na dimensão horizontal) e efetuar uma média espacial, ou seja, a uma determinada cadência, retirar colunas de valores de pixéis para obtenção da média móvel da escala de cinza e gradiente e após efetuar uma média discreta dos mesmos. A cadência utilizada (parâmetro p da equação (3.13)) foi de 9 pixéis, dando um total de 10 pontos de deteção. Esta média espacial é feita dentro da função que é usada para efetuar a deteção da posição da linha de água e consequentemente obter a altura do caudal da ribeira. Foram utilizados 10 pontos de deteção visto que, caso o número de pontos seja baixo, os erros na deteção da linha de água podem ser elevados. Caso seja um número de pontos elevado irá fazer com que o processamento do algoritmo seja mais demorado.

No fluxograma da figura 3.35 são apresentados os passos efetuados de forma a obter uma média espacial da média móvel da escala de cinza e do gradiente.

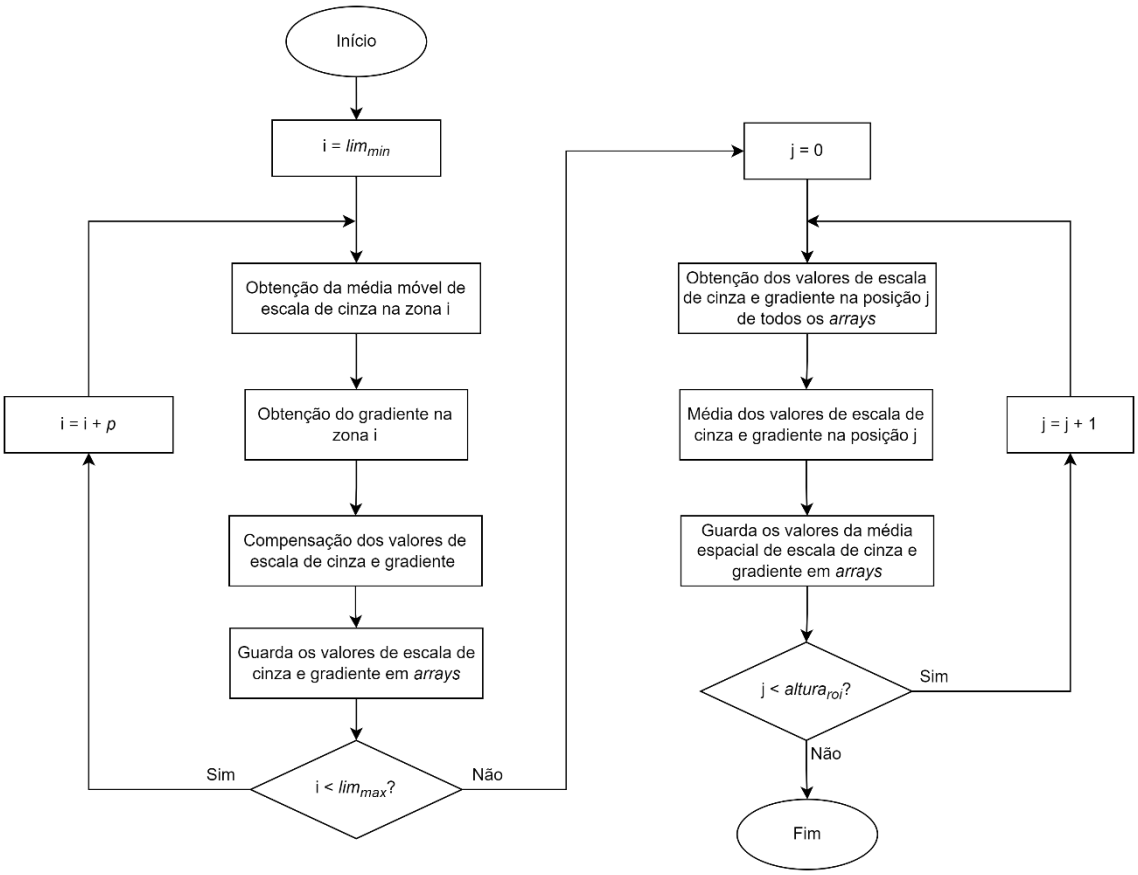


Figura 3.35 - Fluxograma que apresenta os passos até obter a média espacial.

No fluxograma da figura 3.35 primeiramente é definido um limite mínimo (0) e máximo (90), na dimensão horizontal, na região de interesse onde foi possível retirar a informação da escala de cinza e gradiente. Essa informação foi retirada a uma determinada cadência (definida por $p = 9$ pixéis) e compensada, dependendo da zona aonde está a ser retirada a informação, visto que a linha de água apresenta uma determinada inclinação, logo, é necessário compensar essa informação para depois poder extrair a linha de água de forma correta. Depois da aquisição da informação é feito um ciclo, na dimensão vertical, de forma que, a cada posição de pixel seja extraído a informação de todos os *arrays* e efetuado uma média dos valores de escala de cinza e gradiente e o resultado é guardado em *arrays*. Este ciclo termina depois de todos os pixéis sejam percorridos da altura (250 pixéis) da região de interesse.

No desenvolvimento da função foi necessário retirar a informação referente ao declive que a linha de água apresenta maioritariamente nas imagens ao longo do ano. Para isso utilizou-se uma imagem durante o dia, com a zona da linha de água nítida por forma a retirar esse declive. Para a obtenção do declive utilizou-se um *software* de manipulação de imagem chamado de *paint.net*, visto que este permite desenhar uma linha na imagem e permite indicar o declive da mesma [51]. A linha foi desenhada desde o pixel 10 até o pixel 80 (na dimensão horizontal).

Na figura 3.36 é apresentada a imagem utilizada e a linha desenhada na imagem.

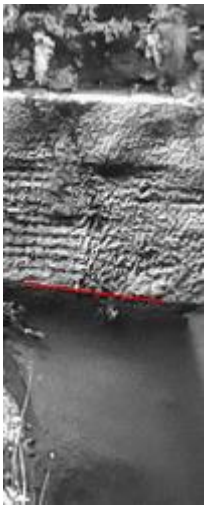


Figura 3.36 - Imagem utilizada para retirar o declive da reta que indica a linha de água.

O declive obtido para a linha de água foi de $m_l = -0{,}1268$. Este declive permitiu alinhar as várias informações (média móvel da escala de cinza e gradiente) retiradas nas várias zonas da imagem. O valor que permitiu efetuar este alinhamento/compensação (Δy), ou seja, o valor que é usado para baixar ou subir as posições (em pixéis) correspondentes aos valores da média móvel da escala de cinza e gradiente é definido por:

$$\Delta y = m_l \Delta x \tag{3.15}$$

em que m_l representa o declive da reta obtido anteriormente e $\Delta x = (ref + 1) - \left(x' + \frac{N_x - 1}{2}\right)$ é a diferença entre a posição, no eixo x, do ponto de deteção aonde é retirado os valores de escala de cinza e uma referência, em pixéis, que é definida. O valor de referência (ref) utilizado foi de 49 (pixel 50 na imagem), que corresponde a metade da largura das imagens que foram utilizadas, mas o utilizador pode alterar esta referência se o pretender.

A função que permitiu efetuar a média espacial é a seguinte:

axis_y, average_grayscale, average_gradient = spatial_average(image, height, lim_min, lim_max, pixels, delta_x, x_reference, slope_value).

Esta função tem como parâmetros a imagem a ser utilizada (*image*), a altura da imagem (*height*), o limite mínimo (*lim_min*) e máximo (*lim_max*), na dimensão horizontal, onde irá começar e terminar a obtenção dos valores dos pixéis para a média espacial (0 e 90, respetivamente), o valor em pixéis (5) usado para retirar um bloco (*pixels*×*pixels*) de valores em escala de cinza para efetuar a média móvel, a cadência (*delta_x*) a que os valores dos pixéis são retirados (9), o pixel (49) utilizado como referência na dimensão horizontal (*x_reference*) e o valor do declive da reta (*slope_value*) referido anteriormente. Tem como saídas o *array* com os pixéis (posição na dimensão vertical) correspondentes a cada valor da média móvel de escala de cinza e gradiente (*axis_y*), o *array* com os valores da média móvel de escala de cinza após a média espacial (*average_grayscale*) e o *array* com os valores do gradiente após a média espacial (*average_gradient*).

Na figura 3.37 é apresentado o exemplo apresentado na figura 3.34, mas agora efetuando a média espacial.

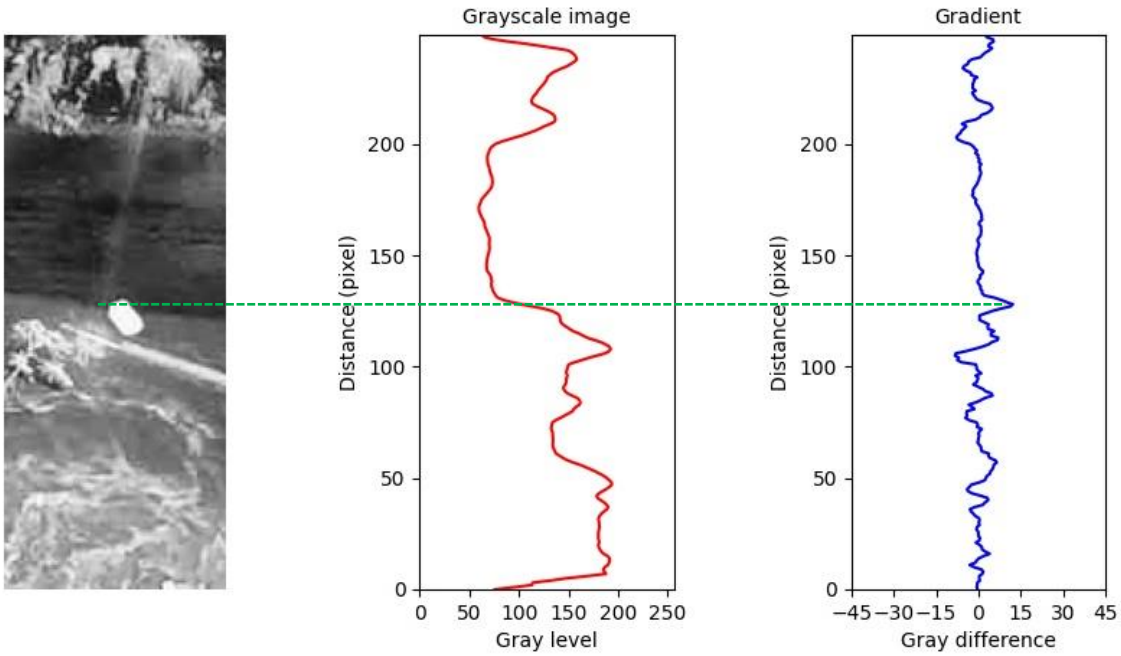


Figura 3.37 - Deteção da linha de água para uma imagem com detritos na água usando 10 pontos de deteção e posterior média espacial.

3.5.2. Deteção da linha de água

O fluxograma da figura 3.38 apresenta os vários procedimentos efetuados na função que permite a obtenção da posição da linha de água e posterior altura do caudal da ribeira.

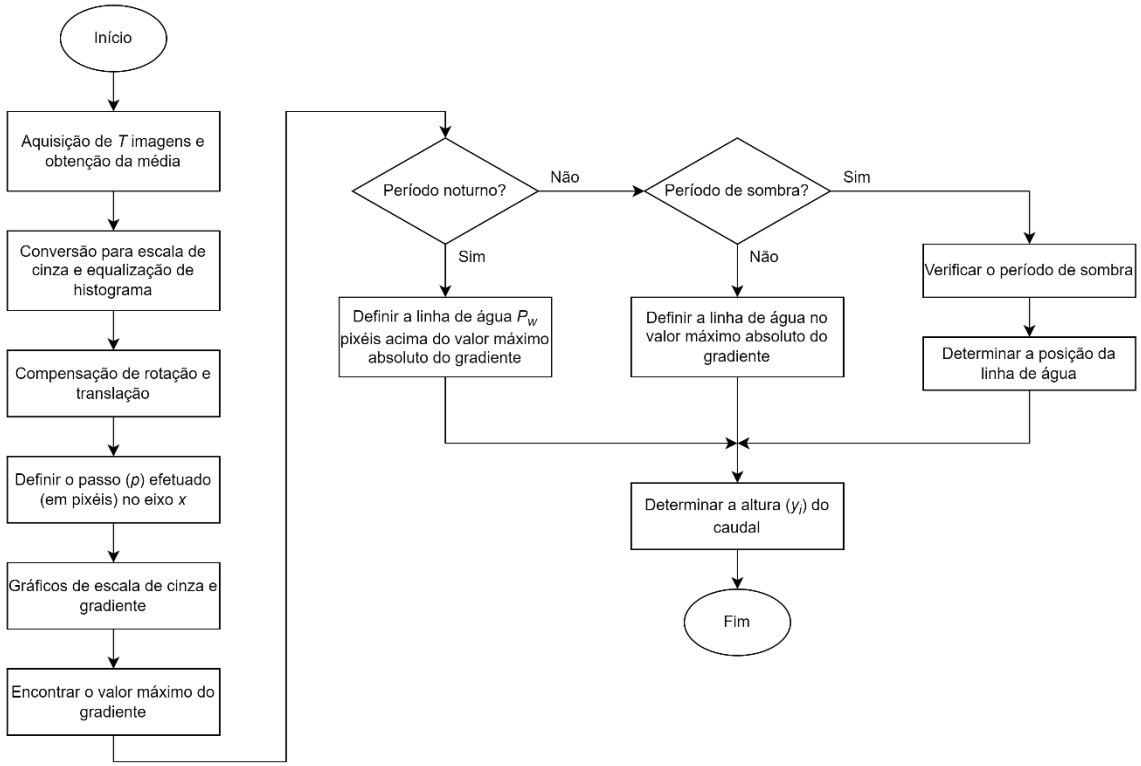


Figura 3.38 - Fluxograma para a obtenção da linha de água.

Primeiramente é efetuada a aquisição de T imagens e a obtenção da média das mesmas. A imagem média é convertida para escala de cinza e é efetuado a equalização de histograma. De seguida faz-se a compensação da rotação e da translação da imagem, como descrito no ponto 3.4.2. Posteriormente foi definido o passo (p) efetuado no eixo x por forma a obter os vários pontos de deteção que permitem efetuar uma média discreta dos gráficos de média móvel de escala de cinza e gradiente. Em seguida, o valor máximo do módulo do gradiente foi determinado. Por fim, dependendo da situação apresentada na região de interesse é efetuado um procedimento por forma a obter a posição da linha de água e posteriormente a altura do caudal da ribeira.

Para imagens captadas durante o dia, podem surgir duas situações. A primeira situação corresponde ao caso em que o valor máximo do módulo do gradiente se situa na posição da linha de água. Isto acontece na maior parte do tempo e o algoritmo procura esse máximo numa zona entre a posição inferior do nível da água (pixel 90) e a parte superior da face de betão (pixel 180). O parâmetro y_i (em pixéis) apresentado na figura 3.31 foi determinado a partir do máximo do módulo do gradiente através da seguinte expressão:

$$y_i = (L_w + 1) + offset \tag{3.16}$$

em que L_w corresponde à posição da linha de água e $offset = abs((ref + 1) \times m_l)$ (ref corresponde ao valor de 49) corresponde ao valor que é necessário somar ao valor da

posição da linha de água para obter o valor da altura do caudal, neste caso na posição inicial da imagem.

Na figura 3.39 é apresentado um exemplo de uma imagem durante o dia com a obtenção da linha de água (linha a branco) através do máximo do módulo do gradiente.

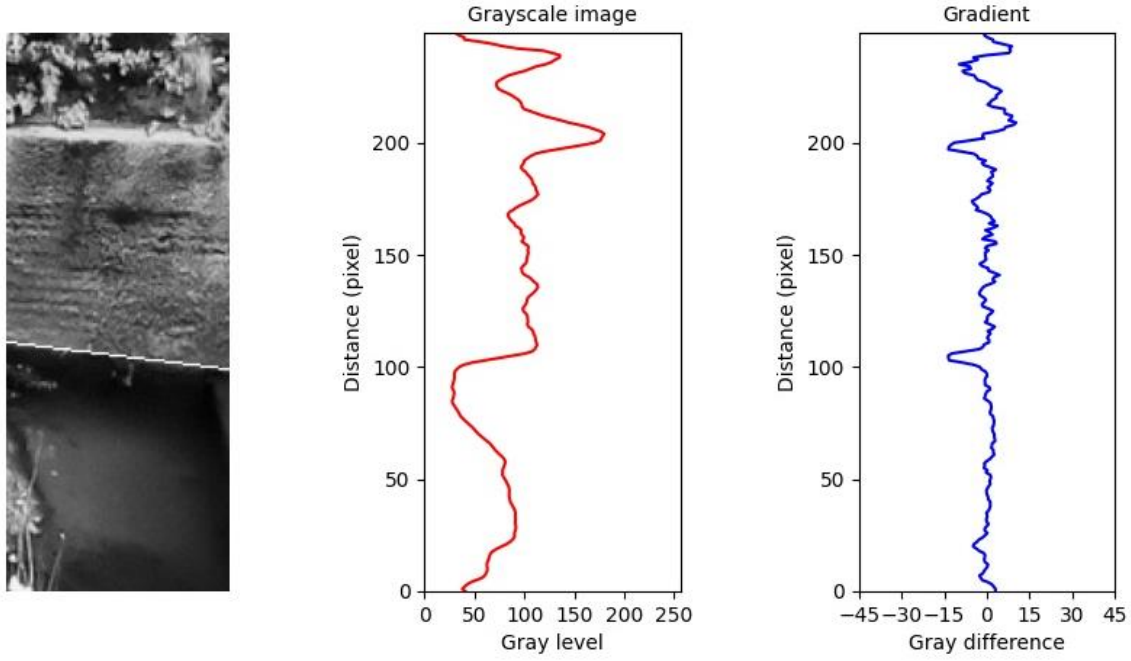


Figura 3.39 - Deteção da linha de água para uma imagem durante o dia.

A outra situação, que pode ocorrer durante um curto período do dia, é que os edifícios, ou até a própria parede da ribeira, podem produzir uma sombra dentro da zona de pesquisa (do pixel 90 a 180). Neste caso, o valor máximo do módulo do gradiente na transição pode coincidir com a fronteira entre a zona iluminada pelo sol e a zona de sombra. No fluxograma da figura 3.38, o intervalo de tempo em que esta situação pode ocorrer é definido pelo “período de sombra”. Para determinar este intervalo de tempo foi necessário efetuar testes e analisar em várias épocas do ano como este intervalo de tempo varia. Na tabela 3.2 são apresentados os resultados obtidos referentes ao ano de 2020.

Tabela 3.2 - Resultados para o "período de sombra" efetuados no ano 2020.

Mês	**1º Período**		**2º Período**		**Mês**	**1º Período**		**2º Período**	
	Início	**Fim**	**Início**	**Fim**		**Início**	**Fim**	**Início**	**Fim**
janeiro	8:30 h	9:30 h	9:20 h	10:00 h	**julho**	9:00 h	9:10 h	12:30 h	13:10 h
fevereiro	8:40 h	9:20 h	10:00 h	10:50 h	**agosto**	9:00 h	9:10 h	11:30 h	12:50 h
março	9:00 h	9:10 h	10:40 h	11:40 h	**setembro**	8:40 h	9:00 h	10:40 h	11:50 h
abril	8:40 h	9:00 h	11:20 h	12:20 h	**outubro**	8:10 h	8:40 h	9:50 h	10:50 h
maio	8:50 h	9:00 h	12:00 h	13:00 h	**novembro**	8:10 h	8:20 h	9:10 h	9:30 h
junho	9:00 h	9:10 h	12:40 h	13:10 h	**dezembro**	8:20 h	8:30 h	9:00 h	9:20 h

Ao analisar as imagens ao longo do ano de 2020 verificou-se a existência de dois períodos onde ocorre a existência de sombra perto da zona da linha de água. Existe um primeiro período de sombra que é provocado pelos prédios, sendo que o segundo período a sombra é provocado pela própria parede da ribeira. Outra situação importante

a referir é que durante estes períodos tipicamente não ocorre a alteração do nível de água, ou seja, o nível mantém-se.

Em relação aos resultados da tabela 3.2 observa-se que os dois períodos de sombra, no contexto de um ano inteiro (neste caso o ano de 2020), coincidem, visto que o primeiro período vai desde as 8:10 h até às 9:30 h e o segundo período vai desde as 9:00 h até às 13:10 h. Assim sendo juntou-se os dois períodos no contexto da implementação do algoritmo, sendo que irá existir certas particularidades que irão distinguir os dois períodos no algoritmo.

Para a implementação do algoritmo foi necessário utilizar a imagem, depois de compensada, com um aumento do brilho, sendo que esta serviu para diferenciar os diferentes períodos. A função que permitiu aumentar o brilho de uma imagem é a seguinte:

image_bright = bright(image, brightness).

A função tem como parâmetros a imagem (*image*) e o brilho pretendido na imagem (*brightness*). Tem como saída a imagem com o brilho aplicado (*image_bright*). Esta função permite aumentar o brilho de uma imagem a cada pixel correspondente à cor azul, verde e vermelho (*RGB – Red, Green and Blue*), multiplicando a cada pixel o valor do brilho pretendido. Neste trabalho o valor utilizado para aumentar o brilho foi de 2, visto que este valor permitiu evidenciar a zona de sombra na região de interesse. Um valor mais elevado irá provocar a perda de informação, não referente à zona de sombra, mas da posição da linha de água.

Na figura 3.40 é apresentado um exemplo de uma imagem depois de equalizada e compensada com aumento do brilho.

Figura 3.40 - Imagem durante o dia no "período de sombra" com aumento do brilho.

De seguida procedeu-se à deteção de picos no gráfico do gradiente, por forma a obter os mais pronunciados, sendo esta deteção efetuada dentro da zona de pesquisa (pixel 90 até 180). É importante referir que para efetuar este processo de deteção de

picos, é necessário converter os valores do gráfico do gradiente, na zona de pesquisa, em valores em módulo.

A função que permitiu a deteção de picos é a seguinte:

gradient_peaks, _ = find_peaks(array_gradient, height = mean_gradient).

Nesta função tem-se como parâmetros o *array* com os valores em módulo do gradiente (*array_gradient*), dentro da zona de pesquisa, e a altura mínima dos picos (*height*) [52]. Para este último valor foi efetuada uma média dos valores do módulo do gradiente (*mean_gradient*), dentro da zona de pesquisa, e o resultado foi usado para a altura mínima dos picos. Este valor permitiu reduzir algum ruído que existia devido à rugosidade da parede ou de detritos existentes na água. Tem-se como saída um *array* com os índices dos picos que satisfazem todas as condições fornecidas (*gradient_peaks*), sendo que neste caso a única condição é a altura do pico.

De seguida obteve-se o pico mais pronunciado, que corresponde ao valor máximo do módulo do gradiente. Para verificar que se está perante uma situação de "período de sombra", após a obtenção do pico mais pronunciado, na imagem global (equalizada e compensada) com o aumento do brilho, foram delimitadas duas zonas acima e abaixo da posição desse pico. Estas zonas tem a dimensão de 200×10 pixéis. Na figura 3.40 está evidenciado a região de interesse e as duas zonas na posição do valor máximo absoluto do gradiente detetado.

Após a obtenção dessas duas zonas foi efetuada uma média dos valores de escala de cinza de cada zona e depois foi feita a diferença da média da zona superior e inferior. Essa diferença é comparada com um valor limite por forma a verificar se está perante um "período de sombra". Para a obtenção do limite foi necessário analisar os valores da diferença para vários dias que contenham a existência de sol, sendo que foi utilizado imagens a uma cadência de 10 minutos. No gráfico da figura 3.41 são apresentados os resultados obtidos.

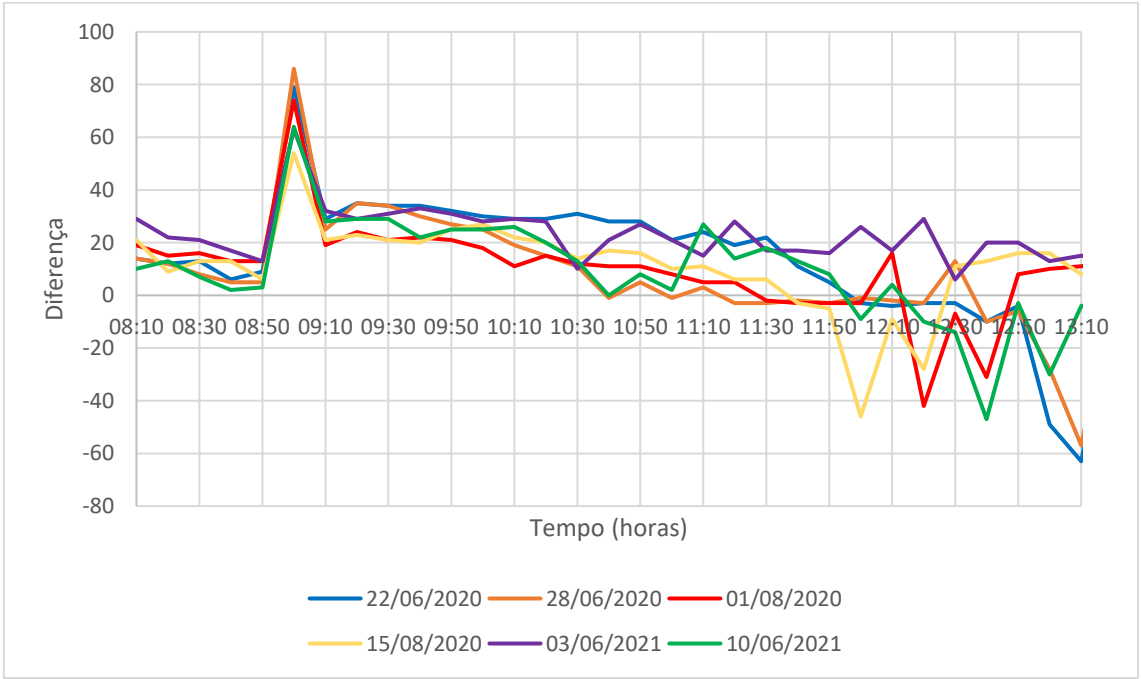


Figura 3.41 - Resultados obtidos para a diferença entre zonas.

Analisando os resultados apresentados na figura 3.41 verificou-se que a ocorrência de sombra dentro da zona de pesquisa da região de interesse apresenta-se em torno das 09:00h obtendo um valor de diferença elevado. Assim sendo, o valor escolhido para o limite de deteção do primeiro “período de sombra” foi de 40 . Caso o valor da diferença obtido seja inferior ao limite, considera-se que não se está perante um “período de sombra” e o valor máximo do módulo do gradiente obtido corresponde à posição da linha de água. Caso a diferença seja superior ao limite, considera-se um “período de sombra” e é efetuada novamente uma pesquisa desde o pixel 90 até um pixel abaixo do pixel correspondente ao valor máximo do módulo do gradiente por forma a verificar se foi detetado mais algum pico. Caso tenha sido detetado um ou mais do que um pico dentro dessa zona de pesquisa, o pico mais pronunciado é o que será considerado como a posição da linha de água. Caso não seja detetado nenhum pico nessa zona de pesquisa isso pode corresponder que a fronteira da zona iluminada pelo sol e a zona da sombra está a coincidir com a posição da linha de água. Assim sendo, o pico correspondente ao valor máximo do módulo do gradiente é o que será considerado para a posição da linha de água.

Na figura 3.42 é apresentado um exemplo de uma imagem no primeiro “período sombra” com a deteção correta da linha de água (linha a branco).

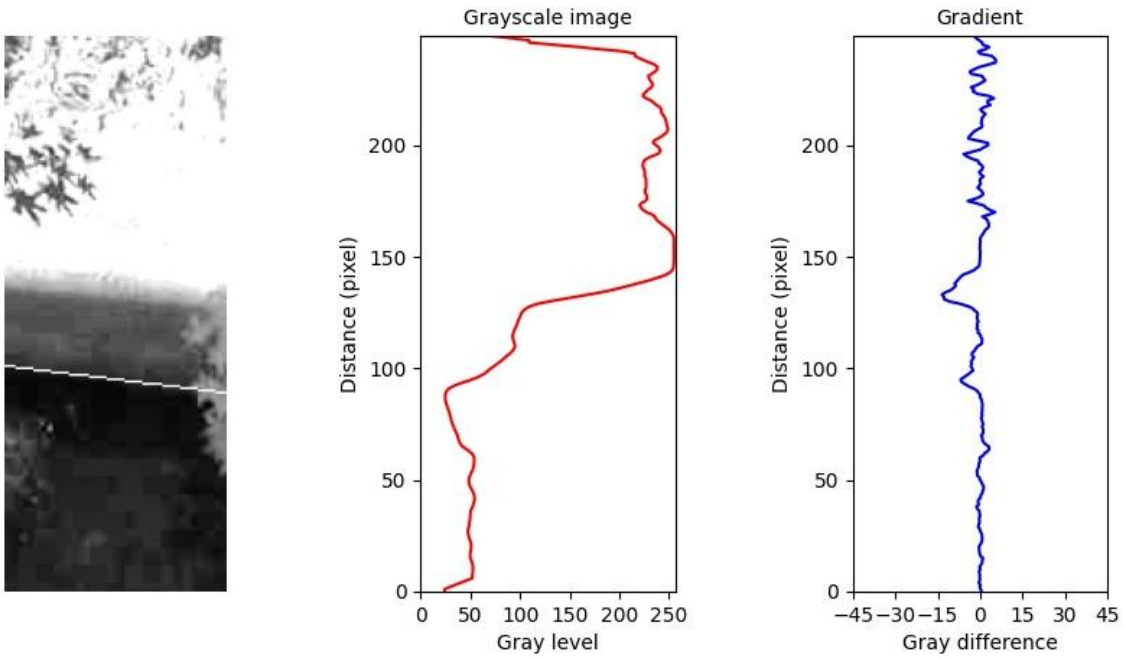


Figura 3.42 - Deteção da linha de água para uma imagem durante o primeiro "período de sombra".

Relativamente ao segundo “período de sombra” apresentado na tabela 3.2 foi necessário adotar um procedimento diferente em relação ao primeiro período. Para a deteção deste período recorreu-se primeiro à diferença entre as duas zonas apresentadas no gráfico da figura 3.41. Durante esse período o valor da diferença tende a ser negativo como é possível ver desde às 12:00 h até às 13:10 h. Assim sendo, o valor escolhido para o limite de deteção do segundo “período de sombra” foi de -20. Após ser detetado o segundo “período de sombra” é efetuada uma pesquisa desde o pixel 90 até ao 180 na região de interesse, sem a aplicação do aumento de brilho na imagem depois de compensada, e são obtidos os vários picos. É importante referir que nesta deteção não foi usado o aumento de brilho na imagem, visto que é possível evidenciar bem a sombra e a linha de água, algo que com o aumento de brilho deixava

de ser menos evidente. Após a deteção dos picos é escolhido o pico mais pronunciado, mas com valor negativo. A escolha recai no pico mais negativo, visto que, neste segundo período, quando a sombra está na água existente na ribeira, a transição entre a parede e a água irá criar um pico negativo no gradiente, enquanto a transição entre a água e a sombra irá criar um pico positivo, logo o pico negativo é o que indica a posição da linha de água.

Na figura 3.43 é apresentado um exemplo de uma imagem no segundo “período sombra” com a deteção correta da linha de água (linha a branco).

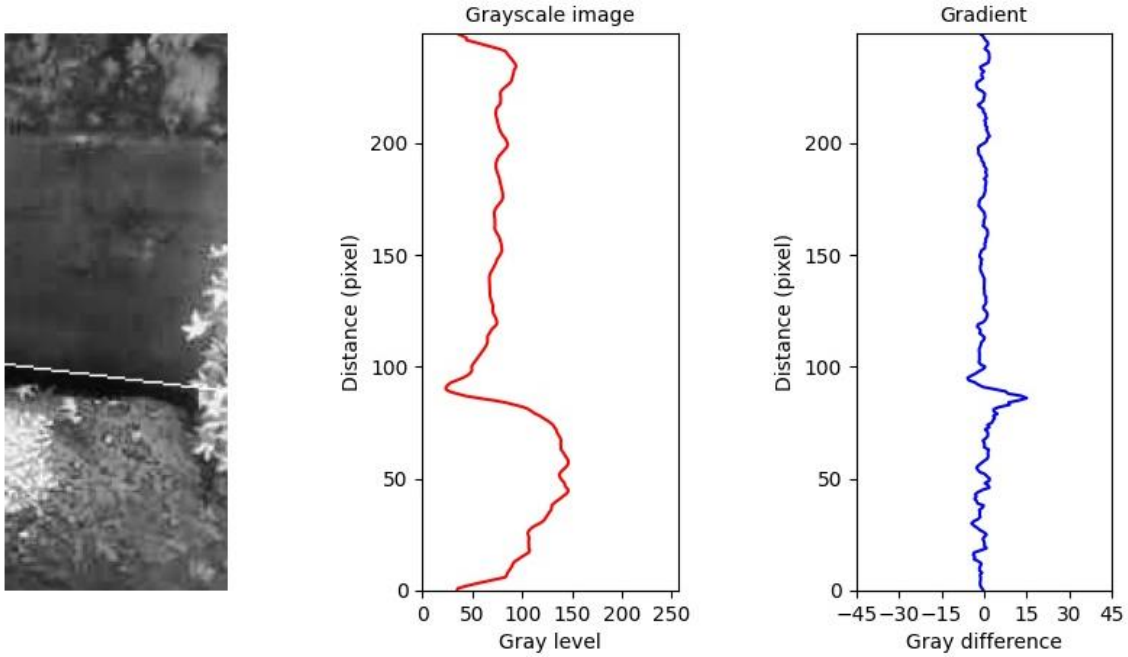


Figura 3.43 - Deteção da linha de água para uma imagem durante o segundo "período de sombra".

Para imagens capturadas durante a noite foi importante ter em atenção que um dos problemas principais tem a ver com a iluminação da região de interesse. Como a ribeira encontra-se num meio urbano, o sistema de iluminação pública tende a ser suficiente para iluminar a região de interesse. Assim sendo, não foi instalado nenhum equipamento de iluminação permitindo minimizar os custos de instalação de um sistema dedicado que iria necessitar de alimentação elétrica.

Apesar da iluminação pública ser suficiente para a análise feita neste trabalho, é necessário ter em atenção que os postes de iluminação estão orientados para avia pública, sendo esperado que na região de interesse existam zonas com sombra no período noturno. Com isto foi necessário distinguir o procedimento para a obtenção da linha de água para imagens durante o dia e durante a noite. Como a aquisição de imagens durante a noite requer diferentes parâmetros na câmara, esta informação foi utilizada para efetuar a distinção entre as duas situações.

Na figura 3.44 são apresentados os gráficos da média móvel de escala de cinza e do gradiente para uma imagem retirada durante a noite.

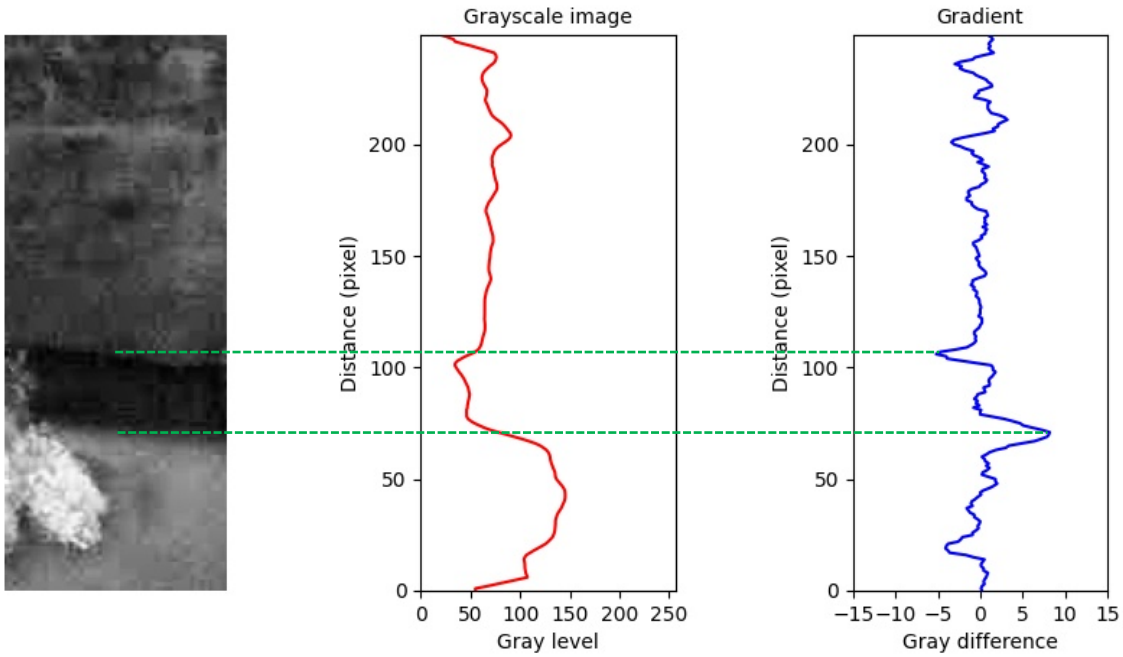


Figura 3.44 - Imagem retirada durante a noite com os gráficos respetivos.

Como pode ser observado na figura 3.44, o valor máximo do módulo do gradiente ocorre na zona de sombra sobre a água e não na linha de água. No entanto, a posição da zona de sombra pode ser usada para efetuar a deteção da linha de água, visto que uma variação do nível da água irá corresponder a uma variação da zona de sombra. A distância da zona de sombra até a linha de água é definida pelo parâmetro P_W medido na dimensão vertical.

Para a obtenção deste parâmetro foi efetuada uma análise para diferentes níveis de água da ribeira por forma a avaliar a variação do mesmo. Para esta análise foram utilizadas cerca de 20 imagens.

No gráfico da figura 3.45 são apresentados os resultados do parâmetro P_W para cada nível.

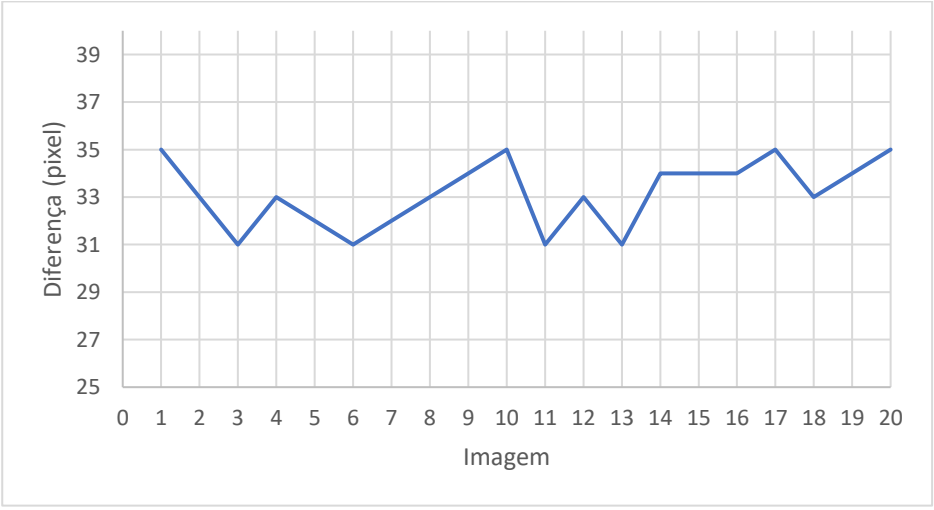


Figura 3.45 - Distância entre a zona de água a sombreado e a linha de água para imagens durante a noite.

Como é possível observar pela figura 3.45, o valor do parâmetro P_W apresenta uma ligeira oscilação, sendo que o valor que mais se adequa à variação é 33 ± 2 pixéis. Assim sendo, será este o valor que será utilizado como referência no algoritmo para o parâmetro P_W.

Na figura 3.46 é apresentado um caso, para imagens durante a noite, em que não é possível detetar a linha de água, mas sim a zona de sombra, sendo esta usada para detetar a posição de linha de água.

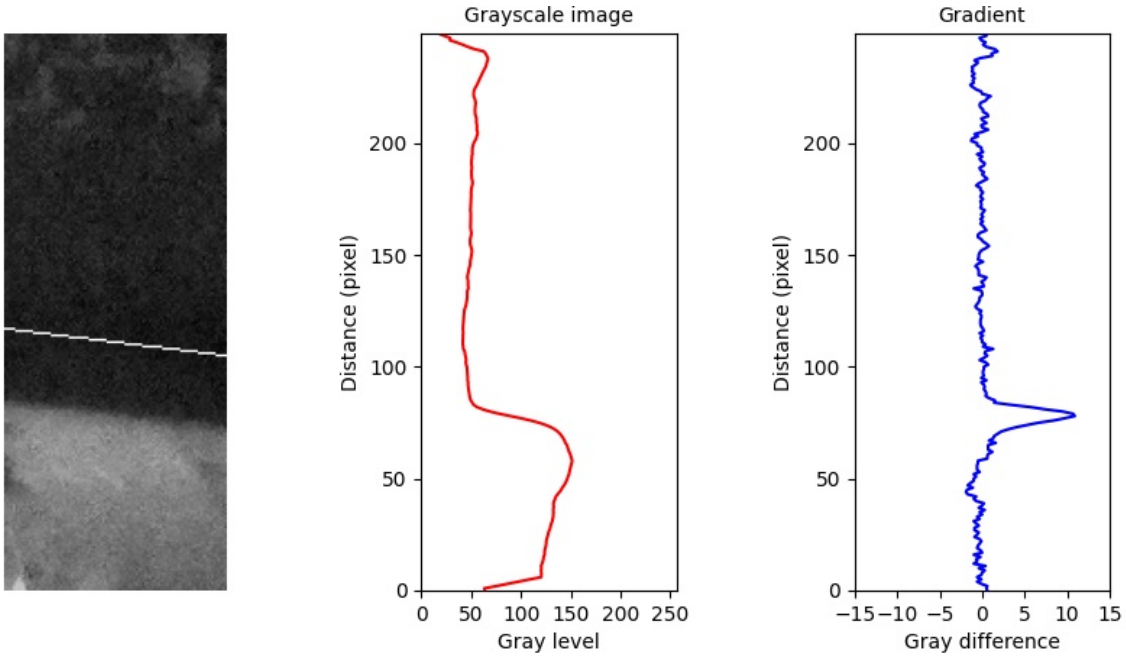


Figura 3.46 - Imagem durante a noite em que a linha de água é detetada através zona de sombra na água.

Por fim, a função implementada que permitiu obter a linha de água para todas as situações descritas anteriormente é a seguinte:

pixel_water_line = detect_water_line(image_bright, array_axis_y, array_gradient, array_gradient_bright, height, width, height_zones, width_zones, lim_min_day, lim_min_night, lim_max, lim_zones_max, lim_zones_min, coord_top_left, type, sun_situation, off_night).

Nesta função tem-se como parâmetros a imagem com o aumento de brilho (*image_bright*) depois de compensada, o *array* com os valores dos pixéis (*array_axis_y*) que corresponde a cada valor de gradiente presente no *array* com os valores do gradiente (*array_gradient*), o *array* com os valores do gradiente (*array_gradient_bright*) para a situação de aumento do brilho, a altura e largura da região de interesse (*height* e *width*, respetivamente), a largura e altura das zonas (*height_zones* e *width_zones*, respetivemente) apresentadas na figura 3.32, os limites de pesquisa da posição da linha de água nos *arrays* durante o dia e noite (*lim_min_day* e *lim_min_night*, respetivamente), o limite máximo de pesquisa (*lim_max*), o limiar máximo e mínimo (*lim_zones_max* e *lim_zones_min*, respetivamente) utilizado para o primeiro e segundo "período de sombra", respetivamente, as coordenadas do ponto superior esquerdo da região de interesse (*coord_top_left*), o tipo de imagem (*type*) que indica se a imagem foi obtida durante o dia ou durante a noite, a variável (*sun_situation*) que indica se a imagem obtida está dentro do "período de sombra" e por fim o *offset* (P_W) a adicionar a posição da linha

de água detetada durante a noite (*off_night*). Como saída tem-se a posição, em pixel, da linha de água (*pixel_water_line*).

No fluxograma da figura 3.47 são descritos os passos efetuados de forma a perceber melhor a função que permite obter a linha de água.

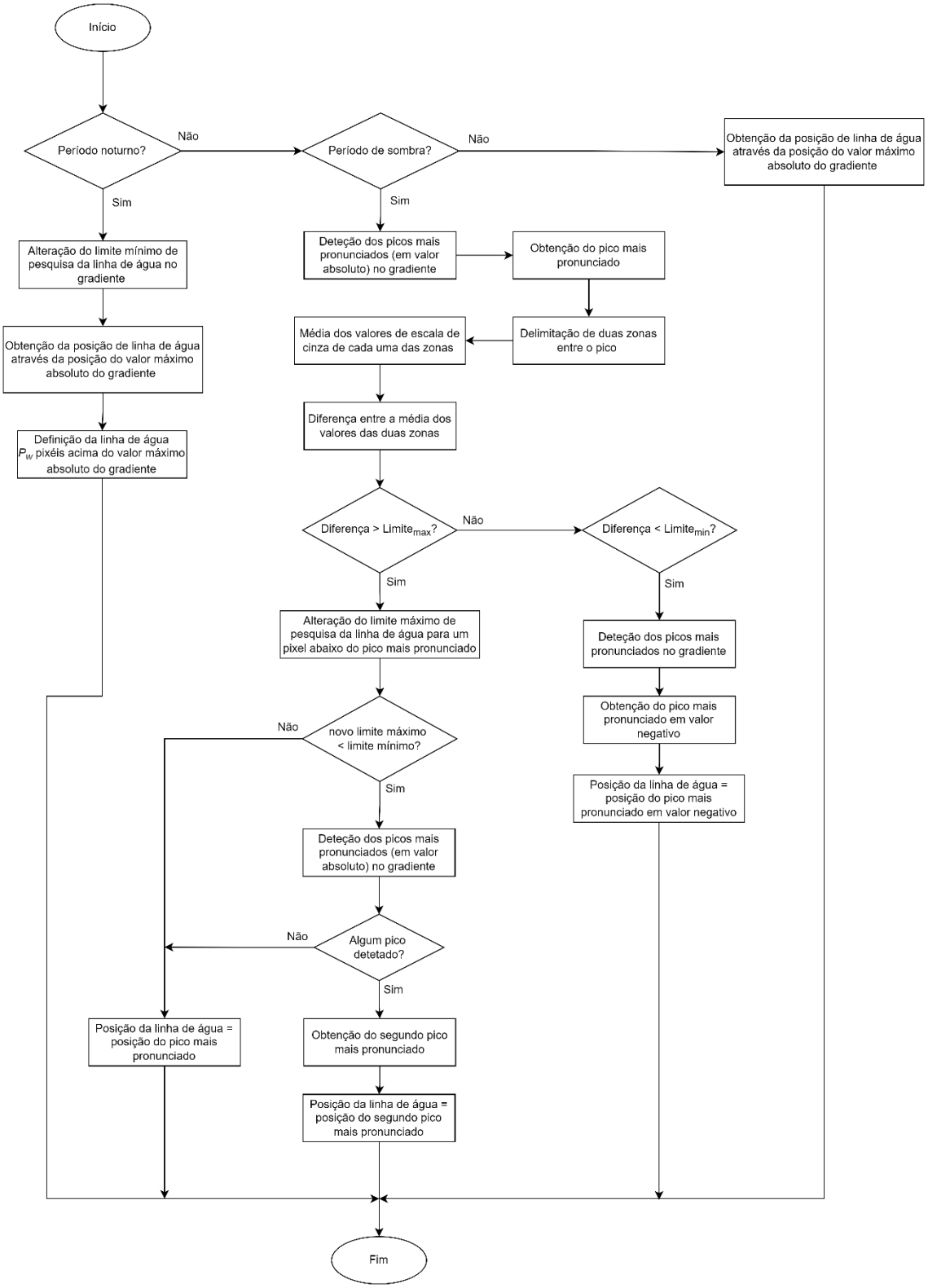


Figura 3.47 - Fluxograma que descreve os passos realizados na função para a deteção da linha de água.

3.5.3. Estimação da altura do caudal

A altura do caudal foi estimada utilizando as equações (3.7) e (3.8) para obter X e Y, com os parâmetros do plano da parede da zona de betão. Foram necessários três parâmetros para este procedimento, sendo estes: a distância d, o ângulo horizontal α_x e o ângulo vertical α_y. Através de medições locais e por simulação foi possível confirmar que o plano da zona de betão era diferente do plano da zona da parede de pedra. Seguindo o procedimento aplicado na secção 3.3, o plano da zona de betão tem os seguintes parâmetros: $d = 24{,}16\,m$, $\alpha_x = 7{,}4°$ and $\alpha_y = 22{,}2°$.

Após isto procedeu-se à obtenção das coordenadas (P'_x, P'_y) apresentado na figura 3.31 na secção 3.5.1. Estas coordenadas têm como referência o sistema de coordenadas $x'y'$. Primeiramente foi obtida a equação da reta da linha de calibração e da linha de água (considerando o sistema $x'y'$), sendo estas, respetivamente:

$$y = m_c\left(x - b_{y'=0}\right) \qquad e \qquad y = m_l x + y_i \tag{3.17}$$

em que $m_c = \frac{1}{0{,}174}$ corresponde ao declive da linha de calibração, $b_{y'=0} = 30$ corresponde a abcissa quando $y' = 0$, $m_l = -0{,}1268$ corresponde ao declive da linha de água e y_i corresponde ao valor obtido pelo algoritmo realizado.

Em relação à linha de calibração, a equação da reta foi efetuada através de medições realizadas no local da ribeira. Como o plano real da imagem tem uma ligeira inclinação em relação à vertical, esta linha de calibração permitiu obter a altura do caudal na vertical correspondente ao plano real da imagem.

De seguida procedeu-se à obtenção do valor de P'_x. Para isso igualou-se ambas as equações (3.17), obtendo a seguinte expressão:

$$P'_x = \frac{\frac{1}{m_c} y_i + b_{y'=0}}{1{,}022} \tag{3.18}$$

Com o valor de P'_x é possível obter o valor de P'_y através de uma das equações (3.17), substituindo o valor de x por P'_x. O valor de P'_y irá corresponder ao valor de P_H apresentado na figura 3.31.

Com o parâmetro P_H determinado, a posição da linha de água, que corresponde às coordenadas (P_x, P_y), foi obtida através do sistema de coordenadas xy apresentado na figura 3.31. Primeiro foi necessário obter as coordenadas (x_{roi}, y_{roi}) apresentado na figura 3.31. Utilizando o sistema de coordenadas xy e o ponto central da imagem (origem do sistema xy) obteve-se as coordenadas do ponto inferior esquerdo da região de interesse, sendo $(350, -380)$. Após isto obteve-se o valor de P_y. O valor de P_y é dado por $P_y = P_H + P_{y0}$, em que P_{y0} corresponde ao valor de y_{roi}, sendo este valor de -380. O valor de P_x foi obtido através da equação da reta da linha de calibração, mas agora em relação ao sistema xy, sendo esta:

$$P_x = \frac{1}{m_c} P_y + x_{roi} + {x'}_{y'=200} \tag{3.19}$$

em que ${x'}_{y'=200} = 116$ corresponde a abcissa quando $y' = 200$. Conhecendo os valores de P_x e P_y é possível obter os valores para x_1 e y_1, respetivamente. Considerando as

equações (3.7) e (3.8), as distâncias no plano da parede são dadas pelas seguintes expressões:

$$X = \frac{dx_1}{x_1 \sin(\alpha_x) + y_1 \cos(\alpha_x) \tan(\alpha_y) + d \cos(\alpha_x)} \tag{3.20}$$

$$Y = \frac{y_1 d - y_1 X \sin(\alpha_x)}{y_1 sin(\alpha_y) + d \cos(\alpha_y)} \tag{3.21}$$

A altura do caudal da ribeira, em metros, foi determinada através da diferença entre Y e Y_0, em que Y_0 corresponde à distância considerada para a altura do caudal zero. A altura do caudal zero corresponde, no sistema $x'y'$, na reta da linha de calibração às coordenadas $P'_{Y_0} = 90$ e $P'_{X_0} = 46$, que corresponde, no sistema xy, às coordenadas $P_{Y_0} = -290$ e $P_{X_0} = 416$. A partir do valor de P_{Y_0} chega-se ao valor de Y_0, sendo este $-5{,}25\ m$.

A função implementada que permite a conversão do valor em pixel para centímetros é a seguinte:

water_level = convert_pixel_cm(value_line_water, slope_water_line, slope_calibration, b_slope_calibration, x_roi, y_roi, x__y_200, pixel_size, img_cropping1, img_cropping2, sensor_size_vertical, img_resolution_vertical, sensor_size_horizontal, img_resolution_horizontal, distance_camera_wall, alpha_y_rad, alpha_x_rad, focal_length, zero_flow_height).

Nesta função tem-se como parâmetros a posição, em pixel, da linha de água (*value_line_water*), o declive (m_l) da reta da linha de água (*slope_water_line*), o declive (m_c) da reta da linha de calibração (*slope_calibration*), o valor da abcissa ($b_{y'=0}$) quando a reta da linha de calibração interseta o eixo y (*b_slope_calibration*), as coordenadas (x_{roi} e y_{roi}) do ponto inferior esquerdo da região de interesse (*x_roi* e *y_roi*), a abcissa ($x'_{y'=200}$) quando o valor de $y' = 200$ (*x__y_200*), o tamanho (T) do pixel (*pixel_size*), o corte (C_1) da imagem devido à alteração da proporção da tela (*img_cropping1*), o corte (C_2) da imagem devido a questões de privacidade (*img_cropping2*), o tamanho (S_V) do sensor na vertical (*sensor_size_vertical*), a resolução (R_V) da imagem na vertical (*img_resolution_vertical*), o tamanho (S_H) do sensor na horizontal (*sensor_size_horizontal*), a resolução (R_H) da imagem na horizontal (*img_resolution_horizontal*), a distância (d) entre a lente da câmara e o plano do objeto (*distance_camera_wall*), o ângulo vertical e horizontal (α_y e α_x) em radianos (*alpha_y_rad* e *alpha_x_rad*), o comprimento (f) focal (*focal_length*) e a altura (Y_0) do caudal zero (*zero_flow_height*). Tem-se como saída o valor do nível de água em centímetros (*water_level*).

Para certas imagens foi muito difícil ou até impossível obter a altura do caudal da ribeira. Nesta situação, os erros cometidos são elevados. Uma filtragem dos dados permitiu minimizar estes erros. Um dos procedimentos que pode ser efetuado nesta situação é a eliminação dos valores que excedam um determinado limiar quando estes são comparados com os resultados anteriores. Pela análise da evolução do nível do caudal ao longo do tempo verificou-se que, na maioria dos casos, a altura do caudal da ribeira poderá aumentar substancialmente num curto período, mas a diminuição da mesma é mais lenta no tempo. Assim sendo, foram analisados os resultados da altura do caudal a uma cadência de 1 minuto e os limites obtidos foram de 26,8 cm (14 pixéis) para o aumento do caudal e -11,4 cm (-6 pixéis) aquando da diminuição do caudal.

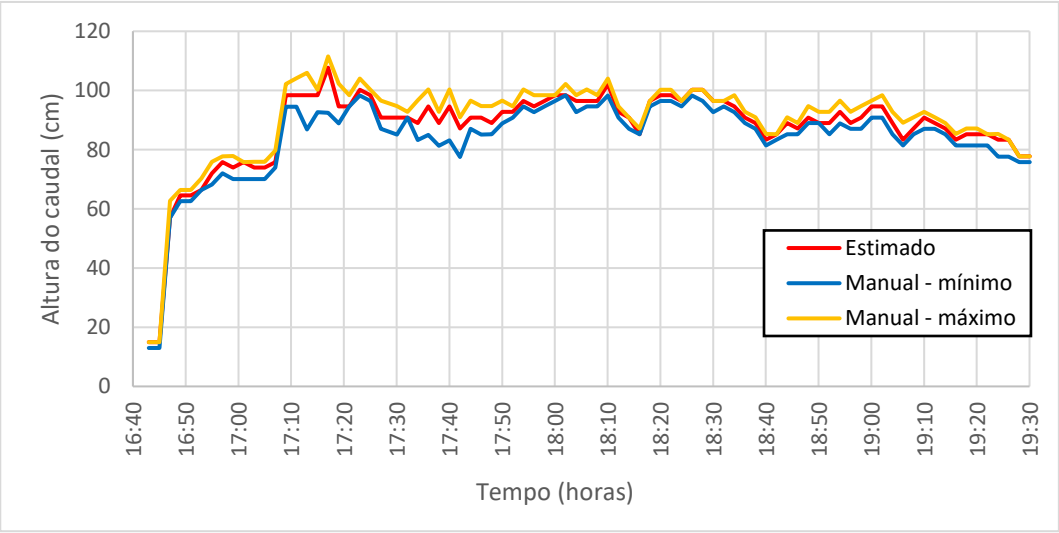
No Anexo A é apresentado o código desenvolvido neste trabalho para a deteção do nível de água.

4. Análise de resultados

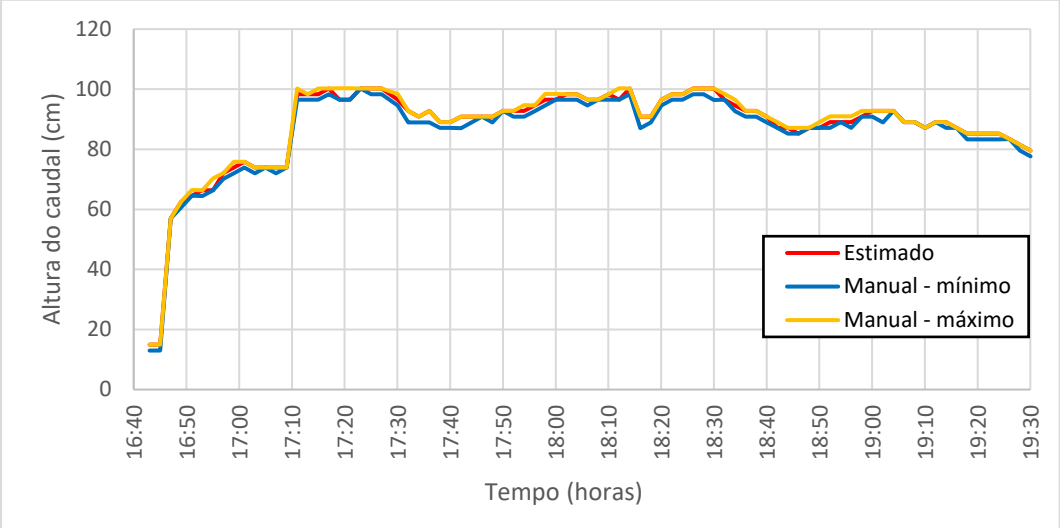
Neste ponto do trabalho serão apresentados os resultados obtidos pelo algoritmo implementado para diferentes situações ocorridas ao longo do tempo. As imagens utilizadas para testar o algoritmo desenvolvido foram as adquiridas entre janeiro 2020 e julho de 2021. Estas imagens foram capturadas a cada 10 minutos na maioria do tempo, sendo que em períodos de precipitação (menos frequente) as imagens foram capturadas a cada minuto, visto que nestes períodos é onde normalmente ocorre a variação do nível de água, estando o restante tempo com baixos nível de caudal e de valor praticamente constante. De todas as imagens adquiridas foram descartadas cerca de 8% das imagens, visto que as mesmas apresentavam imensa vegetação na zona da região de interesse utilizada neste trabalho. Para analisar os resultados de várias situações que ocorreram dentro do período de estudo, foi necessário efetuar uma filtragem das imagens utilizadas, sendo que, da totalidade de imagens adquiridas, foram utilizadas 10558 imagens nesta análise de resultados.

4.1. Imagem individual versus média temporal

Primeiramente foi efetuada uma experiência para verificar as diferenças na exatidão utilizando apenas a imagem individual e utilizado uma média temporal de várias imagens. Na figura 4.1 são apresentados os gráficos com os resultados num período do dia de 3 abril de 2020 em que ocorreu precipitação. As imagens utilizadas não foram afetadas por vestígios de chuva, visto que a precipitação ocorreu nas montanhas. É importante referir que entre as 17:30h e as 19:10h houve a existência de detritos na região de interesse. Os gráficos mostram a comparação entre os valores estimados pelo algoritmo e os valores medidos manualmente a partir das imagens e da régua graduada. Para a média temporal foi obtida uma imagem média a partir de $T = 5$ imagens.



a)



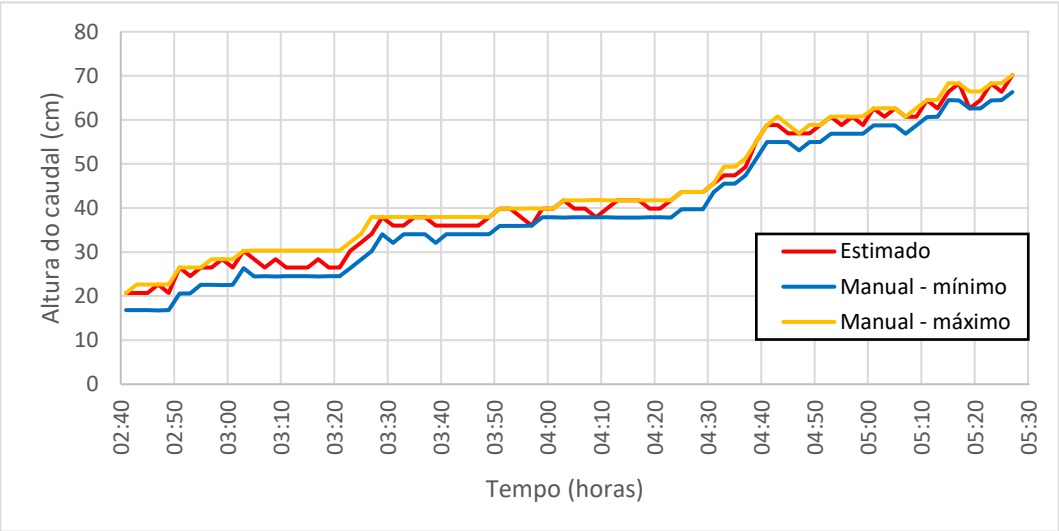
b)

Figura 4.1 – Estimativa do nível de água durante um período do dia 3 de abril de 2020: a) imagem individual; b) imagem média a partir de $T = 5$ imagens.

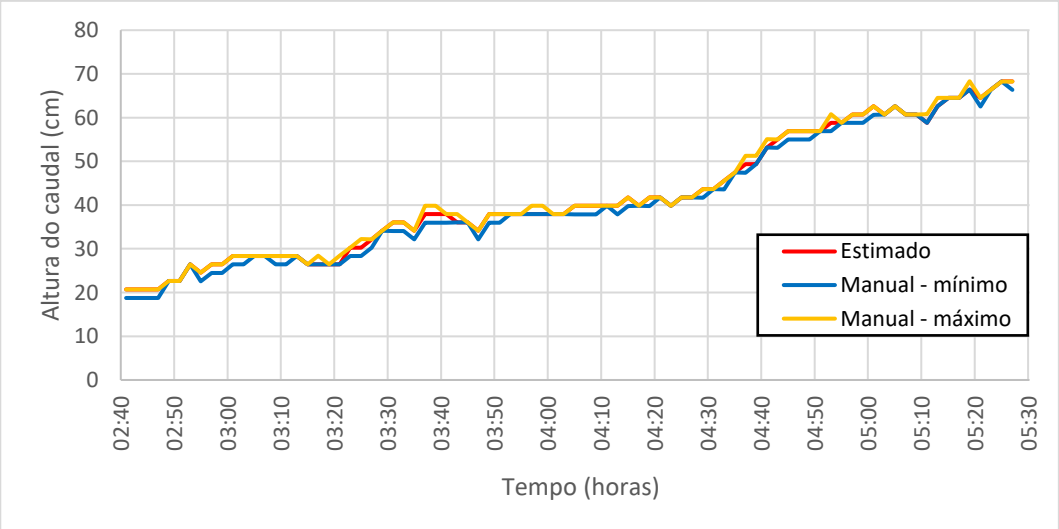
Relativamente ao gráfico da imagem 4.1 a) obteve-se uma ondulação média da água de 6,1 cm e a ondulação máxima foi de 19,1 cm. A exatidão obtida utilizando imagens individuais foi de 1,0 cm. Relativamente ao gráfico da figura 4.1 b) obteve-se uma exatidão de 0,8 cm para uma ondulação média de 2,0 cm. É possível aumentar a exatidão com a média temporal das imagens porque esta permite combinar informações de diferentes posições da linha de água.

Para o período noturno, foi determinado inicialmente na seção 3.5 o parâmetro P_W antes de estimar o nível de água. Após a determinação desse parâmetro (33 ± 2 pixéis) foi necessário obter a exatidão em relação a este parâmetro de forma a acrescentar à exatidão obtida para o nível de água. Assim sendo, a exatidão obtida por este processo foi de 2,1 cm. De seguida procedeu-se ao mesmo método efetuado anteriormente, mas agora durante a noite, para analisar a diferença de exatidão entre a utilização de imagens individuais ou utilizado uma média temporal de várias imagens.

Na figura 4.2 são apresentados os gráficos com os resultados num período da noite no dia 20 de fevereiro de 2021 em que ocorreu ligeira precipitação e aumento do caudal. É importante referir que entre as 03:00h e as 04:00h foi detetada a existência de detritos na região de interesse. As imagens utilizadas foram afetadas por vestígios de chuva, sendo que não afetou a obtenção do nível de água.



a)



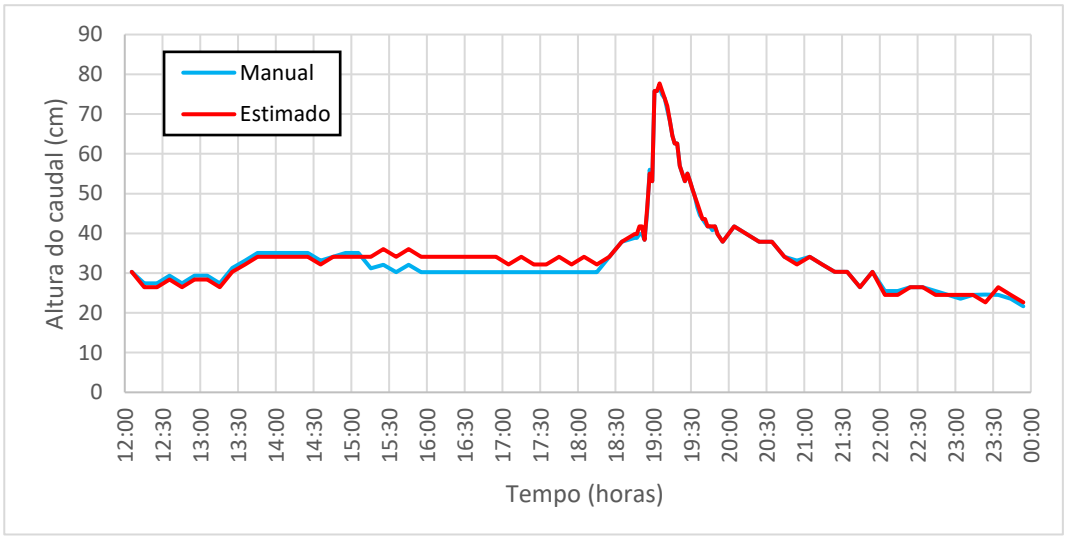
b)

Figura 4.2 - Estimativa do nível de água durante um período da noite do dia 20 de fevereiro de 2021: a) imagem individual; b) imagem média a partir de $T = 5$ imagens.

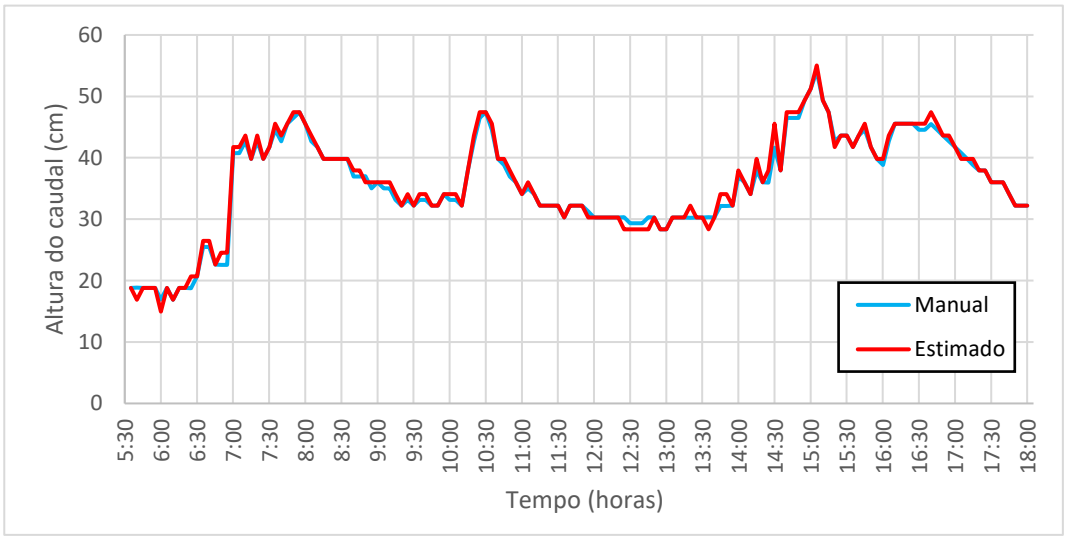
Relativamente ao gráfico da figura 4.2 a) obteve-se uma exatidão de 3,2 cm. A razão pela qual esta exatidão é baixa é devido ao facto que, apenas com imagens individuais, a sombra tende a não ser muito definida, estando a região sombra-água mais desvanecida. Este problema tende a ser mais atenuado com a questão da média temporal como é possível observar no gráfico da figura 4.2 b), em que a exatidão é de 2,6 cm. É importante referir que, com a média temporal das imagens, é possível melhorar a exatidão, mas esta tende a ser menor do que a durante o dia, visto que a obtenção do nível de água durante a noite é efetuada a partir de um método indireto através da sombra sobre a água.

4.2. Precipitação urbana

De seguida procedeu-se à análise de situações em que ocorre precipitação na zona urbana com aumento do caudal. Na figura 4.3 a) e b) são apresentados os resultados obtidos no dia 17 de abril de 2020 e no dia 27 de março de 2021, respetivamente.



a)



b)

Figura 4.3 - Estimativa do nível de água durante períodos de precipitação no dia: a) 17 de abril de 2020; b) 27 de março de 2021.

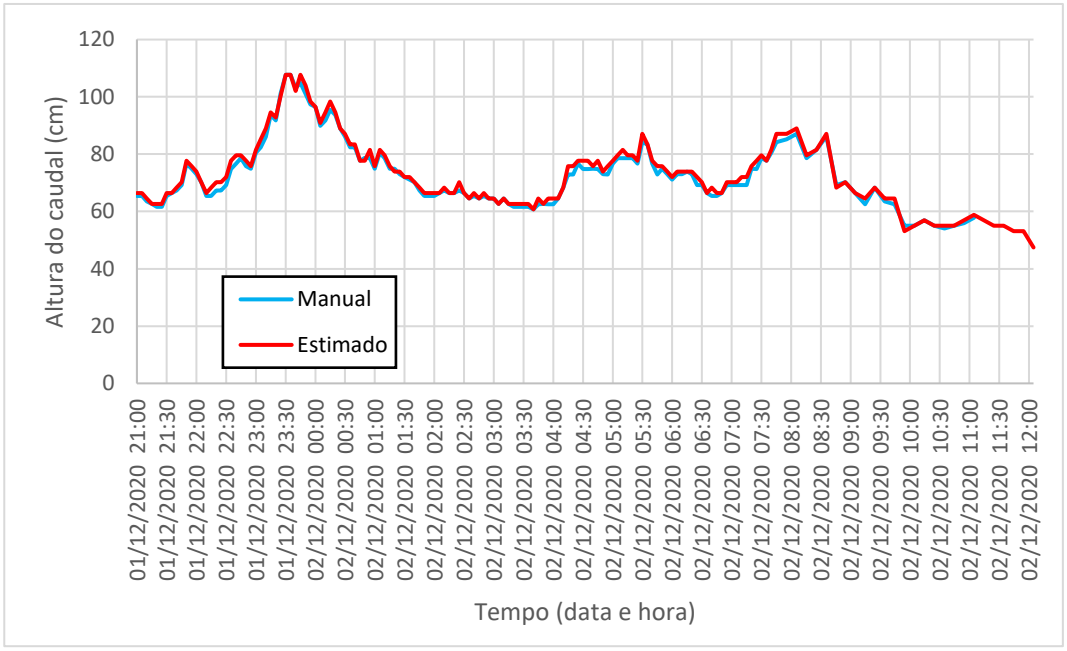
Relativamente ao gráfico da figura 4.3 a) houve um ligeiro aumento do nível de água no período entre 13:15h e 13:45h e depois o nível diminui lentamente. Das 15:15h até as 18:15h é possível observar um erro de 2 a 4 cm devido à humidade existente na parede da ribeira, fazendo com que a obtenção do nível de água por parte do algoritmo esteja acima do valor correto. Após as 18:15h começa a existência de precipitação na zona urbana e o nível de água aumenta drasticamente. Entre as 19:10h até às 20:00h ocorre uma descida acentuada do nível de água, sendo que após as 20:00h até às

00:00h a descida é mais ligeira. Em relação a exatidão, tem-se 1,3 cm durante o dia e 2,5 cm durante a noite.

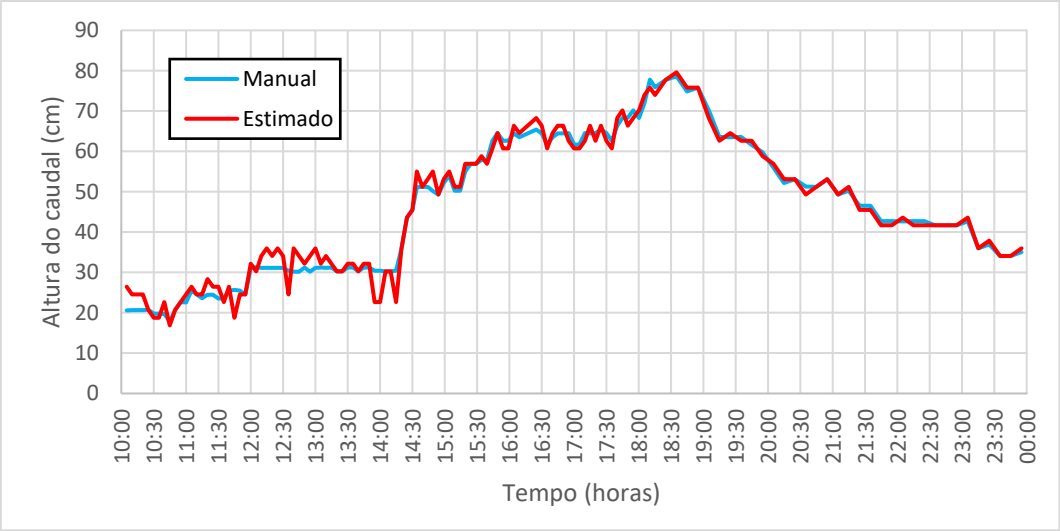
Em relação ao gráfico da figura 4.3 b) houve subidas acentuadas do nível da água nos períodos de 06:45h e 7:15h e de 10:10h e 10:30h, sendo que nestas situações ocorreram precipitação na zona urbana. Após estes dois períodos ocorreu uma descida do nível de água mais ligeira comparada com as subidas. No período de 13:35h e 15:05h ocorre uma subida do nível de água de forma mais incremental ao longo do tempo, sendo que neste período também ocorreu a existência de precipitação. Após isto ocorreu uma ligeira diminuição do caudal seguida de um ligeiro aumento, sendo que este aumento se deve a um aumento da precipitação nas zonas montanhosas ou a algum aumento de um curso de água que esteja conectado à ribeira. Após este aumento ocorreu a diminuição do nível de água na ribeira. Em relação à exatidão, tem-se 0,5 cm durante o dia e 2,8 cm durante a noite. A exatidão durante o dia é maior referente ao resultado obtido ao gráfico da figura 4.1 b) porque nesta situação, apesar de existir precipitação, o aumento do caudal não provocou uma forte ondulação.

4.3. Forte precipitação urbana

Após a obtenção dos resultados referentes a situações com períodos de ligeira precipitação, procedeu-se à análise de resultados para situação onde ocorre uma forte precipitação na zona onde a ribeira está localizada. Na figura 4.4 a) e b) são apresentados os resultados obtidos no dia 1 e 2 de dezembro de 2020 e no dia 25 de dezembro de 2020, respetivamente.



a)



b)

Figura 4.4 - Estimativa do nível de água durante períodos de grande precipitação no dia: a) 1 e 2 de dezembro de 2020; b) 25 de dezembro de 2020.

Relativamente ao gráfico da figura 4.4 a) houve um aumento do caudal das 21:00h até às 23:45h, sendo que ao longo deste aumento ocorreu a existência de eventos de grande precipitação na zona urbana e também a existência de detritos na água. Das 23:45h às 04:00h ocorreu uma descida do caudal para valores próximos dos existentes às 21:00h, sendo que ao longo deste período ocorreu aumentos ligeiros do caudal, sendo estes provavelmente provocados por um aumento de precipitação das zonas montanhosas. Das 04:00h até as 05:30h ocorreu um aumento ligeiro do caudal, existindo também neste período a ocorrência de grande precipitação na zona urbana, sendo que depois as 05:30h até as 07:00h ocorreu a descida do nível de água. Das 07:00h às 08:35h volta a ocorrer um aumento do caudal, seguido de uma descida do nível da água (das 8:35h até as 10:00h) e estabilização do mesmo. Para esta situação tem-se uma exatidão de 0,6 cm durante o dia e 3,3 cm durante a noite.

Em relação ao gráfico da figura 4.4 b) das 10:00h até às 14:00h tem-se várias oscilações no nível do caudal, sendo estas provocadas pela grande precipitação na zona urbana, fazendo com que haja um ligeiro erro na deteção do nível de água. Das 14:00h às 15:00h tem-se uma grande subida do nível de água, sendo que este continua a subir de uma forma menos acentuada até as 18:35h. Durante este período continua a existir ligeiras oscilações provocadas pela grande precipitação. Após as 18:35h tem-se uma descida do nível de forma ligeira até às 00.00h do dia 26 de dezembro. Nesta situação tem-se uma exatidão de 2,0 cm durante o dia e 2,8 cm durante a noite, sendo que as imagens durante a noite são menos afetadas do que durante o dia, pois a qualidade da imagem é mais estável.

4.4. Sombra provocada pelo sol

De seguida procedeu-se à análise de resultados referentes à situação do "período de sombra". Na figura 4.5 são apresentados os resultados obtidos no dia 10 de junho de 2021 durante o "período de sombra".

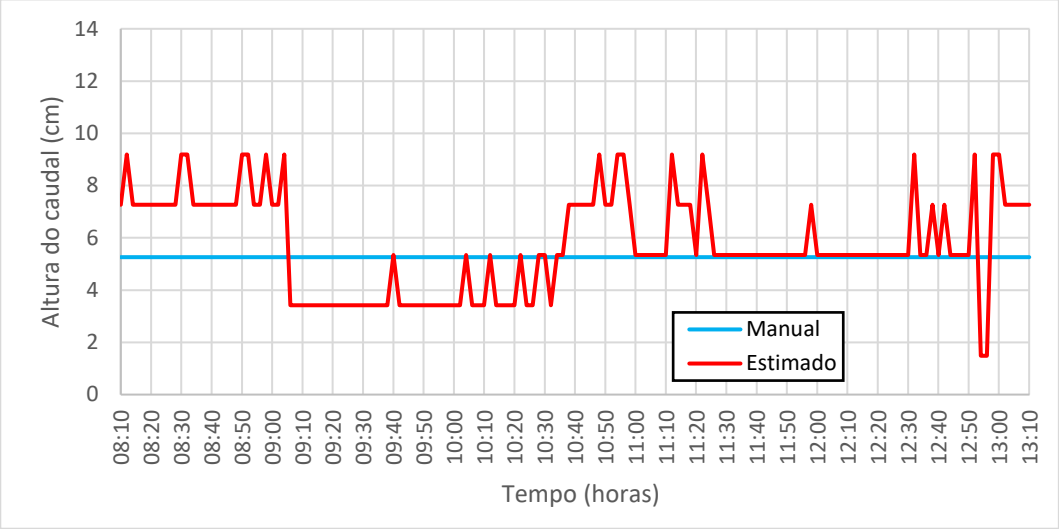
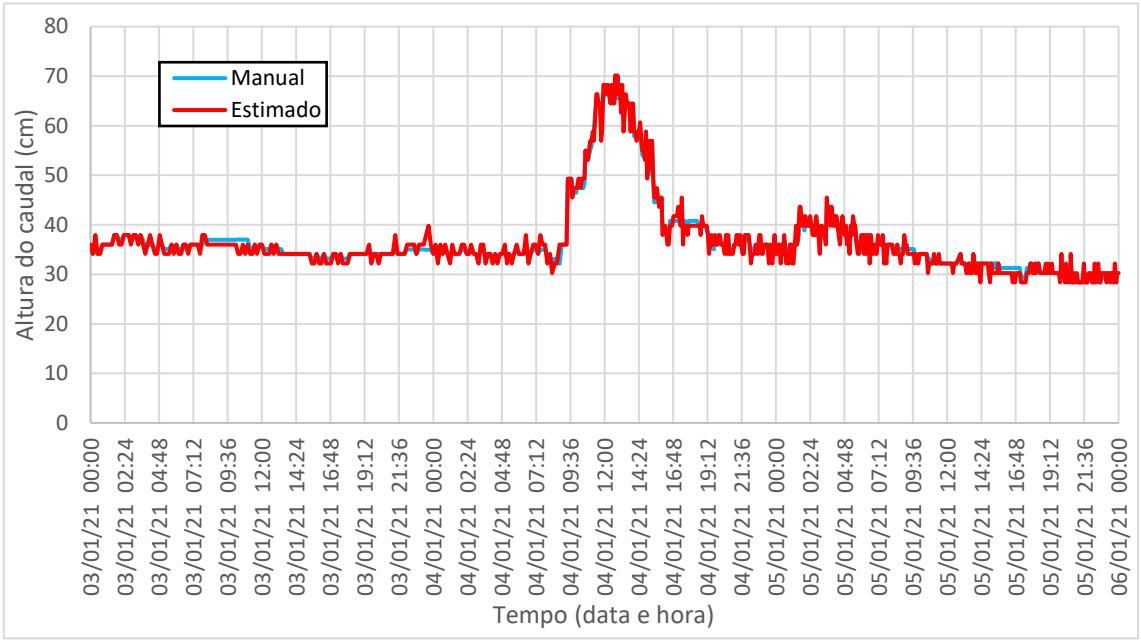


Figura 4.5 – Estimativa do nível de água no dia 10 de junho de 2021 durante o "período de sombra".

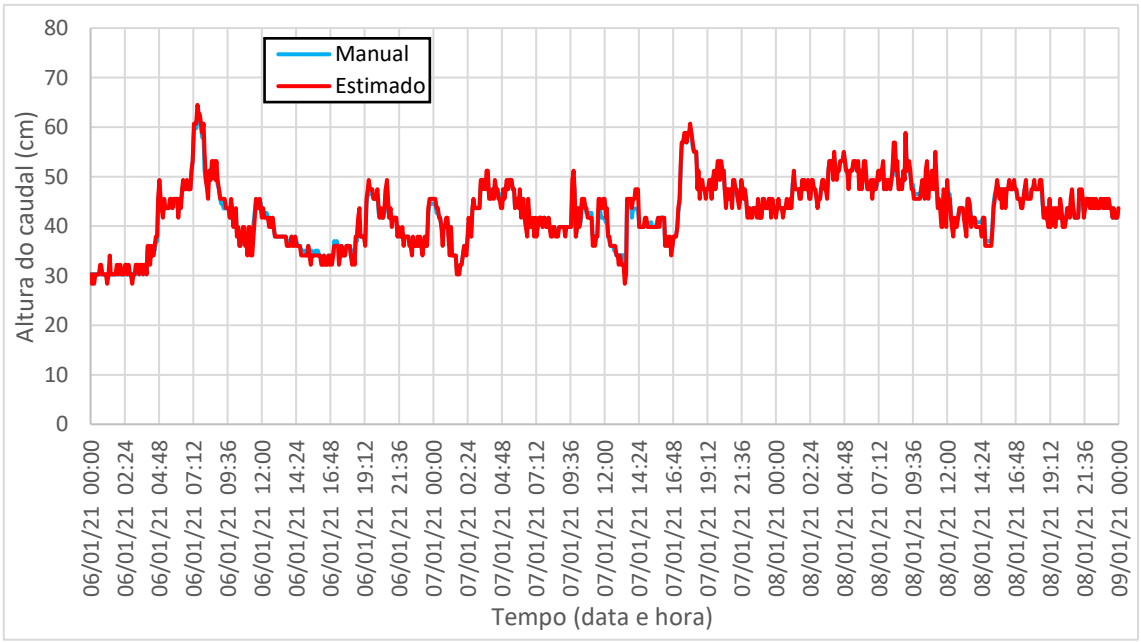
Em relação ao gráfico da figura 4.5, o nível de água ao longo de todo o período de estudo foi de 5,3 cm. Das 8:10h até as 9:05h o erro cometido na deteção do nível foi maior, porque o nível de água ficou muito escuro devido à sombra existente no local de observação. Com o movimento do sol a qualidade da imagem foi-se alterando na zona da linha de água resultando em variações no nível de água detetado. Das 12:40h até as 13:10h ocorreram oscilações da deteção do nível devido à situação do segundo "período de sombra", como foi referido no ponto 3.5. Nesta situação tem-se uma exatidão de 1,5 cm.

4.5. Grande período de observação

Por fim procedeu-se à análise de resultados para um grande período de observação. Na figura 4.6 são apresentados os resultados obtidos entre o dia 3 e 8 de janeiro de 2021.



a)



b)

Figura 4.6 – Estimativa do nível de água para um período mais longo de observação: a) 3 a 5 de janeiro de 2021; b) 6 a 8 de janeiro de 2021.

Relativamente ao gráfico da figura 4.6 a), o nível de água desde às 00:00h do dia 3 de janeiro até às 08:10h do dia 4 de janeiro mantém-se praticamente constante, apresentando nesse período uma exatidão de 0,7 cm durante o dia e 2,7 cm durante a

noite. Durante o dia 4 de janeiro entre as 09:20h até às 12:55h ocorreu um período de grande precipitação levando a um grande aumento do nível de água, sendo que de seguida houve a diminuição do caudal até as 18:25h. Durante o dia 4 de janeiro obteve-se uma exatidão de 1,2 cm com uma ondulação média de 2,3 cm. Durante a noite do dia 4 de janeiro e a madrugada de 5 de janeiro houve um ligeiro aumento de caudal desde a 01:30h até as 05:50h devido à precipitação à montante da ribeira. Neste período noturno obteve-se uma exatidão de 2,9 cm. Durante o dia 5 de janeiro até à 00:00 do dia 6 de janeiro houve uma estabilização do nível de água na ribeira, obtendo-se uma exatidão de 0,7 cm durante o dia e 2,7 cm durante a noite.

Relativamente ao gráfico da figura 4.6 b), no dia 6 de janeiro ocorreu um grande aumento do nível de água desde as 04:00h até às 05:15h, tendo-se estabilizado o nível e voltando a aumentar desde as 07:00h até as 07:30h com a existência de precipitação na zona urbana. Depois dessa hora existe uma diminuição do nível até o anoitecer. Neste período obteve-se uma exatidão de 2,9 cm durante a noite e 0,7 cm durante o dia, com uma ondulação média de 1,1 cm. Durante a noite de 6 de janeiro e a madrugada de 7 de janeiro ocorreram aumentos ligeiros do nível de água devido à precipitação nas zonas montanhosas, obtendo-se uma exatidão de 2,7 cm durante esse período. Estes aumentos do nível de água continuam também durante o dia 7 de janeiro devido à precipitação à montante com uma exatidão de 0,8 cm. Durante a noite do dia 7 de janeiro e a madrugada do dia 8 de janeiro ocorre um ligeiro aumento do caudal com oscilações, mas apesar disso obteve-se uma exatidão de 2,6 cm. Durante o dia 8 de janeiro ocorre uma diminuição do caudal e às 15:05h ocorreu novamente um aumento do nível de água, tendo estabilizado o nível pelo anoitecer. Neste período a exatidão foi de 1 cm. O facto de a exatidão ser um pouco baixa em relação a outros casos já apresentados deve-se à parede de betão estar húmida pela precipitação ocorrida e porque a água estava mais clara neste período (sem inertes), o que faz com que a linha de água seja menos nítida. Apesar desta situação, o erro cometido na deteção não é tão elevado. Durante a noite ocorreu uma descida do caudal e posterior estabilização do nível, obtendo uma exatidão de 2,8 cm.

4.6. Sumarização dos resultados

De forma a sumarizar os resultados, na tabela 4.1 são apresentados os valores da exatidão obtida com base nas diferentes situações ocorridas ao longo do tempo.

Tabela 4.1 - Exatidão em função da situação ocorrida ao longo do tempo.

Período	Situação	Exatidão (cm)
Diurno	Dia normal	0,7
	Ondulação da água de 2 cm e detritos na água	0,8
	Dia ensolarado sem alteração do nível de água	1,5
	Precipitação urbana	1,3
	Grande precipitação urbana	1,6
	Precipitação nas zonas montanhosas	0,8
	Humidade na parede da ribeira	1,0
	Média para o período de observação	**1,1**
Noturno	Noite normal	2,7
	Precipitação urbana	2,7
	Grande precipitação urbana	3,3
	Precipitação nas zonas montanhosas	2,8
	Média para o período de observação	**2,9**

De acordo com os resultados apresentados na tabela 4.1, a exatidão média do algoritmo desenvolvido para a obtenção do nível de água foi de 1,1 cm para imagens durante o dia e 2,9 cm para imagens durante a noite.

De forma a analisar os resultados obtidos pelo algoritmo implementado é apresentado na tabela 4.2 um resumo dos resultados incluindo os resultados apresentados no estado da arte (ponto 2.3).

Tabela 4.2 - Resumo dos resultados do estado da arte e do algoritmo desenvolvido.

Sistemas baseados em imagens	Tema	Exatidão (cm)
Sistemas existentes	Medição do nível de água com dois sistemas baseados em imagem	1,18 e 0,87
	Deteção automática do nível de água que considera o movimento da câmara	1
	Medições do nível de água efetuada por *drones*	Entre 1,2 e 5,1
	Medição do nível de água a montante de um riacho	3
Sistema desenvolvido	Sistema automático de medição do nível de água em ribeiras do Funchal através de videovigilância	Dia – 1,1 Noite – 2,9

Comparando os resultados obtidos com os métodos apresentados no estado da arte (tabela 4.2), verifica-se que a exatidão obtida é próxima dos outros métodos (cerca de 1 cm). No entanto, vários desses métodos utilizam um medidor de água na zona de medição para gerar um forte contraste entre a superfície da água e a parede ou para melhorar a deteção do nível. Além disso, o medidor também serve como referência para as medições de forma a comparar os resultados. Na técnica desenvolvida neste

trabalho, o uso medidor de água tornou-se impossível devido à estrutura da ribeira, visto que para isso seria necessário fixar na parede de modo que o mesmo não se movesse devido às subidas de nível. Também o objetivo do trabalho seria não introduzir nenhum elemento na ribeira para que não ficar dependente do mesmo, caso se queira utilizar o sistema noutra ribeira. No entanto, o principal motivo em não utilizar um medidor de água é que este não resolve o problema da existência de detritos na água, da ondulação e da fraca qualidade das imagens.

Durante o período noturno a iluminação pública, mais concretamente os postes de iluminação, ajudou na deteção do nível de água, reduzindo os custos do sistema, visto não ter sido instalado iluminação artificial. Apenas utilizando a iluminação pública o erro de deteção do nível foi maior quando comparado com a precisão obtida durante o dia. No entanto, é possível melhorar a exatidão durante a noite com iluminação artificial focada na parede onde está localizada a região de interesse utilizada, permitindo reduzir o erro produzido pela deteção indireta do nível, ou seja, a deteção do nível através da sombra ocorrida na água.

Certos métodos também removem imagens obtidas em condições climatéricas adversas, como por exemplo neve, chuva forte, dia ensolarado e até durante o período noturno. No algoritmo desenvolvido todas as imagens obtidas foram utilizadas para obtenção do nível de água, tendo tratado cada uma delas de uma forma adequada, ou seja, com um processo de deteção adequado.

Outra situação que certos métodos apresentam problemas tem a ver com o facto da ondulação da água e a existência de detritos na água à volta do medidor de água provocarem grandes erros na deteção da linha de água. Na técnica desenvolvida são consideradas várias zonas na região de interesse para resolver estes problemas. Neste trabalho foram utilizadas 10 posições de deteção da linha de água para estimar a mesma através de média espacial. Este processo permitiu atenuar e até eliminar muitos dos problemas que outros métodos não conseguiram resolver.

5. Conclusão

Neste capítulo são apresentadas as conclusões finais após a realização deste projeto, bem como sugestões de trabalhos a desenvolver no futuro.

5.1. Conclusões finais

Neste trabalho foi desenvolvido um sistema de baixo custo que permite efetuar a deteção do nível de caudal nas ribeiras do Funchal. Estas ribeiras são constituídas por paredes de pedra ou de betão que permitem conter a água que as percorre. Na maior parte do tempo, o nível de água nas ribeiras tende a ser baixo, mas este pode-se alterar devido a condições climatéricas adversas, aumentando muito rapidamente o nível de água. Durante estes eventos, a qualidade da água altera-se com a presença de detritos na água e o rápido aumento de nível provoca ondulação da água. Devido à precipitação urbana, as imagens adquiridas ficam com vestígios de chuva o que influencia a deteção do nível de água.

No sistema implementado para aquisição de imagens foi utilizado um Raspberry Pi 3 modelo B e uma câmara Pi NoIR V1 para trabalhar durante o dia e durante a noite. Este sistema foi colocado no teto de uma varanda de um prédio em frente à ribeira de Santa Luzia, Funchal. A instalação no prédio permitiu reduzir os custos de instalação e também o impacto ambiental, sendo que a câmara, ao ficar sob a varanda permitiu evitar/reduzir a saturação das imagens devido à luz solar, o efeito do vento e da chuva. Este sistema já estava em funcionamento no início deste trabalho, de forma a recolher informação para testar o algoritmo desenvolvido.

Depois da instalação do sistema foi necessário calibrar a câmara de forma que a imagem obtida tenha a informação necessária para detetar o nível de água. Neste ponto teve-se em consideração eliminar uma parte do topo da imagem por questões de privacidade, visto que a mesma captava uma estrada rodoviária. De seguida foi desenvolvido um sistema de referência que permitiu relacionar uma distância no plano da imagem com uma distância no plano do objeto, ou seja, relacionar os pixéis da imagem com uma distância real da parede da ribeira. O sistema desenvolvido permite, caso o sistema seja implementado noutra ribeira, efetuar a relação imagem-objeto mais facilmente.

Posteriormente começou-se no desenvolvimento do algoritmo de deteção do nível de água. O processo inicia com a aquisição de T imagens e com a obtenção da média das mesmas. Com essa imagem média foi efetuada a conversão da mesma para escala de cinza e aplicou-se a equalização de histograma para equilibrar a imagem em termos de contraste. Um filtro gaussiano foi aplicado para reduzir o ruído e de seguida foi aplicada binarização para realçar uma das bordas superiores da ribeira para efetuar a compensação da imagem em termos de rotação. Depois foi efetuada uma deteção de padrões para efetuar uma compensação da imagem em termos de translação. Caso ocorra um erro na deteção da borda ou nos padrões foi utilizada a região de interesse obtida através dos parâmetros da última imagem em que foi efetuada uma compensação bem-sucedida.

Após a obtenção da região de interesse passou-se para a segunda fase do algoritmo que foi a deteção da altura do nível de água. Primeiramente foi definido um passo no eixo x para obter os vários pontos de deteção que permitiu criar uma média

discreta dos gráficos de média móvel de escala de cinza e gradiente. Em seguida foi determinado o valor máximo do gradiente. Consoante a situação e especificidade da imagem na região de interesse foi efetuado um procedimento de forma a obter a posição da linha de água e posteriormente a altura do caudal da ribeira. Depois disto o valor obtido na imagem foi convertido para centímetros.

Após a implementação do algoritmo, este foi testado usando imagens em diferentes condições climatéricas, de forma a estabelecer uma exatidão da deteção do nível de água. A exatidão teve uma variação entre 0,7 cm e 1,6 cm durante o dia e entre 2,7 cm e 3,3 cm durante a noite. A exatidão média obtida através das diferentes condições climatéricas foi de 1,1 cm durante o dia e 2,9 cm durante a noite. Apesar dos resultados de exatidão obtidos serem mais baixos em comparação com os fornecidos por outros métodos, vários dos métodos utilizam um medidor de água que permite melhorar a exatidão, mas traz diversas desvantagens, uma delas em relação aos detritos flutuam em torno do mesmo, causando erros na deteção. No método desenvolvido não se utilizou esse elemento, apesar a exatidão ser mais baixa, mas permitiu obter o nível de água em situações em que outros métodos não conseguem. Outros métodos descartam as imagens em determinadas condições climatéricas, algo que no trabalho efetuado não se realizou, visto que foi possível obter o nível de água em muitas das situações climatéricas adversas.

Por fim, é de referir que deste trabalho resultou um artigo publicado em revista, estando este no Anexo B.

5.2. Trabalhos futuros

Como recomendações de trabalhos futuros sugere-se a necessidade de melhorar a deteção do nível de água nos casos em que uma grande área da linha de água permanece obstruída pela vegetação, principalmente quando é possível por meio de inspeção detetar certos pontos da linha de água. Uma sugestão consiste em utilizar técnicas de inteligência artificial baseadas em aprendizagem automática.

Também se propõe avaliar a introdução de iluminação artificial durante o período noturno para melhorar o desempenho de deteção durante o período noturno.

Referências

[1] J. Yu e H. Hahn, "Remote Detection and Monitoring of a Water Level Using Narrow Band Channel," *Journal of Information Science and Engineering ,* pp. 71-82, 2010.

[2] D. E. Alsdorf, E. Rodríguez e D. P. Lettenmaier, "Measuring surface water from space," *Reviews of Geophysics,* pp. 1-24, 2007.

[3] P. F. A. da Silva e C. A. da Meneses, Elucidário Madeirense.

[4] R. Quintal, *Aluviões da Madeira. Séculos XIX e XX,* pp. 31-48, 1999.

[5] R. P. de Oliveira, A. B. de Almeida, J. Sousa, M. J. Pereira, M. M. Portela, M. A. Coutinho, R. Ferreira e S. Lopes, "A Avaliação do Risco de Aluviões na Ilha da Madeira," pp. 1-20, 2011.

[6] T. d. C. -. S. R. d. Madeira, "Acompanhamento das medidas de apoio à reconstrução da RAM na sequência da aluvião de 20/02/2012 - Ano de 2010 -," Funchal, 2011.

[7] Região Autónoma da Madeira, "Resolução n.º 805/2017," *JORNAL OFICIAL,* nº 181952/02, p. 50, 2017.

[8] Stevens, "Smart SDI-12 / RS-485 Pressure, Temperature, and Digital Crest Gauge Sensor," [Online]. Available: https://stevenswater.com/products/smart-pt/. [Acedido em 10 setembro 2021].

[9] California Polytechnic State University - Irrigation Training and Research Center, "Water Level Sensor and Datalogger Testing and Demonstration," San Luis Obispo, 1998.

[10] MASSA, "PulStar Series Ultrasonic Sensors," [Online]. Available: https://www.massa.com/industrial/ultrasonic-sensors/pulstar/. [Acedido em 20 setembro 2021].

[11] C. J. Gleason, L. C. Smith, D. C. Finnegan, A. L. LeWinter, L. H. Pitcher e V. W. Chu, "Technical Note: Semi-automated effective width extraction from time-lapse RGB imagery of a remote, braided Greenlandic river," *Hydrology and Earth System Sciences ,* pp. 2963-2969, 2015.

[12] R. Tsubaki, I. Fujita e S. Tsutsumi, "Measurement of the flood discharge of a small-sized rizer using an existing digital video recording system," *Journal of Hyfro-environment Research 5,* pp. 313-321, 2011.

[13] H.-C. Yang, C.-Y. Wang e J.-X. Yang, "Applying image recording and identification for measuring water stages to prevent flood hazards," *Natural Hazards,* pp. 737-754, 2014.

[14] T. Hies, P. S. Babu, Y. Wang, R. Duester, H. S. Eikaas e T. K. Meng, "Enhanced Water-Level Detection By Image Processing," *10th International Conference on Hydroinformatics, Hamburg, Germany,* pp. 1-8, 14-18 julho 2012.

[15] Z. Zhang, Y. Zhou, H. Liu, L. Zhang e H. Wang, “Visual Measurement of Water Level under Complex Illumination Conditions,” *Sensors,* pp. 1-22, 2019.

[16] Y.-T. Lin, Y.-C. Lin e J.-Y. Han, “Automatic water-level detection using single-camera images with varied poses,” *Measurement,* pp. 167-174, 2018.

[17] E. Ridolfi e P. Manciola, “Water Level Measurements from Drones: A Pilot Case Study at a Dam Site,” *Water,* pp. 1-10, 2018.

[18] P. Leduc, P. Ashmore e D. Sjogren, “Technical note: Stage and water width measurement of a mountain stream using a simple time-lapse camera,” *Hydrology and Earth System Sciences,* pp. 1-11, 2018.

[19] A. Criminisi, “Single-View Metrology: Algorithms and Applications,” *Springer-Verlag Berlin Heidelberg,* pp. 224-236, 2002.

[20] OpenCV, “About,” [Online]. Available: https://opencv.org/about/. [Acedido em 29 outubro 2022].

[21] PyPi, “opencv-python 4.6.0.66,” [Online]. Available: https://pypi.org/project/opencv-python/. [Acedido em 29 outubro 2022].

[22] R. P. K. Reddy, D. C. N. e I. R. Reddy, “Canny Scale Edge Detection,” *International Journal of Engineering Trends and Technology,* pp. 1-4, 2015.

[23] A. J. D. M. Dixon K, N. J. e S. G. E. , “Performance Study of Edge Detection Operators,” *International Conference on Embedded Systems ,* pp. 7-11, 2014.

[24] A. S. Ahmed, “Comparative Study Among Sobel, Prewitt and Canny Edge Detection Operators used in Image Processing,” *Journal of Theoretical and Applied Information Technology,* pp. 6517-6525, 2018.

[25] D. A. D. A. e R. T. , “Analytical Comparison between Sobel and Prewitt Edge Detection Techniques,” *International Journal of Scientific & Engineering Research,* pp. 1482-1485, 2016.

[26] R. O. Duda e P. E. Hart, “Use of the Hough Transformation to Detect Lines and Curves in Pictures,” pp. 1-15, 1971.

[27] A. Coste, “CS6640: Image Processing, Project 4, Hough Transform,” 2012.

[28] G. J. Bergues, C. Schürrer e N. Brambilla, “Straight Line Detection Through Sub-pixel Hough Transform,” *CompCom 2019: Intelligent Computing,* pp. 1129-1137, 2019.

[29] OpenCV, “Hough Line Transform,” [Online]. Available: https://opencv24-python-tutorials.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_houghlines/py_houghlines.html. [Acedido em 28 junho 2021].

[30] OpenCV, “Hough Line Transform,” [Online]. Available: https://docs.opencv.org/3.4/d9/db0/tutorial_hough_lines.html. [Acedido em 28 junho 2021].

[31] Raspberry Pi, [Online]. Available: https://www.raspberrypi.org/. [Acedido em 23 07 2022].

[32] Pololu, “Pololu 5V Step-Up/Step-Down Voltage Regulator S18V20F5,” [Online]. Available: https://www.pololu.com/product/2574. [Acedido em 24 agosto 2022].

[33] Raspberry Pi, “Camera,” [Online]. Available: https://www.raspberrypi.org/documentation/accessories/camera.html. [Acedido em 24 agosto 2021].

[34] python, “python,” [Online]. Available: https://www.python.org/. [Acedido em 27 julho 2021].

[35] Visual Studio Code, “Visual Studio Code,” [Online]. Available: https://code.visualstudio.com/. [Acedido em 27 julho 2021].

[36] OpenCV, “OpenCV - Python Tutorials,” [Online]. Available: https://docs.opencv.org/4.5.2/d6/d00/tutorial_py_root.html. [Acedido em 11 agosto 2021].

[37] S. Raj, S. Raj e S. Kumar, “An Improved Histogram Equalization Technique for Image Contrast Enhancement,” pp. 1-7, 2015.

[38] OpenCV, “Histograms - 2: Histogram Equalization,” [Online]. Available: https://docs.opencv.org/master/d5/daf/tutorial_py_histogram_equalization.html. [Acedido em 25 junho 2021].

[39] OpenCV, “Histograms - Image Processing,” [Online]. Available: https://docs.opencv.org/master/d6/dc7/group__imgproc__hist.html#ga7e54091f0c937d49bf84152a16f76d6e. [Acedido em 25 junho 2021].

[40] D. P. Sharma, “Intensity Transformation using Contrast Limited Adaptive Histogram Equalization,” *International Journal of Engineering Research,* pp. 282-285, 2013.

[41] S. Misra, H. Li e J. He, “Gaussian blur (one feature),” em *Machine Learning for Subsurface Characterization*, Gulf Professional Publishing, 2020, p. 412.

[42] OpenCV, “Image Filtering - Image Processing,” [Online]. Available: https://docs.opencv.org/4.5.2/d4/d86/group__imgproc__filter.html#gaabe8c836e97159a9193fb0b11ac52cf1. [Acedido em 25 junho 2021].

[43] P. G. Bathinda e N. K. Garg, “Binarization Techniques used for Grey Scale Images,” *International Journal of Computer Applications,* pp. 8-11, 2013.

[44] OpenCV, “Miscellaneous Image Transformations - Image Processing,” [Online]. Available: https://docs.opencv.org/3.4/d7/d1b/group__imgproc__misc.html#ggaa9e58d2860d4afa658ef70a9b1115576a147222a96556ebc1d948b372bcd7ac59. [Acedido em 23 junho 2021].

[45] OpenCV, “Feature Detection - Image Processing,” [Online]. Available: https://docs.opencv.org/3.4/dd/d1a/group__imgproc__feature.html#ga04723e007ed888ddf11d9ba04e2232de. [Acedido em 25 junho 2021].

[46] B. Batra, S. Singh, J. Sharma e S. M. Arora, “Computacional analysis of edge detection operators,” *International Journal of Applied Research ,* pp. 257-262, 2016.

[47] Scipy, “scipy.ndimage.rotate,” [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.ndimage.rotate.html#scipy-ndimage-rotate. [Acedido em 4 julho 2021].

[48] D. Stoitchkov, “Analysis of Methods for Automated Symbol Recognition in Technical Drawings,” 2018.

[49] OpenCV, “Object Detection - Image Processing,” [Online]. Available: https://docs.opencv.org/4.5.2/df/dfb/group__imgproc__object.html#ga586ebfb0a7fb604b35a23d85391329be. [Acedido em 29 junho 2021].

[50] OpenCV, “Operations on arrays - Core functionality,” [Online]. Available: https://docs.opencv.org/4.5.2/d2/de8/group__core__array.html#ga8873b86a29c5af51cafdcee82f8150a7. [Acedido em 29 junho 2021].

[51] paint.net, “paint.net,” [Online]. Available: https://www.getpaint.net/. [Acedido em 1 agosto 2021].

[52] Scipy, “scipy.signal.find_peaks,” [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.find_peaks.html. [Acedido em 17 agosto 2021].

Anexo A – Algoritmo desenvolvido

```
import os
import numpy as np
from cv2 import ROTATE_180, cv2 as cv
import math
from scipy import ndimage
from matplotlib import pyplot as plt
from scipy.signal import find_peaks
from datetime import datetime


def clean_pastes(directory): # Função que permite limpar ficheiros de um diretório
    for the_file in os.listdir(directory):
        file_path = os.path.join(directory, the_file)
        try:
            if os.path.isfile(file_path):
                os.unlink(file_path)
        except Exception as e:
            print(e)


def analysis_empty_data(dir): # Função que permite eliminar as imagens que não tem nenhum conteúdo (estão vazias)

    for root, dirs, files in os.walk(dir):
        for f in files:

            path = os.path.join(root, f)    # Permite juntar o caminho do diretório com o nome do ficheiro (neste caso, o nome da imagem)

            image = cv.imread(path)         # Leitura da imagem

            # Verificar se a imagem tem conteúdo ou não, se não tiver conteúdo e imagem é eliminada
            if image is None:
                os.remove(path)


def date_hour (name): # Função que permite obter a data e a hora da imagem capturada

    year = str(name[0:4])       # Ano
    month = str(name[5:7])      # Mês
    day = str(name[8:10])       # Dia
    hour = str(name[11:13])     # Hora
    minute = str(name[14:16])   # Minuto
    second = str(name[17:19])   # Segundo

    date_h = "%s-%s-%s_%s-%s-%s" %(year, month, day, hour, minute, second)  # Data e hora
    hour_str = "%s:%s" %(hour, minute)                                      # Hora

    return(date_h, hour_str)


def time_average(ini_dir, dir1, dir2, dir3, num_images): # Função que efetuar a média temporal de um grupo de imagens

    name_images = []    # Array para colocar o nome das imagens usadas para efetuar a média

```

```
    for root, dirs, files in os.walk(dir1):
        for f in files:

            path = os.path.join(root, f)    # Permite juntar o caminho do diretório com o nome do ficheiro (neste caso, o nome da imagem)

            # Executa a função date_hour para obter a data e hora da imagem
            dhour, hour_min = date_hour(f)
            dhour_str = str(dhour)

            image = cv.imread(path) # Leitura da imagem

            name_images = np.append(name_images, dhour_str) # Adicionar o nome da imagem (data e hora) no array

            os.chdir(dir2)
            cv.imwrite('%s.jpg' %dhour_str, image)  # Guarda a imagem no diretório dir2

            if(len(name_images) == num_images): # Caso o número de imagens no array seja igual a variável num_images

                files_average = list([os.path.join(dir2, file) for file in os.listdir(dir2)])   # Obter todos os ficheiros de imagem do diretório dir2

                # Calcular a média das imagens
                # Primeiro é definido o array usado para efetuar a média como float para forçar a conversão dos valores de 8 bits das imagens, senão ocorreria overflow dos valores
                average = cv.imread(files_average[0]).astype(float)

                for file in files_average[1:]:  # Ciclo que percorre todas as imagens que serão usadas para efetuar a média

                    image = cv.imread(file)     # Leitura da imagem
                    average = average + image   # É efetuado a soma de duas imagens, pixel por pixel / canal por canal

                average = average/len(files)    # Média do número de imagens escolhido pelo utilizador

                # Normalizar a imagem, para que as intensidades dos vários pixéis estejam entre 0 e 255
                # Irá tornar a imagem mais clara/nítida sem perder informação
                output = cv.normalize(average, None, 0, 255, cv.NORM_MINMAX)

                os.chdir(dir3)
                cv.imwrite('%s.jpg' %name_images[len(name_images)-1], output)   # Guardar a imagem média no diretório dir3

                os.chdir(dir2)
                os.remove("%s.jpg" %name_images[0]) # Remover a primeira imagem que foi escrita/guardada no diretório dir2

                name_images = np.delete(name_images, 0) # Remover o nome da primeira imagem escrita/guardada do array

    os.chdir(ini_dir)   # Retorna para o diretório inicial


```

```python
def bright(image, bright_value): # Função que permite aumentar o brilho de uma imagem

    blue, green, red = cv.split(image)  # Dividir a imagem nos três canais de cores (azul, verde e vermelho)

    palette = []    # Array que serve para colocar os valores corresppondes, para cada valor de pixel, ao aumento de brilho pretendido

    for i in range(256):    # Ciclo para obter os valores correspondentes ao aumento de brilho pretendido

        temp = i * bright_value # Multiplicação da intensidade do pixel com o valor de brilho pretendido

        if(temp < 0):
            temp = 0
        elif(temp > 255):
            temp = 255

        palette.append(temp)    # Guardar o novo valor de intensidade do pixel no array

    for n in range(np.size(red[:,1])):  # Ciclo que permite, para cada canal de cor, efetuar o aumento de brilho correspondente a cada valor de pixel (linha e coluna, respetivamente)
        for m in range(np.size(red[1,:])):

            red[n,m] = palette[red[n,m]]
            green[n,m] = palette[green[n,m]]
            blue[n,m] = palette[blue[n,m]]

    image = cv.merge([blue, green, red])    # Junção dos três canais de cor para formar a imagem original, mas com o aumento de brilho pretendido

    return(image)   # Retorna a imagem com aumento de brilho


def image_rotation(original_image, image, width, height, points, offset_initial_x_zone, offset_final_x_zone, distance_y_zone, discarded_px):   # Função que permite calibrar a imagem em termos de rotação

    image_gaussian = cv.GaussianBlur(image, (3,3), 0)   # Aplicação do filtro gaussiano (permite a remoção de ruído da imagem)

    binary_image = cv.adaptiveThreshold(image_gaussian, 255, cv.ADAPTIVE_THRESH_GAUSSIAN_C, cv.THRESH_BINARY, 33, 2)    # Efetua a binarização adaptativa

    # Dimensões (largura - w_zone_edge e altura - h_zone_edge) da zona da imagem aonde será aplicado a deteção de bordas
    h_zone_edge = distance_y_zone
    initial_w = offset_initial_x_zone
    final_w = width - offset_final_x_zone
    w_zone_edge = final_w - initial_w

    # Recorte da imagem na zona de interesse para a deteção de bordas
    # Largura: desde 250 até width-100 (neste caso 1280-100=1180 pixéis)
    # Altura: desde 0 até h_zone_edge (neste caso 144 pixéisw)
    cropped_image = binary_image[0:h_zone_edge, initial_w:final_w]
```

```
145
146     edges_canny = cv.Canny(cropped_image, 100, 200) # Aplicação do método Canny para
    deteção de bordas
147
148     # Aplicação da transformada de Hough que permite a deteção de linhas numa imagem
149     lines_image = cv.HoughLines(edges_canny, 1, np.pi/360, points)  # Obtenção das
    várias linhas detetadas (neste caso, os valores de rho e theta) na imagem resultante
    do método de deteção de bordas
150
151     if(lines_image is None):    # Caso não seja detetado nenhuma linha na imagem
152
153         lines_detected = 0  # Variável que indica o número de linhas detetadas
154         error_edge = 1      # Variável que indica se detetou ou não alguma linha (0
    - detetou linhas, 1 - não detetou linhas)
155     else:
156
157         lines_detected = len(lines_image)
158         error_edge = 0
159
160     # Variáveis que irão permitir guardar o valor final de rho e theta
    (correspondente a uma das bordas da ribeira)
161     final_rho = 0
162     final_theta = 0
163
164     if(error_edge == 0):    # Caso tenha sido detetado linhas
165
166         # Arrays que irão permitir guardar os valores de rho e theta de cada uma das
    linhas detetadas
167         rho = []
168         theta = []
169
170         for i in range(discarded_px, h_zone_edge): # Ciclo que permite guardar os
    valores de rho (e consequentemente de theta) por ordem crescente (Nota: não será
    considerado bordas detetadas abaixo dos primeiros 10 pixéis (no eixo vertical))
171             for j in range(lines_detected):
172
173                 temp_rho = lines_image[j-1][0][0]
174
175                 if(temp_rho == i):
176                     rho.append(lines_image[j-1][0][0])
177                     theta.append(lines_image[j-1][0][1])
178
179         lines_detected = len(rho)   # Indica o número de linhas obtidas desde o
    pixel inicial (initial_pixel) até h_zone_edge
180
181         if(lines_detected > 1): # Caso tenha sido detetado mais do que uma linha
182
183             temp_h_middle = []  # Permite guardar o valor no eixo vertical (de cima
    para baixo, a meio da zona da imagem usada para deteção de bordas), em pixéis,
    correspondente a cada linha detetada
184
185             for i in range(lines_detected):
186
187                 alpha = theta[i] - (np.pi/2)    # Obtenção do ângulo (em radianos)
    entre rho e o eixo vertical
188
189                 pixels_add_middle = round(np.sin(alpha)*(w_zone_edge/2))    # Número
    de pixéis a adicionar (ou a subtrair) ao valor obtido no eixo vertical na zona média
    da imagem
190
```

```
                h_zone_middle = round(rho[i]/np.cos(alpha) + pixels_add_middle) # Valor final obtido no eixo vertical na zona média da imagem

                temp_h_middle.append(h_zone_middle) # Colocação do valor anterior no array

            min_h_zone_middle = np.min(temp_h_middle)   # Obtenção do valor mínimo de todas as linhas detetadas

            pos_min_h_zone_middle = np.where(temp_h_middle == min_h_zone_middle)[0] # Posição (ou posições) aonde se encontra o valor mínimo

            if(len(pos_min_h_zone_middle) == 1):    # Caso apenas se tenha detetado um valor mínimo

                final_rho = rho[pos_min_h_zone_middle[0]]       # Valor de rho da linha com o valor mínimo
                final_theta = theta[pos_min_h_zone_middle[0]]   # Valor de theta dad linha com o valor mínimo

            else:   # Caso tenha sido detetado mais do que um valor mínimo igual

                # Arrays que permitem guardar os valores de rho e theta para posteriormente efetuar a média
                mean_rho = []
                mean_theta = []

                for i in range(len(pos_min_h_zone_middle)): # Colocação dos valores mínimos detetados nos arrays correspondentes

                    mean_rho.append(rho[pos_min_h_zone_middle[i]])
                    mean_theta.append(theta[pos_min_h_zone_middle[i]])

                final_rho = np.mean(mean_rho)       # Média do valor de rho
                final_theta = np.mean(mean_theta)   # Média do valor de theta

        elif(lines_detected == 1):  # Caso apenas tenha sido detetado uma linha

            final_rho = rho[0]
            final_theta = theta[0]

        else:   # Caso depois de descartar os primeiros pixeis (dado por initial_pixel) não exista nenhuma linha detetada

            error_edge = 1

        rotation_angle = math.degrees(final_theta) - 90 # Ângulo de rotação usado para rodar a imagem

        image = ndimage.rotate(image, rotation_angle, reshape = True)   # Imagem rodada

        original_image = ndimage.rotate(original_image, rotation_angle, reshape = True) # Imagem original rodada

        rotated_height, rotated_width = image.shape

    final_alpha = final_theta - (np.pi/2)   # Ângulo entre o rho final e o eixo vertical
```

```
    new_rho = 0                                 # Novo valor de rho (correspondente ao início da imagem total inicial (1280*720))

    if(final_rho != 0): # Caso tenha sido detetado uma das bordas da ribeira

        y_rho = round(final_rho/np.cos(abs(final_alpha)))   # Valor y no eixo vertical (ver fig. 3.20 do relatório)

        offset_y_initial_rho = round(initial_w*np.tan(abs(final_alpha))) # Offset (positivo ou negativo) existente no início da imagem (ver fig. 3.20 do relatório)

        if(final_alpha < 0):    # Caso o valor de alpha seja negativo (fig. 3.20 a))

            y_initial_rho = y_rho + offset_y_initial_rho    # Valor y no eixo vertical (considerando o offset)

            new_rho = round(y_initial_rho*np.cos(abs(final_alpha))) # Novo valor de rho no início da imagem

        else:   # Caso o valor de alpha seja positivo (fig. 3.20 b))

            y_initial_rho = y_rho - offset_y_initial_rho    # Valor y no eixo vertical (considerando o offset)

            new_rho = round(y_initial_rho*np.cos(abs(final_alpha))) # Novo valor de rho no início da imagem (sem o offset devido à rotação da imagem)

            offset_rotation = round(width*np.tan(abs(final_alpha))) # Offset devido à rotação da imagem

            new_rho = new_rho + offset_rotation # Novo valor de rho no início da imagem (com o offset devido à rotação da imagem)

    return(image, original_image, new_rho, final_theta, error_edge)


def translation_compensation(original_image, image, first_temp, second_temp, type):
 # Função que permite efetuar a deteção dos padrões para compensação da translação

    # Aplicação da correspondência de padrão (template matching) na imagem (PADRÃO PRINCIPAL)
    res_first_temp = cv.matchTemplate(image, first_temp, cv.TM_CCOEFF_NORMED)   # Aplicação da deteção do padrão principal
    f_min_val, f_max_val, f_min_loc, f_top_left = cv.minMaxLoc(res_first_temp)  # Permite obter as coordenadas (f_top_left) do ponto superior esquerdo do padrão principal

    # Coordenadas do ponto superior esquerdo do padrão principal
    x_ftemp = f_top_left[0]
    y_ftemp = f_top_left[1]

    if(type == 0):  # A verificação do padrão secundário apenas é feito durante o dia

        # Aplicação da correspondência de padrão (template matching) na imagem (PADRÃO SECUNDÁRIO)
        res_second_temp = cv.matchTemplate(image, second_temp, cv.TM_CCOEFF_NORMED) # Aplicação da deteção do padrão secundário
        s_min_val, s_max_val, s_min_loc, s_top_left = cv.minMaxLoc(res_second_temp) # Permite obter as coordenadas (s_top_left) do ponto superior esquerdo do padrão
```

```
secundário
278
279         # Coordenadas do ponto superior esquerdo do padrão secundário
280         x_stemp = s_top_left[0]
281         y_stemp = s_top_left[1]
282
283     else:   # Caso a imagem capturada seja durante a noite (as coordenadas do ponto
superior esquerdo do padrão secundário são colocadas a zero)
284
285         x_stemp = 0
286         y_stemp = 0
287
288     return(original_image, x_ftemp, y_ftemp, x_stemp, y_stemp)
289
290
291 # Função que permite efetuar as verificações necessárias em relação a borda detetada
e aos padrões (principal e secundário)
292 def verification_edge_template(original_image, rot_original_image, image, rot_image,
image_bright, position_edge, x_ftemp, y_ftemp, x_stemp, y_stemp, detection_edge,
ref_edge_temp_inf, ref_edge_temp_sup, ref_temps, delta_edge_temp, delta_temps, type,
sun_situation, off_xtemp, off_ytemp, h_roi, w_roi, theta, old_theta, roi_top_left,
roi_bottom_right):
293
294     if(detection_edge == 0): # Caso tenha sido detetado uma das bordas superiores da
ribeira
295
296         diff_edge_template = y_ftemp - position_edge    # Verificação da distância
entre o padrão principal e uma das bordas (erro de deteção da borda ou padrão
principal)
297
298         if(abs(diff_edge_template - ref_edge_temp_inf) > delta_edge_temp and
abs(diff_edge_template - ref_edge_temp_sup) > delta_edge_temp):    # Verificar se a
distância padrão principal - borda está dentro dos limites definidos
299             error_edge_temp = 1 # Variável que indica se ocorreu erro ou na borda
detetada ou na deteção do padrão principal de acordo com a distância entre os dois
300         else:
301             error_edge_temp = 0
302
303         if(error_edge_temp == 0):   # Caso a borda ou o padrão principal estejam
detetados corretamente
304
305             if(type == 0):  # A verificação do padrão secundário apenas é feito
durante o dia
306
307                 diff_temps = y_ftemp - y_stemp  # Verificação da distência entre
padrões (erro na inclinação da borda detetada ou da deteção do padrão secundário)
308
309                 if(abs(diff_temps - ref_temps) > delta_temps):  # Verificar se a
distência padrão principal - secundário está dentro dos limites definidos
310                     error_temps = 1 # Variável que indica se ocorreu erro ou na
deteção no padrão secundário ou na inclinação da borda detetada de acordo com a
distância entre o padrão principal - secundário
311                 else:
312                     error_temps = 0
313
314                 if(error_temps == 0):   # Caso o padrão secundário foi detetado
corretamente e a inclinação da borda está também correta
315
```

```
316                     coordinates_roi = (x_ftemp + off_xtemp, y_ftemp + off_ytemp)
                # Coordenadas do ponto superior esquerdo da região de interesse (para
deteção do nível de água)
317                     bottom_right_roi = (coordinates_roi[0] + w_roi,
coordinates_roi[1] + h_roi)    # Coordenadas do ponto inferior direito da região de
interesse
318
319                     image_roi = rot_image[coordinates_roi[1]:bottom_right_roi[1],
coordinates_roi[0]:bottom_right_roi[0]]                    # Recorte da imagem na
região de interesse
320                     original_image_roi =
rot_original_image[coordinates_roi[1]:bottom_right_roi[1],
coordinates_roi[0]:bottom_right_roi[0]] # Recorte da imagem original na região de
interesse
321
322                     rot_original_image_copy = rot_original_image.copy() # Efetuar
uma cópia da imagem original depois de rodada
323                     cv.rectangle(rot_original_image_copy, coordinates_roi,
bottom_right_roi, (0,0,255), 1) # Desenhar o local da região de interesse na imagem
total
324
325                     if(sun_situation == 1): # Quando se está perante o "período de
sombra"
326
327                         # Efetua a rotação da imagem
328                         rotation_angle = math.degrees(theta) - 90
                    # Ângulo de rotação usado para rodar a imagem
329                         rot_img_bright = ndimage.rotate(image_bright,
rotation_angle, reshape = True)   # Imagem com brilho rodada
330
331                         image_bright_roi =
rot_img_bright[coordinates_roi[1]:bottom_right_roi[1],
coordinates_roi[0]:bottom_right_roi[0]]   # Recorte da imagem com brilho na região
de interesse
332
333                     else:
334
335                         rot_img_bright = 0
336                         image_bright_roi = 0
337
338                     compensation = 1    # Variável a indicar se houve ou não uma boa
compensação sem erros detetados ('1' - sem erros, '0' - com erros)
339
340                 else:
341
342                     if(old_theta != 0 and roi_top_left != 0 and roi_bottom_right !=
0): # Caso se tenha os dados referentes à última imagem com uma compensação bem-
sucedida
343
344                         # Efetua a rotação da imagem
345                         rotation_angle = math.degrees(old_theta) - 90
                    # Ângulo de rotação usado para rodar a imagem
346                         image = ndimage.rotate(image, rotation_angle, reshape =
True)                   # Imagem rodada
347                         original_image = ndimage.rotate(original_image,
rotation_angle, reshape = True) # Imagem original rodada
348
349                         # Obtenção da região de interesse
350                         image_roi = image[roi_top_left[1]:roi_bottom_right[1],
roi_top_left[0]:roi_bottom_right[0]]                    # Recorte da imagem na
```

```
região de interesse
                        original_image_roi =
original_image[roi_top_left[1]:roi_bottom_right[1],
roi_top_left[0]:roi_bottom_right[0]]   # Recorte da imagem original na região de
interesse

                        rot_original_image_copy = original_image.copy() # Efetuar
uma cópia da imagem original depois de rodada
                        cv.rectangle(rot_original_image_copy, roi_top_left,
roi_bottom_right, (0,0,255), 1) # Desenhar o local da região de interesse na imagem
total

                        coordinates_roi = 0
                        bottom_right_roi = 0
                        error_temps = 0

                        if(sun_situation == 1): # Quando se está perante o "período
de sombra"

                            rot_img_bright = ndimage.rotate(image_bright,
rotation_angle, reshape = True)   # Imagem com brilho rodada

                            image_bright_roi =
rot_img_bright[roi_top_left[1]:roi_bottom_right[1],
roi_top_left[0]:roi_bottom_right[0]]   # Recorte da imagem com brilho na região de
interesse

                        else:

                            rot_img_bright = 0
                            image_bright_roi = 0

                        compensation = 0    # Variável a indicar se houve ou não uma
boa compensação sem erros detetados ('1' - sem erros, '0' - com erros)

                    else:

                        coordinates_roi = 0
                        bottom_right_roi = 0
                        image_roi = 0
                        original_image_roi = 0
                        rot_img_bright = 0
                        image_bright_roi = 0
                        compensation = 0
                        rot_original_image_copy = 0

            elif(type == 1):    # Quando se está perante uma imagem durante a noite

                coordinates_roi = (x_ftemp + off_xtemp, y_ftemp + off_ytemp)
      # Coordenadas do ponto superior esquerdo da região de interesse (para deteção
do nível de água)
                bottom_right_roi = (coordinates_roi[0] + w_roi, coordinates_roi[1] +
h_roi) # Coordenadas do ponto inferior direito da região de interesse

                image_roi = rot_image[coordinates_roi[1]:bottom_right_roi[1],
coordinates_roi[0]:bottom_right_roi[0]]                    # Recorte da imagem na
região de interesse
                original_image_roi =
rot_original_image[coordinates_roi[1]:bottom_right_roi[1],
```

```
coordinates_roi[0]:bottom_right_roi[0]] # Recorte da imagem original na região de interesse
391
392                 rot_original_image_copy = rot_original_image.copy() # Efetuar uma cópia da imagem original depois de rodada
393                 cv.rectangle(rot_original_image_copy, coordinates_roi, bottom_right_roi, (0,0,255), 1) # Desenhar o local da região de interesse na imagem total
394
395                 error_temps = 0
396                 rot_img_bright = 0
397                 image_bright_roi = 0
398
399                 compensation = 1
400
401         else:
402
403             if(old_theta != 0 and roi_top_left != 0 and roi_bottom_right != 0): # Caso se tenha os dados referentes à última imagem com uma compensação bem-sucedida
404
405                 # Efetua a rotação da imagem
406                 rotation_angle = math.degrees(old_theta) - 90
                # Ângulo de rotação usado para rodar a imagem
407                 image = ndimage.rotate(image, rotation_angle, reshape = True)
                # Imagem rodada
408                 original_image = ndimage.rotate(original_image, rotation_angle, reshape = True) # Imagem original rodada
409
410                 # Obtenção da região de interesse
411                 image_roi = image[roi_top_left[1]:roi_bottom_right[1], roi_top_left[0]:roi_bottom_right[0]]                     # Recorte da imagem na região de interesse
412                 original_image_roi = original_image[roi_top_left[1]:roi_bottom_right[1], roi_top_left[0]:roi_bottom_right[0]]   # Recorte da imagem original na região de interesse
413
414                 rot_original_image_copy = original_image.copy() # Efetuar uma cópia da imagem original depois de rodada
415                 cv.rectangle(rot_original_image_copy, roi_top_left, roi_bottom_right, (0,0,255), 1) # Desenhar o local da região de interesse na imagem total
416
417                 coordinates_roi = 0
418                 bottom_right_roi = 0
419                 error_edge_temp = 0
420                 error_temps = 0
421
422                 if(sun_situation == 1): # Quando se está perante um "período de sombra"
423
424                     rot_img_bright = ndimage.rotate(image, rotation_angle, reshape = True)  # Imagem com brilho rodada
425
426                     image_bright_roi = rot_img_bright[roi_top_left[1]:roi_bottom_right[1], roi_top_left[0]:roi_bottom_right[0]] # Recorte da imagem com brilho na região de interesse
427
428                 else:
```

```
429
430                     rot_img_bright = 0
431                     image_bright_roi = 0
432
433                 compensation = 0
434
435             else:
436
437                 coordinates_roi = 0
438                 bottom_right_roi = 0
439                 image_roi = 0
440                 original_image_roi = 0
441                 rot_img_bright = 0
442                 image_bright_roi = 0
443                 error_temps = 0
444                 compensation = 0
445                 rot_original_image_copy = 0
446
447     else:
448
449         if(old_theta != 0 and roi_top_left != 0 and roi_bottom_right != 0): # Caso
    se tenha os dados referentes à última imagem com uma compensação bem-sucedida
450
451             # Efetua a rotação da imagem
452             rotation_angle = math.degrees(old_theta) - 90
        # Ângulo de rotação usado para rodar a imagem
453             image = ndimage.rotate(image, rotation_angle, reshape = True)
        # Imagem rodada
454             original_image = ndimage.rotate(original_image, rotation_angle, reshape
    = True) # Imagem original rodada
455
456             # Obtenção da região de interesse
457             image_roi = image[roi_top_left[1]:roi_bottom_right[1],
    roi_top_left[0]:roi_bottom_right[0]]                    # Recorte da imagem na
    região de interesse
458             original_image_roi = original_image[roi_top_left[1]:roi_bottom_right[1],
    roi_top_left[0]:roi_bottom_right[0]]   # Recorte da imagem original na região de
    interesse
459
460             rot_original_image_copy = original_image.copy() # Efetuar uma cópia da
    imagem original depois de rodada
461             cv.rectangle(rot_original_image_copy, roi_top_left, roi_bottom_right,
    (0,0,255), 1) # Desenhar o local da região de interesse na imagem total
462
463             coordinates_roi = 0
464             bottom_right_roi = 0
465             detection_edge = 0
466             error_edge_temp = 0
467             error_temps = 0
468
469             if(sun_situation == 1): # Quando se está perante o "período de sombra"
470
471                 rot_img_bright = ndimage.rotate(image_bright, rotation_angle,
    reshape = True)   # Imagem com brilho rodada
472
473                 image_bright_roi =
    rot_img_bright[roi_top_left[1]:roi_bottom_right[1],
    roi_top_left[0]:roi_bottom_right[0]]   # Recorte da imagem com brilho na região de
    interesse
474
```

```
            else:

                rot_img_bright = 0
                image_bright_roi = 0

            compensation = 0

        else:

            coordinates_roi = 0
            bottom_right_roi = 0
            image_roi = 0
            original_image_roi = 0
            rot_img_bright = 0
            image_bright_roi = 0
            error_edge_temp = 0
            error_temps = 0
            compensation = 0
            rot_original_image_copy = 0

    return(rot_original_image_copy, image_roi, original_image_roi, rot_img_bright, image_bright_roi, coordinates_roi, bottom_right_roi, detection_edge, error_edge_temp, error_temps, compensation)


def average_grayscale(image, height, img_zone, pixels): # Função que permite obter uma média (pixels*pixels) da escala de cinza de uma parte da imagem (img_zone) e coloca-a num array

    new_height = (height - pixels) + 1  # Valor máximo do pixel da imagem utilizado que depende do número de pixéis (pixels) que serão utilizados para efetuar a média

    array_grayscale = []    # Array que permite guardar os resultados da média da escala de cinza

    for i in range(new_height): # Ciclo que permite percorrer os pixéis (na dimensão vertical) da imagem até o valor máximo definido

        temp_average = []   # Array que permite guardar os valores da escala de cinza das várias regiões (pixels*pixels) para efetuar a média

        for j in range(pixels): # Ciclo que permite percorrer os pixéis na dimensão vertical

            for k in range(img_zone, img_zone + pixels):    # Ciclo que permite percorrer os pixéis na dimensão horizontal

                temp_average.append(int(image[i+j, k])) # Guarda o valor de escala de cinza doos pixéis selecionados

        average = round(np.mean(temp_average))  # Efetua a média dos valores da escala de cinza

        array_grayscale.append(average) # Coloca o resultado da média no array

    array_axis_y = []   # Eixo vertical (y) que permite colocar os valores da média num gráfico

    offset_y = int((pixels-1)/2)    # Valor do offset (que depende do parâmetro pixels) que permite enquadrar os pixéis da imagem ao resultado da média da escala de
```

```
cinza
521
522     for i in range(new_height): # Ciclo que permite criar o array de pixéis, sendo
que cada um irá corresponder a um determinado valor de escala de cinza (média)
523         array_axis_y.append(i+offset_y)
524
525     return(array_grayscale, array_axis_y)    # Retorna o array com os valores da
média da escala de cinza e o array com os pixéis correspondentes aos valores da
média
526
527
528 def gradient(array_grayscale):  # Função que permite obter o gradiente dos valores
de escala de cinza da imagem
529
530     array_gradient = [] # Array que permite guardar os valores do gradiente
531
532     for i in range(len(array_grayscale)):   # Ciclo que permite percorrer o array
com os valores da escala de cinza
533
534         if(i == len(array_grayscale)-1):    # Condição que permite que, quando se
chegue ao último valor do array, o graidente seja efetuado com ele mesmo, dando
sempre zero, permitindo que o comprimento do array seja igual ao array de escala de
cinza
535             j = i
536         else:
537             j = i+1
538
539         value_gradient = int(array_grayscale[j] - array_grayscale[i])   # Efetua o
gradiente (diferença de dois valores consecutivos)
540
541         array_gradient.append(value_gradient)   # Guarda o resultados do gradiente
no array
542
543     return(array_gradient)  # Retorna o array com os valores do gradiente
544
545
546 def spatial_average(image, height, lim_min, lim_max, pixels, delta_x, x_reference,
slope_value):   # Função que permite efetuar a média espacial dos vários gráficos de
escala de cinza e gradiente
547
548     # Arrays para colocar os valores da média espacial de escala de cinza e
gradiente
549     array_grayscale = []
550     array_axis_y = []
551     array_gradient = []
552
553     points = 0  # Vairável que permite contar quantos pontos/zonas serão utilizadas
de acordo com o delta x escolhido
554
555     for i in range(lim_min, lim_max, delta_x):
556
557         # Arrays temporários
558         temp_grayscale = []
559         temp_axis_y = []
560         temp_gradient = []
561
562         temp_grayscale, temp_axis_y = average_grayscale(image, height, i, pixels)
# Executa a função para obter a média dos valores de escala de cinza de uma
determinada zona da imagem
563
```

```
564         temp_gradient = gradient(temp_grayscale) # Executa a função para obter o
    gradiente de uma determinada zona da imagem
565
566         pixel_position = round(i + ((pixels-1)/2))  # Indica a posição (no eixo x)
    aonde fica situados os valores após a média móvel
567
568         # Permite efetuar a compensação da posição dos valores (escala de cinza,
    gradiente) nos pixéis
569         if(pixel_position < x_reference or pixel_position > x_reference):
570             offset_axis = round(slope_value*((x_reference + 1) - pixel_position))
571         else:
572             offset_axis = 0
573
574         for j in range(len(temp_axis_y)):   # Compensação da posição dos valores
575             temp_axis_y[j] = temp_axis_y[j] + offset_axis
576
577         # Efetuar a reversão dos arrays
578         temp_grayscale_rev = temp_grayscale[::-1]
579         temp_gradient_rev = temp_gradient[::-1]
580
581         array_grayscale.append(temp_grayscale_rev)
582         array_axis_y.append(temp_axis_y)
583         array_gradient.append(temp_gradient_rev)
584
585         points = points + 1
586
587     # Arrays para guardar a média espacial (escala de cinza e gradiente)
588     array_average_index = []
589     array_average_gray = []
590     array_average_grad = []
591
592     for i in range(height): # Ciclo que permite ir de pixel em pixel por forma a
    efetuar a média dos vários valores
593
594         # Variáveis que permitem guardar os vários valores de escala de cinza e
    gradiente
595         value_grayscale = 0
596         value_gradient = 0
597
598         for j in range(points): # Ciclo que permite ir de zona em zona (depende do
    delta x utilizado)
599
600             for k in range(len(array_axis_y[0])):   # Ciclo que percorre linha a
    linha os arrays que contém os valores de escala de cinza e gradiente
601
602                 if (array_axis_y[j][k] == i):
603
604                     # Adiciona os valores de escala de cinza e gradiente
605                     value_grayscale = value_grayscale + array_grayscale[j][k]
606                     value_gradient = value_gradient + array_gradient[j][k]
607
608         # Efetua a média dos valores de escala de cinza e gradiente (o resultado
    terá três casas decimais)
609         value_grayscale = round(value_grayscale/points, 3)
610         value_gradient = round(value_gradient/points, 3)
611
612         # Colocação dos resultados das médias nos arrays correspondentes
613         array_average_index.append(i)
614         array_average_gray.append(value_grayscale)
615         array_average_grad.append(value_gradient)
```

```
616
617     return(array_average_index, array_average_gray, array_average_grad) # Retorna os
    arrays com a média espacial da escala de cinza e gradiente e o arrays com a posição
    dos pixéis
618
619
620 # Função que permite a deteção da posição da linha de água para os diferentes tipos
    de imagens
621 def detect_water_line(image_bright, array_axis_y, array_gradient,
    array_gradient_bright, h_roi, w_roi, height_zones, width_zones, lim_min_day,
    lim_min_night, lim_max, lim_zones_max, lim_zones_min, coord_top_left, type,
    sun_situation, off_night):
622
623     # Arrays temporários para guardar os valores do pixéis e os valores do gradiente
    correspondentes
624     temp_gradient = []
625     temp_gradient_bright = []
626     temp_gradient_original = []
627     temp_gradient_original_bright = []
628     temp_axis_y = []
629
630     process_sun = 0 # Variável que indica qual dos processos é efetuado no "período
    de sombra"
631
632     if(type == 0 and sun_situation == 0):   # Obtenção da posição da linha de água
    durante o dia (excluíndo o "período de sombra")
633
634         for i in range(lim_min_day, lim_max + 1):   # Ciclo que permite guardar os
    valores, dentro da janela definida, do valor absoluto do gradiente e pixéis
    correspondentes
635
636             temp_gradient.append(abs(array_gradient[i]))
637             temp_axis_y.append(array_axis_y[i])
638
639         index_value_max_grad = np.argmax(temp_gradient) # Obtenção da posição no
    array do valor mais elevado (em termos absolutos) do gradiente
640         water_line = temp_axis_y[index_value_max_grad]  # Obtenção da posição do
    pixel que corresponde ao valor mais elevado (em termos absolutos) do gradiente
641
642     elif(type == 0 and sun_situation == 1):   # Obtenção da posição da linha de água
    durante o "período de sombra"
643
644         for i in range(lim_min_day, lim_max + 1):   # Ciclo que permite guardar os
    valores, dentro da janela definida, do valor original e absoluto do gradiente e
    pixéis correspondentes
645
646             temp_gradient_bright.append(abs(array_gradient_bright[i]))
647             temp_gradient_original_bright.append(array_gradient_bright[i])
648             temp_axis_y.append(array_axis_y[i])
649
650         mean_gradient_bright = np.mean(temp_gradient_bright)  # Obtenção da média
    dos valores absolutos do gradiente
651
652         gradient_peaks, _ = find_peaks(temp_gradient_bright, height =
    mean_gradient_bright)   # Efetua a deteção de picos no array com os valores do
    gradiente
653
654         # Arrays temporários para colocar a posição dos picos detetados e o valor
    original e absoluto do gradiente que corresponde aos picos detetados
655         detected_peaks = []
```

```
656         value_detected_peaks_abs = []
657         value_detected_peaks = []
658 
659         for i in range(len(gradient_peaks)):    # Ciclo que indica os picos
existentes dentro da janela definida e seus valores original e absoluto do gradiente
660 
661             detected_peaks.append(temp_axis_y[gradient_peaks[i]])
662 
 value_detected_peaks.append(temp_gradient_original_bright[gradient_peaks[i]])
663             value_detected_peaks_abs.append(temp_gradient_bright[gradient_peaks[i]])
664 
665         index_value_max_grad = np.argmax(value_detected_peaks_abs) # Obtenção da
posição no array do valor mais elevado (em termos absolutos) dos picos detetados no
gradiente
666         pixel_shadow = detected_peaks[index_value_max_grad]  # Obtenção da posição
do pixel que corresponde ao valor mais elevado (em termos absolutos) dos picos
detetados no gradiente
667 
668         pos_pixel_shadow = (h_roi - 1) - pixel_shadow  # Posição (considerando de
baixo para cima na imagem), em pixel, que corresponde ao valor mais elevado (em
termos absolutos) do gradiente
669 
670         offset = round((width_zones - w_roi)/2) # Offset que serve para enquadrar a
zona de pixéis inferior e superior ao valor obtido anteriormente (pixel_shadow)
671 
672         # Coordenadas do ponto superior esquerdo da zona de pixéis inferior e
superior ao valor obtido anteriormente (pixel_shadow)
673         coord_zone_higher_tl = (coord_top_left[0] - offset, coord_top_left[1] +
pos_pixel_shadow - height_zones)
674         coord_zone_bottom_tl = (coord_top_left[0] - offset, coord_top_left[1] +
pos_pixel_shadow)
675 
676         # Coordenadas do ponto inferior direito da zona de pixéis inferior e
superior ao valor obtido anteriormente (pixel_shadow)
677         coord_zone_higher_dr = (coord_zone_higher_tl[0] + width_zones,
coord_zone_higher_tl[1] + height_zones)
678         coord_zone_bottom_dr = (coord_zone_bottom_tl[0] + width_zones,
coord_zone_bottom_tl[1] + height_zones)
679 
680         # Recorte da zona de pixéis inferior e superior ao valor obtido
anteriormente (pixel_shadow)
681         zone_higher = image_bright[coord_zone_higher_tl[1]:coord_zone_higher_dr[1],
coord_zone_higher_tl[0]:coord_zone_higher_dr[0]]
682         zone_bottom = image_bright[coord_zone_bottom_tl[1]:coord_zone_bottom_dr[1],
coord_zone_bottom_tl[0]:coord_zone_bottom_dr[0]]
683 
684         difference_zones = round(np.mean(zone_higher) - np.mean(zone_bottom))   #
Diferença entre a média das duas zonas
685 
686         if(difference_zones > lim_zones_max):   # Quando se está presente de uma
sombra na imagem provocado pelos prédios
687 
688             process_sun = 1 # Indica que efetuou-se o processo quando se está
presente nesta situação (variável que serve para apresentar os gráficos
correspondentes deste processo)
689 
690             new_lim_max = pixel_shadow - 1  # Novo limite máximo de pesquisa pelo
segundo maior valor (em termos absolutos) do gradiente
691 
```

```
692             if(new_lim_max < lim_min_day):  # Caso o novo limite máximo seja menor
que o limite mínimo de pesquisa (durante o dia)
693
694                 water_line = pixel_shadow   # A linha de água é igual a posição que
corresponde ao valor mais elevado (em termos absolutos) do gradiente
695
696             else: # Caso contrário é efetuada a pesquisa do segundo maior valor (em
termos absolutos) do gradiente e sua posição
697
698                 # Arrays temporários para colocação dos picos detetados e valores
correspondentes
699                 new_detected_peaks = []
700                 value_new_detected_peaks_abs = []
701
702                 for i in range(len(detected_peaks)):    # Ciclo que indica os picos
existentes dentro da janela definida e seu valor absoluto do gradiente
703
704                     if(detected_peaks[i] >= lim_min_day and detected_peaks[i] <=
new_lim_max):  # Caso o pico detetado esteja dentro da janela definida é guardado a
posição e o valor absoluto do pico
705
706                         new_detected_peaks.append(detected_peaks[i])
707
 value_new_detected_peaks_abs.append(abs(array_gradient_bright[detected_peaks[i]]))
708
709                 if(len(new_detected_peaks) == 0):   # Caso não tenha sido detetado
nenhum pico dentro da janela definida
710
711                     water_line = pixel_shadow
712
713                 else:   # Caso contrário procura o maior valor (em termos absolutos)
do gradiente e sua posição
714
715                     index_value_max_grad = np.argmax(value_new_detected_peaks_abs)
 # Obtenção da posição no array do maior valor (em termos absolutos) dos picos
detetados no gradiente
716                     water_line = new_detected_peaks[index_value_max_grad]
# Obtenção da posição do pixel que corresponde ao maior valor (em termos absolutos)
dos picos detetados no gradiente
717
718         elif(difference_zones < lim_zones_min): # Quando se está presente de uma
sombra na imagem provocado pela própria parede da ribeira
719
720             for i in range(lim_min_day, lim_max + 1):   # Ciclo que permite guardar
os valores, dentro da janela definida, do valor original e absoluto do gradiente e
pixéis correspondentes
721
722                 temp_gradient.append(abs(array_gradient[i]))
723                 temp_gradient_original.append(array_gradient[i])
724                 temp_axis_y.append(array_axis_y[i])
725
726             mean_gradient = np.mean(temp_gradient)  # Obtenção da média dos valores
absolutos do gradiente
727
728             gradient_peaks, _ = find_peaks(temp_gradient, height = mean_gradient)
# Efetua a deteção de picos no array com os valores do gradiente
729
730             # Arrays temporários para colocar a posição dos picos detetados e o
valor original e absoluto do gradiente que corresponde aos picos detetados
731             detected_peaks = []
```

```
732             value_detected_peaks = []
733 
734             for i in range(len(gradient_peaks)):    # Ciclo que indica os picos
existentes dentro da janela definida e seu valor no gradiente
735 
736                 detected_peaks.append(temp_axis_y[gradient_peaks[i]])
737 
 value_detected_peaks.append(temp_gradient_original[gradient_peaks[i]])
738 
739             index_value_min_grad = np.argmin(value_detected_peaks)  # Obtenção da
posição no array do menor valor dos picos detetados no gradiente
740             water_line = detected_peaks[index_value_min_grad]       # Obtenção da
posição do pixel que corresponde ao menor valor dos picos detetados no gradiente
741 
742         else:
743 
744             for i in range(lim_min_day, lim_max + 1):   # Ciclo que permite guardar
os valores, dentro da janela utilizada, do valor absoluto do gradiente e pixéis
correspondentes
745 
746                 temp_gradient.append(abs(array_gradient[i]))
747                 temp_axis_y.append(array_axis_y[i])
748 
749             index_value_max_grad = np.argmax(temp_gradient) # Obtenção da posição no
array do valor mais elevado (em termos absolutos) do gradiente
750             water_line = temp_axis_y[index_value_max_grad]  # Obtenção da posição do
pixel que corresponde ao valor mais elevado (em termos absolutos) do gradiente
751 
752     elif(type == 1):  # Obtenção da posição da linha de água durante a noite
753 
754         for i in range(lim_min_night, lim_max + 1):   # Ciclo que permite guardar os
valores, dentro da janela utilizada, do valor absoluto do gradiente e pixéis
correspondentes
755 
756             temp_gradient.append(abs(array_gradient[i]))
757             temp_axis_y.append(array_axis_y[i])
758 
759         index_value_max_grad = np.argmax(temp_gradient) # Obter a posição no array
do valor mais elevado (em termos absolutos) do gradiente
760         water_line = temp_axis_y[index_value_max_grad]  # Obtenção da posição do
pixel que corresponde ao valor mais elevado (em termos absolutos) do gradiente
761 
762         water_line = water_line + off_night # Soma do valor de offset na posição de
linha de água obtida devido à questão da sombra na água durante a noite
763 
764     if(type == 0 and sun_situation == 0 or type == 1):  # Caso a imagem tenha sido
capturada durante o dia, sem estar inserida no "período de sombra", ou durante a
noite
765 
766         difference_zones = 0
767 
768     return(water_line, process_sun)
769 
770 
771 def convert_pixel_cm(yi, s_water_line, s_calibrate, b_s_calibrate, roi_x, P_y0,
x_y_200, T, C_1, C_2, S_V, R_V, S_H, R_H, d, a_y, a_x, f, Y0):   # Função que
permite converter o valor da altura do nível de água de pixéis para centímetros
772 
773     P_x = (1/s_calibrate*yi + b_s_calibrate)/1.022   # Cálulo do valor de P'x
```

```
    P_y = s_water_line*P_x + yi                     # Cálculo do valor de P'y que corresponde ao valor de P_H

    Py = P_y + P_y0                             # Cálculo do valor de Py
    Px = (1/s_calibrate)*Py + roi_x + x_y_200 # Cálculo do valor de Px

    y1 = (Py * T * C_1 * C_2 * S_V * d) / (R_V * f) # Cálculo do valor de y1 que corresponde ao ponto, no eixo vertical, do nível de água no plano da imagem
    x1 = (Px * T * S_H * d) / (R_H *f)              # Cálculo do valor de x1 que corresponde ao ponto, no eixo horizontal, do nível de água no plano da imagem

    X = (d * x1) / (x1*np.sin(a_x) + y1*np.cos(a_x)*np.tan(a_y) + d*np.cos(a_x))
 # Cálculo do valor de X que corresponde, no eixo horizontal, do nível de água no plano real
    Y = (y1*d - y1*X*np.sin(a_x)) / (y1*np.sin(a_y) + d*np.cos(a_y))
 # Cálculo do valor de Y que corresponde, no eixo vertical, do nível de água no plano real

    height_water_level = round((Y - Y0)*100, 1)   # Cálculo da altura do nível de água em centímetros

    return(height_water_level)



# Diretórios utilizados no algoritmo
inicial_directory = os.getcwd()                                         # É o diretório (C:\Project_Detect_Water_Line) que contém todos os ficheiros utilizados neste algoritmo (incluíndo este ficheiro de python)
directory1 = r'C:\Project_Detect_Water_Line\1_Images'                   # Contém as imagens a utilizar no algoritmo
directory2 = r'C:\Project_Detect_Water_Line\2_Group_Images_for_Average' # Contém o grupo de imagens a utilizar para efetuar média temporal
directory3 = r'C:\Project_Detect_Water_Line\3_Images_After_Average'     # Contém as imagens após efetuar a média temporal
directory4 = r'C:\Project_Detect_Water_Line\4_Results'                  # Contém os gráficos com os resultados obtidos
directory5 = r'C:\Project_Detect_Water_Line\5_Region_of_Interest'       # Contém a região de interesse com a linha de água obtida
directory6 = r'C:\Project_Detect_Water_Line\6_Total_Image_With_ROI'     # Contém a imagem total (após efetuar a rotação) com a zona da região de interesse desenhada



def main():

    # Variáveis utilizadas no algoritmo
    image_type = 0      # Variável que indica o tipo de imagens que estão a ser utilizadas (0 - dia, 1 - noite)

    sun_period = 0                  # Variável que indica se as imagens utilizadas estão ('1') ou não ('0') dentro do "período de sombra"
    initial_sun_period = "08:10"    # Variável que indica o início do "período de sombra"
    final_sun_period = "13:10"      # Variável que indica o fim do "período de sombra"

    num_img_avg = 1     # Número de imagens a utilizar para efetuar a média temporal das mesmas

```

```
    brightness = 2.0    # Variável que indica o valor do brilho a aplicar a imagem

    offset_initial_x_edge_detection = 250    # Offset de forma a descartar os primeiros 250 pixéis (eixo horizontal) da zona de deteção de bordas na imagem
    offset_final_x_edge_detection = 100      # Offset de forma a descartar os últimos 100 pixéis (eixo horizontal) da zona de deteção de bordas na imagem
    distance_y_edge_detection = 144          # Tamanho da zona de deteção de bordas na imagem no eixo vertical
    line_points = 270                        # Variável que indica o número de interseções (pontos) necessários para detetar uma borda
    num_discarded_pixels = 10                # Número de pixéis (no eixo vertical) a descartar na deteção de bordas

    difference_edge_inf_temp = 186.5    # Valor de referência da distância entre o padrão principal e a borda inferior da ribeira
    difference_edge_sup_temp = 200.5    # Valor de referência da distância entre o padrão principal e a borda superior da ribeira
    delta_diff_edge_temp = 2            # Valor do delta em que o valor da distância entre uma das bordas e o padrão principal pode variar
    difference_templates = 32.5         # Valor de referência da distância entre o padrão principal e o secundário
    delta_diff_templates = 3            # Valor do delta em que o valor da distância entre o padrão principal e o secundário pode variar
    offset_x_temp = 420                 # Valor do offset (no eixo horizontal) a efetuar de forma a obter a região de interesse
    offset_y_temp = 200                 # Valor do offset (no eixo vertical) a efetuar de forma a obter a região de interesse
    height_roi = 250                    # Variável que indica a altura da região de interesse
    width_roi = 100                     # Variável que indica a largura da região de interesse

    coord_roi_top_left_prev = 0         # Guarda as coordenadas da posição superior esquerda da região de interesse anterior
    coord_roi_bottom_right_prev = 0     # Guarda as coordenadas da posição inferior direita da região de interesse anterior
    previous_theta = 0                  # Guarda o valor de theta anterior

    initial_zone = 0                    # Coluna (de pixéis) inicial aonde irá começar a ser retirado valores para obter a média da escala de cinza e gradiente
    final_zone = width_roi - 10         # Coluna (de pixéis) final aonde irá ser retirado os últimos valores para obter a média da escala de cinza e gradiente


    pixels_average_gray = 5      # Número de pontos que será utilizado, na vertical e horizontal (5*5 = 25), para efetuar a média dos valores de escala de cinza
    step_pixel = 9               # Variável que indica a cadência (na dimensão horizontal) que é retirado os valores de escala de cinza para posterior processamento
    pixel_reference = 49         # Valor utilizado com referência para efetuar o alinhamento dos vários arrays de escala de cinza e gradiente
    slope_water_line = -0.1268   # Declive que indica a tendência da linha de água em diferentes níveis de altura do caudal da ribeira

    pixel_lim_min_day = 90      # Offset mínimo aonde começa a pesquisa da linha de água durante o dia
    pixel_lim_min_night = 50    # Offset mínimo aonde começa a pesquisa da linha de água durante a noite
    pixel_lim_max = 180         # Offset máximo aonde acaba a pesquisa da linha de água durante o dia e noite
```

```
846
847     h_zones_sun_period = 10     # Altura do retângulo de píxeis que será utilizado
    durante o "período de sombra"
848     w_zones_sun_period = 200    # Largura do retângulo de píxeis que será utilizado
    durante o "período de sombra"
849
850     limit_zones_max_sun_period = 40     # Limite máximo que indica quando é que se
    está perante o primeiro "período de sombra"
851     limit_zones_min_sun_period = -20    # Limite mínimo que indica quando é que se
    está perante o segundo "período de sombra"
852
853     offset_night = 33   # Offset utilizado para somar ao valor obtido da linha de
    sombra (durante o período da noite) para obter a linha de água
854
855     slope_calibration = 1/0.174         # Variável que corresponde ao declive da
    reta da linha de calibração
856     b_slope_calibration = 30            # Variável que corresponde à abcissa quando
    a reta da linha de calibração interseta o eixo y
857     x_roi = 350                         # Variável que corresponde ao ponto, neste
    caso no eixo x, inferior esquerdo da região de interesse, visto do centro da imagem
858     y_roi = -380                        # Variável que corresponde à origem da
    região de interesse na imagem, no eixo vertical
859     x__y_200 = 116                      # Variável que corresponde à abcissa quando
    o valor de y'=200
860     pixel_size = 0.0014                 # Variável que corresponde ao tamanho do
    pixel em milímetros
861     img_cropping1 = 0.75                # Variável que corresponde ao corte da
    imagem devido à alteração da proporção da tela
862     img_cropping2 = 0.85                # Variável que corresponde ao segundo corte
    da imagem devido à estrada rodoviária
863     sensor_size_vertical = 1944         # Variável que corresponde ao tamanho do
    sensor na vertical em pixéis
864     sensor_size_horizontal = 2592       # Variável que corresponde ao tamanho do
    sensor na horizontal em pixéis
865     img_resolution_vertical = 720       # Variável que corresponde à resolução da
    imagem na vertical em pixéis
866     img_resolution_horizontal = 1280    # Variável que corresponde à resolução da
    imagem na horizontal em pixéis
867     distance_camera_wall = 24.16        # Variável que corresponde à distância entre
    a camera e a parede de betão em metros
868     alpha_x = 7.4                       # Variável que corresponde ao valor do
    ângulo alpha x em graus
869     alpha_y = 22.2                      # Variável que corresponde ao valor do
    ângulo alpha y em graus
870     focal_length = 3.60                 # Variável que corresponde ao valor do
    comprimento focal em milímetros
871     zero_flow_height = -5.25            # Variável que corresponde ao valor da
    altura do caudal zero em metros
872
873     alpha_x_rad = np.radians(alpha_x)      # Variável que corresponde ao valor do
    ângulo alpha x em radianos
874     alpha_y_rad = np.radians(alpha_y)      # Variável que corresponde ao valor do
    ângulo alpha y em radianos
875
876
877     # Limpeza dos ficheiros que estão nos diretórios
878     clean_pastes(directory2)
879     clean_pastes(directory3)
880     clean_pastes(directory4)
881     clean_pastes(directory5)
```

```
882     clean_pastes(directory6)
883
884
885     # Leitura dos padrões (principal e secundário) e obtenção das dimensões dos
    mesmos
886     first_template = cv.imread("Main_Template.jpg", 0)
887     second_template = cv.imread("Secondary_Template.jpg", 0)
888
889     archive_results = open('C:\Project_Detect_Water_Line\Results.txt', 'w+')    #
    Criação do ficheiro de texto que irá guardar a data e hora da imagem captada e o
    respetivo nível de água
890
891     analysis_empty_data(directory1) # Efetua a função que elimina imagens sem
    conteúdo
892
893     time_average(inicial_directory, directory1, directory2, directory3, num_img_avg)
      # Efetua a média temporal das imagens
894
895
896     for root, dirs, files in os.walk(directory3):
897         for f in files:
898
899             path = os.path.join(root, f)    # Permite juntar o caminho do diretório
    com o nome do ficheiro (neste caso, o nome da imagem)
900
901             # Executa a função date_hour para obter a data e hora da imagem
902             dhour, hour_min = date_hour(f)
903             dhour_str = str(dhour)
904
905             print(dhour_str)
906
907             if(hour_min >= initial_sun_period and hour_min <= final_sun_period):
     # Verificar se as imagens estão dentro do "período de sombra"
908                 sun_period = 1
909             else:
910                 sun_period = 0
911
912             image = cv.imread(path) # Leitura da imagem
913
914             image_gray = cv.cvtColor(image, cv.COLOR_BGR2GRAY)  # Passagem da imagem
    a cores para escala de cinza
915
916             w, h = image_gray.shape[::-1]   # Dimensões (w - largura, h - altura) da
    imagem
917
918             # CLAHE (Equalização de histograma adaptativo de contraste limitado)
919             clahe = cv.createCLAHE(clipLimit = 2.0, tileGridSize = (8,8))
920             image_clahe = clahe.apply(image_gray)
921
922             if(sun_period == 1):    # Verificar se as imagens estão dentro de o
    "período de sombra" que provoca sombra na região de interesse da imagem
923
924                 image_bright = bright(image, brightness)    # Função que permite
    aumentar o brilho de uma imagem
925
926                 image_gray_bright = cv.cvtColor(image_bright, cv.COLOR_BGR2GRAY)  #
    Passagem da imagem a cores para escala de cinza
927
928                 # CLAHE (Equalização de histograma adaptativo de contraste limitado)
929                 clahe_bright = cv.createCLAHE(clipLimit = 2.0, tileGridSize = (8,8))
```

```
930                 image_clahe_bright = clahe_bright.apply(image_gray_bright)
931 
932             else:
933                 image_clahe_bright = 0
934 
935             # Função que efetuar a calibração em termos de rotação da imagem
936             rotated_image_clahe, rotated_original_image, rotated_rho, rotated_theta,
    error_detection_edge = image_rotation(image, image_clahe, w, h, line_points,
    offset_initial_x_edge_detection, offset_final_x_edge_detection,
    distance_y_edge_detection, num_discarded_pixels)
937 
938             # Função que efetua a compensação da translação
939             rot_comp_original_image, coord_x_ftemp, coord_y_ftemp, coord_x_stemp,
    coord_y_stemp = translation_compensation(rotated_original_image,
    rotated_image_clahe, first_template, second_template, image_type)
940 
941             # Função que permite efetuar as verificações necessárias em relação a
    borda detetada e aos padrões (principal e secundário)
942             org_image, roi, original_roi, rotated_image_clahe_bright, roi_bright,
    coord_roi_top_left, coord_roi_bottom_right, error_detection_edge,
    error_edge_template, error_templates, good_verification =
    verification_edge_template(image, rot_comp_original_image, image_clahe,
    rotated_image_clahe, image_clahe_bright, rotated_rho, coord_x_ftemp, coord_y_ftemp,
    coord_x_stemp, coord_y_stemp, error_detection_edge, difference_edge_inf_temp,
    difference_edge_sup_temp, difference_templates, delta_diff_edge_temp,
    delta_diff_templates, image_type, sun_period, offset_x_temp, offset_y_temp,
    height_roi, width_roi, rotated_theta, previous_theta, coord_roi_top_left_prev,
    coord_roi_bottom_right_prev)
943 
944             if(good_verification == 1): # Caso ocorra uma boa compensação em termos
    de rotação e translação
945 
946                 previous_theta = rotated_theta
947                 coord_roi_top_left_prev = coord_roi_top_left
948                 coord_roi_bottom_right_prev = coord_roi_bottom_right
949 
950 
951             if(error_detection_edge == 0 and error_edge_template == 0 and
    error_templates == 0):    # Caso não ocorra nenhum erro
952 
953                 if(sun_period == 1):
954 
955                     # Função que efetua a média espacial dos valores de escala de
    cinza e gradiente
956                     axis_y, average_grayscale_bright, average_gradient_bright =
    spatial_average(roi_bright, height_roi, initial_zone, final_zone,
    pixels_average_gray, step_pixel, pixel_reference, slope_water_line)
957 
958                 axis_y, average_grayscale, average_gradient = spatial_average(roi,
    height_roi, initial_zone, final_zone, pixels_average_gray, step_pixel,
    pixel_reference, slope_water_line)
959 
960 
961                 if(sun_period == 1):
962 
963                     # Função que permite, para qualquer situação, obter a linha de
    água, na zona central da imagem, em pixéis
964                     pixel_water_line, sun_process =
    detect_water_line(rotated_image_clahe_bright, axis_y, average_gradient,
    average_gradient_bright, height_roi, width_roi, h_zones_sun_period,
```

```
    w_zones_sun_period, pixel_lim_min_day, pixel_lim_min_night, pixel_lim_max,
    limit_zones_max_sun_period, limit_zones_min_sun_period, coord_roi_top_left_prev,
    image_type, sun_period, offset_night)
965
966                 else:
967
968                     rotated_image_clahe_bright = 0
969                     average_gradient_bright = 0
970
971                     pixel_water_line, sun_process =
    detect_water_line(rotated_image_clahe_bright, axis_y, average_gradient,
    average_gradient_bright, height_roi, width_roi, h_zones_sun_period,
    w_zones_sun_period, pixel_lim_min_day, pixel_lim_min_night, pixel_lim_max,
    limit_zones_max_sun_period, limit_zones_min_sun_period, coord_roi_top_left_prev,
    image_type, sun_period, offset_night)
972
973
974                 if(image_type == 1):# Situação durante a noite
975
976                     pixel_water_line = pixel_water_line + offset_night  # Somar ao
    valor obtido da função anterior o offset de forma a obter a linha de água na zona
    central da imagem
977
978                 # Efetuar o desenho da linha de água na imagem roi
979                 position_water_line = (height_roi - 1) - pixel_water_line
980                 offset_y_initial = abs(round((pixel_reference +
    1)*slope_water_line))
981                 offset_y_final = abs(round(((width_roi - 1) -
    pixel_reference)*slope_water_line))
982
983                 if(sun_period == 1):
984
985                     roi_bright_copy = roi_bright.copy()
986                     cv.line(roi_bright_copy, (0, position_water_line -
    offset_y_initial), ((width_roi - 1), position_water_line + offset_y_final), 0, 1)
     # Desenhar a linha de água na região de interesse
987
988                 roi_copy = roi.copy()
989                 cv.line(roi_copy, (0, position_water_line - offset_y_initial),
    ((width_roi - 1), position_water_line + offset_y_final), 0, 1)   # Desenhar a linha
    de água na região de interesse
990
991
992                 # Valor da linha de água no início da imagem (x = 1)
993                 value_line_water = (pixel_water_line + 1) + offset_y_initial
994                 print(value_line_water)
995
996
997                 # Função que permite converter de pixéis para centímetros
998                 water_level = convert_pixel_cm(value_line_water, slope_water_line,
    slope_calibration, b_slope_calibration, x_roi, y_roi, x__y_200, pixel_size,
    img_cropping1, img_cropping2, sensor_size_vertical, img_resolution_vertical,
    sensor_size_horizontal, img_resolution_horizontal, distance_camera_wall,
    alpha_y_rad, alpha_x_rad, focal_length, zero_flow_height)
999                 print(water_level)
1000
1001
1002                archive_results.write(str(dhour) + "," + str(value_line_water) + ','
    + str(water_level) + "\n")  # Escrita no ficheiro de texto a data e hora da imagem e
    o valor respetivo (em píxeis) da linha de água
```

```
1003
1004
1005                # Efetuar o desenho da linha de água na imagem roi original
1006                offset_y_original = abs(round(25*slope_water_line))
1007
1008
1009                # Desenhar uma linha, na vertical, na zona central da imagem
(referência utilizada) e uma na linha de água deste o centro até o centro - 25
pixéis
1010                original_roi_copy = original_roi.copy()
1011                cv.line(original_roi_copy, (pixel_reference - 25,
position_water_line - offset_y_original), (pixel_reference, position_water_line),
(0,0,255), 1)
1012                cv.line(original_roi_copy, (pixel_reference, 0), (pixel_reference,
height_roi -1 ), (0,0,255), 1)
1013
1014
1015                # Construção dos gráficos
1016                plt.subplot(131)
1017
1018                if(sun_period == 1 and sun_process == 1):   # Caso se esteja
presente no primeiro "período de sombra"
1019                    plt.imshow(roi_bright_copy, cmap = 'gray')  # Região de
interesse da imagem do primeiro "período de sombra"
1020                else:
1021                    plt.imshow(roi_copy, cmap = 'gray') # Região de interesse da
imagem para as restantes situações
1022
1023                plt.axis('off')
1024
1025                plt.subplot(132)
1026
1027                if(sun_period == 1 and sun_process == 1):   # Caso se esteja
presente no primeiro "período de sombra"
1028                    plt.plot(average_grayscale_bright, axis_y, color = 'red')   #
Gráfico com os valores da média móvel de escala de cinza para o primeiro "período de
sombra"
1029                else:
1030                    plt.plot(average_grayscale, axis_y, color = 'red')  # Gráfico
com os valores da média móvel de escala de cinza para as restantes situações
1031
1032                plt.ylim(0, (height_roi-1))
1033                plt.xlim(0, 256)
1034                plt.xticks([0, 50, 100, 150, 200, 250])
1035                plt.yticks([0, 50, 100, 150, 200])
1036                plt.title('Grayscale image', fontsize = 10)
1037                plt.xlabel('Gray level', fontsize = 10)
1038                plt.ylabel('Distance (pixel)', fontsize = 10)
1039
1040                plt.subplot(133)
1041
1042                if(sun_period == 1 and sun_process == 1):   # Caso se esteja
presente no primeiro "período de sombra"
1043                    plt.plot(average_gradient_bright, axis_y, color = 'blue')   #
Gráfico com os valores do gradiente para o primeiro "período de sombra"
1044                else:
1045                    plt.plot(average_gradient, axis_y, color = 'blue')  # Gráfico
com os valores do gradiente para as restantes situações
1046
1047                plt.ylim(0, (height_roi-1))
```

```
1048
1049                if(image_type == 0):
1050                    plt.xlim(-45, 45)
1051                else:
1052                    plt.xlim(-15, 15)   # Para as imagens noturnas
1053
1054                if(image_type == 0):
1055                    plt.xticks([-45, -30, -15, 0, 15, 30, 45])
1056                else:
1057                    plt.xticks([-15, -10, -5, 0, 5, 10, 15]) # Para as imagens
    noturnas
1058
1059                plt.yticks([0, 50, 100, 150, 200])
1060                plt.title('Gradient', fontsize = 10)
1061                plt.xlabel('Gray difference', fontsize = 10)
1062                plt.ylabel('Distance (pixel)', fontsize = 10)
1063
1064                plt.subplots_adjust(wspace = 0.7)
1065
1066                # Permite a criação de uma imagem com os vários gráficos anteriores
1067                figure = plt.gcf()
1068                figure.set_size_inches(10, 5)
1069
1070                # Guarda a imagem com os vários gráficos no diretório
1071                os.chdir(directory4)
1072                plt.savefig('%s.jpg' %dhour_str, dpi = 100)
1073
1074                plt.clf()
1075
1076
1077                os.chdir(directory5)
1078                cv.imwrite("%s_roi.jpg" %dhour_str, original_roi_copy)   # Guarda a
    região de interesse com a linha de água desenhada no diretório
1079
1080                os.chdir(directory6)
1081                cv.imwrite("%s.jpg" %dhour_str, org_image)   # Guarda a imagem com a
    borda, os padrões e a região de interesse desenhada
1082
1083
1084            elif(error_detection_edge == 1):    # Caso tenha ocorrido um erro na
    deteção de uma das bordas superiores da ribeira
1085                print(dhour_str + ": " + "Erro - Não foi possível detetar uma das
    bordas superiores da ribeira")
1086
1087            elif(error_edge_template == 1): # Caso tenha ocorrido um erro na posição
    da borda detetada ou na deteção do padrão principal
1088                print(dhour_str + ": " + "Erro - Deteção incorreta do padrão
    principal ou da borda superior da ribeira")
1089
1090            elif(error_templates == 1): # Caso tenha ocorrido um erro na deteção do
    padrão secundário ou na inclinação da borda detetada
1091                print(dhour_str + ": " + "Erro - Deteção incorreta do padrão
    secundário ou inclinação incorreta da borda superior da ribeira detetada")
1092
1093    archive_results.close() # Fechar o ficheiro de texto
1094
1095
1096 # Função main
1097 main()
```

sensors

MDPI

Article

Measurement of Water Level in Urban Streams under Bad Weather Conditions

Joaquim Amândio Azevedo * and João André Brás

Faculty of Exact Sciences and Engineering, University of Madeira, 9020-105 Funchal, Portugal; jbras1998@hotmail.com
* Correspondence: jara@uma.pt

Abstract: Flood control and water resources management require monitoring the water level in rivers and streams. Water level measurement techniques increasingly consider image processing procedures. Most of the systems use a staff gauge to support the waterline detection. However, these techniques can fail when applied to urban stream channels due to water undulation, debris on the water surface, and traces of rain captured by the camera, and other adverse effects on images can be quite dramatic on the results. The importance of considering these effects is that they are usually associated with the variation in the water level with the occurrence of rain. The technique proposed in this work uses a larger detection zone to minimize the effects that tend to obstruct the waterline. The developed system uses an infrared camera to operate during the day and night. Images acquired in different weather conditions helped to evaluate the proposed technique. The water level measurement accuracy was about 1.8 cm for images taken during the day and 2.8 cm for images taken at night. During short periods of heavy rain, the accuracy was 2.6 cm for the daytime and 3.4 cm for the nighttime. Infrared lighting can improve detection accuracy at night. The developed technique provides good accuracy under different weather conditions by combining information from various detection positions to deal with waterline detection issues.

Keywords: urban monitoring; water stream channels; water level measurement; image processing

Citation: Azevedo, J.A.; Brás, J.A. Measurement of Water Level in Urban Streams under Bad Weather Conditions. *Sensors* **2021**, *21*, 7157. https://doi.org/10.3390/s21217157

Academic Editors: Aime' Lay-Ekuakille and Ali Khenchaf

Received: 26 August 2021
Accepted: 25 October 2021
Published: 28 October 2021

1. Introduction

Monitoring the water level in rivers, streams, and reservoirs has several applications, such as flood control, water flow measurement, and water resources management [1–5]. The techniques typically employed to measure the water level are based on float or pressure sensors, ultrasonic water meters, satellite-based systems, and image-based systems [6]. Owing to the risk of flooding, it is essential to control the flow of urban streams in hydrographic basins with a large population close to mountainous areas. In heavy rainfall situations, the water can drag large amounts of sediment and organic matter, rendering contact measurement systems unusable. Ultrasonic systems are simple to install, but they have several disadvantages, namely those related to water turbulence [7,8]. Satellite systems do not provide sufficient spatial or temporal resolution, especially for small water streams [9,10]. Image-based systems are a viable alternative to measure water level due to their low cost and easy installation beside a river or near houses [11–13]. Image analysis techniques to estimate the water level seem to be the most suitable for urban stream channels. While some works propose unmanned aerial vehicles with image acquisition capabilities for water monitoring [14–16], a fixed system is best suited to the context of this work.

Numerous works propose extraction of the waterline in image-based systems using a staff gauge to support the measurements. Hies et al. [17] applied an edge detection algorithm and the Hough transform [18] to detect the waterline over a white ruler located in the wall of an urban stream channel. Lo et al. [4] used images captured every minute to monitor the water level in urban riverine areas. They discarded images with low contrast or low brightness from the analysis. A water ruler on a bridge pier made it possible to monitor

the water level. The authors also proposed the use of virtual markers when there is no ruler in the monitored zone. Lin et al. [19] determined an average image from successive images to reduce noise and applied the Hough transform to identify the waterline. They reported an accuracy of 1 cm when using single-camera images. Zhang et al. [20] proposed a system based on an infrared video camera to solve problems of poor visibility, image distortions, and ambient noise in water level measurements with staff gauges. A photovoltaic system of 200 W and a wind generator of 300 W supported by 12 V @300 Ah batteries provided the power to the camera and communication system. The method proposed in [21] deals with different illumination conditions of the water gauge. The authors used the difference between two adjacent regions of interest in the gray image, first with coarse regions to detect a zone for the waterline and then with fine positioning of the waterline. Xu et al. [22] proposed to improve the waterline detection accuracy by identifying the characters on the staff gauge image through a neural network. Image recognition with a staff gauge is also used in [23,24], obtaining a measurement error of 0.9 cm.

Some image-based water level measurement systems do not use staff gauges. The land region of the stream channel may have some texture allowing for the discrimination of the water region [25]. Griesbaum et al. [26] extracted the waterline along a building facade during flood events. Ridolfi et al. [27] proposed a method to obtain the waterline in dam reservoirs, where the high contrast between the concrete face and the water helped the decision. They applied the Canny method [28] to detect the water level. In the context of mountain streams, Young et al. [29] used several vertical rocks where a clear edge allowed the definition of the water margin. They manually removed images without a clear edge at the water margin from the detection process. Leduc et al. [30] considered a different method to obtain the water level of a mountain river, but they also removed images taken under bad weather conditions, like those obtained during rainfall events. Eltner et al. [31] deployed ground control points to provide a reference for the image data. To obtain the waterline, they used time-lapse images to highlight the water regions due to moving water.

Most of the techniques proposed in the literature to measure the water level in rivers and streams use staff gauges. The methods employed can be edge detection of the waterline, image thresholding to recognize the water surface, or character recognition on the staff gauge. Some techniques have used successive frames of images to improve waterline detection. Nevertheless, the presence of debris in the water obstructing the staff gauge and insufficient illumination make the process of water-level measurement difficult. In many cases, existing techniques discard images with low contrast or insufficient brightness. However, these images can be captured at night or during periods of rain in situations where water level measurement is more important. Several studies show high accuracy in detecting the water level, some of them providing results around 1 cm. While this is true for cases where the water has almost no undulation, other situations may impose lower accuracies owing to water level fluctuations. For instance, the work presented in [26] indicated an accuracy of 5 cm for water undulation of ± 10 cm. This issue is rather important in periods of heavy precipitation. It should also be considered that raindrops on the camera lens could affect images because of outdoor installation.

In this work, we proposed a technique based on an image system to measure the water level in urban stream channels. Walls often flank these narrow-width streams. Heavy rainfall occurring in the surrounding mountains can cause rapid changes in the water level. In these situations, the water can drag large amounts of sediment and organic matter. These occurrences change the water quality and floating debris, making it difficult to use existing techniques to measure the water level. Therefore, we developed a new technique to deal with the debris on the water or obstacles in the waterline and consider different weather conditions. The technique does not require a staff gauge. A new approach became necessary to relate the image plane to the object plane that simplifies the parameterization needed to measure the water level. The image acquisition system is of low cost, autonomous in energy supply, and it makes use of the easy access to communication facilities normally

found in urban areas. It also enables local processing to launch alerts, if desired, and internet access to send images to a remote server.

2. Materials and Methods

2.1. Measurement System

Flash floods have given rise to the greatest natural disasters on Madeira Island, with significant loss of human life. Given the orography of the island, with the highest point at 1862 m, heavy rains have caused strong water flows in the streams of the city of Funchal. From the beginning of the 19th century to the end of 2010, 38 flash floods were recorded on Madeira Island [32]. About 1000 people died in the flash flood of 1803, mostly in Funchal. More recently, the flash flood of 20 February 2010 resulted in more than 45 deaths. The weather station near Funchal recorded an accumulated rainfall above 4000 mm between October 2009 and February 2010, with some days recording a precipitation above 100 mm [33].

Figure 1 shows an image of the channel used in the experimental setup to support the development of the proposed technique. This is one of the three main water streams of Funchal with a high potential risk of flooding. The figure also illustrates the region of Madeira where the study took place. Stone or concrete walls typically flank these urban streams. The installation of a staff gauge to provide a reference system for the waterline detection proved difficult or impossible due to the water flow being too strong during heavy rain events. Thus, this created a reference system obtained from natural existing control points on the channel wall.

Figure 1. Urban stream channel image of the experimental setup.

We developed a low-cost image acquisition system based on a Raspberry Pi 3 model B and a Pi NoIR camera V1 [34]. This infrared camera allows for daytime as well as nighttime operation with different luminosity conditions. The camera was installed on the ceiling of a balcony in a building facing the stream. This method of installing the camera has several advantages. As the camera is under a balcony, and therefore protected from the rain, the lens becomes sheltered from raindrops. This installation also protects the camera from direct sunlight, which would saturate the image, and eliminates the need for a mast to suspend the camera, minimizing the environmental impact. The system used the Wi-Fi network of the house, avoiding the installation of a dedicated communication system. In the power supply for the camera, options were not limited to the house's electrical system but also to a renewable energy system. For this study, we installed an 80 W solar panel on

the building terrace for power supply and a 100 Ah @12V battery for energy storage. This solution makes the system autonomous in terms of power consumption.

2.2. Camera Calibration

The main parameters specified by the manufacturer for the Pi NoIR camera V1 are the resolution of 2592 × 1944 pixels, the pixel size of 1.4 µm × 1.4 µm, and the focal length of 3.6 mm. For camera calibration, we evaluated the focal length from experimental data. The dimension of an object in a plane parallel to the camera plane was given by

$$w = \frac{P \times T \times d}{f} = \frac{\omega_s \times d}{f} \quad (1)$$

where P is the dimension of the object in pixels, T is the pixel size in mm, d is the distance between the camera lens and the plane of the object in meters, w is a dimension of the object in meters, w_s is the image dimension of the object in mm, and f is the focal length in mm. We determined the focal length considering (1) with known distances, which yielded a result very close to the value provided by the manufacturer.

We found it necessary to determine some local parameters to define the region of interest (ROI) used to measure the water level. Figure 2 shows an image taken by the camera, with a resolution of 1280 × 720. In Pi camera V1, the change in the aspect ratio to achieve this resolution corresponds to 75% of the full sensor size in the vertical dimension. For data processing, the image should include the entire vertical region of the wall and part of the water zone. From initial experiments, we found that the wall was nine meters high and had an inclination of 5.8° related to the vertical. As the water stream is between two streets and for privacy reasons, an additional 15% cut to the top of the images became necessary to avoid capturing vehicles. Then, the images were converted to the resolution of 1280 × 720. The region of interest used in the water level measurements is marked in Figure 2 by a red rectangle. The figure also includes the control points used in this work, defined by the lines and the green rectangle. The region of interest included all vertical zones of the concrete face. From measurements, we determined that the concrete was two meters high.

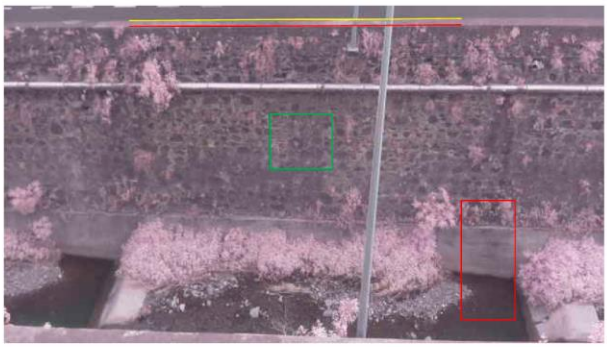


Figure 2. Image taken by the camera with the used control points (lines and green rectangle) and the ROI (red rectangle).

Measurements proved to be a difficult task due to harsh access to the water stream. Thus, we determined the distance between the camera and the stream wall by an indirect method. Figure 3 shows the reference system used in the calculations. The plane XY is the object plane and xy is the image plane. The goal was to obtain the distance d, the horizontal

angle α_x, and the vertical angle α_y to characterize the wall plane. These angles were defined in the xz and yz planes, respectively.

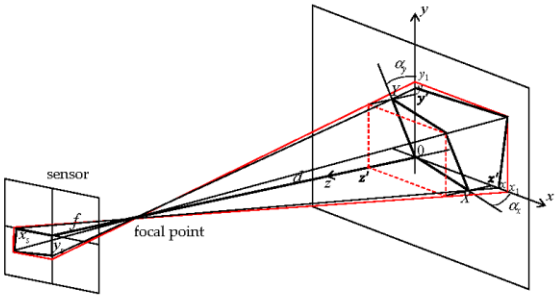


Figure 3. Reference system used to relate a point in the image plane to a point in the object plane.

The horizontal and vertical distances were obtained from (1), giving

$$x = x_s \frac{d}{f} \qquad y = y_s \frac{d}{f} \tag{2}$$

with

$$x_s = P_x \times T \times \frac{S_H}{R_H} \qquad y_s = P_y \times T \times C_1 \times C_2 \times \frac{S_V}{R_V} \tag{3}$$

where S_H is the horizontal sensor size (2592 pixels), S_V is the vertical sensor size (1944 pixels), R_H is the horizontal image resolution (1280 pixels), R_V is the vertical image resolution (720 pixels), C_1 is the image cut due to the aspect ratio change (0.75), and C_2 is the second image cut (0.85). The development of a new approach proved necessary to relate the image plane to the object plane. The goal was to simplify the parametrization required to obtain the water level, given by the distance from the stream wall to the camera, the horizontal angle between both planes, and the vertical angle between both. Using the geometric representation of Figure 3 and considering $x'/(d\text{-}z') = x/d$, the distance X can be obtained from x using the expression

$$X = \frac{xd}{d\cos(\alpha_x) + x\sin(\alpha_x)} \tag{4}$$

Similarly, Y is given by

$$Y = \frac{yd}{d\cos(\alpha_y) + y\sin(\alpha_y)} \tag{5}$$

For calibration purposes when performing actual measurements in the ROI, we placed a ten-meter graduated strip vertically on the wall in different horizontal positions. These measurements also provided values to assess the error made by the proposed technique in measuring the water level. From two distances obtained in the image in opposite directions around the origin, two vertical values y_a and y_b were defined using (2). The graduated strip made it possible to obtain the corresponding actual distances Y_a and Y_b. With (5) and these two distances, the unknowns d and α_y could be determined by solving a system of two equations. The results were d = 23.68 m and α_y = 27.2°. Substituting d into (4), α_x was determined from known values of x and X, giving 7.4°.

To relate a point in the xy plane with a point in the XY plane, we derived the equation of the XY plane using the general form $ax + by + cz + d = 0$. The constants a, b, c, and d

were obtained using three points of the plane. The points were (0,0,0), (0,y',z'), and (x',0,z'), resulting in the following plane equation in the xyz reference system:

$$z = \tan(\alpha_x)x + \tan(\alpha_y)y \tag{6}$$

For a point (X,Y) in the object plane, a point (x_1,y_1) in the image plane is given by

$$x_1 = \frac{Xd\cos(\alpha_x)}{d - X\sin(\alpha_x) - Y\sin(\alpha_y)} \tag{7}$$

$$y_1 = \frac{Yd\cos(\alpha_y)}{d - X\sin(\alpha_x) - Y\sin(\alpha_y)} \tag{8}$$

A point in the object plane can be obtained from a point in the image plane by solving these equations for X and Y.

2.3. Camera Motion Compensation

Strong wind speeds can cause small camera movements. In addition, camera position may vary over time due to its reinstallation, which results in minor changes to the calibration parameters determined in the previous section. To obtain a stable ROI, it was necessary to calculate a camera motion compensation before any measurement of the water level. The motion compensation included image rotation and translation. We considered the line defined by the upper edge of the stream wall as a set of control points to determine the camera rotation about the initial conditions. Another control point was the coordinates of a template used to compensate for the translation motion. Figure 4 shows an example of a template applied in the compensation procedure. A green rectangle represents the template shown in Figure 2. Appropriate characteristics are necessary for the template to be detectable. To generalize the procedure proposed in this work, we removed the metallic tubes observed in Figure 2 from the template options. In any case, using this type of object increases the success rate of the template matching procedure.

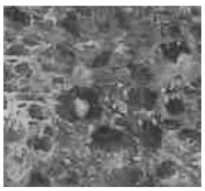


Figure 4. Template used in camera compensation.

The flowchart shown in Figure 5 describes the camera motion compensation procedure. Python software allowed for data processing with the support of the OpenCV library [35]. The images acquired by the camera were converted to grayscale. We applied the Contrast Limited Adaptive Histogram Equalization (CLAHE) method [36] to highlight the wall features. The edge detection procedure started by applying a Gaussian filter to reduce noise and applying the binarization process to convert the grayscale image into a binary image. We applied this procedure to the area around the top of the wall containing the desired edges. For edge detection, we chose the Canny method because it provides the best performance among all edge detectors [37]. The Hough transform proved to be the most effective to identify the straight lines of the edges within the area of interest.

The edge detection procedure aims to detect the yellow or red line represented in Figure 2. In many situations, it detected both edges. In this case, the compensation procedure considered was the upper edge. In some cases, such as at night, this edge was not detected, and the second edge was necessarily used. As the two lines were parallel, the

image rotation due to the camera motion was determined. Next, we applied a template matching method to detect the coordinates of the image given in Figure 4. This procedure allowed us to obtain the translation of the image caused by the camera movement. Six matching methods are available in the OpenCV library to search the template in the input image. The best results were obtained with the Normalized Correlation Coefficient Matching method. Finally, the coordinates of the template allowed for defining the ROI in the input image.

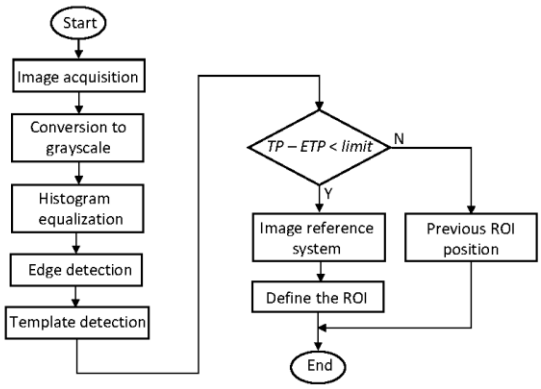


Figure 5. Flowchart for the camera movement compensation.

In most cases, the algorithm detected the correct wall edge and the correct template position. However, for images taken under very difficult lighting conditions, incorrect detection of one of these parameters or both may occur. In this case, the error made in detecting the waterline when using an incorrect ROI setting can be high. To avoid this situation, we determined the distance between the edge and the position of the template (TP) and compared it with the expected position of the template (ETP). When this difference exceeded a certain limit, we applied the last successful compensation to the ROI. As the algorithm does not know which edge it detects, the comparison represented in the flowchart served for the two values of the *limit* parameter. We determined this parameter by measuring the distance between the edge and the position of the template for various images acquired in different situations. It may also happen that the algorithm detects the edge with a small error in the slope and the template position correctly. This situation can lead to errors in setting the ROI. The application of a second template proved to be useful to minimize this effect.

2.4. Waterline Detection

The waterline detection procedure started by defining a ROI around the water boundary, as shown in Figure 2. We set this region at a certain distance from the center of the image. However, small camera movements can result in an image center different from that obtained in the calibration process, which can cause large errors in the waterline detection. Another way was to define the ROI using a reference point of the stream wall around the center. This point can be the coordinates of the template used for camera compensation. Figure 6 shows the image reference system defined to support the waterline detection. A vertical line in the object plane is seen in perspective in the image plane. We used the line within the ROI to detect the waterline position, defined by the point (x_1, y_1).

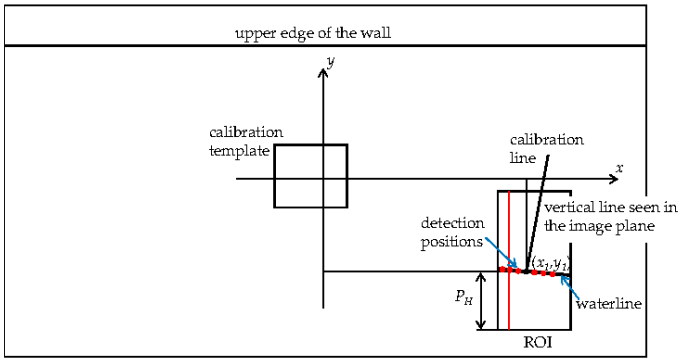


Figure 6. Schematic of the image plane.

Figure 7 shows the ROI for various images taken under different conditions. The stream flow can be characterized by shallow water most of the time, resulting in images like the one shown in Figure 7a taken during the day. Figure 7b shows a typical image taken at night. Rain events affect image quality, as shown in Figure 7c. Figure 7d is a typical situation that occurs during periods of rain, with water undulation. Another situation is the existence of debris on the water surface, as shown in Figure 7e. Figure 7f shows an example with a shadow effect within the ROI created by buildings on sunny days. As can be seen, edge detection methods are not suitable for obtaining the waterline because of the image and water quality.

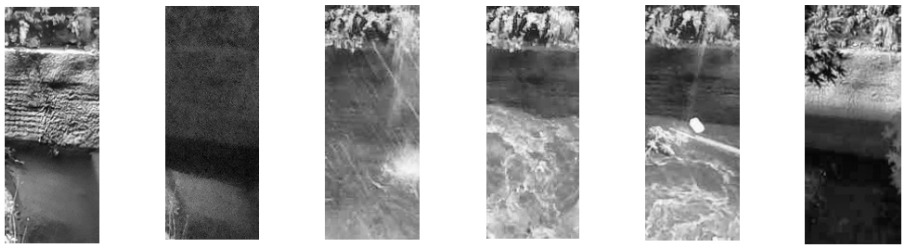


Figure 7. Different conditions of the waterline detection: (a) taken during the day; (b) taken at night; (c) image with traces of rain; (d) water undulation; (e) debris in the water; (f) shaded zone on the ROI.

To minimize some of the effects observed in the images of Figure 7, we considered T images captured with a time difference of three seconds between them to obtain an average image. By converting the image to grayscale and applying histogram equalization, it became possible to highlight the waterline. This line was determined by detecting the transition between the water and the stream wall. We also defined a reference system for the ROI to support the waterline detection procedure, where (x', y') is a point in the ROI image with the origin at the lower-left pixel. The application of a moving average filter allowed reducing noise effects on the image. For a position x', represented in Figure 6 by a red line, the grayscale profile was given by

$$P_1\left(x' + \frac{N_x - 1}{2}, y' + \frac{N_y - 1}{2}\right) = \frac{1}{N_x N_y} \sum_{n=1}^{N_x} \sum_{m=1}^{N_y} P(x', y'), 1 \leq y' \leq P_y - N_y \quad (9)$$

where $P(x',y')$ is the pixel at position (x',y'), N_x is the number of horizontal pixels, N_y is the number of vertical pixels, and P_y is the number of pixels in the vertical dimension of the ROI. In addition, we determined the gradient of the grayscale profile to detect the water boundary by the maximum absolute value of the gradient.

The problem of using a single detection position was that, in many situations, the maximum absolute value of the gradient did not match the position of the waterline. Irregularities in the wall, debris on the water surface, traces of rain captured by the camera, water undulation, and other effects can create a maximum gradient at the wrong position. Using a larger waterline zone solved this problem and improved detection. For the experimental setup, we surveyed the waterline at S equidistant positions (detection positions) in a dimension of about two meters. We added the gradients of the grayscale profiles considering the slope of the waterline to enhance its detection (Figure 6). For this, we measured the slope m of this line, giving a relationship between y' and x' of the form $\Delta y' = m\,\Delta x'$. The gradients of the grayscale profiles were determined at S positions of x'. The sum of gradients considered the slope of y' to highlight the waterline values and to minimize the effects that degrade the waterline detection. In other urban stream locations, the waterline inside the ROI may have a different shape. The applied procedure uses the detection positions defined on the curve created by the waterline. Figure 8 shows the result obtained with one detection position (Figure 8a) and ten detection positions (Figure 8b) for an image with debris on the water surface. For $S = 1$, the maximum absolute value of the gradient was drastically affected by the water quality. As shown in Figure 8b, it was possible to detect the waterline with several detection positions, despite the existence of floating debris around the sensing zone.

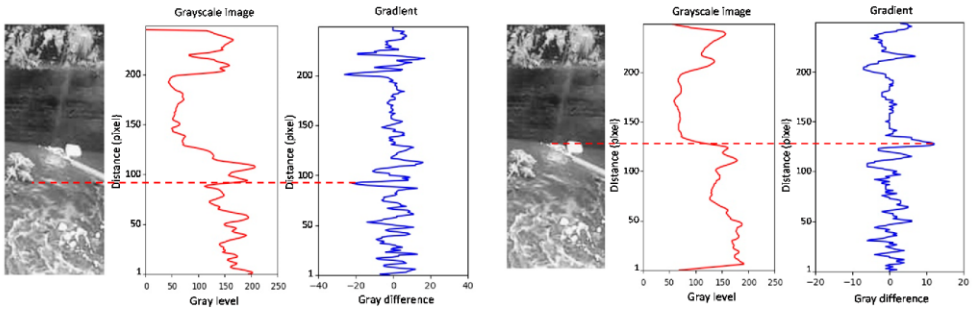


Figure 8. Waterline detection for an image with debris in the water: (**a**) $S = 1$; (**b**) $S = 10$.

For images taken at night, it became necessary to consider the ROI lighting issues. As the water stream is in an urban environment, the street lighting system may be sufficient to illuminate the area of interest. In other cases, we can employ infrared lighting. In this work, we did not install any equipment to light the water zone, minimizing the costs of installing a dedicated system that requires a power supply. With the streetlights facing to the street, shadow zones might be visible within the ROI created by the stream wall. Thus, it was necessary to distinguish the procedure for obtaining the waterline for images taken at night from those taken during the day. The knowledge that night image acquisition requires different camera parameters allowed us to distinguish the two cases. Figure 9 shows the grayscale profile and the gradient for two images taken at night. Figure 9a illustrates an image with a detectable waterline. As can be observed, the maximum absolute value of the gradient occurred on the line created by the wall shadow over the water and not on the waterline. However, this position can aid in the detection of the water level. A variation in the water level has a corresponding variation in the shadow edge. The distance between

the waterline and the shadow line was practically constant and defined by the parameter P_W measured in the vertical of the image. Figure 9b shows a case where the waterline was not detected, and the shadow line was necessary to detect the water boundary.

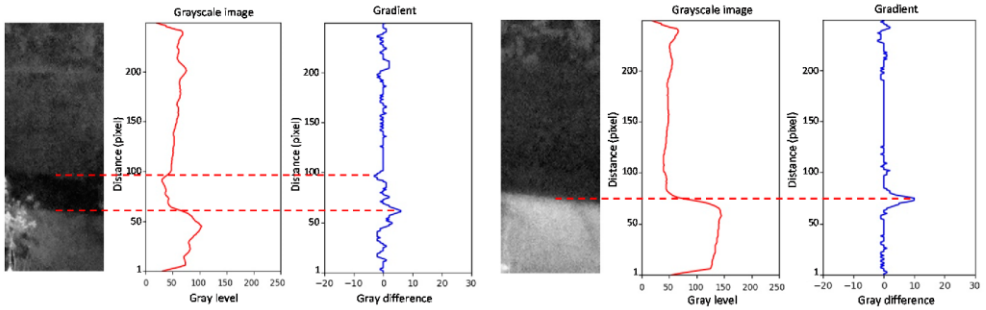


Figure 9. Waterline detection image taken at night: (**a**) the waterline is visible; (**b**) the waterline is invisible.

Figure 10 shows the flowchart of the procedure to extract the waterline. The initial operations were the acquisition of T images to obtain the average image, conversion to grayscale, and histogram equalization. We defined S positions in the horizontal dimension of the ROI to detect the waterline. The grayscale profile was determined using (9) for each of the S values of x' as well as the corresponding gradient functions. Then, we obtained the maximum absolute value by summing the gradients.

Two situations arose for images taken during the day. The first one was the case in which the maximum absolute value of the gradient corresponded to the waterline. This situation happened most of the time and the algorithm searched for this maximum in the concrete face. We determined the parameter P_H (in pixels) shown in Figure 6 from this maximum. However, for a short period during the day, shadows of buildings may appear in the ROI. In this case, the maximum absolute value of the gradient can occur in the transition between the sunlit area and the shaded area. In the flowchart of Figure 10, the "Shadow Period" defines the time interval in which this situation can happen, and we determined this period from initials measurements. To determine the waterline position, the algorithm searched for two peaks corresponding to the highest absolute values of the gradient. To assess whether the shadow of buildings affected the waterline detection procedure, we employed a technique to verify the conditions of existence of a shadow episode within the ROI. As sunlight produces a bright image in the sunlit zone, we determined, by image processing, the brightness of a small band above each of the gradient peaks and the brightness of a band below those peaks. This operation allowed us to compare the brightness of the image produced in the sunlit area with that of the shaded area. If any peak produced a difference in brightness above a threshold, this corresponded to a shadow transition. In that case, the other peak defined the position of the waterline. The threshold obtained came from measurements made on several captured images. Otherwise, we defined the waterline by the maximum absolute value of the gradient.

2.5. Water Level Estimation

To estimate the water level, we resorted to using Equations (7) and (8) to obtain X and Y, with the parameters of the wall plane of the concrete zone. We needed three parameters for this procedure, the distance d, the horizontal angle α_x, and the vertical angle α_y. Through measurements and simulation, we confirmed that the plane of the concrete zone was different from the plane of the stone zone. Following the procedure applied in the

calibration section, the concrete plane had the following parameters: d = 24.16 m, $\alpha_x = 7.4°$, and $\alpha_y = 22.2°$.

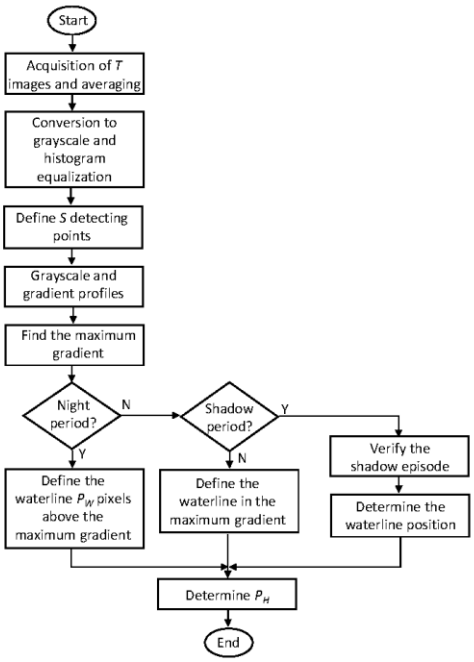


Figure 10. Flowchart for waterline detection.

With the P_H parameter determined in the previous section, we obtained the position of the waterline in the reference system represented in Figure 6. The vertical distance is $P_y = P_H - P_{y0}$, where P_{y0} is the origin of the ROI in the vertical dimension of the image. P_x was determined from the following expression:

$$P_y = m_l P_x + K \tag{10}$$

where m_l and K are the calibration line parameters. Knowing P_x and P_y, x_1 and y_1 were calculated, respectively. Considering Equations (7) and (8), the distances in the wall plane are given by

$$X = \frac{dx_1}{x_1 \sin(\alpha_x) + y_1 \cos(\alpha_x) \tan(\alpha_y) + d\cos(\alpha_x)} \tag{11}$$

$$Y = \frac{y_1 X \cos(\alpha_x)}{x_1 \cos(\alpha_y)} \tag{12}$$

Finally, we determined the water level through the difference between Y and Y_0, where Y_0 is the distance considered for the zero-water level.

For some images, it was very difficult or impossible to measure the water level. In such cases, large errors could occur. We used data filtering to minimize the error effects. One procedure was to remove values that exceed a certain threshold when compared to previous results. To support this decision, we noticed that water flow could increase

suddenly but decreases more slowly. The second peak of the gradient can also be applied to replace the wrong value if it does not exceed the defined threshold. The reason was the high probability of the waterline being there.

3. Results and Discussion

Images captured over several months allowed us to evaluate the developed technique. The stream water had very low levels for long periods, especially in summer. There was a noticeable variation in the water level only during episodes of rainfall in the surrounding mountains.

Figure 11a,b show the results for images taken every two minutes on 3 April 2020. The rain event started during the day. The images were not affected by traces of rain because the precipitation occurred on the mountain. The camera motion compensation procedure allowed a correct configuration of the ROI, meaning a correct detection of the wall edge and the template. To evaluate the water undulation magnitude, Figure 11a shows the case for the water level measurement with a single image and $S = 10$ positions. Equation (9) defined $N_x = N_y = 5$. The graph shows the comparison between the values estimated by the proposed technique and the values measured manually from the images and the graduated strip. The average water undulation was 5.7 cm and the maximum undulation was 15.3 cm. The accuracy of the water level estimation was 0.9 cm. There was a lot of debris floating around the waterline between 5:30 pm and 7:10 pm. Temporal and spatial averages enabled us to minimize its effects in the estimation of the water level. Figure 11b shows the results with the average image determined from $T = 5$ images. The accuracy of the water level estimation was 0.8 cm for water undulation of 1.8 cm. This accuracy was possible because the technique combines information from different positions of the waterline to increase the success rate.

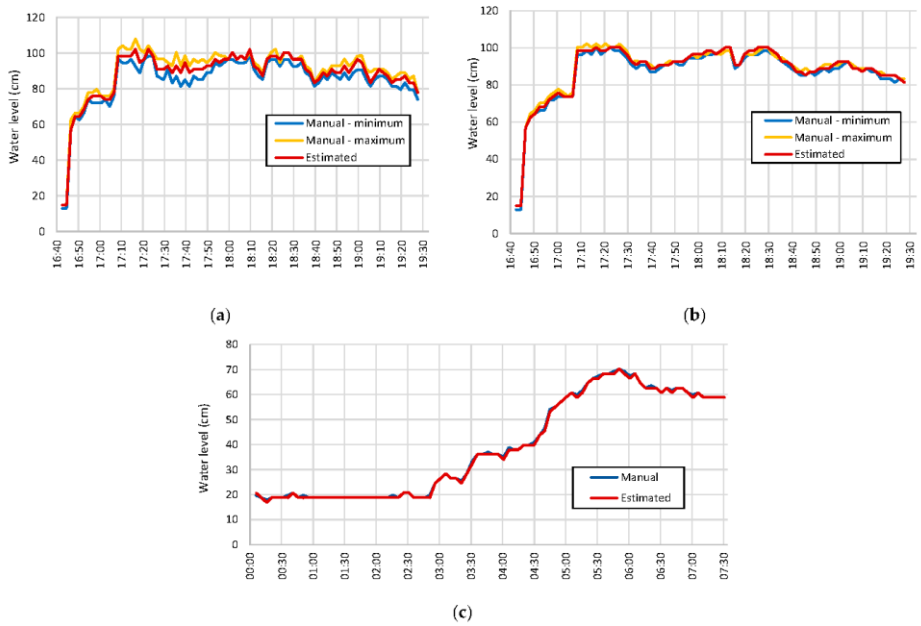


Figure 11. Water level estimation: (**a**) images taken during the day with $T = 1$; (**b**) image taken during the day with $T = 5$; (**c**) images taken at night.

For the night period, we determined the parameter P_W before estimating the water level. To obtain this parameter, we measured the shadow width for various water levels. The result was $P_W = 32 \pm 2$ pixels obtained with data from twenty cases. The accuracy of the water level estimation by this process was 2.2 cm. The technique applied to images acquired at night for the same day resulted in an accuracy of 2.6 cm. The error obtained for this period was higher because of the indirect estimation of the water level. Figure 11c shows the results for the case of images acquired every 5 min on 20 February 2021, with a rain event starting at night. The graph considers images acquired every 5 min. In this case, manual values correspond to results obtained by the shadow line. The accuracy of the water level estimation was 2.4 cm.

Figure 12a shows the results obtained on 17 April 2020. The water underwent a small increase in level on a sunny day for the period between 1:20 pm and 1:40 pm and then slowly decreased. We may observe an error of 3 to 4 cm during this period because of the humidity on the wall, which makes the water level decision be slightly higher than the correct one. After 6:45 pm, the rain reached the urban area and the flow increased drastically. The accuracy of the estimate was 1.2 cm for the daytime and 2.8 cm for the nighttime. Another test was to evaluate the technique in periods with heavy rain, producing rain traces on acquired images. Figure 12b shows the results of an episode of this type on 27 March 2021. The accuracy had similar values to those obtained previously. Evaluating the results of two heavy rain events that occurred in December 2020 during the day, the accuracy was 1.8 cm. Rain events occurring at night produced an accuracy of 2.8 cm, being less affected than during the day as the image quality was more stable. Evaluating several heavy rain events, the accuracy obtained during short periods was about 2.6 cm for images taken during the day and 3.4 cm for images taken at night.

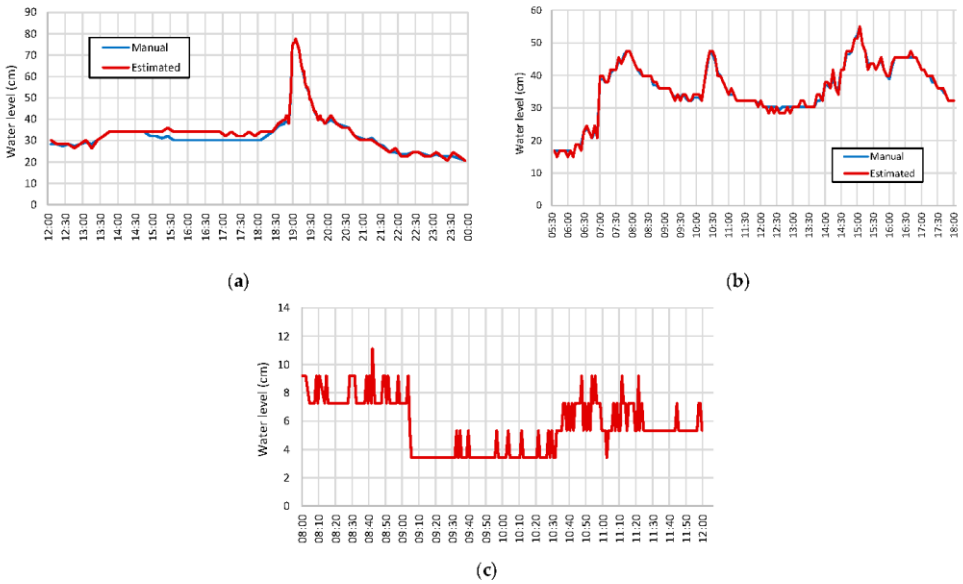


Figure 12. Water level estimation for images for different situations: (**a**) day with a change in the weather conditions; (**b**) day with heavy rain episodes; (**c**) shadow episode on the ROI.

Another situation that required evaluation was the occurrence of shadows of buildings within the ROI. In the flowchart of Figure 10, we refer to this episode as the Shadow period.

For the site of the experimental setup, this period occurred in the morning and lasted about 15 min. Considering data collected over a year, the determined shadow period was between 8:10 am and 9:10 am, because the shadow created by the buildings varied with the movement of the sun. It is also worth noting that the shadow episodes coincided with periods of shallow water. Figure 12c shows results for 10 June 2021 with a shadow episode occurring between 8:50 and 9:05 a.m. The water level was 5.3 cm in the time interval between 8:00 and 12:00. Before 9:05 a.m., larger errors occurred because the waterline area became too dark in the shaded zone. Due to the movement of the sun, different image qualities resulted in variations in the estimated water level. The accuracy obtained for these results was 1.6 cm. Figure 13 shows the results obtained for a longer period of observation. Images acquired between 3 and 5 January 2021 aided in estimating the water level. There was an episode of heavy rain starting at 9:00 a.m. on day 4 and some water level fluctuations during the night of day 5 originated by the rain fall in the surrounding mountains.

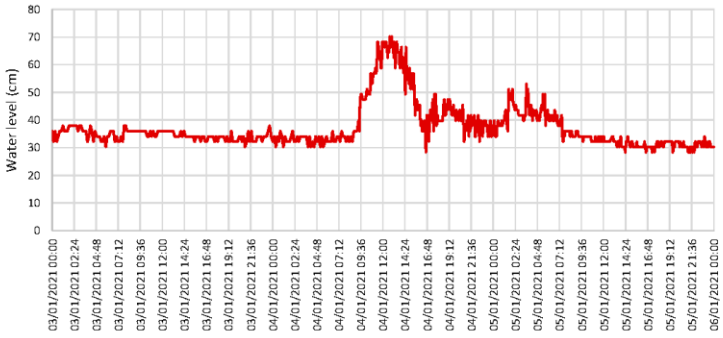


Figure 13. Water level estimation for a longer period of observation.

According to the results, the accuracy of the proposed method for obtaining the water level was 1.8 cm for daytime images and 2.8 for nighttime images. The accuracy at night was lower but varied less. Various methods show accuracies of about 1 cm [19–24]. However, these methods use a staff gauge to create a strong contrast between the water surface and the stream wall, as well as to serve as a reference for measurements. Furthermore, in some cases, they used dedicated lighting at nighttime. For the water stream in question, the use of a staff gauge turned out to be impossible. The luminosity provided by streetlights helped in the image acquisition at night, which reduced system installation costs. This solution produced a larger detection error when compared with the accuracy achieved with images taken during the day. However, improving accuracy is possible via the installation of infrared lighting on the wall in front of the ROI. This solution would remove the error produced by indirect detection. To summarize, Table 1 shows the accuracy obtained based on the experimental data for different water stream situations.

Some methods also remove images taken during adverse weather conditions, such as those with low contrast and brightness, and acquired in periods of heavy rain [4]. In the proposed technique, all taken images to detect the water level and filtering eliminated random errors. Even with low image contrast, the waterline remained detectable. An average water undulation of 5.7 cm and debris on the water surface resulted in an accuracy of 0.9 cm. Existing methods also do not allow measuring the water level properly in the presence of debris on the water around the staff gauge. The proposed technique considers several detection positions to overcome this problem. There were at least ten detection positions over a width of about two meters to estimate the waterline. This procedure is compatible with the use of staff gauges. Effectively, a way to improve accuracy in the

presence of objects obstructing the waterline could be through multiple staff gauges with a specified distance between them. Afterwards, we can apply the proposed technique.

Table 1. Accuracy as a function of the water stream situation.

	Situation	Accuracy (cm)
Daytime	Water undulation of 1.8 cm and debris in the water	0.8
	Water undulation of 5.7 cm and debris in the water	0.9
	Shallow water, sunny day	1.6
	Rain	1.3
	Heavy rain	2.6
	Average for the observation period	1.8
Nighttime	Water undulation of 1.8 cm and debris in the water	2.6
	Rain	2.7
	Heavy rain	3.4
	Average for the observation period	2.8

4. Conclusions

We developed a monitoring system to measure the water level in urban stream channels. Stone or concrete walls typically surround these channels to contain the water. The water is at a low level most of the time, but it can change quickly with rain events. These degrade water quality, cause water undulation, give rise to debris floating on the water surface, create traces of rain in the acquired images, and so on. The developed system used a Raspberry Pi and a Pi NoIR camera to operate day and night. Installation took place on a building facing the water stream wall. This installation allowed for attaining a low cost and low environmental impact solution for monitoring the water level. Placing the camera under the balcony avoided image saturation due to direct sunlight. The location also reduced the wind and rain effect on the camera. Wall features provided natural control points for camera calibration and reference for measurements. Compared with other methods, a larger detection zone allowed for minimizing the effects that make it difficult to detect the waterline. Using various image quality situations, we established the accuracy of the water level. We determined the accuracy from experimental data for different water stream situations influenced by weather conditions. Accuracies ranged from 0.8 cm to 2.6 cm for daytime and from 2.6 to 3.4 cm for nighttime. Average accuracies of 1.8 cm for the day and 2.8 cm for the night were determined by averaging the results obtained from various data periods. Although the results may be lower than those provided by some works, we did not use a staff gauge and the technique can be applied in periods where other methods tend to fail. Future work is needed to improve the water level detection in cases where a large area of the waterline remains obstructed by vegetation, mainly when it is possible by inspection to detect some points of the water boundary.

Author Contributions: Conceptualization, J.A.A.; methodology, J.A.A.; software, J.A.B.; validation, J.A.A. and J.A.B.; formal analysis, J.A.A.; investigation, J.A.A.; resources, J.A.A.; data curation, J.A.A.; writing—original draft preparation, J.A.A.; writing—review and editing, J.A.A. and J.A.B.; supervision, J.A.A.; project administration, J.A.A.; funding acquisition, J.A.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research was co-financed by the program INTERREG V-A, Spain-Portugal, MAC 2014–2020, VIMETRI-MAC "Sistema de vigilancia meteorológica para el seguimiento de riesgos medio ambientales (Meteorological monitoring system for tracking environmental risks)", grant number VIMETRI-MAC/3.5b/065. This research has been partially supported by Center for Research in Mathematics and Applications (CIMA) related with the Statistics, Stochastic Processes and Applications (SSPA) group, through the grant UIDB/04674/2020 of FCT–Fundação para a Ciência e a Tecnologia, Portugal.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The authors wish to thank to Filipe Santos for his support to this work.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Chen, Y.; Chiu, C. An efficient method of discharge measurement in tidal streams. *J. Hydrol.* **2002**, *265*, 212–224. [CrossRef]
2. Hapuarachchi, H.A.P.; Wang, Q.J.; Pagano, T.C. A review of advances in flash flood forecasting. *Hydrol. Process.* **2011**, *25*, 2771–2784. [CrossRef]
3. Chen, Y. Flood discharge measurement of a mountain river—Nanshih River in Taiwan. *Hydrol. Earth Syst. Sci.* **2013**, *17*, 1951–1962. [CrossRef]
4. Lo, S.; Wu, J.; Lin, F.; Hsu, C. Visual Sensing for Urban Flood Monitoring. *Sensors* **2015**, *15*, 20006–20029. [CrossRef]
5. Fujita, I. Discharge Measurements of Snowmelt Flood by Space-Time Image Velocimetry during the Night Using Far-Infrared Camera. *Water* **2017**, *9*, 269. [CrossRef]
6. Bradley, A.A.; Kruger, A.; Meselhe, E.A.; Muste, M.V.I. Flow measurement in streams using video imagery. *Water Resour. Res.* **2002**, *38*, 51. [CrossRef]
7. Yorke, T.H.; Oberg, K.A. Measuring river velocity and discharge with acoustic Doppler profilers. *Flow Meas. Instrum.* **2002**, *13*, 191–195. [CrossRef]
8. Yu, J.; Hahn, H. Remote Detection and Monitoring of a Water Level Using Narrow Band Channel. *J. Inf. Sci. Eng.* **2010**, *26*, 71–82. [CrossRef]
9. Alsdorf, D.E.; Rodríguez, E.; Lettenmaier, D.P. Measuring surface water from space. *Rev. Geophys.* **2007**, *45*, RG2002. [CrossRef]
10. Gleason, C.J.; Smith, L.C.; Finnegan, D.C.; LeWinter, A.L.; Pitcher, L.H.; Chu, V.W. Technical Note: Semi-automated effective width extraction from time-lapse RGB imagery of a remote, braided Greenlandic river. *Hydrol. Earth Syst. Sci.* **2015**, *19*, 2963–2969. [CrossRef]
11. Tsubaki, R.; Fujita, I.; Tsutsumi, S. Measurement of the flood discharge of a small-sized river using an existing digital video recording system. *J. Hydro-Environ. Res.* **2011**, *5*, 313–321. [CrossRef]
12. Yang, H.; Wang, C.; Yang, J. Applying image recording and identification for measuring water stages to prevent flood hazards. *Nat. Hazards* **2014**, *74*, 737–754. [CrossRef]
13. Zhen, Z.; Yang, Z.; Yuchou, L.; Youjie, Y.; Xurui, L. IP camera-based LSPIV system for on-line monitoring of river flow. In Proceedings of the 2017 IEEE 13th International Conference on Electronic Measurement & Instruments, Guangzhou, China, 20–22 October 2017; pp. 357–363. [CrossRef]
14. Tauro, F.; Porfiri, M.; Grimaldi, S. Surface flow measurements from drones. *J. Hydrol.* **2016**, *540*, 240–245. [CrossRef]
15. Langhammer, J.; Bernsteinová, J.; Mirijovský, J. Building a High-Precision 2D Hydrodynamic Flood Model Using UAV Photogrammetry and Sensor Network Monitoring. *Water* **2017**, *9*, 861. [CrossRef]
16. Bandini, F.; Jakobsen, J.; Olesen, D.; Reyna-Gutierrez, J.A.; Bauer-Gottwein, P. Measuring water level in rivers and lakes from lightweight Unmanned Aerial Vehicles. *J. Hydrol.* **2017**, *548*, 237–250. [CrossRef]
17. Hies, T.; Babu, P.S.; Wang, Y.; Duester, R.; Eikaas, H.S.; Meng, T.K. Enhanced water-level detection by image processing. In Proceedings of the 10th International Conference on Hydroinformatics, Hamburg, Germany, 14–18 July 2012.
18. Duda, R.O.; Hart, P.E. Use of the Hough transformation to detect lines and curves in pictures. *Commun. ACM* **1972**, *15*, 11–15. [CrossRef]
19. Lin, Y.; Lin, Y.; Han, J. Automatic water-level detection using single-camera images with varied poses. *Measurement* **2018**, *127*, 167–174. [CrossRef]
20. Zhang, Z.; Zhou, Y.; Liu, H.; Gao, H. In-situ water level measurement using NIR-imaging video camera. *Flow Meas. Instrum.* **2019**, *67*, 95–106. [CrossRef]
21. Zhang, Z.; Zhou, Y.; Liu, H.; Zhang, L.; Wang, H. Visual Measurement of Water Level under Complex Illumination Conditions. *Sensors* **2019**, *19*, 4141. [CrossRef]
22. Xu, Z.; Feng, J.; Zhang, Z.; Duan, C. Water Level Estimation Based on Image of Staff Gauge in Smart City. In Proceedings of the 2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovations, Guangzhou, China, 8–12 October 2018; pp. 1341–1345. [CrossRef]
23. Guo, S.; Zhang, Y.; Liu, Y. A Water-Level Measurement Method Using Sparse Representation. *Autom. Control Comput. Sci.* **2020**, *54*, 302–312. [CrossRef]
24. Chen, G.; Bai, K.; Lin, Z.; Liao, X.; Liu, S.; Lin, Z.; Zhang, Q.; Jia, X. Method on water level ruler reading recognition based on image processing. *Signal Image Video Process.* **2021**, *54*, 33–41. [CrossRef]
25. Udomsiri, S.; Iwahashi, M. Design of FIR Filter for Water Level Detection. *World Acad. Sci. Eng. Technol.* **2008**, *24*, 47–52. [CrossRef]
26. Griesbaum, L.; Marx, S.; Höfle, B. Direct local building inundation depth determination in 3-D point clouds generated from user-generated flood images. *Nat. Hazards Earth Syst. Sci.* **2017**, *17*, 1191–1201. [CrossRef]

27. Ridolfi, E.; Manciola, P. Water Level Measurements from Drones: A Pilot Case Study at a Dam Site. *Water* **2018**, *10*, 297. [CrossRef]
28. Canny, J. A Computational Approach to Edge Detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **1987**, *11*, 184–203.
29. Young, D.S.; Hart, J.K.; Martinez, K. Image analysis techniques to estimate river discharge using time-lapse cameras in remote locations. *Comput. Geosci.* **2015**, *76*, 1–10. [CrossRef]
30. Leduc, P.; Ashmore, P.; Sjogren, D. Technical note: Stage and water width measurement of a mountain stream using a simple time-lapse camera. *Hydrol. Earth Syst. Sci.* **2018**, *22*, 1–11. [CrossRef]
31. Eltner, A.; Elias, M.; Sardemann, H.; Spieler, D. Automatic Image-Based Water Stage Measurement for Long-Term Observations in Ungauged Catchments. *Water Resour. Res.* **2018**, *54*, 10362–10371. [CrossRef]
32. Quintal, R. Aluviões da Madeira—Séculos XIX e XX. *Territorium* **1999**, *6*, 6–28. [CrossRef]
33. Oliveira, R.P.; Almeida, A.B.; Sousa, J.; Pereira, M.J.; Portela, M.M.; Coutinho, M.A.; Ferreira, R.; Lopes, S. Avaliação do Risco de Aluviões na Ilha da Madeira. In Proceedings of the 10° Simpósio de Hidráulica e Recursos Hídricos dos Países de Língua Oficial Portuguesa (SILUSBA), Pernambuco, Brasil, 26–29 September 2011.
34. Raspberry Pi. Available online: https://www.raspberrypi.org (accessed on 7 June 2021).
35. Bradski, G. The OpenCV library. *Dobb's J. Softw. Tools Prof. Program.* **2000**, *25*, 122–125.
36. Pizer, M.S.; Amburn, E.P.; Austin, J.D.; Cromartie, R.; Geselowitz, A.; Greer, T.; Romeny, B.H.; Zimmerman, J.B.; Zuiderveld, K. Adaptive histogram equalization and its variations. *Comput. Vis. Graph. Image Process.* **1987**, *39*, 355–368. [CrossRef]
37. Batra, B.; Singh, S.; Sharma, J.; Arora, S.M. Computational analysis of edge detection operators. *Int. J. Appl. Res.* **2016**, *2*, 257–262.