Engenharia Eletrotécnica e de Computadores

Sistema de Visão Artificial para Identificação de Sinais de Trânsito

Robótica

09-12-2022

Instituto Politécnico do Cávado e do Ave

João Sampaio 18611
Rui Ribeiro 16419
Tiago Carvalho 18601

INSTITUTO POLITÉCNICO
DO CÁVADO E DO AVE
ESCOLA SUPERIOR DE TECNOLOGIA

Resumo

Este relatório tem por objetivo a avaliação da componente prática de Robótica, relativamente ao processamento de imagem. Este subtema da unidade curricular, serve para identificar objetos por exemplo, através da captura de imagens em tempo real e identificação do tipo de objeto, no caso sinais de trânsito.

Foi utilizada uma webcam que captura a imagem, e a partir dessa imagem, não ideal, é feito o processamento, sendo necessário aplicar técnicas do melhoramento da imagem de forma a ignorar tudo o que não interessa para a identificação dos sinais (ruído).

Como resultado, foi possível distinguir todos os sinais corretamente, e perceber todo o processo que é necessário executar, desde o melhoramento da imagem até á caracterização do sinal.

Índice

Introdução 6
1. Processamento de Imagem 7
1.1. Função regionprops 9
1.2. Função imclose 10
2. Desenvolvimento do projeto 12
2.1. Segmentação por Thresholding 12
2.2. Captura da imagem 13
2.3. Aplicação do threshold na imagem 14
2.4. Caraterísticas do sinal 18
2.5. Distinção dos sinais 19
2.5.1. Sinais Amarelos 21
2.5.2. Sinais Azuis 23
2.5.3. Sinais Vermelhos 25
Conclusão 27
Bibliografia 29

Índice de Figuras

Figura 1 - Sinais de trânsito utilizados 6
Figura 2 – Diagrama exemplo do processamento de uma imagem 7
Figura 3 – Espaço de cor RGB e suas componentes 8
Figura 4 - Espaço de cor HSV e suas componentes 8
Figura 5 – Processo de dilatação 10
Figura 7 – Processo de fecho 11
Figura 6 - Processo de erosão 11
Figura 8 – Realização da segmentação de um sinal utilizando a ferramenta Color Thresholder 12
Figura 9 – Ruído detetado em qualquer threshold 14
Figura 10 – Sinal detetado no threshold com ruído de ambiente 15
Figura 11 – Sinal detetado sem ruído de ambiente 16
Figura 12 – Na direita temos a imagem binária após threshold, e na esquerda foi aplicado um fecho 16
Figura 13 – Identificação da forma 18
Figura 14 – Resultado da multiplicação da imagem RGB com a binária 20
Figura 15 – Imagem multiplicada do sinal de semáforo 21
Figura 16 – Imagem após a aplicação do threshold vermelho / objeto detetado 22
Figura 17 – Divisão do objeto no seu centroide 23
Figura 18 – Threshold preto para detetar o objeto dentro do sinal 25
Figura 19 - Divisão do objeto no seu centroide 26

Introdução

No âmbito da unidade curricular de Robótica do curso de Engenharia Eletrotécnica e de Computadores, da Escola Superior de Tecnologia IPCA, foi proposta a realização de um trabalho prático de modo a explorar técnicas de processamento de imagem.

O trabalho prático tem como objetivo colocar em prática os conhecimentos adquiridos ao longo da unidade curricular, desenvolvendo também a programação em Matlab.

O trabalho consiste na obtenção de imagens RGB capturadas utilizando uma webcam, que posteriormente serão processadas através de processos de segmentação por threshold de cor, operadores morfológicos, remoção de ruído e identificação de objetos/blobs. Todo este processo é realizado em tempo real.

As imagens capturadas e processadas são de sinais de trânsito, que posteriormente, recorrendo a todas às técnicas supracitadas, serão distinguidos entre eles através da sua cor , forma e símbolo.

O presente relatório irá clarificar todo o processo de desenvolvimento realizado pelo grupo de trabalho.

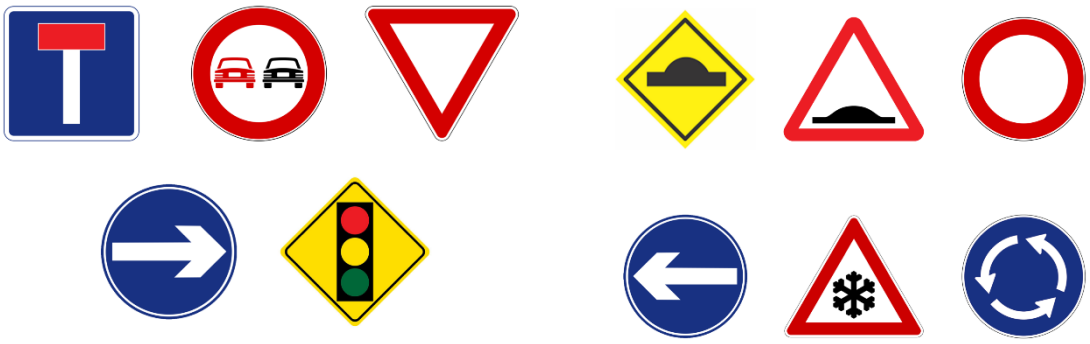


Figura 1 - Sinais de trânsito utilizados

1. Processamento de Imagem

O processamento de imagem pode ser entendido como um processo de manipulação e análise de informação visual. Assim qualquer operação que melhore a qualidade, corrija, analise ou altere uma imagem pode ser considerada processamento de imagem.

No caso de um sistema de visão por computador, este é constituído por hardware e software, onde em conjunto efetuam operações de aquisição armazenamento, processamento e visualização de imagem.

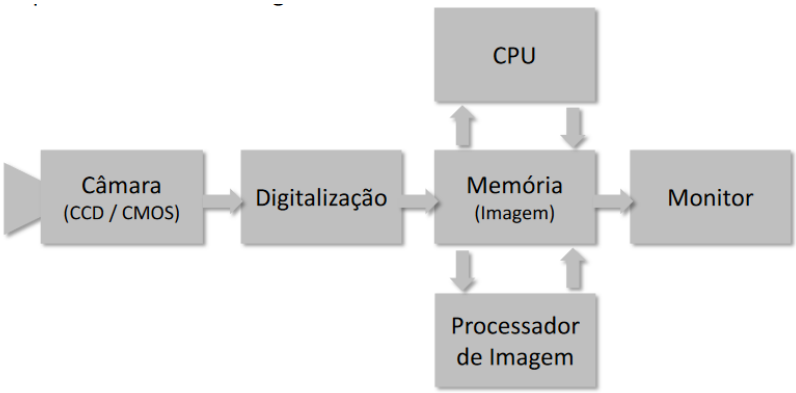


Figura 2 – Diagrama exemplo do processamento de uma imagem

A imagem pode ser de três tipos:

- Binária, onde existem apenas dois níveis de cor, o preto e o branco.
- Escalas de cinzento, onde existem mais de dois níveis de cor, mas apenas um canal de cor.
- A cores, onde existe mais de um canal por cor.

A análise da cor é normalmente efetuada recorrendo ao espaço de cor RGB, ou transformações deste, por exemplo o HSV.

Um espaço de cor é um método pelo qual se torna possível especificar, criar ou visualizar uma cor, esta cor é usualmente especificada utilizando três parâmetros/coordenadas. Exemplos de espaços de cor podem ser RGB (Red, Geen e Blue), HSV (Hue, Saturation, Value), entre outros.

O espaço de cor RGB é constituído por 3 componentes, o vermelho, verde e azul, normalmente uma imagem digital a cores é representada por estes 3 componentes sendo que para cada canal de cor é codificada uma matriz individual com o valor da cor em questão, num determinado pixel.

O espaço HSV, é constituído por 3 componentes, a tonalidade, saturação e o valor, sendo que este espaço proporciona um método intuitivo de especificar a cor. Neste método é possível selecionar a tonalidade, e realizar ajustes na saturação e intensidade. A separação entre a luz e a cor, traz vantagens relativamente ao espaço RGB quando se pretende realizar operações sobre cores.

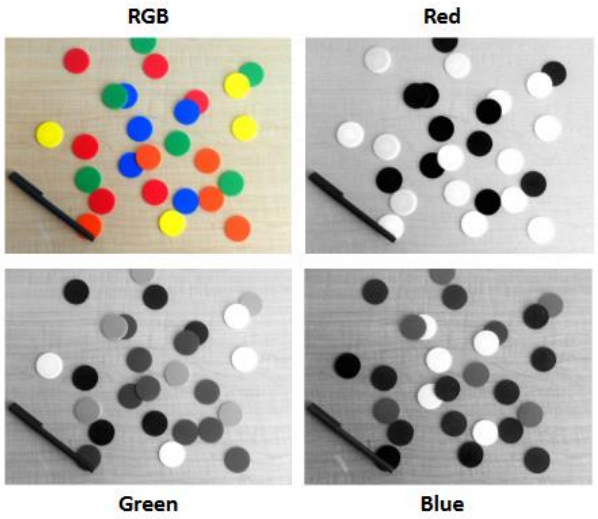


Figura 3 – Espaço de cor RGB e suas componentes

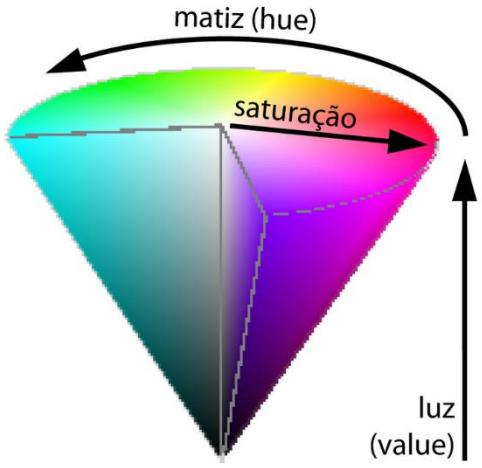


Figura 4 - Espaço de cor HSV e suas componentes

1.1. Função regionprops

A função regionprops() é utilizada para realizar a medição das propriedades de uma imagem.

$$\text{stats} = \text{regionprops}(\text{BW}, \text{properties})$$

Como parâmetros de entrada é necessário enviar uma imagem binária (preto e branco ou 0 e 1), representada por BW no exemplo anterior, e as propriedades que queremos obter valores, como por exemplo:

- Área – corresponde á área para cada objeto identificado na imagem em 2D;
- Centroide- centro de massa da região avaliar, que será o sinal de trânsito;
- Comprimento do eixo principal – comprimento do eixo principal com início no ponto centroide;
- Comprimento do eixo secundário – comprimento do eixo secundário com início no ponto centroide;
- Circularidade – qualidade de aquilo que é circular. Percorre todos os objetos detetados e indica se estes apresentam uma forma circular;
- Lista de Id de pixéis – cria uma lista de todos os objetos. Esses objetos são definidos pela sua conectividade de pixéis.
- Entre outros, que não foram utilizados no desenvolvimento deste projeto.

Estes valores que serão retornados da função vão ser aplicados em diferentes fórmulas de maneira a distinguir formas e remoção de ruído. Todos esses aspetos serão explicados e exemplificados no presente relatório.

1.2. Função imclose

A técnica realiza dois processos distintos, inicialmente um processo de dilatação e outro de erosão, e é basicamente utilizada para preencher falhas dentro de regiões de primeiro plano, minimizando as alterações nas restantes regiões.

Primeiro a função passa pela dilatação que consiste em adicionar pixéis aos limites da região segmentada, fazendo aumentar a sua área e preenchendo as zonas do seu interior, ou seja, pensando numa matriz de 3x3, se a direita/esquerda ou acima/baixo do centro houver um pixel da imagem dentro da zona estruturante(matriz),o centro passa a colocar esse pixel como sendo do primeiro plano. Segue o exemplo.

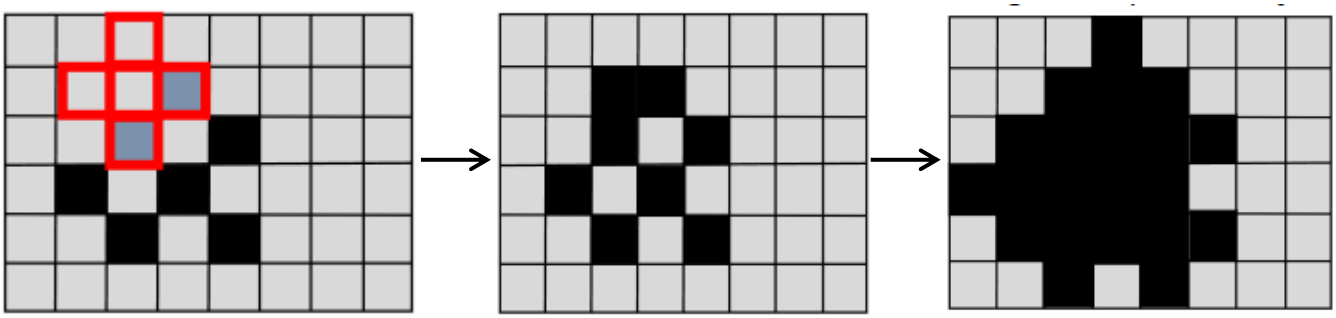


Figura 5 – Processo de dilatação

Em seguida a função procede a realizar uma erosão, que consiste em remover pixéis aos limites de uma região segmentada, diminuindo a área e eliminando as regiões cuja dimensão seja inferior a do elemento estruturante, ou seja, considerando o mesmo elemento estruturante da dilatação, desta vez, se pelo menos um pixel que não pertença ao primeiro plano se incluir dentro do elemento estruturante, então o pixel central será eliminado, como se pode observar pelas seguintes imagens .

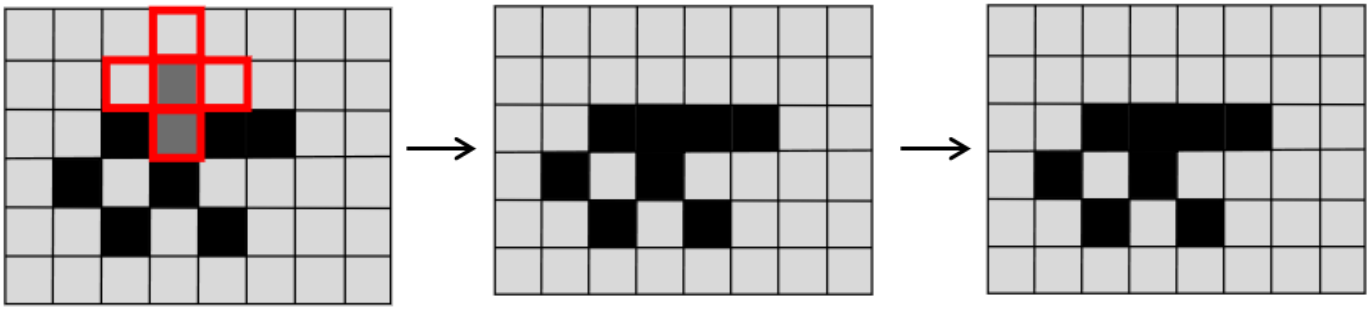


Figura 7 - Processo de erosão

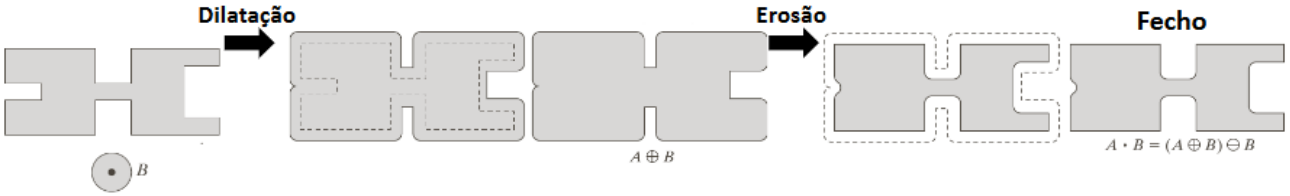


Figura 6 – Processo de fecho

2. Desenvolvimento do projeto

2.1. Segmentação por Thresholding

De forma a realizar a segmentação por thresholding das imagens correspondentes aos sinais, recorremos a uma ferramenta/aplicação disponibilizada pelo Matlab, de nome Color Thresholder. O código é criado automaticamente e exportado para uma função.

O espaço de cor escolhido foi o espaço HSV, pois como explicado anteriormente, este espaço permite realizar a separação entre a luz e a cor, aspeto crucial para a realização da segmentação neste projeto.

A ferramenta de threshold irá automatizar e facilitar processo, em relação ao processo manual utilizando a função roicolor(), atribuindo os valores de threshold mínimo e máximo.

Com a ferramenta é possível realizar as alterações necessárias nos valores de Hue, Saturation e Value em tempo real, apresentando do lado esquerdo uma amostra da segmentação.

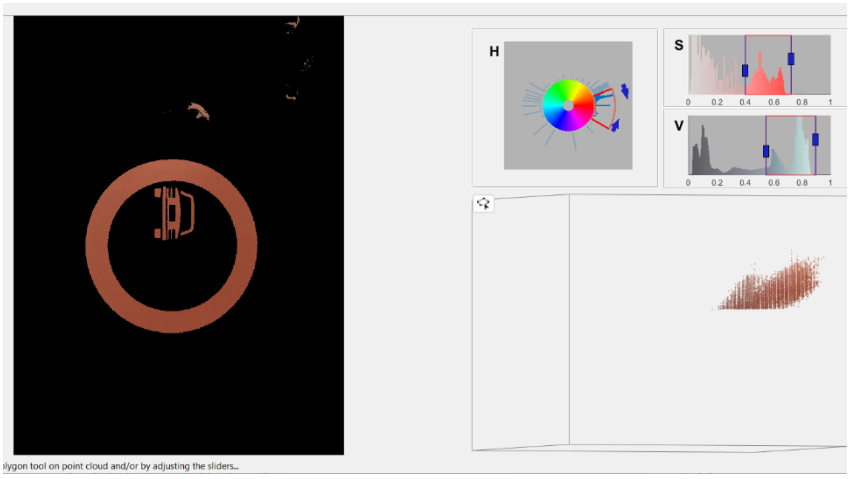


Figura 8 – Realização da segmentação de um sinal utilizando a ferramenta Color Thresholder

De forma a melhorar ao máximo a obtenção da segmentação, tiramos uma foto de cada sinal recorrendo á câmara utilizada no projeto (câmara do computador), permitindo assim obter uma imagem semelhante ao que irá ser capturada durante a execução do programa.

2.2. Captura da imagem

A função de captura de imagem irá possibilitar tirar uma foto ao sinal, que por sua vez será processada e o sinal identificado.

Inicialmente existe uma limpeza de todas as entradas de vídeo identificadas, de forma a libertar a câmara para poder novamente utilizá-la ao correr novamente o programa.

De seguida identificamos qual câmara utilizar, na eventualidade de existirem mais, qual o sistema operativo que está a ser utilizado e o tipo de imagem que será capturada, juntamente com a resolução.

Para realizar uma captura de imagem contínua é necessário definir um frame-rate (quantidade de frames por segundo), realizar uma captura continua para permitir a deteção em tempo real e definir o tipo de imagem que irá ser criado, neste caso RGB.

Para realizar uma captura em tempo real, foi criado um loop infinito que nunca será quebrado durante a execução do programa.

Dentro dessa função um temporizador é criado que irá executar a função processamento durante a execução do timer (BusyMode).

2.3. Aplicação do threshold na imagem

De forma a realizar a diferenciação entre sinais, procedemos inicialmente a uma separação pelas diferentes cores.

Para o efeito utilizamos uma máquina de estados que irá aplicar os thresholds para as cores amarelo, azul e vermelho nessa respetiva sequência.

A deteção do sinal apenas acontecerá no seu threshold específico, no entanto, visto que se trata de uma sequência, certos sinais serão inicialmente expostos a diferentes thresholds, que irão detetar sempre algum ruído do ambiente em que o projeto está inserido.

A imagem seguinte representa a tentativa de identificação de um sinal azul na máscara amarela. Como esperado não foi detetado qualquer sinal, no entanto foi detetado algum ruído.

Figura 9 – Ruído detetado em qualquer threshold

De forma a ignorar esse ruído (visto que não é o sinal), recorremos á função regionprops () que irá transformar todos os blobs/ruído indesejado em objetos que possuem características, como por exemplo a área.

As áreas de todos os objetos na figura serão registadas num vetor de tamanho igual há quantidade de objetos encontrados, por uma coluna. A maior área (supostamente o sinal de trânsito) será registada noutro vetor, incluindo a sua posição no primeiro vetor mencionado.

Após essa informação estar devidamente guardada nos vetores o programa procede á comparação entre eles, realizando uma eliminação de todos os objetos identificados, não incluindo o objeto correspondente ao suposto sinal de trânsito (maior área).

Para o programa ser capaz de distinguir se um sinal foi encontrado ou não, é necessária a criação de uma comparação conforme o seu valor de área.

A comparação tem um valor central, e para obter esse valor foram realizados vários testes, aproximando ou afastando o sinal da câmara, de modo a registar os valores de área. Esse será sempre superado pela área de um sinal de trânsito e nunca superado pela área do ruído que possa existir no ambiente de trabalho.

A imagem seguinte exemplifica a deteção de um sinal azul no seu respetivo threshold. Verificamos que o sinal foi encontrado e que existe ruído de ambiente.

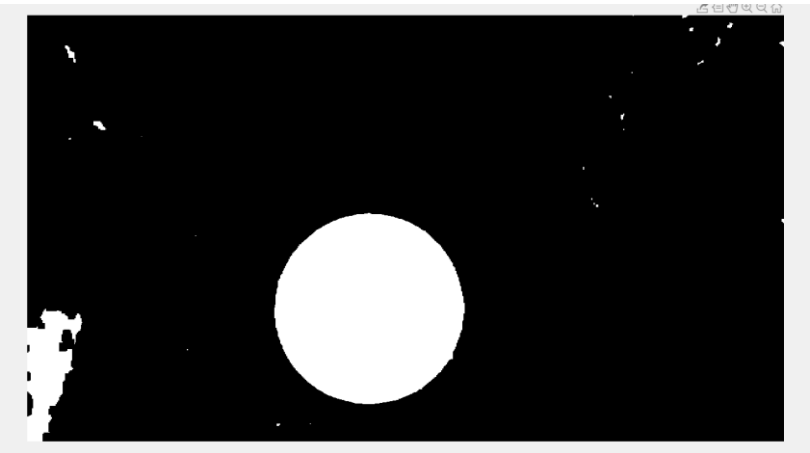


Figura 10 – Sinal detetado no threshold com ruído de ambiente

A imagem seguinte exemplifica o resultado do processo de remoção de ruído. Se a área deste objeto fosse inferior ao valor central definido, o programa iria considerar o seguinte círculo branco como ruído, e não um sinal. Neste caso, detetou o sinal de trânsito

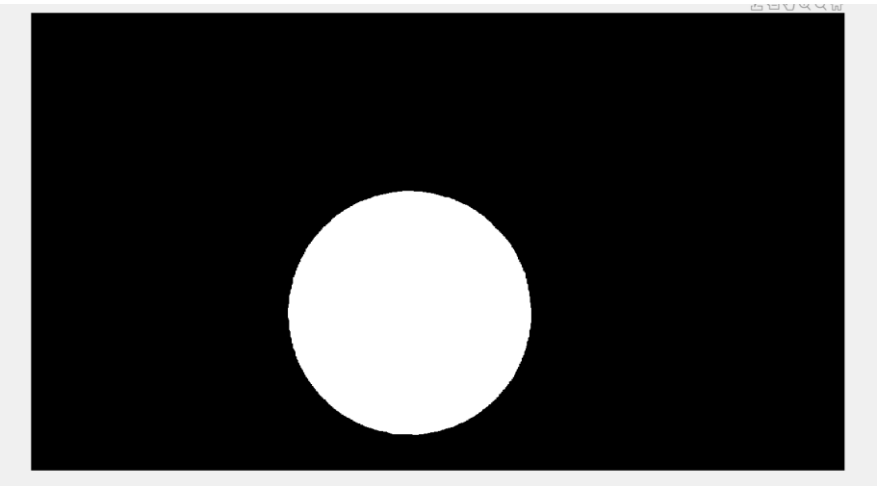


Figura 11 – Sinal detetado sem ruído de ambiente

Devido a ruído o contorno do sinal também poderá ser afetado, causando um erro nos cálculos. Esse erro poderá ser a diferença entre um sinal ser considerado circular ou triangular, por exemplo.

Para colmatar este problema recorremos á técnica de processamento de imagem chamada fecho (função imclose()).

Como podemos verificar na imagem seguinte, o sinal tem algum ruído no exemplo da esquerda, que é corrigido no exemplo da direita.

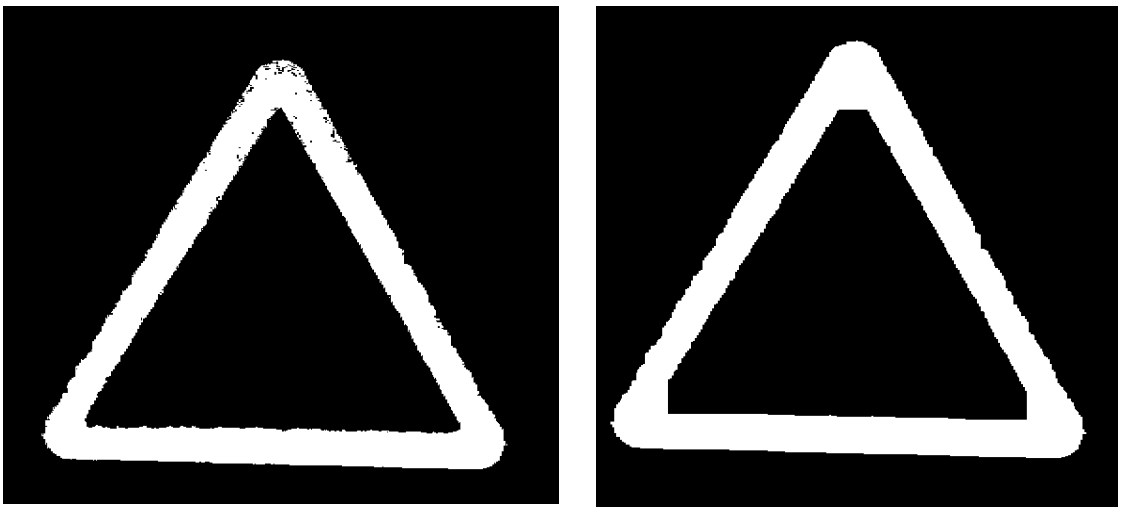


Figura 12 – Na direita temos a imagem binária após threshold, e na esquerda foi aplicado um fecho

Após a segmentação da imagem para qualquer uma das cores, se algum sinal for encontrado, uma flag será ativada e o programa irá abandonar a máquina de estados, para depois proceder ao respetivo processamento da imagem. Se tal não for o caso, o programa executa infinitamente, até detetar um sinal.

2.4. Caraterísticas do sinal

O segundo aspeto mais importante para a distinção dos vários sinais de trânsito é a obtenção das características dos mesmos. Este processo irá permitir obter informações como:

- Área;
- Centroide;
- Comprimento do eixo principal;
- Comprimento do eixo secundário;
- Circularidade.

Todas as caraterísticas listadas serão essenciais para retirar a informação relativa ao seu contorno/forma.

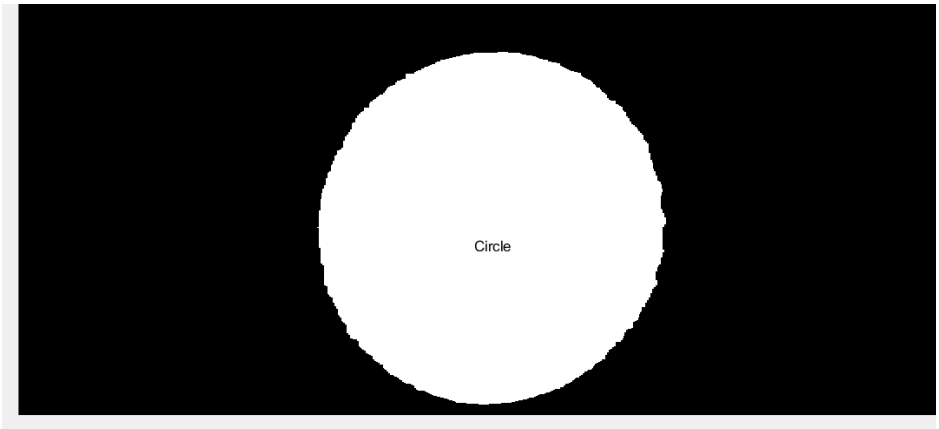


Figura 13 – Identificação da forma

Mais tarde essa informação será utilizada para distinguir entres os sinais possíveis, conforme a sua cor que já fora distinguida anteriormente no threshold.

2.5. Distinção dos sinais

Os sinais de trânsito são inicialmente distinguidos conforme as duas características explicadas anteriormente:

- Cor
- Forma

A cor é distinguida recorrendo a segmentação por threshold de cor, e a forma é obtida através de fórmulas onde os seus parâmetros são extraídos recorrendo á função regionprops().

A cada sinal apenas corresponde uma cor e uma forma, ou seja, duas condições têm de ser respeitadas. A condição de cor é verdadeira quando a flag da máscara é ativada, e a condição da forma é ativada quando a função whichShape () retorna 1, para a forma correspondente.

Flags de cor:

- maskYellow – ativa quando o sinal é de cor amarela;
- maskBlue – ativa quando o sinal é de cor azul;
- maskRed – ativa quando o sinal é de cor vermelha.

Função whichShape(parâmetro da lista):

- isTriangle – retorna 1 se for triângulo;
- isCircle – retorna 1 se for círculo;
- isRectangle – retorna 1 se for retângulo;
- isSquare – retorna 1 se for quadrado;
- isPentagono – retorna 1 se for pentágono;

No entanto, existe mais uma condição após as duas características anteriores, que se trata do interior do sinal, ou seja, o símbolo.

Existem vários sinais com cores iguais e formas iguais, como por exemplo o sinal de obrigação de circundar a rotunda, o sinal de obrigatório virar á direita e obrigatório virar á esquerda.

De forma a permitir aplicar novamente técnicas de processamento de imagem, objetivando isolar o símbolo no interior, recorremos á multiplicação da imagem inicial RGB com a imagem binária sem ruído.

Este procedimento isola apenas o sinal de trânsito, na imagem inicial, como podemos ver na imagem seguinte.

Figura 14 – Resultado da multiplicação da imagem RGB com a binária

Após essa multiplicação, é possível realizar novamente uma segmentação por threshold, e se necessário, mais operações como erosão e eliminação de ruído.

2.5.1. Sinais Amarelos

Após a ativação da flag amarela, limpeza da imagem e multiplicação entre imagem RGB e binária, temos como resultado uma imagem onde apenas consta o sinal amarelo.

Existem apenas dois sinais amarelos que são o sinal de aviso de semáforo e o sinal de aviso de lomba.

Estes sinais contêm no seu interior diferentes cores, como por exemplo, o círculo vermelho do sinal de trânsito e o símbolo da lomba.

De forma a distinguir entres os dois, escolhemos aplicar um threshold vermelho na imagem multiplicada. Desta forma vamos isolar o círculo vermelho no sinal de trânsito e através da função regionprops(), detetar se algum objeto foi criado.

Um objeto apenas será criado se algum pixel (ou pixéis), estiver dentro dos valores de threshold, ou seja, se for vermelho. O mesmo não acontece no sinal de aviso de lomba, visto que o símbolo interior é de cor preta.

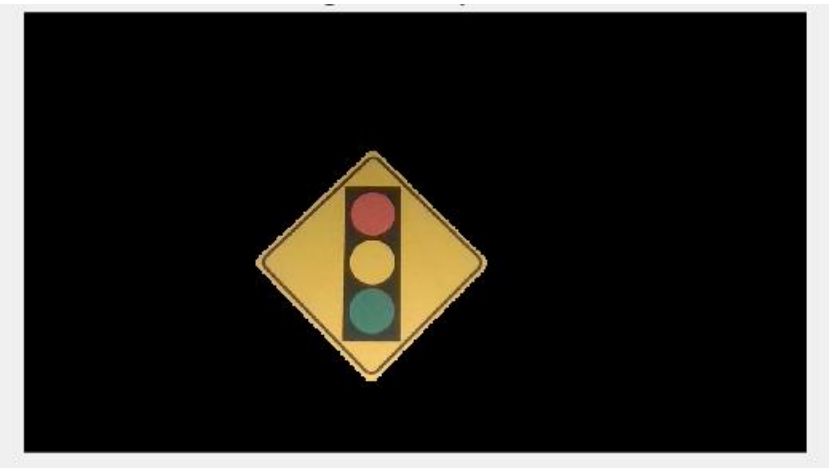


Figura 15 – Imagem multiplicada do sinal de semáforo

Escolhemos aplicar um threshold vermelho, no entanto poderíamos aplicar um threshold verde para detetar o círculo verde.

Um threshold preto já não seria viável, porque além de detetar o símbolo da lomba no sinal respetivo, detetaria também o losango e o retângulo preto que se encontra inserido no sinal de aviso de semáforo.

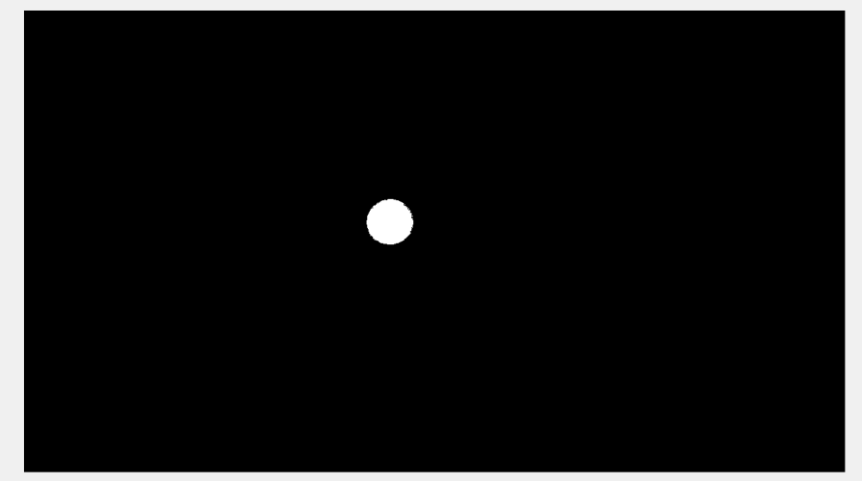


Figura 16 – Imagem após a aplicação do threshold vermelho / objeto detetado

2.5.2. Sinais Azuis

Após a ativação da flag azul, limpeza da imagem e multiplicação entre imagem RGB e binária, temos como resultado uma imagem onde apenas consta o sinal azul.

Existem 4 sinais azuis, onde 3 deles são circulares e o seu objeto interior é branco. Apenas o sinal de via pública sem saída apresenta um retângulo vermelho no seu interior, no entanto este sinal é facilmente distinguido dos outros através da sua forma, que é quadrada. Poderíamos também realizar um threshold vermelho para detetar esse retângulo no interior.

De forma a distinguir os 3 outros sinais, aplicamos um threshold branco, seguido de uma erosão, com o intuito de eliminar o círculo branco no interior do sinal, apenas restando as setas.

No sinal de obrigação de rotunda existem 3 setas, logo, recorrendo á função regionprops() podemos identificar todos os objetos e então separar o sinal de rotunda dos de direção esquerda e direita.

Para distinguir entre os dois sinais de direção diferentes, aplicamos um algoritmo que cria uma linha imaginária vertical no eixo do X, dividindo a imagem em duas partes. Essa linha interseta o centroide do objeto e vai permitir realizar uma contagem dos pixéis no lado direito e esquerdo.

A imagem seguinte serve de exemplo da divisão do objeto (a linha foi criada utilizando a ferramenta de inserir forma e não corresponde a uma linha criada no Matlab).

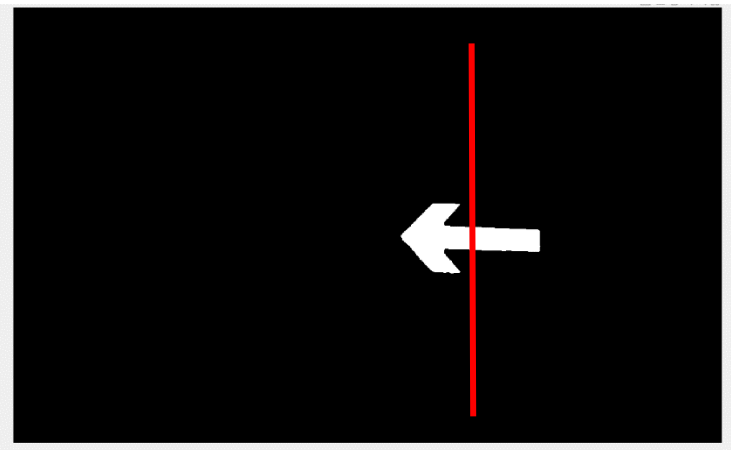


Figura 17 – Divisão do objeto no seu centroide

Como podemos facilmente verificar a seta apresenta uma quantidade maior de pixéis na sua cabeça, o que permite encontrar a sua orientação de forma simples, recorrendo a duas equações que irão quantificar os pixéis dos dois lados.

Após retirados os valores, uma comparação é realizada. Se a metade esquerda for superior á direita, o sinal corresponde ao sinal da figura anterior, ou seja, o sinal de direção esquerda. O contrário acontece com o sinal de direção direita.

2.5.3. Sinais Vermelhos

Após a ativação da flag vermelha, limpeza da imagem e multiplicação entre imagem RGB e binária, temos como resultado uma imagem onde apenas consta o sinal vermelho.

Existem 5 sinais vermelhos diferentes, onde 3 deles são triangulares e 2 são circulares. Facilmente distinguimos recorrendo á sua forma, os sinais circulares dos triangulares.

Para os sinais circulares é aplicado um filtro preto, visto que no sinal de proibição de ultrapassagem existem 2 carros, um deles preto e outro vermelho. Se o threshold utilizado fosse o vermelho, a distinção entre este sinal e o de proibição seria mais complicada.

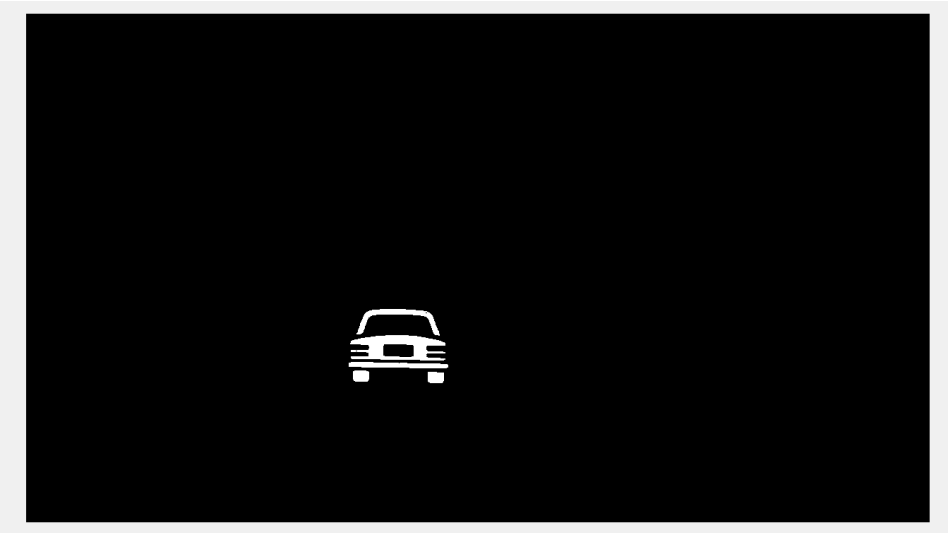


Figura 18 – Threshold preto para detetar o objeto dentro do sinal

Com o threshold preto facilmente isolamos o carro preto dentro do sinal de ultrapassagem, e para o sinal de proibição normal, nenhum objeto seria detetado no interior.

Para os sinais triangulares recorremos também ao mesmo threshold preto utilizado anteriormente, de forma a isolar o símbolo da neve e o símbolo da lomba.

Tal como para os sinais azuis de direção, utilizamos o mesmo algoritmo de divisão do objeto a meio, no entanto, em vez de uma linha no eixo do X, recorremos a uma linha horizontal no eixo do Y.

Como o objeto da neve é simétrico, distinguimos esse sinal do da lomba com uma simples comparação entre as metades superior e inferior.

Considerando que o ambiente de trabalho não é perfeito e que a folha ou a câmara pode estar ligeiramente inclinada, realizamos vários testes para obter um valor de segurança. Assim em vez de esperar que o sinal da neve seja perfeitamente simétrico no seu símbolo, existe um valor de segurança de 300 pixéis.

No sinal da lomba, existe uma diferença bastante maior entre a parte superior e inferior do seu objeto, logo não existirá confusão entre os dois.

A imagem seguinte serve de exemplo da divisão do objeto (a linha foi criada utilizando a ferramenta de inserir forma e não corresponde a uma linha criada no Matlab).

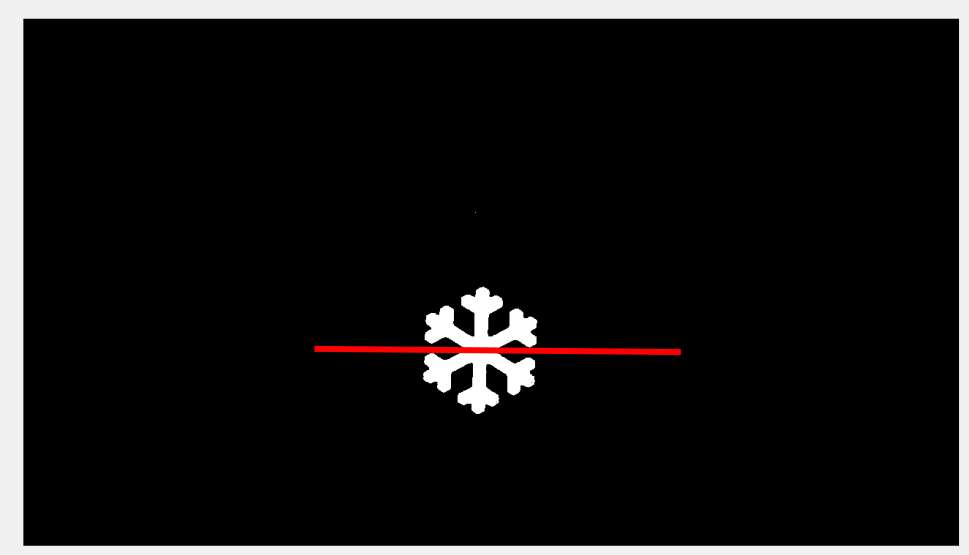
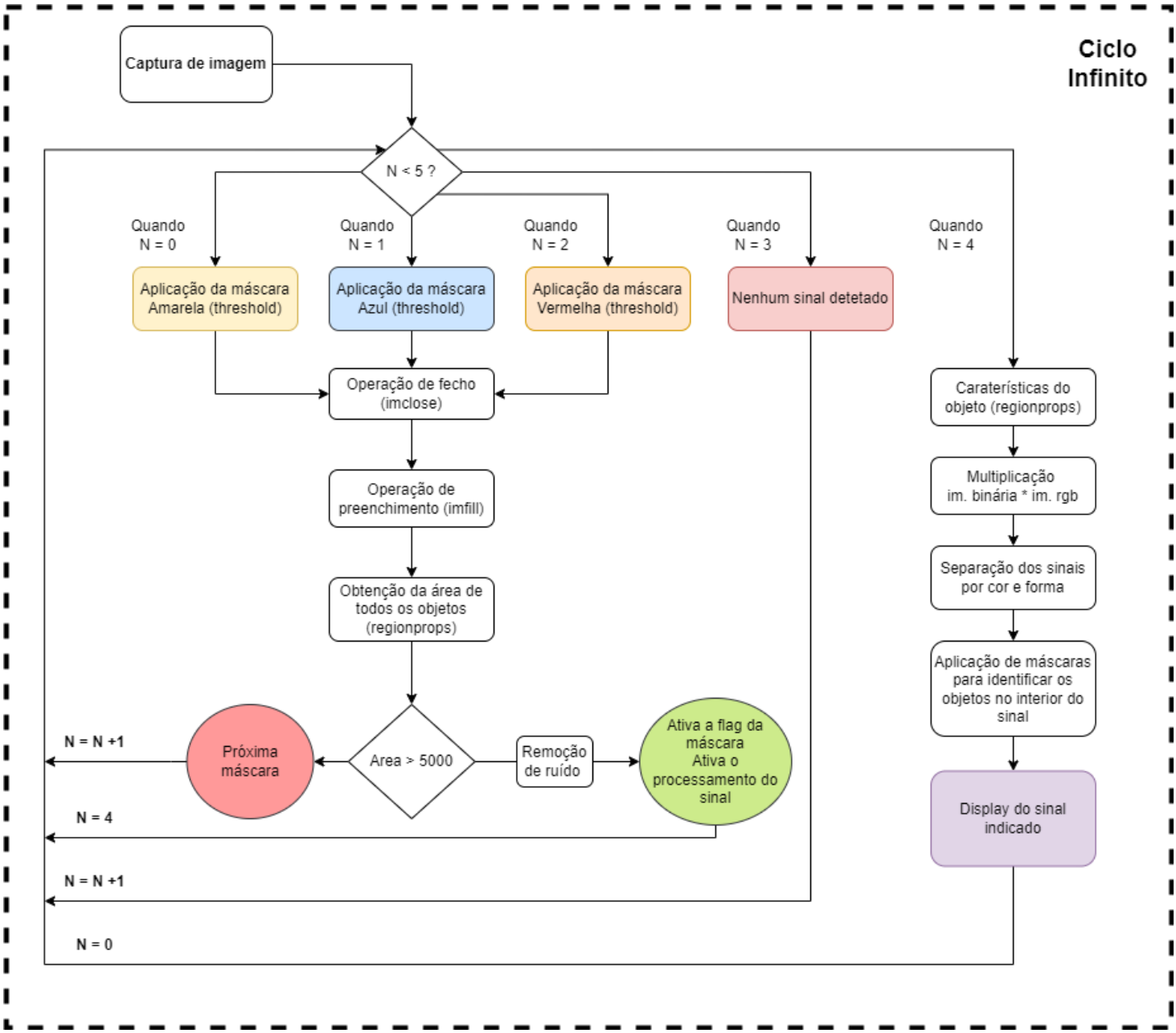


Figura 19 - Divisão do objeto no seu centroide

Para o sinal de perigo geral, ao realizar o threshold preto, nenhum objeto será detetado.

3. Diagrama do projeto



Conclusão

Com a realização deste trabalho prático, foram aplicados diversos conhecimentos obtidos no decorrer da unidade curricular.

O trabalho contribuiu para o desenvolvimento de novas competências, tais como, aplicação de técnicas de processamento de imagem e programação em Matlab.

Ao longo do desenvolvimento tornou-se cada vez mais evidente que o aspeto mais importante para a identificação dos sinais de trânsito, seria o ambiente de trabalho.

Como por exemplo, a luminosidade e estabilidade do sinal e da câmara, ou seja, o sinal necessita de estar diretamente em frente á camara sem qualquer inclinação, e também deve conter uma luz uniforme sobre ele.

A qualidade e caraterísticas da câmara também são bastante importantes, de modo a reduzir o ruído na imagem, reflexão da luz e contraste.

Todo o desenvolvimento do trabalho prático teve em conta esses aspetos, ou seja, se existir uma alteração do ambiente de trabalho, existe a possibilidade de o programa ser afetado.

Fora esse aspeto, o grupo foi capaz de implementar uma boa solução, objetivando sempre a simplificação do processo, a clareza e funcionalidade do código criado, completando todos os objetivos propostos pelo docente.

Bibliografia

https://www.mathworks.com/help/images/ref/regionprops.html

https://www.mathworks.com/help/images/ref/colorthresholder-app.html