Detetor de Objetos Baseado em Informação de Cor

Gabriel Silva (85129) & Gonçalo Vítor(85119)

***Resumo* – O artigo aqui escrito descreve o trabalho feito pelos autores no desenvolvimento duma aplicação simples, usando OpenCV, no âmbito da unidade curricular de Computação Visual, que visa detectar e seguir, num vídeo, um dado objeto, baseando-se na sua cor, permitindo ao utilizador decidir a cor a ser seguida. O relatório começa por introduzir o problema e o modelo de cor HSV e apresenta depois a aplicação e como foi construída.**

***Abstract* – The hereby article depicts the work done by its authors in the development of a simple application, using OpenCV, for the curricular unit Computação Visual(Visual Computation). The goal of the application is to detect and follow an object in a video, based on its color. The color to be followed is specified by the user. The report starts by introducing the problem and the HSV color model and proceeds to present the application and it was built.**

I. Introdução

Numa imagem, um objeto, ainda que só de uma cor, nunca tem todos os pixels exatamente da mesma cor, por exemplo, por causa da luz que nele incide, que pode fazer uma certa parte mais clara e outra um pouco mais escura. Ainda assim, conseguimos claramente classificar tal objeto como de uma só cor, se este realmente for de uma só cor. Num computador, no entanto, isto não é tão simples. Parece óbvio que é necessário definir um certo intervalo de cor que representa a cor do objeto em questão, mas como definimos esse intervalo? Olhemos para o padrão de cor mais conhecido, RGB. Em RGB, definimos uma cor pela quantidade de vermelho, verde e azul que ela tem. O problema deste esquema, para o nosso caso, é que cada valor de R, G e B é uma cor, e variar os três, ou só dois, ou apenas um nem sempre corresponde ao intervalo de cores esperado. No entanto, existe um outro modelo de cores que resolve este problema, chamado HSV. HSV significa Hue (tom), Saturation (saturação) e Value (valor). A grande vantagem deste esquema face a RGB é o Hue, que representa sozinho as várias cores que existem, como vermelho, verde, amarelo, azul, etc, enquanto que saturação e valor definem o brilho da cor. Neste modelo é mais simples criar intervalos para encontrar a cor desejada. O valor de Hue varia entre 0 e 360 graus e, por exemplo, cores como o vermelho estão entre os valores 0 e 60. Misturamos isso com um intervalo de brilho, por exemplo, de 50 a 100%, e temos um intervalo de vermelhos bem definido.

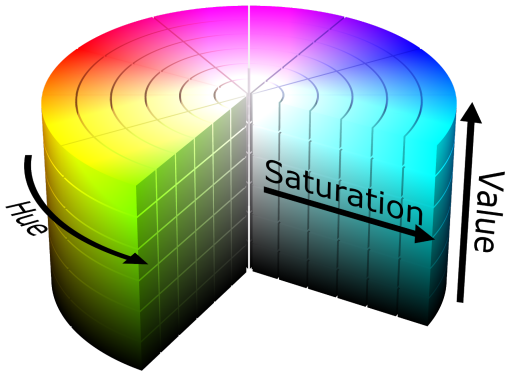

Figura 1: HSV Color Model

II. A Aplicação

A aplicação é bastante simples do ponto de vista do utilizador. Ao correr a aplicação a câmera de

vídeo do dispositivo do utilizador é ligada e começa uma captura de vídeo que é apresentada ao utilizador.

Para além desta janela é mostrada uma janela com várias *tracks bars* para manipular os valores de Hue, Saturação e Valor para escolher a cor que se quer detectar na captura de vídeo. Para além disso, pode-se carregar no vídeo e os valores das *track bars* são atualizados automaticamente mediante a cor do pixel que foi "selecionado" na imagem. Ainda nesta janela, é dada ao utilizador a possibilidade de escolher se quer ou não fazer com que a aplicação ignore o *background* do vídeo, caso o utilizador queira seguir apenas um objeto na eventualidade de haver outros objetos da mesma cor no fundo do vídeo.

O utilizador pode ainda ver outra janela onde vê apenas o resultado da aplicação, isto é, o vídeo original filtrado de forma a que apenas se vejam objetos da cor indicada e ainda outras duas janelas, uma com uma máscara e outra com o resultado blured. Falaremos do que é a máscara mais à frente.

É ainda possível usar, em vez da câmera do dispositivo para fazer uma captura de vídeo ao vivo, um vídeo, indicando o seu nome como argumento na execução do programa (o ficheiro deve estar na mesma pasta que o ficheiro executável).

Figura 2: Janela das *Track Bars*

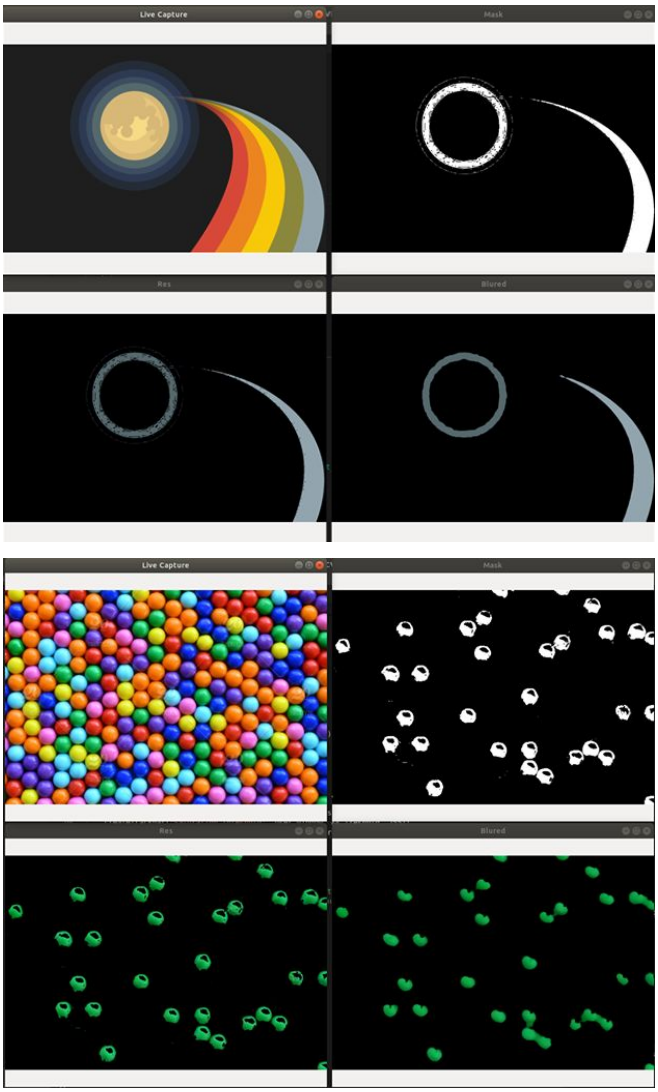

Figura 3 e 4: Janelas da captura, da máscara, resultado e resultado blured.

Nestes testes que mostramos aqui usámos uma imagem, simplesmente para testar a aplicação numa imagem com mais definição, visto que as câmeras dos nossos dispositivos não são muito boas, mas a aplicação funciona com vídeos e os resultados são iguais.

III. Construção da Aplicação

O primeiro aspecto relevante na construção do aplicação é ser capaz de fazer uma captura de vídeo. Isto é bastante simples de fazer em

OpenCV, dado que a biblioteca já tem métodos que o suportam. Fazer uma captura de vídeo é tão simples como “abrir” um dispositivo ou vídeo. A partir daí basta ler, num ciclo, um frame de cada vez, e fazer o que quisermos com ele. A coisa mais básica que fazemos com cada frame é mostrá-lo ao utilizador, para que possa ver o que está a ser capturado e compará-lo com o resultado.

A parte principal do projeto é a seleção de cor. Como mencionado anteriormente, o utilizador pode fazê-lo de duas maneiras: com as *track bars* presentes numa das janelas, ou clicando no vídeo, num pixel da cor desejada. Na realidade, o utilizador não escolhe uma cor exata, mas sim um intervalo de cores e a aplicação filtra a captura de vídeo, para que apenas sejam mostrados no resultado os pixels que se encontram nesse intervalo de cor. É por isso que existem duas *track bars* para cada valor de HSV, um para o limite inferior e outro para o limite inferior. No caso em que o utilizador escolhe a cor carregando no vídeo, o programa cria estes intervalos usando valores the *threshold* default, que nós achamos adequados, ou definidos pelo utilizador na janela com as várias *track bars*. Note-se que se o utilizador não escolher a cor ao carregar na captura, tem de escolher um limite superior e inferior à partida, pelo que as *track bars* de *thresholds* não fazem nada neste caso.

A filtragem mencionada anteriormente é feita com recurso à função de OpencV *inRange*. Esta função usa uma imagem e limites de cor superior e inferior e permite criar uma máscara, onde apenas os os pixels da imagem fornecida que se encontram dentro dos limites têm cor (o pixel em questão fica a branco). É de notar que a imagem usada tem de ser primeiro convertida para HSV, dado que os limites estão em HSV. Esta máscara pode depois ser aplicada à imagem original, o que irá resultar numa imagem em que só vemos aquilo que está dentro dos limites, mas, desta vez com a cor que tem na imagem original.

Para remover o background, a biblioteca OpenCV também tem já funcionalidades que permitem criar uma máscara em que existem os pixels do *foreground*. Esta máscara é misturada com a máscara referida acima para remover o *background* do resultado.

IV. Resultados

Há um fator muito importante que afeta a qualidade dos resultados produzidos pela aplicação: focos na imagem com um brilho muito diferente do resto, por causa da iluminação. Isto faz com que pixels do mesmo objeto, por terem um brilho tão diferente deixem de estar dentro dos limites usados.

Apesar de normalmente, especialmente em capturas com câmeras de baixa qualidade, haver algum ruído, aplicar um *blur* ao resultado não melhora muito os resultados, não compensando, em muitas situações, a troca entre o detalhe da imagem e a qualidade do resultado.

Ambos os membros do grupo contribuíram igualmente no desenvolvimento do projeto.

Nota

Na pasta que enviámos encontram-se: o código (ficheiro .cpp), este relatório, o binário executável e um vídeo. O programa pode ser testado com esse vídeo.