Controlo do Cursor do Rato em OpenCV

Márcia Pires, 88747 Tomás Martins, 89286

***Abstract* – This report describes in detail the development of an application to detect and track, in real time or through a pre-recorded video, a given object, thus controlling the cursor movements on the computer screen; different colored objects allow you to select different options about files, folders or applications. The application was developed in Python using an OpenCV library. The report starts by introducing the idea for the application as well as the objectives defined for it. Next, it presents the main points for the development of the project and how it was implemented, and the available features are described as well. Finally, the results obtained are presented along with some observations.**

***Resumo* – Este relatório descreve de forma detalhada o desenvolvimento de uma aplicação para detetar e seguir, em tempo real ou através de um vídeo pré-gravado, um dado objeto controlando, desse modo, os deslocamentos do cursor no ecrã do computador; objetos de diferentes cores permitem selecionar diferentes opções sobre ficheiros, pastas ou aplicações. A aplicação foi desenvolvida em Python utilizando a biblioteca OpenCV. O relatório começa por introduzir a ideia para a aplicação bem como os objetivos estabelecidos para a mesma. De seguida, são apresentados os principais pontos para o desenvolvimento do projeto e como foi feita a sua implementação, e também são descritas as funcionalidades disponíveis. Finalmente, são apresentados os resultados obtidos juntamente com algumas observações.**

I. INTRODUÇÃO

Desenvolvida no âmbito da cadeira de Computação Visual, a aplicação consiste na deteção de diferentes objetos em tempo real ou, se o utilizador preferir, através de um vídeo pré-gravado. Após a deteção de um objeto, é feito o seguimento dos seus movimentos que são então refletidos no deslocamento do cursor no ecrã do computador. Objetos de cores diferentes produzem um resultado diferente sobre o cursor sendo possível: movê-lo, fazer scroll para cima e para baixo, clicar no botão do lado esquerdo/direito e manter o botão pressionado.

De modo a tornar o nosso projeto robusto e completo, realizámos uma pesquisa detalhada de maneira a compreender quais as funcionalidades que pretendíamos e faziam sentido implementar, tirando o melhor partido daquilo que foi aprendido nas aulas práticas ao longo do semestre, relativas à biblioteca de OpenCV.

II. DESCRIÇÃO DO CONTEÚDO DOS FICHEIROS

A. Módulos

Por forma a ser feita uma boa implementação do projeto, foi necessário estruturar de forma organizada e clara o mesmo, tendo sido o resultado obtido o conjunto de ficheiros em seguida descritos de forma bastante sucinta:

- **program.py** — Ficheiro base para a utilização da aplicação. É nele que está implementada toda a mecânica que permite a deteção e seguimento dos objetos através ou da leitura de um vídeo, ou da leitura direta de webcam.
- **calibrator.py** — Ficheiro que permite ao utilizador descobrir os melhores valores para o range do HSV para um determinado objeto.
- **result.avi** — Ficheiro de vídeo resultante da gravação, se o utilizador assim optar.

III. Funcionalidades

Esta secção tem como objetivo detalhar todas as funcionalidades relevantes da aplicação.

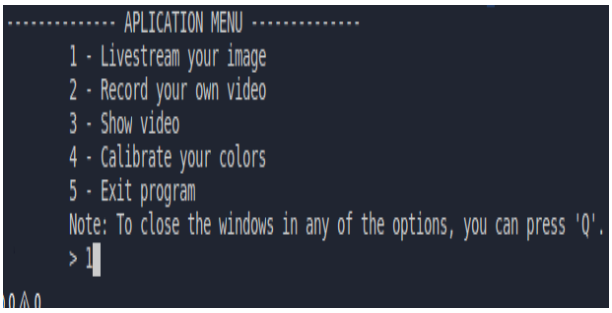

Fig. 1 - Menu inicial da aplicação.

Ao iniciar a aplicação o utilizador irá encontrar um menu, tal como é possível observar na Figura 1. **Através dele, o utilizador tem acesso às seguintes opções:**

1. Utilizar a aplicação recorrendo ao uso da webcam, em tempo real.
2. Gravar um vídeo.
3. Utilizar o vídeo previamente gravado para correr na aplicação.
4. Abrir o programa de calibração de cores.
5. Sair do programa.

A. Em tempo real

Nesta funcionalidade da aplicação, serão abertas 3 janelas distintas:

- **Video:** Nesta janela, é mostrado ao utilizador a imagem que está a ser captada em tempo real, pela própria câmera do computador. Também é possivel para o utilizador ver rectângulos, a indicar quais os objetos que estão a ser identificados para o controlo da aplicação, da cor dos próprios objetos (assim como uma indicação por texto de qual a cor identificada), bem como uma linha azul que segue os últimos movimentos do utilizador durante o controlo da translação do rato.
- **Mask:** Nesta janela, é mostrada uma máscara aplicada sobre a imagem anterior. Aparece então a branco, todos os objetos que estão a ser identificados no momento para o controlo da aplicação.
- **Bitwise:** Por último, nesta janela, é possível observar o resultado de uma operação *and* entre a imagem captada na janela Video e a captada na janela Mask, ou seja, aqui aparece apenas os objetos captados pela máscara na sua cor original.

B. Gravar um vídeo

Assim que o utilizador indicar que pretende selecionar esta opção, vai ser então aberta uma janela que irá começar a gravar um vídeo pela câmera do utilizador. A gravação deste vídeo pode ser terminada a qualquer altura, premindo a tecla *"Q"*. Este vídeo será então guardado no diretório do programa, sobre o nome de *"result.avi"*.

C. Utilizar um vídeo previamente gravado

Esta secção é bastante semelhante à primeira, com a particularidade de que esta, em alternativa a processar imagens em tempo real, irá abrir as mesmas 3 janelas mencionadas anteriormente sendo que as imagens terão a sua origem no vídeo gravado previamente, na secção 2. Os movimentos de objetos com as cores predeterminadas que apareçam no vídeo, serão então usados para manipular o cursor do computador.

D. Blur

O utilizador tem ainda a opção de escolher se deseja que o processamento do vídeo, quer seja em tempo real ou pré-gravado, tenha ou não *Blur*. Se assim o desejar, é então utilizado o método **Gaussian Blurring**, através do uso da função *cv2.GaussianBlur()*. Desta forma conseguimos demonstrar que, mesmo aplicando um certo nível de *blur*, é possível fazer a deteção das cores dos objetos corretamente.

Fig. 2 - Aplicação com *blur* aplicado.

Fig. 3 - Aplicação sem *blur* aplicado.

E. Calibrar as cores

Nesta janela, para além de ser aberto as 3 janelas já mencionadas na secção 1, é aberta ainda uma quarta janela entitulada "Test your colors". Nesta janela, poderemos modificar os valores da Lower HSV(L - H, L - S, L - V) e da Upper HSV (U - H, U - S, U - V) e observar nas restantes janelas, em tempo real, a máscara aplicada à câmera a ser alterada. Com esta funcionalidade, da qual iremos aprofundar mais à frente, é nos permitido calibrar o *range* das cores HSV das máscaras para determinados objetos do nosso ambiente.

F. Como calibrar?

No início do nosso ficheiro *program.py*, estão presentes os valores de HSV para cada uma das cores utilizadas na aplicação. No entanto, devido a problemas de poluição visual nos nossos ambientes de trabalho, foi necessário aprimorar os valores das cores dos objetos que pretendíamos utilizar em concreto. Isto porque, certos objetos de uma determinada cor, por vezes não eram reconhecidos pelo programa pois nem sempre estavam no espectro adequado. Assim, para se proceder então à calibração, é preciso executarmos a secção 4 do nosso programa e ir ajustando os valores de HSV referidos anteriormente, até apenas o nosso objeto em questão estar visível na imagem. De seguida, no ficheiro *program.py*, na secção do HSV das cores, procurar o nome da cor cujos valores queremos substituir. Vamos tomar como exemplo a cor azul, que tem como *default*, os valores: BLUE_COLOR_LOWER = [110, 100, 100] e BLUE_COLOR_UPPER = [130, 255, 255]. A primeira lista, corresponde então aos valores HSV Lower e a segunda lista, por sua vez, corresponde aos HSV Upper. Dentro da cada lista, o 1º elemento

corresponde ao valor **H**, o 2º corresponde ao valor **S** e o 3º corresponde ao valor **V**. Ficaremos então com 2 listas no formato de: BLUE_COLOR_LOWER = [L-H, L-S, L-V] e BLUE_COLOR_UPPER = [U-H, U-S, U-V]. Após esta pequena edição, concluimos então o processo de calibração para determinada cor.

G. Modos de controlo do cursor do rato

Para proceder então à manipulação do movimento do cursor do rato, temos à disposição as seguintes opções:

- **Cor azul**: permite controlar o movimento do cursor do rato;
- **Cor verde**: permite controlar o clique do botão esquerdo do rato;
- **Cor laranja**: permite fazer *scroll* para baixo;
- **Cor amarelo**: permite fazer *scroll* para cima;
- **Combinação da cor verde + amarelo**: permite controlar o clique do botão direito do rato;
- **Combinação da cor verde + laranja**: permite manter o botão esquerdo pressionado, fazendo com que seja possível arrastar items.

IV. Implementação

Para podermos implementar tais funcionalidades, recorremos às bibliotecas de **OpenCV(cv2)**, **numpy** e **pynput**.

A primeira, permitiu-nos realizar operações que envolvessem OpenCV e manipulação de imagem. A segunda, permitiu-nos executar operações de matrizes, de modo a aplicar as máscaras. A terceira, permitiu-nos controlar o sistema, mais especificamente, o cursor do rato.

Começámos então por calibrar o programa, tal como mencionado anteriormente, para as cores Azul, Laranja, Verde e Amarelo, utilizando os seguintes objetos:

Fig. 4 - Objetos utilizados para manipular o sistema.

De seguida, damos ao utilizador a escolha de qual das secções quer executar.

Para reproduzirmos então a manipulação em tempo real, é iniciada uma captura de ecrã *(VideoCapture(0))*, enquanto para utilizar um vídeo pré-gravado, é feito uma captura do ficheiro desejado, *(VideoCapture("result.avi")*. Assim que a captura é iniciada e o tamanho da janela definido, começamos por aplicar a máscara e se o utilizador quiser, o *Blur* da imagem. O *Blur* é então obtido através da função *cv2.GaussianBlur()*.

A máscara será calculada através da função *cv2.inRange()* ao qual passamos os parâmetros de Lower e Upper do HSV da cor pretendida, que nos irá devolver a máscara daquela cor. Para finalizar, ainda submetemos a máscara a um processo de *morphologyEx*, tanto de Open como de Close, de maneira a limparmos as falhas e ruído que possam haver na máscara.

Será aplicada uma máscara para cada cor, pois isso permitiu-nos saber que cores distintas estão presentes no ecrã. Para tal, realizámos uma soma entre todos os valores da matriz da máscara de cada cor e se o resultado for maior que zero, então essa cor está presente na imagem.

Após verificar que imagens estão presentes, para cada mácara, através da função de *cv2.rectangle()*, conseguimos inserir um rectângulo sobre o objeto de cada cor.

Para o controlo do movimento do rato, no entanto, só é possivel encontrar um rectângulo e são ainda realizados cáculos para determinar o centro do rectângulo, de modo a obtermos a posição do cursor na janela de vídeo e por último é feita uma regra de 3 simples em que convertemos a posição do rato relativa à janela, para a posição relativa ao ecrã do computador.

Se houver a cor azul (que controla o movimento do rato), serão então guardadas todas as posições do cursor numa lista, que depois iremos passar à função *cv2.line()*, de maneira a que possamos desenhar uma linha que mostre o percurso recente do cursor. Esta lista guarda até 50 posições, até começar a descartar a posição mais antiga.

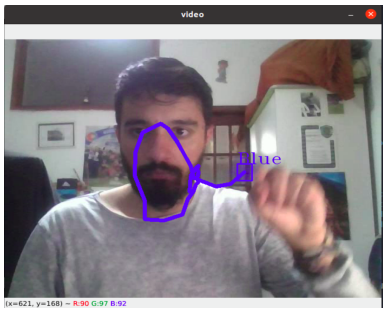

Fig. 5 - Manipular o cursor através da cor azul.

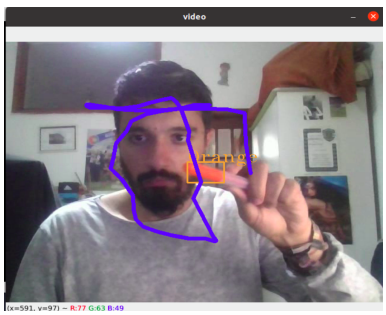

Fig. 6 - Manipular o scroll através da cor laranja.

Para a gravação do vídeo, na secção 2 é iniciada uma captura do ecrã, e através da função *cv2.VideoWriter()*, somos capazes de guardar o vídeo a ser capturado.

V. CONCLUSÃO

Tal como esperado, a realização desta aplicação permitiu-nos aprofundar e aplicar os conhecimentos aprendidos nas aulas práticas de Computação Visual, desafiando as nossas capacidades de implementação dos conceitos para obter as funcionalidades pretendidas.

Consideramos ter alcançado com sucesso os objetivos que nos propusémos a realizar, tendo conseguido como produto final uma aplicação completa e robusta. No entanto, também foi necessário superar algumas dificuldades, nomeadamente a deteção das cores dos objetos corretamente pois ambos estávamos a trabalhar num ambiente com bastante poluição visual, o que dificultava o foco no objeto pretendido e também o facto de existir muita oscilação de luz natural durante o dia, levando várias vezes a recorrer a luz artificial para iluminar o ambiente, o que alterava as cores dos objetos.

No final, o balanço é bastante positivo e a percentagem de divisão do trabalho é de 50/50.

VI. REFERÊNCIAS

- Joaquim Madeira e Paulo Dias, através de código adaptado dos guiões das aulas
- `https://docs.opencv.org/master/index.html`