Relatório do trabalho da disciplina de Introdução à Visão por Computador

Fase 3 – Algoritmos de Deteção de Objetos

Henrique Azevedo - 23488

José Lourenço - 23496

Engenharia em Desenvolvimento de Jogos Digitais

Dezembro de 2022

ESCOLA SUPERIOR DE TECNOLOGIA IPCA

Índice

Introdução

Contextualização

Este projeto diz respeito à proposta de trabalho da unidade curricular de Introdução à Visão por Computador (IVC), no âmbito do 2º ano da Licenciatura em Engenharia em Desenvolvimento de Jogos Digitais, no Instituto Politécnico do Cávado e Ave, realizado durante o período letivo de 2022/2023.

A partir da utilização do código jogo *Breakout*, previamente dado pelo docente, objetiva-se, com o uso de algoritmos de deteção de objetos, e com o uso da câmara, seja possível detetar um objeto, e com o mesmo, realizar as ações necessárias para o jogo.

Nesta terceira e última fase do projeto foi proposto a colocação do *paddle* do jogo *Breakout* a mover, utilizando assim, alguns métodos de algoritmia de deteção de objetos que foram lecionadas ao longo do decorrer das aulas.

Concluindo, este relatório vem a fazer explicação profunda de todos os métodos adotados que contribuíram para criação da proposta elaborada para a fase 3, com conceitos teóricos, problemas e resultados.

Desenvolvimento

Para a boa realização desta fase do projeto, foi necessária a revisão de vários conceitos teóricos sobre processamento de imagem que foram previamente lecionados e assim criar um melhor entendimento e encontrar a melhor forma de resolver o projeto proposto.

Conceitos Teóricos

RGB (*to Gray*)

O espaço de cor RGB faz uso de três cores primárias (vermelho, verde e azul) para produzir o conjunto de cores visíveis. É um sistema de cores aditivo, baseado na teoria tricromática.

O espaço de cor RGB é assim constituído por 3 componentes:

- Vermelho (RED);
- Verde (GREEN);
- Azul (BLUE);

Numa profundidade de cor de 24bpp (bits por pixel), cada um dos três componentes R, G e B é definida por uma variável de 8 bits, que pode tomar valores entre 0 e 255.

O valor 0 (zero) indica que a referida componente de cor não tem qualquer contribuição para a representação da cor final. O valor 255 indica que a contribuição dessa componente será máxima.

Para converter imagens RGB para Gray é primeiro necessário passar a imagem de RGB para BGR (azul, verde e vermelho) e de seguida converter para Gray.

Deteção de Objetos (*Face Detection*)

Deteção de objetos, neste caso de faces, é uma técnica que encontra caras humanas em imagens digitais, tendo como exemplo é a deteção de caras quando se tira uma fotografia no telemóvel. Assim, deteção de faces apenas deteta a presença de faces na imagem, mas não as reconhece

Assim, a deteção de faces é conseguida através do uso de *classifiers*. Estes são algoritmos que avaliam se uma imagem dada é positiva (com cara) ou negativa (sem cara). Então, para os *classifiers* conseguirem realizar a sua tarefa, têm de ser "treinados" com centenas de imagens com e sem faces.

Com o uso de *OpenCV*, que já contêm *classifiers* treinados previamente, é possível elaborar a deteção com mais facilidade. Usando-se *Haar Cascades Classifiers*.

Haar feature-based cascade classifiers:

Haar Cascades Classifiers, através de métodos de *"machine leraning"* sobre deteção visual de objetos, é capaz de processar imagens extremamente rápido e com alto nível de deteção.

Isto provêm de três razões:

- *Haar classifiers* emprega o conceito *"Integral Image"* que deixa que as suas características sejam utilizadas com muita rapidez;
- O algoritmo de aprendizagem é baseado em *"AdaBoost"*, que seleciona um pequeno número, de um grande grupo, de características dando *classifiers* com grande eficiência;
- *Classifiers* mais complexos, são juntos para formar uma *"cascade"* que descarta qualquer região da imagem que não contenha caras, assim usando maior processamento em regiões com o objeto em questão.

Descrição

Para a elaboração desta última fase do projeto prático, começou-se por criar uma classe de nome *"camInput"* que terá todos os algoritmos utilizados.

Dentro desta classe, em primeiro, inicia-se a câmara, através da função de *OpenCV "cv.VideoCapture()"*, utilizando nesta, a variável *"cap"*, seguindo da abertura da mesma numa janela com o uso do método *"showcam"* usando a função *"cap.open(0)"*.

```
def __init__(self):
    self.cap = cv.VideoCapture()

def showcam(self):
    if not self.cap.isOpened():
        self.cap.open(0)
```

Figura 1 - Implementação de Código 1

De seguida para a realização da proposta foi criado o método *"Face_Detection"*, onde, de início se declara um valor para a posição que mais tarde será usada para o jogo. Sendo este o valor de *"MIDDLE"* (1).

Com a ideia de melhorar a experiência do jogador, é invertida a imagem para que esta esteja a par com a visão do jogador.

Sendo a deteção de caras com o uso de *Haar Cascades* o método escolhido para a realização desta fase, é necessária a transformação da *frame* da imagem para *grayscale*, pois este só funciona com imagens nesse estado, com o uso da função *"cv.cvtColor(frame, cv.COLOR_BGR2GRAY)"*, que transforma uma imagem RGB em imagem cinzenta. Assim, de seguida é carregador o *classifier* pretendido através da função *"cv.CascadeClassifier()"*, que será o *Haar Cascade* para a deteção de caras (*"haarcascade_frontalface_alt.xm"*), que é obtido através do *OpenCV*. Com estes parâmetros deteta-se a imagem com o uso da função *"detectMultiScale(…)"* na variável com o valor obtido na função anterior.

Na função anterior é necessário a colocação da imagem em cinzento, *scaleFactor* e *minNeighbors. scaleFactor* define quão a imagem é reduzida por cada *scale*, no nosso caso foi usado o valor de *scale* 1.1, que significa que a imagem é reduzida por 10%. *minNeighbors* define a qualidade da deteção: maiores valores resultam em menos deteções, mas maior qualidade, tendo sido usado o valor 4 no nosso código.

```
def Face_detection(self):
    position = MovementDirection.MIDDLE
    cap = self.cap

    ret, frame = cap.read()
    frame = frame[:, ::-1, :]

    grey = cv.cvtColor(frame, cv.COLOR_BGR2GRAY)
    face_cascade = cv.CascadeClassifier(
        r"C:\...\haarcascade_frontalface_alt.xml")
    faces = face_cascade.detectMultiScale(grey, 1.1, 4)
```

Figura 2 - Implementação de Código 2

Foi também necessária uma cópia do *frame* original da imagem para ser possível a mostragem do vídeo.

Agora, para as funcionalidades do jogo é necessário a caracterização da posição da face detetada na imagem. Assim, começa-se por definir um centro da imagem para futuras comparações.

Com a criação de um *For Loop* consegue-se desenhar um retângulo de cor verde e também detetar o centro desse mesmo retângulo. Então comparando a posição central do retângulo (posição da cara na imagem) com o centro da imagem é possível descobrir se a cara está à direita, à esquerda ou até no centro da imagem, por fim retornando à posição sempre que o *loop* é corrido (ou seja, ao longo que o jogo decorra)

```
image = frame.copy()
image_center = image.shape[1]/2
for (x, y, w, h) in faces:
    cv.rectangle(image, (x, y), (x+w, y+h), (0, 255, 0), 4)
    retanglecenter = (x+x+w)/2
    cv.imshow('Face Detection', image)
    if (retanglecenter > image_center - 30) & (retanglecenter < image_center + 30):
        position = MovementDirection.MIDDLE
    elif(retanglecenter < image_center -30):
        position= MovementDirection.LEFT
    elif (retanglecenter > image_center + 30):
        position= MovementDirection.RIGHT

return position
```

Figura 3 - Implementação de Código 3

Por fim é definida a posição da face e seguinte movimentação do *paddle* no método *"game_loop"*, com a chamada do método previamente descrito.

```
MovementDirection = self.camInput.Face_detection()
```

Figura 4 - Implementação de Código 4

Assim, terminando com o método *"destroyWindow"*, com o propósito de fechar a janela da camara quando o jogo não está a decorrer.

```
def destroyWindow(self):
    cap = self.cap

    if cap.isOpened():
        cap.release()
        cv.destroyWindow("Face Detection")
```

Figura 5 - Implementação de Código 5

Problemas de Codificação

Com os conteúdos lecionados pelo docente ao longo das aulas o grupo conseguiu superar esta terceira e última fase do projeto sem quaisquer problemas.

Resultados

No final da implementação de todo o algoritmo no código, conseguiu-se obter o resultado pedido no enunciado da última fase.

Ao iniciar o programa, surge uma janela que apresenta o jogo *Breakout*, como ser pode ver imagem seguinte

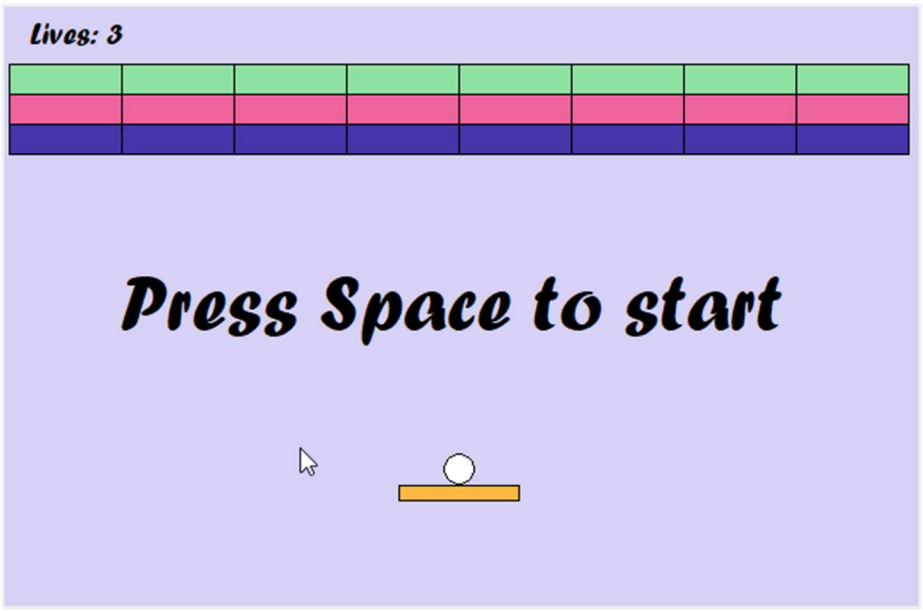

Figura 6 - Jogo *Breakout*

Assim para iniciar o jogo é necessário carregar a tecla espaço, o que fará com que abra uma janela com a camara se abra.

Nesta janela é possível fazer o reconhecimento da face do jogador, que será utilizado para mover o *paddle* no jogo. Assim, quando a cara é detetada na parte direita da imagem, o *paddle* irá mover-se para a direita e vice-versa, assim como quando não é detetada qualquer face ou até se esta estiver no centro da imagem, o *paddle* não se move.

Nas seguintes imagens é possível interpretar a deteção da face e deslocação do *paddle*:

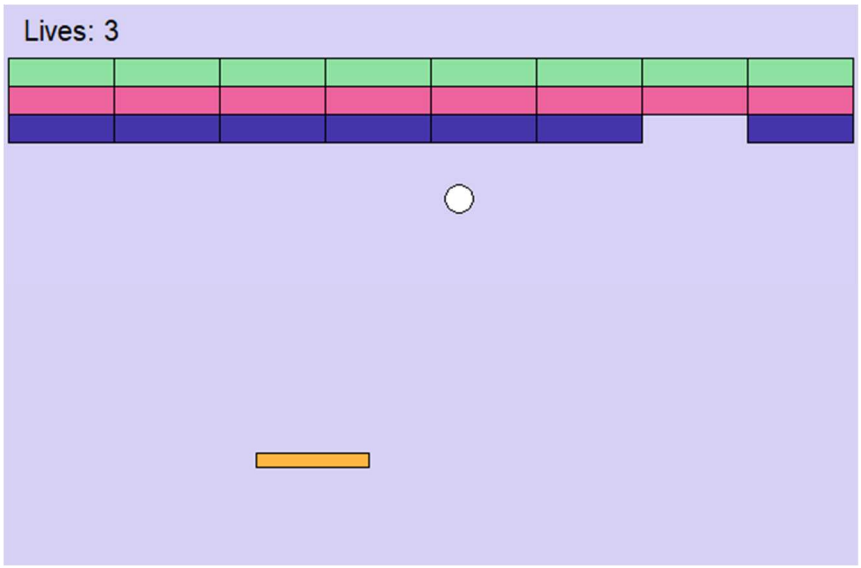

Figura 7 - Partida de Breakout

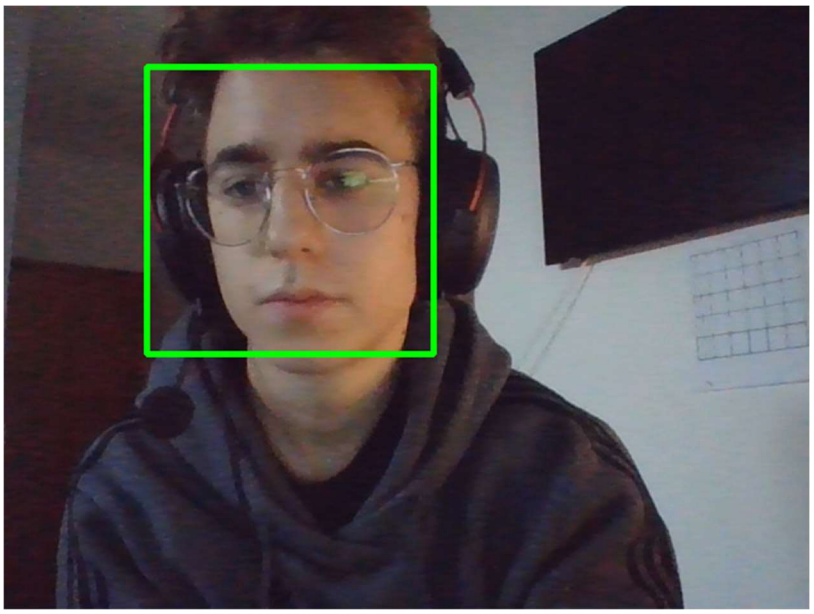

Figura 8 – Deteção da Face

Conclusão

Com o final desta última fase entende-se que foi possível a aprendizagem de vários conhecimentos sobre algoritmos de deteção de objetos e sua respetiva implementação, tendo havido a grande contribuição dos exercícios realizados em aula pelo docente.

Sendo uma área nova foi necessário um maior empenho no estudo de informação para a melhor resolução do projeto, tal como discussões entre o grupo para decidir o melhor meio de trabalho, tendo escolhido o método que mais nos deu conforto na sua realização.

Ao longo desta fase não surgiram muitas situações com dificuldades ou desentendimentos, mas mesmo assim foi possível evoluir com grupo e melhorar as nossas capacidades críticas e práticas e assim preparar-nos para futuros projetos.

Assim damos como concluído o projeto da cadeira de Introdução à Visão por Computador.

Bibliografia

https://pyimagesearch.com/2021/04/05/opencv-face-detection-with-haar-cascades/

https://www.datacamp.com/tutorial/face-detection-python-opencv