Introdução à Computação Visual **(Prazo para submissão: 15/06/2022 23:55)**

Trapalho Prático 2: Realidade Aumentada

Professores: Erickson R. Nascimento

Política da Disciplina: Leia todas as instruções abaixo cuidadosamente antes de começar a fazer o trabalho, e antes de fazer a submissão.

- O trabalho é individual. Se qualquer indício de cópia for constatado, a nota final será anulada.
- A submissão deve ser feita em formato de um Notebook Python executável através do Moodle. Se o Notebook não executar a nota final será anulada.
- Todas as fontes de material precisam ser citadas. O código de conduta da UFMG será seguido à risca.
- Não serão aceitos trabalhos atrasados. O Moodle irá fechar a submissão após o prazo de entrega.
- O trabalho deverá usar o código de esqueleto como estrutura da implementação. Qualquer outra estrutura acarretará na anulação da nota final.

Objetivo: O objetivo deste trabalho é detectar e localizar alvos nos quadros (*frames*) de um vídeo e inserir na cena objetos tridimensionais acima de cada alvo detectado. Para este trabalho, o objeto tridimensional será **um cubo tridimensional** e um **modelo 3D (o Pikachu)**. A Figura 1 mostra um exemplo de como um frame do vídeo final deve ficar após o processamento.

- Vídeo sobre realidade aumentada que pode ajudá-los na implementação do TP: https://youtu.be/1z0Sga8_RxE

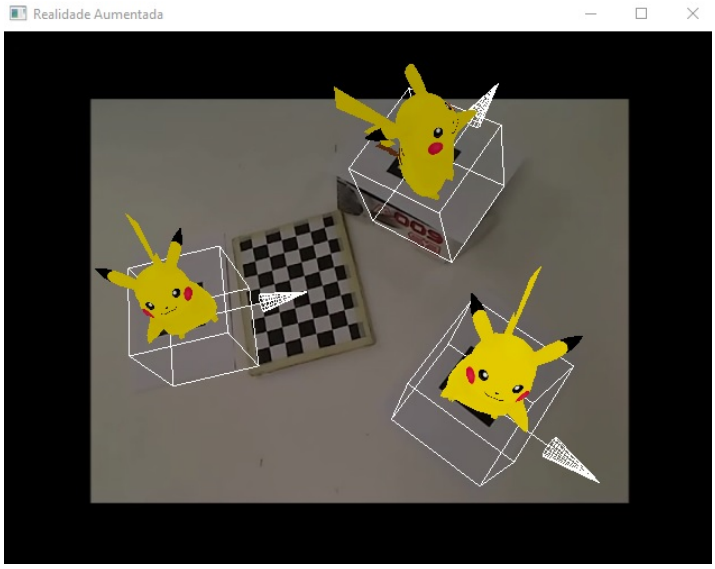

Figura 1: Exemplo do resultado de um frame processado após a inclusão de objetos tridimensionais acima de cada um dos alvos detectados.

O que deve ser feito

O trabalho deverá ser implementado em um Notebook Python e as decisões de implementação deverão ser documentadas no próprio Notebook. Siga os passos abaixo na sua implementação:

(a) (b) (c)

Figura 2: (a) Frame com todos os alvos detectados; (b) Imagem do alvo no qual o objeto 3D deve ser incluído sobre sua superfície; (c) Exemplo de um Frame do vídeo com o objeto 3D e um cubo foi incluído.

1. O primeiro passo será a calibração da câmera utilizando o vídeo com o tabuleiro de xadrez como alvo de calibração. Nesta etapa será realizada a estimação dos parâmetros intrínsecos da câmera. Para fazer a calibração, a(o) aluna(o) poderá utilizar o toolbox de Jean-Yves Bouguet para Matlab/Octave (versão 5.2.0) [1] [2]. A Figura 2-b mostra um frame do vídeo com alvo de calibração (o tabuleiro de xadrez).

2. Depois de estimada a matriz de parâmetros intrínsecos, o aluno deverá implementar um método de detecção e localização da posição e orientação do alvo em cada frame do vídeo. Pode-se utilizar funções da OpenCV para ler vídeos, imagens, calcular a matriz de homografia, recortar e colar, binarizar, detectar bordas (`Canny()`), detectar quinas (`cornerHarris()`) e detecar poligonos (`findContours()`). A Figura 2-a mostra um frame do vídeo da cena com os alvos localizados. **Atenção: não tentem usar detectores de keypoints e fazer a correspondência de descritores.** A abordagem com keypoints não funcionará bem, pois o alvo não possui textura. Para localização da orientação do alvo, use uma estratégia de casamento de templates como visto em aula.

3. Neste passo você deverá codificar uma função para obter a pose da câmera (parâmetros extrínsecos). Para isso, use a função `solvePnP()` da OpenCV como mostrado em sala de aula.

4. Finalmente um cubo (objeto tridimensional) deverá ser incluído na cena na posição o orientado de cada alvo detectado (**a orientação do cubo deve ser mostrada**). **A renderização deverá ser feita utilizando somente as funções da OpenGL (o aluno não deve usar funções da OpenCV para renderizar objetos na cena)**. O objeto deve ser inserido acima da superfície do objeto mostrado na Figura 1. O objeto 3D deve ser incluído em todos os frames do vídeo.

   Para usar o modelo 3D do Pikachu, você deverá usar o código no arquivo `objloader.py`. Instale a biblioteca PyGame e para carregar o modelo use o comando:

   ```
   obj_pikachu = OBJ("Pikachu.obj", swapyz=True)
   ```

   E para renderizar o modelo use:

   ```
   glCallList(obj_pikachu.gl_list)
   ```

5. O modelo do Pikachu deverá ser animado sendo rotacionado em cada frame em torno do eixo Z (veja o vídeo `tp2-icv-resultado.mp4` que mostra um exemplo do resultado esperado). Note que o cubo não deve ser rotacionado, apenas o modelo do Pikachu. O modelo nos alvos indicados com 1 e 3 (Figura 2-a) deverá ser rotacionado no sentido anti-horário e o modelo que será posicionado no alvo indicado com 2 deverá será rotacionado no sentido horário.

6. Os alvos detectados devem ser mostrados como na Figura 2-(a).

[1] http://www.vision.caltech.edu/bouguetj/calib_doc/htmls/example.html
[2] https://github.com/nghiaho12/camera_calibration_toolbox_octave

Avaliação

- Documentação: 10%;
- Detecção do alvo: 20%;
- Renderização do Cubo: 25%;
- Renderização do Pikachu com rotação: 45%.

Bibliotecas a serem utilizadas

Você deverá criar um ambiente Python para este trabalho prático. Seu ambiente deve conter as as versões das seguintes bibliotecas:

- OpenGL 3.1.5;
- OpenCV 4.4.0;
- PIL 7.2.0;
- Numpy 1.18.5;
- Python 3.8.3;
- PyGame 1.9.6.

O que deve ser entregue

Deverá ser entregue um arquivo "nome-completo-do-aluno.ipynb" com:

- Um link de um vídeo no Youtube de no **máximo TRÊS minutos** com a/o aluna/aluno explicando as principais decisões tomadas no trabalho. **A aluna/aluno deve aparecer na imagem.** A nota do trabalho será anulada na ausência do vídeo.
- A implementação do Notebook Python que possa ser executado. **Notebooks que não executarem receberão nota** 0.
- Cada uma das funções implementadas deve ser documentada adequadamente em texto no Notebook, ou seja, as funções devem possuir a descrição de sua função bem como a justificativa da decisão de implementação. Note que NÃO é pedido código com comentários, mas texto descrevendo e discutindo as decisões de implemetnação. A matriz de parâmetros intrínsecos estimada deverá ser incluída no Notebook.

O que será disponibilizado

- Um vídeo no qual aparece o alvo de calibração e os alvos para localizar as posições e orientações onde devem ser inseridos os objetos 3D: entrada.avi. As dimensões dos quadrados no tabuleiro de xadrez são $3cm \times 3cm$. Notem que esse vídeo deve ser utilizar para primeiro calibrar a câmera e depois para incluir o objeto 3D;
- Imagem jpg do alvo (`alvo.jp`);
- Código em Python para carregar o modelo 3D do Pikachu (`objloader.py`);
- Modelo 3D com textura do Pikachu no formato OBJ (`pikachu_obj.zip`).