Projeto Demonstrativo 1 - Explorando OpenCV - Princípios de Visão Computacional - Turma A - 25/03/2018

Thúlio Noslen Silva Santos
14/0164090
Universidade de Brasília
thulionoslen_@hotmail.com

1. Objetivos

Desenvolvimento de uma aplicação que explore e implemente algoritmos usando a plataforma OpenCV. A aplicação deve conseguir:

1. Abrir uma imagem do tipo JPG, permitir que o usuário que clique em um pixel com o mouse e mostrar as informações de cor e posição do pixel;

2. Marcar de vermelho todos os pixels com menos de 13 tons de distância do pixel selecionado no item 1, usando a distância euclidiana no espaço cromático;

3. Realizar os itens anteriores em um vídeo em formato AVI ou x264;

4. Realizar os itens anteriores com quadros vindos de *streaming* de vídeo de uma webcam.

2. Introdução

2.1. Visao Computacional

A Visão Computacional é uma área interdisciplinar que lida com como os computadores podem adquirir entendimento de alto nível a partir de imagens e vídeos digitais. Sub-domínios da visão computacional incluem reconstrução de cenas, detecção de eventos, rastreamento em vídeo, detecção de objetos, estimação de pontos em 3D, etc.

2.2. OpenCV

OpenCV (*Open Source Computer Vision Library*) é uma biblioteca de software de visão computacional e aprendizado de máquina de código aberto. A biblioteca foi construída para prover uma infraestrutura comum para aplicações em visão computacional e acelerar o uso de percepção de máquina em aplicações comerciais.

A biblioteca tem uma variedade de algoritmos otimizados que podem ser usados para detectar e reconhecer faces, identificar objetos, rastrear movimentos de câmera, produzir nuvens de pontos 3D a partir de imagens estéreo, seguir movimento dos olhos, etc.

Ela tem interfaces em C++, Python, Java e MATLAB e suporta Windows, Linux, Android e MAC OS como sistemas operacionais, e a biblioteca tende a ser focada em visão em tempo real.

2.3. Python

Python é uma linguagem de programação interpretada, de alto nível e de propósito geral. Esta é a linguagem de escolha para este trabalho. Existem pacotes para otimização de cálculo numérico em Python que facilitam o desenvolvimento de aplicações com OpenCV, como scipy e numpy.

2.4. Representaçao de imagens no computador

Usando OpenCV e Python, as imagens são representadas em memória por arrays do numpy. As imagens são matrizes, bidimensionais ou tridimensionais a depender do número de canais. Operações em imagens, como alterarações nos pixels individuais, podem ser feitas à mão usando-se funções do numpy e do Python ou por funções do OpenCV, se disponíveis.

3. Materiais e Métodos

Os materiais e equipamentos utilizados foram computadores comuns, e o programa foi testado num sistema Ubuntu 16 64-bits. Para desenvolver a aplicação, foi usado o Visual Studio Code. As bibliotecas usadas foram OpenCV 3.4.0 e numpy 1.14.2 e a versão do Python usada foi a 3.5. Outros pacotes usados que vem por padrão em distribuições do Python 3.5 são argparse, time e os. Foram utilizados tutorias de Python[1], OpenCV[2] e numpy[3] na elaboração do projeto.

3.1. Organizaçao dos arquivos do projeto

Um único arquivo foi desenvolvido, main.py. Este arquivo pode ser executado a partir da linha de comando com

```
python main.py --mode MODE_OPT
```

Em que –mode e MODE-OPT são argumentos para o programa que permitem selecionar um modo de execução. A fim de que fossem atendidos os 4 requisistos da aplicação, foram desenvolvidos 3 modos: modo imagem, modo vídeo e modo webcam.

O programa foi enviado com duas imagens, image1.jpg e image2.jpg, e um vídeo, video1.mp4.

3.2. Modo de imagem (requisitos 1 e 2)

Para executar o modo de imagem do programa, basta entrar na pasta em que o arquivo está localizado e executar o comando:

```
python main.py --image IM_PATH
```

Em que IM_PATH é uma string tal como images/image1.jpg que indica o caminho da imagem no disco. Um exemplo pode ser conferido na seção 4.1.

3.3. Modo de vídeo (requisito 3)

Para executar o modo de vídeo do programa, novamente basta entrar na pasta em que o arquivo está localizado e executar o comando:

```
python main.py --video VID_PATH
```

Tal como antes, VID_PATH é uma string que indica o caminho para o vídeo, tal como videos/video1.mp4. Um exemplo pode ser conferido na seção 4.2.

3.4. Modo de webcam (requisito 4)

Para executar o modo de vídeo do programa, novamente basta entrar na pasta em que o arquivo está localizado e executar o comando:

```
python main.py --webcam [CAM_DEV]
```

Neste caso, CAM_DEV é um inteiro que indica qual dispositivo de câmera utilizar, e é um argumento opcional. Caso não seja dado, será o usado o dispositivo 0, a câmera padrão. Um exemplo de uso pode ser conferido na seção 4.3.

3.5. Redimensionamento de imagens

No programa vem incluído um redimensionamento das imagens ou quadros de entrada, caso queria se utilizar imagens muito grandes ou diminuir o tamanho de vídeos para aumento de performance. Para utilizar o redimensionamento, basta incluir o argumento –resize RSZ_STD no comando, em que RSZ_STD é um dos padrões de qualidade implementados, que são o qhd (quasi-HD), hd (HD) e fhd (full-HD). Note que o redimensionador mantém o aspect ratio da imagem, mas garante que a imagem será sempre menor ou igual às dimensões do padrão dado. Note que este argumento pode ser utilizado nos três modos de execução. Um exemplo de uso do redimensionamento será dado na seção 4.1.

Figura 1. Imagem image1.jpg original.

3.6. Detalhes adicionais

Mais detalhes podem ser conferidos no arquivo-fonte do projeto, em que há vários detalhamentos de mais baixo nível explicando o funcionamento de cada função. Também podem ser conferidos detalhes sobre os argumentos na linha de comando com:

```
python main.py --help
```

4. Resultados

4.1. Requisitos 1 e 2

Para teste dos requisitos 1 e 2, foram usadas as imagens image1.jpg e image2.jpg disponibilizadas, sendo que image1.jpg é uma imagem em tom de cinza e image2.jpg é uma imagem colorida. Primeiro, a imagem image1.jpg foi aberta com o comando:

```
python main.py \
--image images/image1.jpg
```

Ao clicar no ombro da Lena na figura 1, a posição retornada foi (x, y): (343, 432), e a intensidade I: 209. O resultado da coloração dos pixels foi a figura 2.

Usando agora uma imagem colorida, image2.jpg, foi usado o seguinte comando com redimensionamento para o padrão qHD:

```
python main.py \
--image images/image2.jpg \
--resize qhd
```

Ao clicar na letra W de WOULD da figura 3, que está em branco, obteve-se a posição (x, y): (526, 126) e dados RGB: [254 254 254], e foi gerada a imagem da figura 4.

Figura 2. Imagem image1.jpg ao clicar no ombro da Lena.

Figura 3. Imagem image2.jpg original.

Figura 4. Imagem image2.jpg ao clicar na letra W de WOULD.

4.2. Requisito 3

Para teste do requisito 3, foi usado o vídeo video1.mp4. Ele foi aberto na aplicação com o comando:

Figura 5. Quadro do vídeo video1.mp4 original.

Figura 6. Quadro do vídeo video1.mp4 depois de clicar no céu.

```
python main.py \
--video videos/video1.mp4
```

Foi capturado o quadro na figura 5, um clique foi dado no céu da imagem, resultando na saída (x, y): (1172, 138) e RGB: [224 229 253] e novamente um quadro foi capturado, mas com o quadro contendo partes coloridas de vermelho, que pode ser visto na figura 6.

4.3. Requisito 4

Por fim, para teste do requisito 4, foi utilizada a webcam padrão de um notebook. Para usá-la na aplicação, foi usado o comando:

```
python main.py --webcam 0
```

Similar ao requisito 3, foi capturado o quadro dado na figura 7, um clique foi dado na porta na imagem, resultando na saída (x, y): (586, 148) e RGB: [54 13 0] e novamente um quadro foi capturado, que pode ser visto na figura 8.

5. Discussão e Conclusões

Como pode ser visto nos resultados, o programa opera conforme esperado. Além disso, há pouco impacto de performance tanto nas imagens quanto nos vídeos, já que nenhuma operação foi implementada usando-se laços for. Os

Figura 7. Quadro da webcam original.

Figura 8. Quadro da webcam depois de clicar na porta.

únicos laços utilizados foram os que mantém o programa rodando enquanto a tecla q não for pressionada.

Referências

[1] 3.5.5 documentation. https://docs.python.org/3.5/. Accessed: 2018-03-25.

[2] Opencv: Opencv-python tutorials. https://docs.opencv.org/3.4.1/d6/d00/tutorial_py_root.html. Accessed: 2018-03-25.

[3] Overview — numpy v1.14 manual. https://docs.scipy.org/doc/numpy-1.14.0/. Accessed: 2018-03-25.