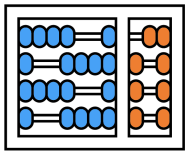

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

MARCOS DA MATA SOUSA (221519)

Relatório - Trabalho 5

CAMPINAS
2022

1. Introdução

Neste relatório é apresentado um método para detecção de movimento em vídeo através da verificação da diferença entre quadros adjacentes. No caso em que os quadros sejam iguais, não haverão mudanças aparentes na imagem. Caso contrário, as regiões de mudanças (aquelas em que houve movimento) serão destacadas. Grande parte do processo foi realizada com a utilização de ferramentas da biblioteca *OpenCV* em linguagem *Python*.

2. Processo de detecção

O vídeo utilizado no experimento é uma abertura de desenho animado, o Pica-Pau.
A captura do vídeo foi realizada com auxílio da biblioteca *OpenCV*, através do método *VideoCapture*.

A partir da captura do vídeo, foi possível "manusear" cada um dos frames através do método *read*, que retorna o próximo quadro do vídeo a cada chamada. Além disso, é retornado um valor booleano (verdadeiro ou falso) que assume valor *True* quando *read* é capaz de retornar o próximo quadro do vídeo. Caso contrário, o valor retornado é *False*, significando que não há mais quadros para serem retornados e, dessa forma, o vídeo chegou ao fim.

Dessa forma, guarda-se os retornos das duas primeiras chamadas de *read* em duas variáveis, *frame1* e *frame2* e, a partir daí, cada quadro é comparado com o seu anterior dentro de um *loop* que é interrompido quando o valor booleano citado acima é *False*. Ao final de cada ciclo do *loop*, *frame1* e *frame2* são atualizados novamente com a utilização do método *read*. No núcleo do *loop* são realizados os seguintes processos (os métodos utilizados são apresentados entre parênteses, todos da biblioteca *OpenCV*):

- Obtêm-se a diferença entre *frame1* e *frame2* (*absdiff*)
- Converte-se a diferença obtida para uma imagem monocromática (níveis de cinza) (*cvtColor*)
- Aplica-se uma suavização gaussiana (filtro de tamanho 5x5) na diferença, a fim de remover eventuais ruídos (*GaussianBlur*)
- Aplica-se um processo de limiarização na diferença, escolhendo-se um intervalo adequado. Aqui, o intervalo escolhido foi [25, 255] (*threshold*)
- Aplica-se uma dilatação na imagem limiarizada, de tal modo que pixels de bordas dos objetos são adicionados (*dilate*)
- Encontra-se os contornos da imagem já dilatada e calcula-se a sua área. (*findContours* e *drawContours*)
- Para contornos que possuem área maior que um limiar (neste caso, adotei 1000), desenha-se um retângulo que envolve cada um destes. (*contourArea e boundingRect*)
- Exibe-se a imagem resultante em tela.

3. Resultados

Antes de apresentar os resultados propriamente ditos, para fins de comparação, abaixo são apresentados dois pares de quadros adjacentes. Note que, por se tratar de uma animação, a mudança de quadros ocorre de maneira lenta, de modo a fornecer sensação de movimento dos desenhos. Dessa forma, as imagens possuem pouquíssimas diferenças visíveis à primeira vista, apesar de haverem diferenças.

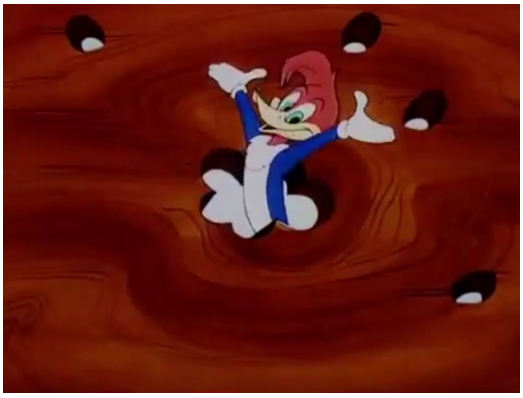

Imagem 1: Frame anterior

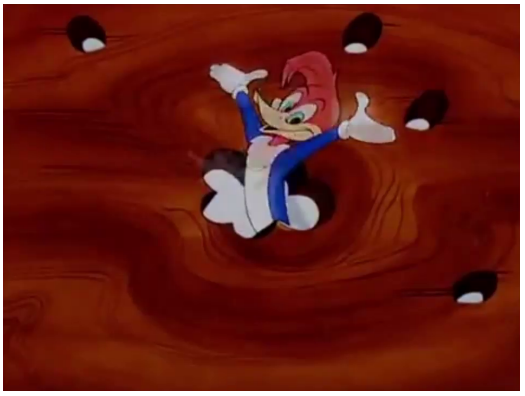

Imagem 2: Frame atual

Imagem 3: Frame anterior 2

Imagem 4: Frame atual 2

Com o início do processo, obtêm-se a diferença entre as duas imagens e converte-se o resultado deste em niveis de cinza (A diferença, bem como outros resultados dos passos do processamento podem ser encontrados no diretório durante a execução do algoritmo. Note que se executado de uma vez, todos os resultados presentes no diretório serão referentes ao último par de frames, uma vez que os resultados são sobrescritos a cada par de frames):

Figura 5: Diferença do primeiro par

Figura 6: Diferença do segundo par

Como dito anteriormente, após o processo de conversão da imagem diferença para níveis de cinza, aplica-se um *threshold*, um filtro gaussiano, bem como uma dilatação na mesma. Antes do resultado final, observamos as imagens sob a aplicação destes três processamentos, processamentos estes que deixaram mais evidentes as diferenças entre os pares quadros originais (principalmente a dilatação)..

Figura 7: Par 1 com threshold, filtro e dilatação

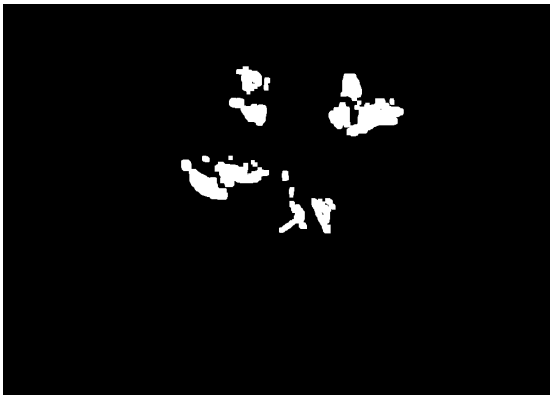

Figura 8: Par 2 com threshold, filtro e dilatação

Por fim, podemos obter e desenhar os contornos da imagem dilata, bem como posicionar retângulos que envolvem os objetos que possuem área maior que 1000 (um valor arbitrário adotado para este trabalho, poderia ser mudado). Abaixo podemos observar os resultados finais de todo o processamento:

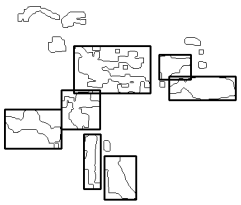

Figura 9: Resultado 1

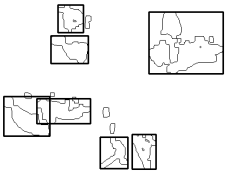

Figura 10: Resultado 2

Como esperado, foi possível obter uma imagem com os contornos da diferença dilatada e binarizada, bem como os retângulos sobrepostos sobre regiões de contorno cujas áreas são maiores que 1000. Vale ressaltar que para uma imagem diferença cujos quadros anterior e atual são iguais, o resultado seria uma imagem branca, sem contornos e retângulos. (A menos que se considere as quatro bordas da imagem como uma região de contorno).

Por fim, conclui-se que, através dos passos descritos neste relatório, o procedimento de detecção de movimentos em vídeo mostrou-se bem sucedido, uma vez que as pequenas diferenças vistas entre os pares de quadros adjacentes são evidenciadas ao fim do processamento. Além disso, o posicionamento dos triângulos nas áreas onde há movimentos significativos também ocorreu com sucesso, de modo que, através das observações dos resultados, vemos que os contornos com maiores áreas são envoltos pelos mesmos, enquanto os pequenos contornos, não.