Remoção de Fundo e Segmentação de Objetos em vídeo via PCA

Aluno: Victor Pereira de Lima

Disciplina: Computação Científica e Análise de Dados (COCADA)

Período: 2025.1

Link para o repositório (Github):
https://github.com/VictorPLima/Projeto_Final_COCADA_2025-1

Introdução

A ideia deste projeto veio de uma aula de inspirações de projetos onde o professor apresentou uma das aplicações de álgebra linear expostas no curso Computational Linear Algebra, esta aplicação era a de Background Removal with Robust PCA, onde a decomposição por Análise de Componentes Principais (PCA) é utilizada para obter e remover o fundo em uma filmagem de rua com pessoas andando. Em situações como esta a câmera normalmente encontra-se estática filmando apenas um certo ângulo, o que resulta em imagens com fundo praticamente constante, onde a única variabilidade fora a iluminação ocorre devido a objetos em movimento (veículos, pessoas, animais, …), como a cada componente da PCA temos a variância do que é decomposto mais explicada, podemos tirar proveito disso para buscar formas de decompor as imagens de um vídeo para obter seus detalhes principais e tentar segmentá-los.

Transformando vídeos em matrizes

O projeto foi desenvolvido no Google Colab utilizando a linguagem de programação Python e suas bibliotecas numéricas (NumPy, scikit-learn), de visualização de dados (matplotlib), e de tratamento de imagens e vídeos (PIL, moviepy, OpenCV). O vídeo utilizado neste projeto encontra-se no repositório com o nome “v1.mp4” e é uma gravação de 50 segundos onde pessoas são registradas caminhando na rua, como observa-se na Imagem 1.

Imagem 1: Frame do vídeo na posição 0:17 / 0:50.

Fonte: https://nbviewer.org/github/fastai/numerical-linear-algebra/blob/master/nbs/3.%20Background%20Removal%20with%20Robust%20PCA.ipynb#Load-and-view-the-data

As imagens são representadas no computador como matrizes de pixels, onde cada entrada possui as cores do pixel codificadas, no caso de imagens coloridas temos que as cores são representadas em código RGB (*Red, Green, Blue*), onde temos números entre 0 e 255 para indicar o quanto de vermelho, verde, e azul tem numa mesma cor e assim o computador conseguir representá-la, consequentemente, uma imagem colorida de dimensão mxn é representada por três matrizes mxn, cada uma para uma das cores RGB, conforme é mostrado abaixo.

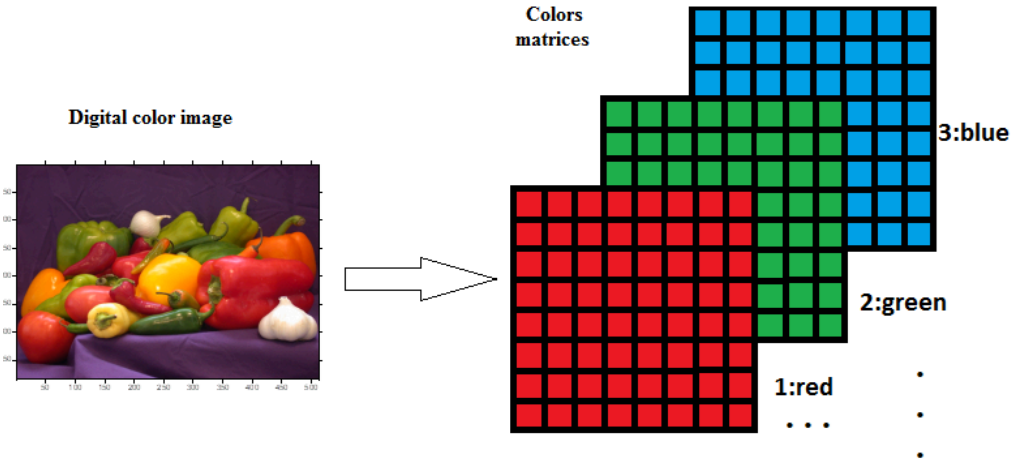


Imagem 2: Representação digital de imagem em código RGB.

Fonte: https://www.researchgate.net/figure/Color-image-representation_fig1_371071395

Um vídeo é composto por várias imagens (frames) obtidas por segundo, pelo menos 30 por segundo (30 fps) normalmente, supondo uma resolução básica de 360p (dimensão 640x360 pixels) e considerando que um pixel RGB requer 3 bytes (quantos bits tem um pixel), um vídeo de 1min requer (60s)*(30fps)*(640*360 pixel/frame)*(3 byte/pixel) = 1.244.160.000 bytes ≈ 1,24 GB, ou seja, trabalhar com vídeo requer muito armazenamento, para mitigar esse problema reduzimos a dimensão dos frames e os passamos para preto e branco (*grayscale*) onde temos apenas um 1 byte por pixel e uma matriz para codificar cor em vez de três.

A PCA busca decompor uma matriz A conforme $A_{mxn} = B_{mxk}C_{kxn}$, onde B, a matriz de componentes principais, possui como vetores coluna os k autovetores associados aos k maiores autovalores da matriz de covariância AA^T ou A^TA ordenados decrescentemente, e C possui as projeções dos vetores coluna de A nas componentes contidas em B. Portanto, para usar a PCA precisamos de uma só matriz (com os dados padronizados, ou centrados na média) que represente todo o vídeo, como este nada mais é do que uma lista (array) de matrizes de pixels (frames) ordenada no tempo, podemos ver o vídeo como uma matriz onde cada vetor coluna é um frame e a quantidade de colunas é a de frames, no entanto, um frame ainda é uma matriz, e não um vetor, para contornar este problema "achatamos" cada frame em um vetor usando a função flatten() do NumPy, conforme abaixo.

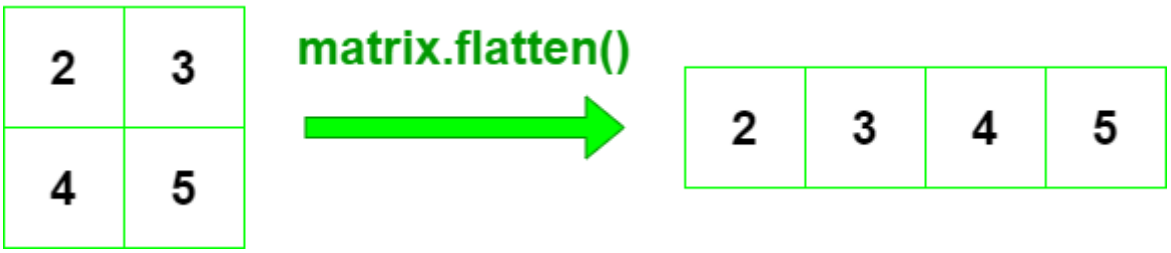
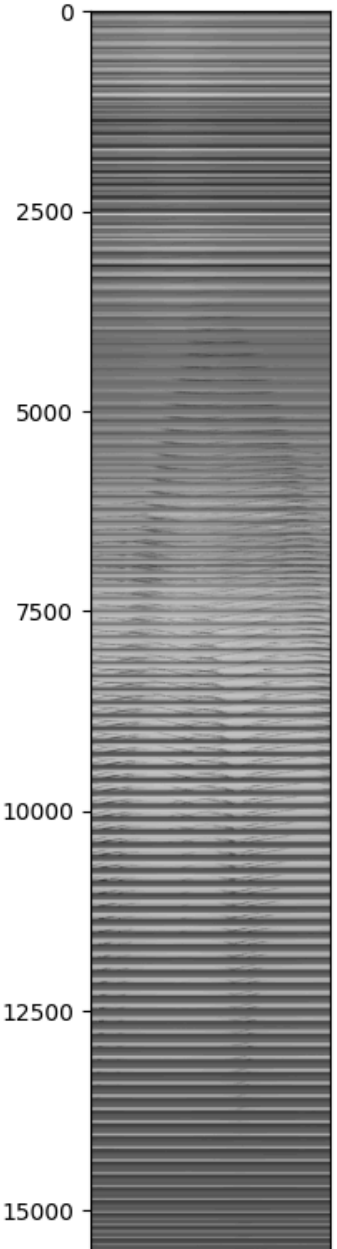


Imagem 3: Achatando matrizes em vetores por meio da função flatten do NumPy.
Fonte: https://www.geeksforgeeks.org/python/flatten-a-matrix-in-python-using-numpy/

Definimos uma taxa de fps para escolher quantos frames pegaremos por segundo do vídeo, já que pegar todos os frames pode ser muito custoso quando lidamos com vídeos com altas taxas de fps (>60 fps), e onde muitas vezes não há mudanças significativas entre frames consecutivos. Feito isso, tornamos cada vetor-frame (como chamaremos frames achatados em vetores) em vetor coluna de uma matriz D e temos a mesma plotada na Imagem 4.



O vídeo original tem duração de 50s e dimensão de 320x240 pixels, para trabalharmos com ele reduzimos sua dimensão em 50% e pegamos seus frames em uma taxa de 60fps, portanto, passamos a trabalhar com (50*60) frames de dimensão 160x120 pixels, ou seja, 3000 matrizes 120x160, que tornam-se os 3000 vetores colunas de dimensão 19200x1 (120*160=19200) que compõem D, que como podemos ver ao lado possui dimensão 19200x3000. A plotagem de D utilizando matplotlib toma 17s no Colab, uma vez que mesmo reduzindo em 50% os frames ainda temos como resultado uma matriz de 19200*3000B=57,6MB, como consequência é recomendado sempre salvá-la e trabalhar sobre cópias, para evitar ter que gerá-la novamente, o que pode levar muito tempo e exigir muito dos recursos de hardware (ou seja, também muito gasto de energia) dependendo do tamanho do vídeo.

Ao observarmos D e suas colunas, que correspondem aos frames do vídeo a cada (1/60)s, podemos reparar que há um grande padrão de fundo que não muda, e padrões sutis em forma de curvas definidas por sombras que mudam de posição ao longo das colunas de D, ou seja, ao longo do tempo. Acreditamos que este padrão de fundo é justamente o fundo da imagem contido em um vetor coluna

que se repete horizontalmente em D, e as sombras são os objetos (pessoas) em movimento, como o fundo pode ser visto como um único vetor que engloba a maior parte de variabilidade de D, ele deve corresponder à primeira componente principal (PCA 1) de D, mas será que somente com uma componente principal é possível obter mesmo toda a variância contida no fundo?

Imagem 4: Matriz de dados do vídeo (D) plotada.

Porque a PCA é interessante para imagens e vídeos?

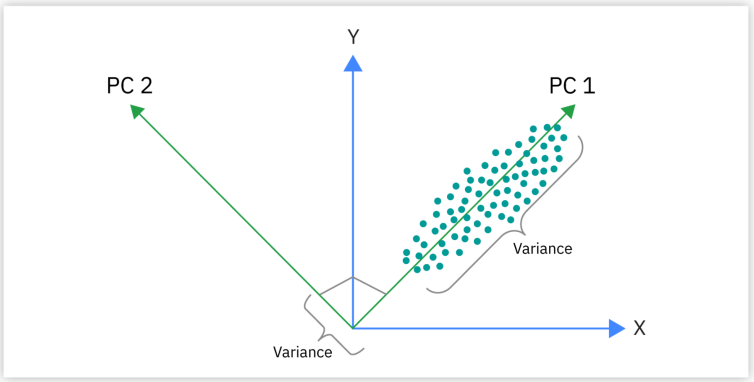


Imagem 5: Visualização da PCA.
Fonte: https://www.ibm.com/br-pt/think/topics/principal-component-analysis

A PCA, conforme observado acima, busca reduzir a dimensionalidade de um conjunto de dados buscando os vetores, neste caso as componentes principais (“PC”), que melhor representam os dados ao explicar a variância dos mesmos. O exemplo da Imagem 6 visa expor melhor isso ao tentar usar PCA para decompor a matriz de pixels em *grayscale* de uma imagem de tabuleiro de xadrez. Observando os padrões da imagem e lembrando que neste padrão de codificação de cores preto e branco correspondem a 0 e 255, respectivamente,

vemos que esta é simplesmente composta de vetores coluna que seguem um padrão de branco e preto ($v_{bp} = [255, 0, \ldots, 255, 0]^T$), preto e branco ($v_{pb} = [0, 255, \ldots, 0, 255]^T$), e apenas preto (vetor nulo), sendo que os dois primeiros vetores podem ser obtidos um a partir do outro por meio da expressão $v_{bp} = [255, 255, \ldots, 255, 255]^T - v_{pb}$, e o último ao multiplicar por 0 (zero) qualquer vetor, ou seja, faz sentido imaginar que a partir de apenas duas componentes principais seja possível representar o tabuleiro e toda a sua variância, o que é confirmado abaixo, onde somando as variâncias explicadas por PCA 1 e 2 chegamos ao acumulado de 99,96% da imagem original, praticamente 100%, e a partir de PCA 3 temos os dados originais completamente explicados. É interessante notar que a imagem, de dimensão 128x128 pixels corresponde a uma matriz de posto 128, ou seja, que poderia ser decomposta em até 128 componentes, mas com apenas 3 (aproximadamente **2,34%**) foi possível representar 100% da imagem, isto se deve à natureza da mesma, que por possuir um padrão simples, torna-se "explicada" com facilidade pela análise de componentes principais.

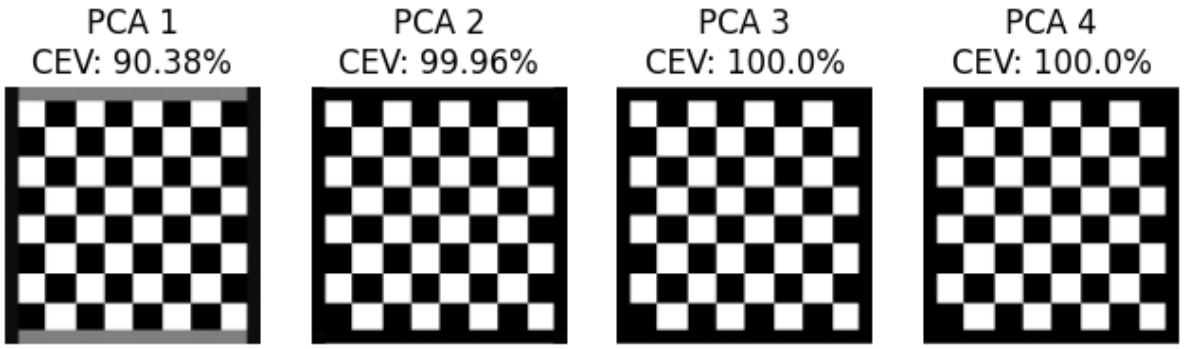


Imagem 6: Exemplo de uso da PCA em imagem de tabuleiro de xadrez 128x128px. CEV corresponde a *Cumulative Explained Variance*.

Ao observar as componentes principais do tabuleiro de xadrez na Imagem 7 observamos que os padrões de branco e preto realmente aparecem como parte dos vetores, e buscando analisar semanticamente vemos que PC1 representa a borda do tabuleiro nos extremos (127, cinza) e o xadrez no meio (255 e 0, branco e preto), PC2 as bordas pretas e a matriz de transição entre branco e preto (v_{bp}) e preto e branco (v_{pb}) ao fazer-se $PC2 - PC1$, e PC3 a complementar de PC2 que deve ser utilizada provavelmente para ajustar as cores da borda, já que $0.5PC3 - PC1$ faz as bordas cinzas se tornarem pretas sem mudar o padrão interno do xadrez.

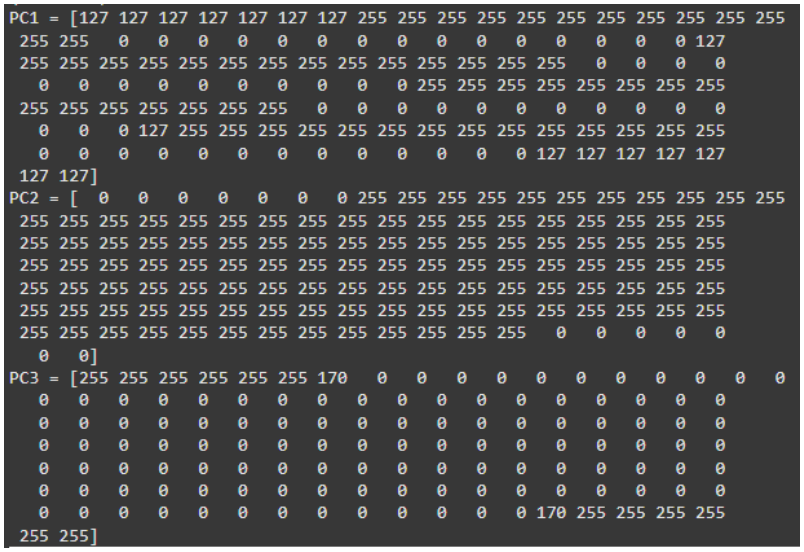


Imagem 7: PCA 1, 2, e 3 com valores inteiros em *grayscale* obtidas para a imagem do tabuleiro de xadrez.

Finalmente, vamos analisar o quando da variância da nossa matriz de dados do vídeo (D) é explicada por suas componentes, como sua dimensão é 19200x3000, significa que tem posto 3000 e com isso no máximo 3000 componentes, no entanto, boas notícias surgem da interpretação do gráfico da Imagem 8: PCA 1 explica praticamente **90%** da variância, e a partir de PCA 350 100% é explicado, ou seja, com apenas 350/3000≈**11,67%** das componentes principais conseguimos representar 100% do vídeo. As porcentagens obtidas realmente nos fazem crer que PCA 1 poderá representar o fundo da imagem.

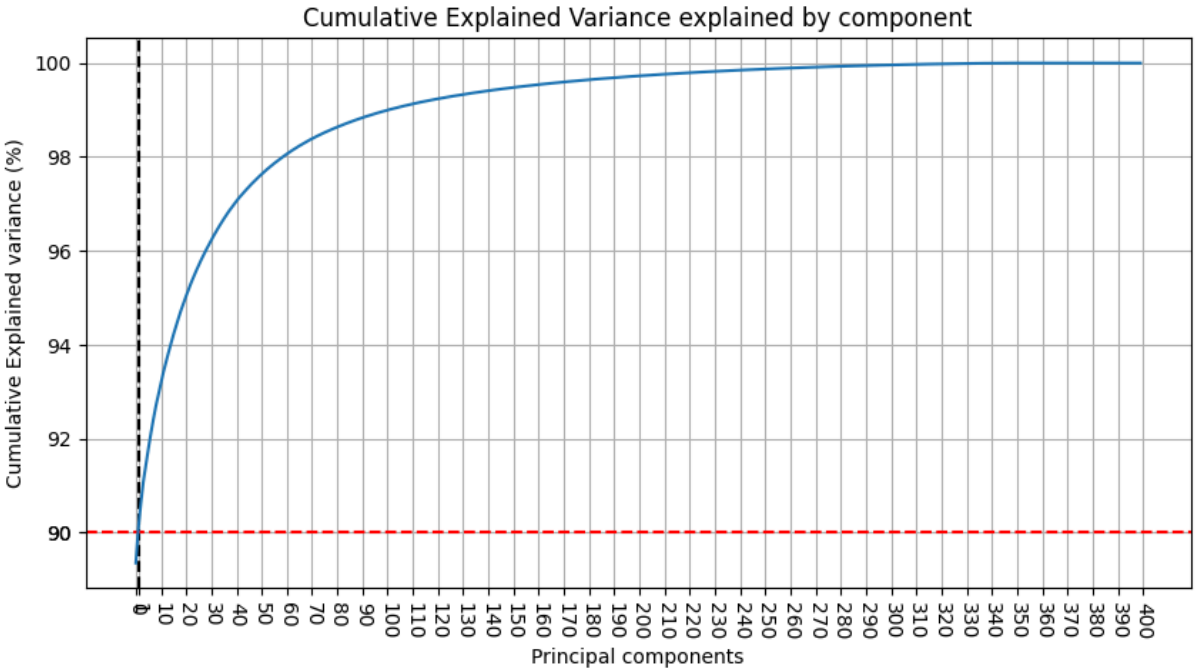


Imagem 8: Gráfico da porcentagem da variância explicada cumulativamente pelas componentes principais de D.

Remoção de Fundo e geração de vídeo sem o mesmo

Chegamos finalmente à etapa principal deste projeto. Tendo D, plotada na Imagem 4, calculamos PCA1, a matriz gerada a partir da primeira componente principal de D, e observando o resultado de sua plotagem na Imagem 9 encontramos uma matriz que corresponde a D sem as sombras observadas, como D e PCA1 são matrizes de mesma dimensão, basta executar $D - PCA1$ para obter justamente apenas as sombras, que acreditamos ser os objetos em movimento.

D **PCA1** **People**

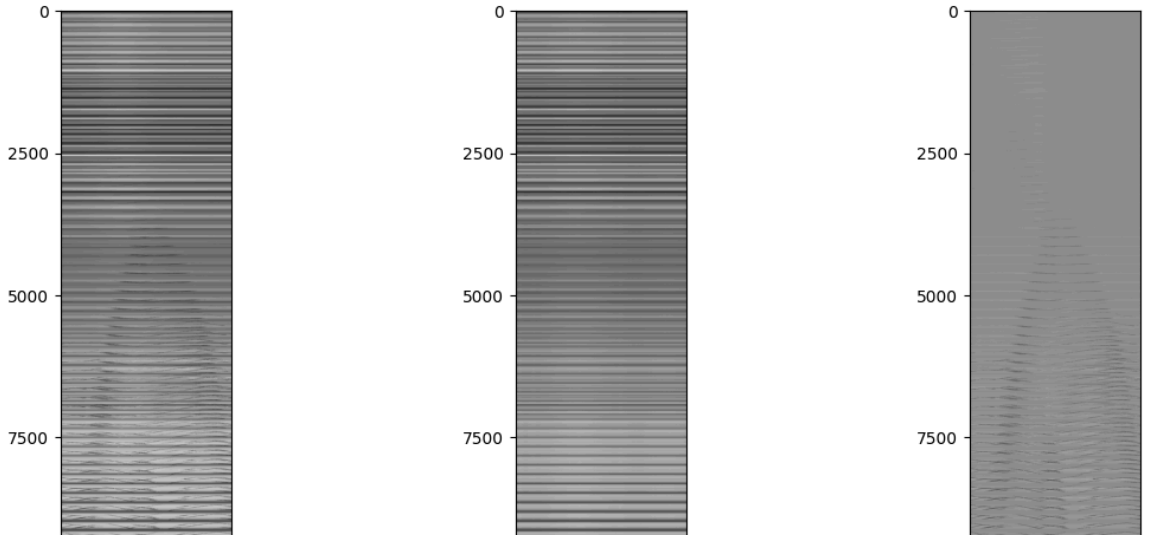


Imagem 9: Operações matriciais de remoção de fundo.

Agora vamos "desachatar" o vetor-frame da posição correspondente ao momento 0:10 (10s de filme) obtido das matrizes D, PCA1, e $D - PCA1$ (People) para ver se alcançamos os resultados desejados. Para obter o vetor-frame do segundo S do filme basta pegar o vetor coluna da coluna (S*60), pois obtemos 60fps para montar nossa matriz de vetores-frames.

Observando os resultados plotados expostos na Imagem 10 vemos que em grande parte alcançamos nosso objetivo: conseguimos remover o fundo e evidenciar as pessoas. É curioso notar que PCA1 explica a variância de tudo que é constante entre os frames, tanto que o gramado é completamente removido na imagem sem fundo, mas como na filmagem uma das pessoas para por um tempo no meio da estrada para conversar e também outras passam e se mantém em posições específicas, vemos seus vultos em PCA1, consequentemente, como diminuímos PCA1 de D, o resultado gera as pessoas dos vultos com cores mais claras, afinal, parte da cor delas está em PCA1. Outro fenômeno interessante é que ainda é possível observar parte da estrada na terceira plotagem, isso se dá pela variabilidade gerada nos pontos da estrada onde as pessoas mais passam, fazendo com que PCA1 não consiga explicar toda a sua variância, e parte dela "sobre" em $D - PCA1$. Usando mais componentes podemos reduzir esse efeito ao obter mais da variância de todos os elementos de fundo, simulamos isso com 1, 2, 3, e 20 componentes principais, conforme pode ser observado na Imagem 11, e percebemos

que com quanto mais componentes explicando o fundo, também mais dos objetos em movimento (pessoas) também é explicado, e na imagem resultante do processo estes aparecem mais apagados, como buscamos um resultado em que eles permaneçam mais nítidos faz mais sentido executar a remoção de fundo apenas com PCA 1, mesmo ainda apresentando pequenos traços do fundo.

Imagem 10: Imagem do frame no momento 0:10 original, apenas com fundo, e sem fundo, respectivamente (da esquerda para a direita).

Agora que temos um método de remoção de fundo definido apenas precisamos recompor ("desachatar") os vetores-frames de nossa matriz **People** e com isso temos um array de frames que usamos para gerar o vídeo original com fundo removido (v1_minus_bg.mp4) com o auxílio da biblioteca OpenCV. Podemos observar o resultado na Imagem 12.

Ao dar *play* no vídeo observamos que o centro, objetos de mão, e pessoas vestidas com cores destoantes (pessoa de casaco azul claro e chapéu/boné branco que aparece em 0:16) do padrão geral apresentam pixels pretos na gravação com fundo removido, isto se deve provavelmente pelo fato de corresponderem à maior variabilidade de D, necessitando de mais componentes principais para serem representados, e como PCA 1 não os engloba, ao ser subtraída de D, acaba gerando pixels pretos onde eles se encontram.

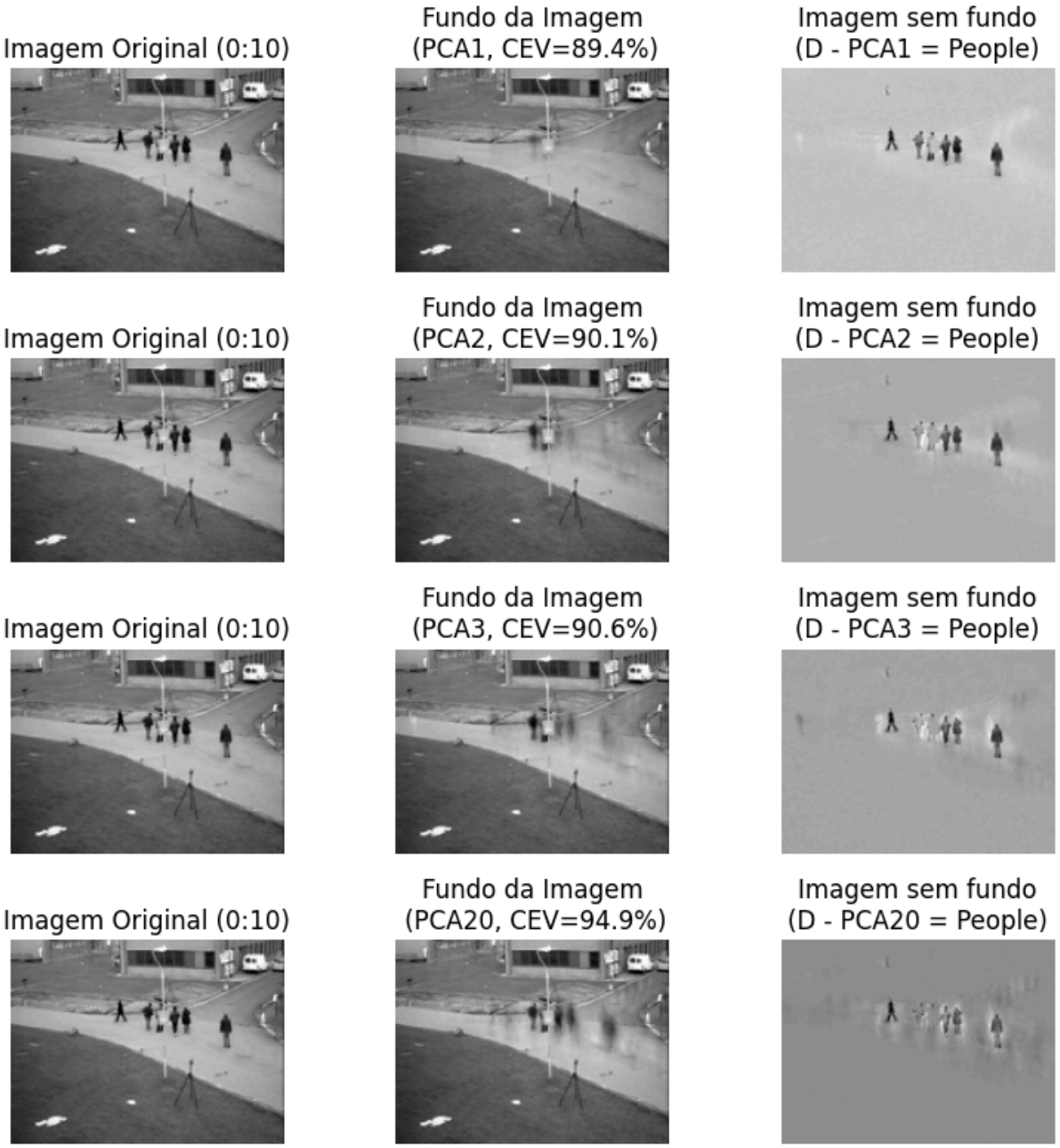


Imagem 11: Remoção de fundo utilizando 1, 2, 3, e 20 componentes principais de D.

Imagem 12: Vídeo original com fundo removido.

Segmentando as objetos em vídeo por meio da PCA

Ressaltar objetos em movimento em gravações é muito útil para observar dinâmicas de comportamento, seja de pessoas, animais, veículos, microorganismos, entre outros, mas certas mudanças podem ser importantes e com isso surge a necessidade de detectá-las, como situações de violência, padrões diferentes de migração de aves, acidentes de trânsito. Para detectar algum elemento é necessário primeiramente segmentá-lo, para que posteriormente seja possível analisar como as suas partes e todo interagem com o ambiente. Tratando de imagens, as partes são pixels, então a tarefa se reduz a encontrar as coordenadas destes para cada elemento de interesse, o que pode ser uma tarefa muito difícil, já que dentre os objetos contidos num vídeo um grande conjunto está estático, este corresponde justamente ao fundo, e eis que surge a ideia: se removermos o fundo o que sobra em destaque são justamente os pixels dos elementos dinâmicos, e a partir deles temos as coordenadas que compõem os objetos.

O objetivo agora é encontrar pixels que se destacam em frames com fundo removido, como o da Imagem 10, ao observá-lo percebemos que quase toda a imagem está numa mesma cor cinza claro onde era o fundo, enquanto as pessoas possuem cores destoantes. O vídeo está em *grayscale*, então os valores de cor estão entre 0 (preto) e 255 (branco), o cinza claro está mais próximo de 255, então, se plotarmos um histograma dos valores de escala de cinza dos pixels de um frame, devemos encontrar uma distribuição centrada perto deste valor, o que observamos na Imagem 13.

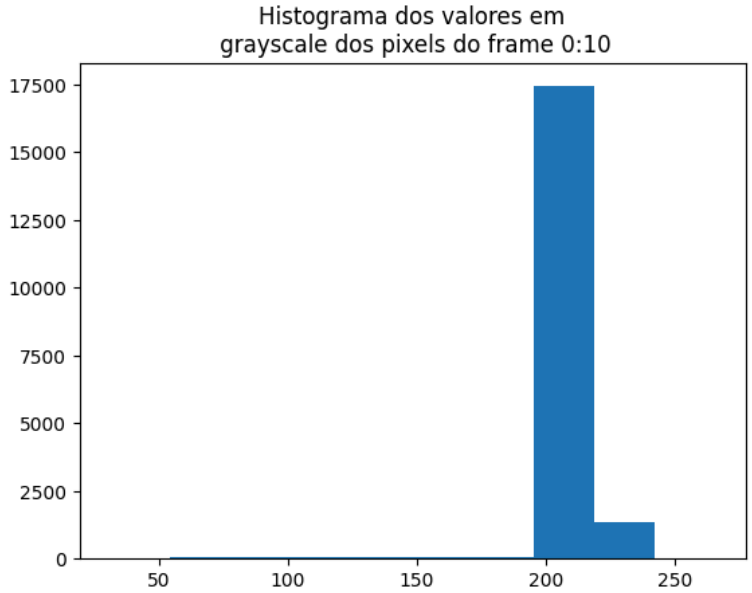


Imagem 13: Histograma dos valores de cor do frame 0:10.

Observando o histograma acima encontramos uma distribuição centrada em 200, valor de cor do fundo, e com uma sutil assimetria à esquerda, onde encontramos uma cauda com valores mais próximos do 0 (preto), esses valores correspondem justamente aos das cores de pixel das pessoas observados em *zoom* (Imagem 14), ou seja, obtendo as posições dos pixels com cores contidas no extremo da cauda esquerda da distribuição conseguimos segmentar as pessoas no frame.

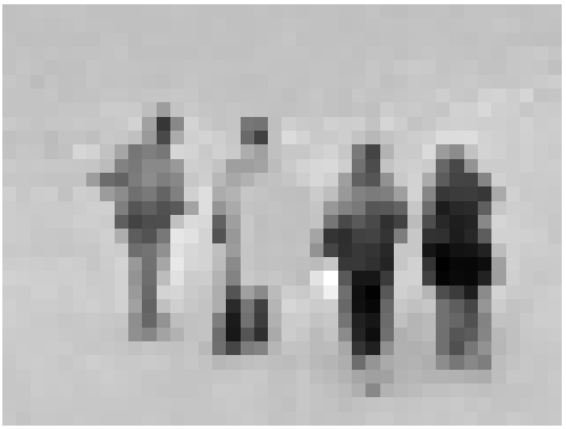


Imagem 14: Cores dos pixels das pessoas no frame 0:10 observadas de perto.

Experimentamos com diferentes quantis criando um canvas branco e pintando de preto na posição dos pixels de cor abaixo do quantil, e com o q(0,02) obtivemos a melhor segmentação, observada na Imagem 15.

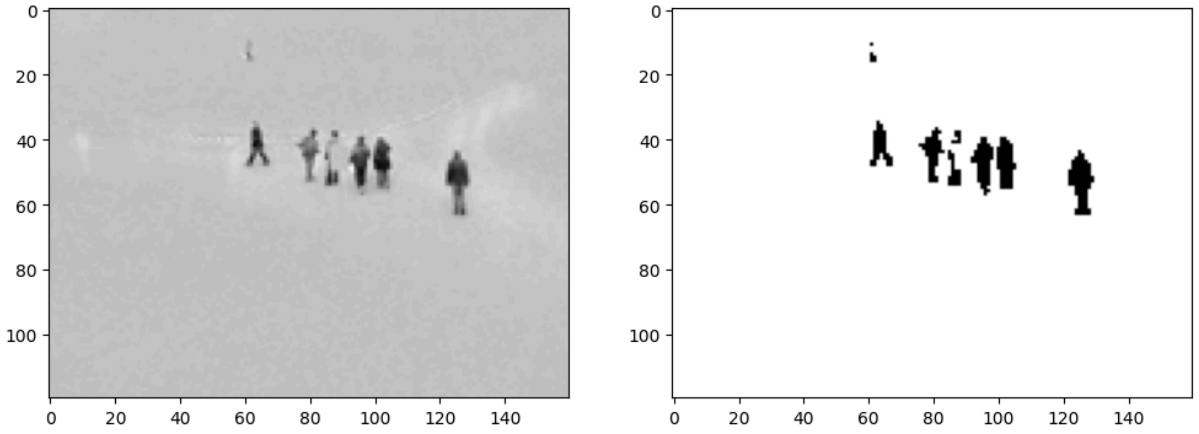


Imagem 15: Frame 0:10 com fundo removido (esquerda), e pessoas dele segmentadas em canvas branco (direita).

Conclusão

Este trabalho trouxe muitas explorações do potencial da PCA para diferentes aplicações, com ela é possível reduzir a quantidade de informação utilizada para representar diferentes dados e ainda mantendo um alto nível de qualidade, como vimos nas Imagens 6 e 8 com um número relativamente pequeno de componentes é possível representar 100% da variância observada, mas há uma dependência da natureza do que é observado, como vimos na imagem do tabuleiro de xadrez, dados com padrões simples e bem definidos são facilmente explicados pela PCA, e por isso que ela justamente funciona tão bem para imagens e vídeos, porque estes possuem muitos elementos e padrões redundantes que acabam implicando em baixa variabilidade e, por conseguinte, em um nível de variância qua acaba conseguindo ser explicado por poucas componentes principais.

As metas deste trabalho foram em grande parte alcançadas, no entanto certos detalhes impedem a generalização desta solução, fazendo com que haja uma parte “artesanal” na tarefa de remoção de fundo e segmentação de objetos dinâmicos, esta parte consiste dos ajustes para adequar-se à natureza dos vídeos e imagens, como por exemplo observar o quanto de variabilidade surge em elementos estáticos devido a fatores de gravação como iluminação, e os relativos ao comportamento de elementos em movimento, como foi no caso da remoção incompleta da imagem da estrada devido à caminhada das pessoas trazer variância. O método que definimos para segmentar as pessoas também dependeu de ajustes e observação para definir um quantil adequado que, apesar de ter gerado resultados satisfatórios, pode não gerar

para outros cenários. Além desses desafios, é inegável o de gerenciamento de memória, pois trabalhar apenas com um vídeo de resolução baixa e de 50s foi o suficiente para em certos momentos utilizar toda a RAM disponibilizada no ambiente do Colab, no entanto, muito trabalho é paralelizável, como por exemplo a transformação de frames em vetores coluna, a subtração das entradas de D pelas de PCA1, a segmentação de objetos em cada frame, a junção de frames em vídeo, e talvez até a plotagem de imagens.

Por fim, podemos dizer que este projeto trouxe muitos conhecimentos e de fato maior entendimento da análise de componentes principais (PCA), além de expor em que situações esta se mostra mais adequada e eficiente e servir de inspiração para usá-la alinhada a outros métodos, buscando resultados cada vez mais interessantes.

Referências

- Apostila de COCADA
- Notas de Aula
- fast.ai course: Computational Linear Algebra
- Notebook Background Removal with Robust PCA compartilhado por e-mail pelo professor
- The Magic of Principal Component Analysis through Image Compression
- Aulas de Probabilidade e Estatística
- https://www.ime.usp.br/~mbranco/MedidasdeAssimetria_2014.pdf