Aplicando projeções perspectiva e geométrica (trabalho 4)

Introdução ao Processamento de Imagem Digital

Randerson A. Lemos (103897) 2022-1S

1 Introdução

Encontrar a correspondência entre pixels de diferentes imagens que capturam a mesma informação de ângulos, posições e perspectivas diferentes é uma problema muito comum abordado pelo processamento de imagens. Esses problemas são genericamente denominados de problemas de Registro e sua resolução normalmente se desenvolve pelo uso dos conhecimento de álgebra linear e suas ideias de espaços vetoriais e transformações entre tais espaços por meio de matrizes de transformação. Para exemplificar em termos práticos os problemas de registro, observe a Figura 1 a seguir retirada do material de aula.

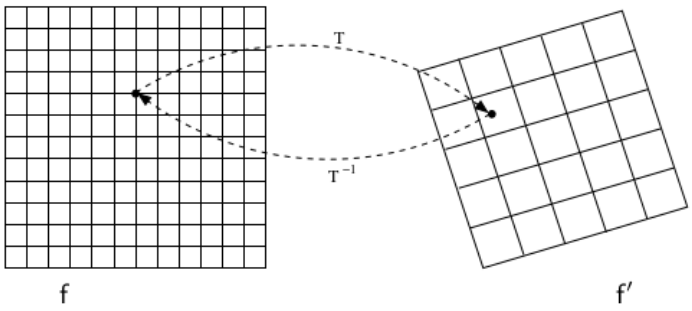

Figura 1: Aplicação da matriz de transformação $\mathbf{T}$ entre as imagens $\mathbf{f}$ e $\mathbf{f}'$ (**fonte**: material de aula).

Nesse figura, temos duas imagens $\mathbf{f}$ e $\mathbf{f}'$ cujo pixels estão relacionados pela matriz de transformação $\mathbf{T}$, ou seja

$$\mathbf{f}' = \mathbf{T}\mathbf{f}. \tag{1}$$

Na Equação 1, a matriz de transformação $\mathbf{T}$ está fazendo o mapeamento da imagem $\mathbf{f}$ para a imagem $\mathbf{f}'$. Tal mapeamento pode ser representado como $\mathbf{T} : \mathbf{f} \mapsto \mathbf{f}'$. Um aspecto importante que precisa ser levado em conta nos problemas de registro é o da natureza discreta das informações presentes, isto é: das imagens digitais. Devido a essa natureza, o mapeamento direto $\mathbf{T} : \mathbf{f} \mapsto \mathbf{f}'$ nem sempre vai conduzir um pixel válido da imagem de origem $\mathbf{f}$ para um pixel válido da imagem de destino $\mathbf{f}'$, uma vez que depois de aplicada a matriz de transformação não há garantias que o pixel transformado seja discreto e esteja contido dento do domínio da imagem de destino $\mathbf{f}'$. Para contornar esse aspecto prático da aplicação das matrizes de transformação em domínios discretos, a solução encontrada foi fazer uso da transformação inversa e de métodos de interpolação, isto é:

$$\mathbf{f} = \mathbf{T}^{-1}\mathbf{f}'. \tag{2}$$

A aplicação da transformação inversa garante que todos os pixels da imagem de destino $\mathbf{f}'$ sejam considerados no problema de encontrar seu correspondente na imagem de origem $\mathbf{f}$ e a aplicação das técnicas de interpolação garante que todos os pixels transformados estejam associados a algum pixel válido da imagem de origem $\mathbf{f}$. Existem diversas técnicas de interpolação, sendo algumas das mais conhecidas as seguintes: Interpolação pelo Vizinho Mais Próximo, Interpolação Bilinear, Interpolação Bicúbica e Interpolação por Polinômios de Lagrange. Essas interpolações são amplamente discutidas no material da disciplina como também no documento de instruções desse trabalho.

Os problemas de registro são principalmente organizados em problemas de transformações geométricas e problemas de transformações projetivas. Ambos esses problemas serão discutidos em mais detalhes a seguir.

1.1 Transformações geométricas

Os problemas de transformação geométricas consistem de uma transformação espacial e uma interpolação. A transformação espacial está associada as transformações afins de rotação, escala, translações, espelhamento e

cisalhamento. Nessas transformações, o paralelismo das linhas e curvas da imagem transformada é preservado, mas dimensões lineares, de área ou volumétricas não. Em coordenadas homogêneas e considerando o caso bidimensional, essas transformações podem ser generalizadas por

$$\begin{bmatrix} X' \\ Y' \\ W \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} X \\ Y \\ W \end{bmatrix}. \tag{3}$$

Cada uma das transformações de rotação, escala, translação, espelhamento e cisalhamento terão uma matriz de transformação específica (a matriz com os coeficientes $a, b, \cdots, i$). A seguir, vamos conhecer a forma das matrizes das transformações de escala, translação e rotação.

1.1.1 Transformação de Escala

A transformação de escala para o caso 2D pode ser realizada a partir da matriz de transformação em coordenadas homogêneas apresenta a seguir.

$$\begin{bmatrix} X' \\ Y' \\ W \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ W \end{bmatrix}. \tag{4}$$

Então, para aplicar um aumento ou redução de escala da imagem, o usuário precisa definir o valor de escala S_x a ser aplicado ao longo do eixo-x e o valor de escala S_y a ser aplicado ao longo do eixo-y.

1.1.2 Transformação de Translação

A transformação de translação para o caso 2D pode ser realizada a partir da matriz de transformação em coordenadas homogêneas apresenta a seguir.

$$\begin{bmatrix} X' \\ Y' \\ W \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ W \end{bmatrix}. \tag{5}$$

Então, para aplicar uma translação na imagem, o usuário precisa definir o valor de translação t_x a ser apicado ao longo do eixo-x e o valor de translação t_y a ser aplicado ao longo do eixo-y.

1.1.3 Transformação de Rotação

A transformação de rotação para o caso 2D pode ser realizada a partir da matriz de transformação em coordenadas homogêneas apresenta a seguir.

$$\begin{bmatrix} X' \\ Y' \\ W \end{bmatrix} = \begin{bmatrix} cos(\alpha) & -sin(\alpha) & 0 \\ sin(\alpha) & cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ W \end{bmatrix}. \tag{6}$$

Então, para aplicar uma rotação na imagem, o usuário precisa definir o valor de rotação α a ser aplicado na imagem.

1.2 Transformações projetivas

As transformações projetivas estão associadas ao processo de projeção de pontos dispostos tridimensionalmente no espaço em um plano bidimensional. Esses transformações são principalmente do tipo ortográfica ou perspectiva. A projeção ortográfica é uma transformação que realiza o mapeamento de pontos tridimensionais sobre o plano da imagem, tal que os pontos são projetados ao longo de linhas paralelas na imagem (informação extraída do material de aula). Já as projeções perspectivas, apesar de haver o mesmo mapeamento de um espaço tridimensional para um espaço bidimensional (um plano), o tamanho do objecto projeto assim como algumas razões de suas medidas lineares e de área são dependentes do aumento da distância do centro de projeção. Essa dependência está associada a percepção de profundida do sistema visual humano. A ideia das transformações projetivas está apresentada na Figura 2.

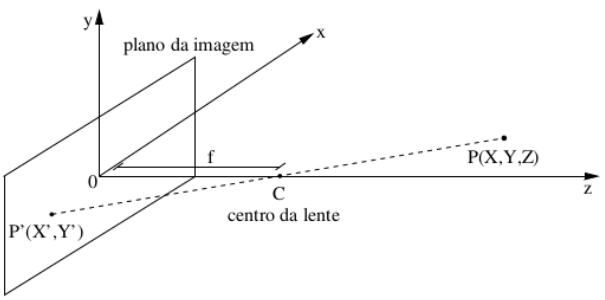

Figura 2: Modelagem geométrica de transformação projetiva (**fonte**: material de aula).

Existem algumas maneiras de se calcular a matriz de transformação de uma transformação projetiva. No caso particular das projeções perspectivas, umas das formas de ser obter tal matriz é a partir de quatro pontos não colineares e a resolução das seguintes equações

$$X' = \frac{aX + bY + c}{iX + jY + 1} \qquad Y' = \frac{fX + eY + f}{iX + jY + 1} \tag{7}$$

considerando-se os quatro pontos não colineares em questão. Os valores obtidos da resolução do sistema de equações para as variáveis a, b, c, d, e, i, j são os coeficientes da matriz de transformação

$$\begin{bmatrix} a & b & c \\ d & e & f \\ i & j & 1 \end{bmatrix}$$

da projeção perspectiva em questão.

2 Solução

A solução utiliza a linguagem de programação Python e conta com o auxílio do gerenciador de projetos e pacotes Conda. Assumindo que o usuário tenha o Conda instalado em sua máquina, a configuração do projeto pode ser feita pela execução do comando `conda env create -f environment.yml` a partir da pasta do projeto **trab3**. Esse comando cria o ambiente de trabalho **mc920-trab3** e instala os seguintes módulos: opencv, numpy, scipy, pandas, matplotlib. Finalizada a configuração do ambiente de trabalho em questão, o usuário deve executar o comando `source source.sh`[1] para carregar as variáveis de ambiente adequadas e, assim, poder usar os programas do projeto dentro do próprio ambiente de trabalho recém configurado.

Dos arquivos presentes na pasta do projeto **trab3**, destacam-se as pastas **assets**, **classes**, **tex** e o programa **main.py**. A pasta **assets** contém vídeos no formato png ou mpg que podem ser utilizados para a aplicação das técnicas de identificação de transições abruptas implementadas. A pasta **classes** contém arquivos com as implementações das diferentes técnicas de identificação de transições abruptas em formato de classes. Os arquivos das classes das técnicas são: **blocksdifferences.py**, **edgesdifferences.py**, **histogramdifferences.py**, e **pixelsdifferences.py**. A pasta **tex** contém os arquivos Latex deste relatório. O programa **main.py** contém as implementações necessárias para aplicar as técnicas de identificação de transições abruptas no vídeo de entrada fornecido pelo usuário. Informações pertinentes das implementações serão apresentadas a seguir.

2.1 Main.py

O programa **main.py** é responsável pela aplicação de todas as técnicas consideradas de identificação de transições abruptas entre quadros consecutivos de vídeos. Para ser executado, esse programa precisa receber o parâmetro de entrada **video_entrada**:

- ao parâmetro **video_entrada** deve-se fornecer o nome do arquivo do vídeo a ser utilizado para aplicação das técnicas de identificação de transições abruptas.

O programa **main.py** disponibiliza tanto um vídeo no formato *mp4* dos quadros considerados como pertencentes a transições abruptas como também gráficos em formato png da evolução dos valores de distância utilizados para determinação de transições abruptas ou não. Todo esse conteúdo é salvo dentro da pasta **out** que está localizada dentro da pasta do projeto **trab3**.

[1] O comando que configura o ambiente de trabalho mc920-trab3 precisa ser executado apenas um vez. Assim sendo, depois que este ambiente está configurado, o usuário precisa apenas executar o comando `source source.sh`

Um exemplo de como executar o programa **main.py** utilizando os recursos contidos dentro do próprio projeto é: `python3 main.py -video_entrada=assets/lisa.mpg`.

O programa **main.py** executa quatro funções principais, as quais são: **main_pixels_differences(path, stem)**; **main_blocks_differences(path, stem)**; **main_histog_differences(path, stem)**; e **main_edges_differences(path, stem)**. Cada uma dessas funções é responsável por executar a técnica de identificação de transições abruptas que encapsula várias vezes utilizando diferentes combinações de parâmetros de entrada (tolerâncias e limiares). Essas funções também estão encarregadas de salvar todo o conteúdo gerado (os vídeos e gráficos) com nomes intuitivos dentro da pasta **out**. A seguir olharemos para essas funções e, principalmente, para as técnicas de identificação de transições abruptas associadas.

2.2 main_pixels_differences(path, stem)

Esse função roda a classe **PixelsDifferences** sobre as tuplas resultantes do produto cartesiano entre os conjuntos $MPNDS \times MPNNS$ cujos elementos são

$$MPNDS = \{0.1; 0.2; 0.3; 0.4; 0.5; 0.6; 0.7\}$$

e

$$MPNNS = \{0.1; 0.2; 0.3; 0.4; 0.5; 0.6; 0.7\}.$$

MPNDS é sigla de *Maximum Pixel Normalized Distances* e MPNNS é sigla de *Maximum Pixel Normalized Numbers*. Os valores do conjunto MPNDS fazem referência a tolerância T_1 utilizada para verificar se dois pixels são ou não significativamente diferentes. Os valores do conjunto NPNNS fazem referência ao limiar T_2, que é utilizado para determinar se o número final de pixels considerados diferentes é suficiente para que os quadros consecutivos sejam classificados como tendo uma transição abrupta. Ambos os valores dos conjuntos em questão apresentam domínio pertencente ao intervalo $[0, 1]$ porque tanto o calculo da tolerância T_1 quando do limiar T_2 e feito de maneira normalizada. Para comparação com a tolerância T_1, o cômputo da distância entre pixels é feita da seguinte maneira

$$d_{xy} = |Q_i(x, y) - Q_{i+1}(x, y)|/255,$$

em que $Q_i(x, y)$ é o valor de intensidade do pixel da posição (x, y) do i-ésimo quadro em escala de cinza do vídeo analisado. Como a diferença máxima entre pixels em escala de cinza representados com 1 bite é de 255, o valor da variável d sempre vai pertencer ao intervalo $[0, 1]$. Para comparação com o limiar T_2 o cômputo do numero de pixels que apresentam transições abruptas é feita assim

$$D_i = \sum_{x=0}^{x=M} \sum_{y=0}^{y=N} (1 \text{ se } d_{xy} > T_1 \text{ caso contrario } 0.$$

Caso D_i, isto é o número de pixels considerados diferentes de acordo com a tolerância T_1 entre o i-ésimo e (i+1)-ésimo quadros, for superior que o limiar T_2, o quadro (i+1)-ésimo é classificado como contendo uma transição abrupta.

2.3 main_blocks_differences(path, stem)

Esse função roda a classe **BlocksDifferences** sobre as tuplas resultantes do produto cartesiano entre os conjuntos $MBNDS \times MBNNS$ cujos elementos são

$$MBNDS = \{0.1; 0.2; 0.3; 0.4; 0.5; 0.6; 0.7\}$$

e

$$MBNNS = \{0.1; 0.2; 0.3; 0.4; 0.5; 0.6; 0.7\}.$$

MBNDS é sigla de *Maximum Block Normalized Distances* e MBNNS é sigla de *Maximum Block Normalized Numbers*. Os valores do conjunto MBNDS fazem referência a tolerância T_1 utilizada para verificar se dois blocos são ou não significativamente diferentes. Os valores do conjunto NBNNS fazem referência ao limiar T_2, que é utilizado para determinar se o número final de blocos considerados diferentes é suficiente para que os quadros consecutivos sejam classificados como tendo uma transição abrupta. Ambos os valores dos conjuntos em questão apresentam domínio pertencente ao intervalo $[0, 1]$ porque tanto o calculo da tolerância T_1 quando do limiar T_2 e feito de maneira normalizada.

Para comparação com a tolerância T_1 o cômputo da distância entre blocos é feita da seguinte maneira

$$d_a i = \sum_{w=0}^{8} \sum_{k=0}^{8} (B_{a_i}(w, k) - B_{a_{i+1}}(w, k))^2/255^2,$$

em que $d_a i$ é a distância quadrática normalizada dos a-ésimos blocos entre os quadros i e $i+1$. Logo $B_{a_i}(w,k)$ é o valor em escala de cinza do pixel que está na posição (w,k) do a-ésimo bloco da i-ésima imagem. Os valores de $d_a i$ estão contidos no intervalo $[0,1]$ (o termo 255^2 é um fator de normalização).

Para comparação com o limiar T_2, o número total de blocos que violam a tolerância T_1 é levantado e, se esse valor ultrapassar o limiar T_2, os quadros consecutivos utilizados na comparação por meio dos seus blocos são considerados como possuidores de transições abruptas. Os valore utilizados em T_2 estão normalizados pelo número total de blocos que cabe nos quadros do vídeo analisado e por isso também estão contidos dentro do intervalo $[0,1]$.

2.3.1 Detalhes de implementação e execução

Como requisitado no trabalho, a classe **BlocksDifferences** é capaz de trabalhar com blocos de 8×8 ou 16×16. No entanto para os resultados mostrados, apenas execuções com blocos de tamanho 8×8 foram considerados. Sobre o problema do casamento exato ou não entre as dimensões dos blocos (que pode ser de 8×8 ou 16×16) e as dimensões da imagem o tratamento dado foi: Começar sempre a subdivisão da imagem em blocos a partir da posição $(0,0)$ e, caso ocorra, as possíveis bordas presentes na parte da direita ou na parte de baixo da imagem são desconsideradas.

2.4 main_histog_differences(path, stem)

Esse função roda a classe **HistogramDifferences** sobre o conjunto dos valores

$$\alpha = \{3,4,5,6\},$$

que são os valores sugeridos na descrição do trabalho. Nenhuma modificação na forma de se calcular a diferença dos histogramas de quadros consecutivos foi realizada de modo que tal cálculo já está explicitado na parte da secção de introdução que trata desse técnica de identificação de mudanças abruptas entre quadros. No entanto vale destacar que os histogramas foram normalizados de modo que a soma dos valores de seus *bins* totaliza 1.

2.4.1 Detalhes de implementação e execução

Para o cálculo dos histogramas, optou-se por utilizar uma implementação própria ao invés de usar alguma função disponibilizada pelos módulos do opencv ou do numpy.

2.5 main_edges_differences(path, stem)

Essa função roda a classe **EdgesDifferences** sobre o conjuto de valores

$$MENDS = [0.010, 0.015, 0.020, 0.025, 0.030, 0.035].$$

MENDS é sigla de *Maximum Edges Normalized Difference*. Os valores desse conjunto estão dentro do itervalo $[0,1]$ e eles representam a diferença máxima normalizada de bordas entre quadros consecutivos tolerada. A normalização é feita considerando a diferença máxima de bordas entre quadros consecutivos possível que ocorreria quando um quadro não teria nenhuma borda e o outro quadro teria apenas 'bordas'. Logo a diferença máxima possível, nessa situação, seria igual ao número de pixels dos quadros do vídeo analisado.

2.5.1 Detalhes de implementação e execução

Para o cálculo do mapa de bordas, foi utilizado o famoso algoritmo Canny que foi desenvolvido por John F. Canny. Esse algoritmo é bastante popular devido a sua versatilidade e robustez. Ele contempla etapas de redução de ruídos, determinação das intensidades do gradiente da imagem, e aplicação de supressão não máxima e teste de hipóteses. Utilizou-se a versão do algoritmo disponibilizada pelo módulo do opencv *Canny* com os parâmetros de entrada fixos. O parâmetro de valor mínimo de 100 e o parâmetro de valor máximo de 200.

Mesmo o algoritmo de detecção de bordas de Canny já conter um etapa de suavização de ruídos, optou-se antes de aplicá-lo, suavizar o quadro de interesse bom um filtro Gaussiano (passa-baixo) de dimensão 5×5.

3 Resultado

Todos os resultados apresentados foram gerados pelo processamento do vídeo **lisa.mpg** que está dentro da pasta **assets**. Devido ao grande número de combinações de parâmetros das quatro técnicas de identificação de mudanças abruptas entre quadros considerados e, consequentemente, o grade número de conteúdo gerado (entre

vídeos e gráficos) serão apresentados aqui apenas um subconjunto desses materiais (caso haja interesse do leitor em verificar a totalidade desse conteúdo basta acessá-los dentro da pasta **out**). Aquim, serão apresentados os gráficos gerados que evidenciam o quão diferentes são dois quadros consecutivos e o número de quadros que foram considerados como detenedores de uma transição abrupta.

3.1 Diferenças entre pixels