Correspondências

Fábia Isabella Pires Enembreck
Programa de Pós-Graduação em Ciência da Computação
Universidade Tecnológica Federal do Paraná
Ponta Grossa-PR, Brasil
Email: fabia.isape@gmail.com

João Eduardo Kozan Silva Fornazari
Bacharelado em Ciência da Computação
Universidade Tecnológica Federal do Paraná
Ponta Grossa-PR, Brasil
Email: joaofornazari@alunos.utfpr.edu.br

1. Introdução

É de longa data que o ser humano busca e compara características de objetos, geralmente com objetivo de qualificá-los para replicar ou descartar. Conforme a tecnologia avançou, comparações se tornaram cada vez mais necessárias e complexas, e a gama de objetivos se expandiu, o que apontou a necessidade de um estudo específico desta área.

Surge então a área de Correspondências, ou *Matching*, que pode ser compreendida como "correspondência em padrões, cor, design; complementares"ou "igual em quantidade; equivalente", de acordo com o Dicionário de Oxford[1].

A tarefa de realizar Correspondências com sucesso não é fácil. Suponhamos que queremos buscar uma bola de futebol laranja nas ruas de Nova Jersey. Para detectar com sucesso uma bola de futebol laranja, é necessário verificar se existem bolas em determinada imagem, se essas bolas são laranjas, se são do tamanho de uma bola de futebol e se possuem as características de uma bola de futebol. São quatro condições complexas de se satisfazer, uma vez que as imagens podem sofrer diversas transformações de iluminação, perspectiva ou escala, por exemplo. Desta forma, um mesmo objeto pode estar muito diferente de uma imagem para outra.

Neste trabalho, serão explicadas duas técnicas para encontrar correspondências em imagens, uma com base em um modelo e outra com base em descritores invariantes a transformações nas imagens. A Figura 1 apresenta um exemplo de correspondência usando *feature matching*, método explicado na Seção 3.2.

Figura 1: Exemplo de correspondências, retirado de [1]

1. Disponível em: en.oxforddictionaries.com/definition/matching

2. Trabalhos relacionados

A Google, em janeiro de 2018, lançou uma ferramenta no aplicativo Google Arts & Culture[2], exclusivo para iOS, que procurava o rosto do usuário em obras artísticas. A funcionalidade estava disponível apenas para os Estados Unidos e chamou muita atenção da mídia. Outros trabalhos com *face matching* são os de Yang [2] que envolvem rostos equivalentes em 2D e 3D, e de Satoh [3] que envolve reconhecimento facial em vídeos.

Técnicas que encontram correspondência por características (ou *feature matching*) podem ser aplicadas em diversas áreas, como por exemplo na segurança, para reconhecimento de digitais [4], e na medicina [5] para localização de veias nos dedos.

Correspondência de modelos (ou *template matching*) é muito utilizada para reconhecimento de objetos, e, dentre os trabalhos existentes nesse âmbito, pode-se citar o de Lee [6] que detecta nódulos pulmonares e o de Pereira [7], que reforça o anti-plágio de imagens com marcas d'água.

Vale citar também os trabalhos de Jain [8] que detecta objetos utilizando modelos deformáveis como base, e o de Schaffalitzky [9] que detecta lugares que aparecem em filmes, possibilitando assim filtrar filmes por localizações específicas.

3. Técnicas para Correspondências

A tarefa de encontrar correspondências em imagens consiste em uma área da visão computacional que busca encontrar recursos equivalentes ou similares entre imagens de uma mesma cena ou objeto. Nesta seção são apresentadas abordagens adotadas para encontrar correspondências em imagens digitais.

3.1. Template Matching

Uma técnica para se encontrar correspondências em imagens digitais consiste em realizar as correspondências por *templates* ou seja, um modelo de objeto. Assim, um

2. Disponível em: https://itunes.apple.com/us/app/google-arts-culture/id1050970557?mt=8

objeto desconhecido na imagem pode ser rotulado como o modelo desejado se eles forem correspondentes [10], [11].

Um exemplo básico para explicar como funciona o *Template Matching* pode ser observado na Figura 2, retirada de [10]. Neste caso, temos um conjunto de imagens binárias para a busca de triângulos. Assim, o modelo é representado como um triângulo (Figura 2-b) para ser comparado com os objetos da Figura 2-a. Vale ressaltar que nem sempre teremos uma correspondência exata, uma vez que podem ocorrem variações no modelo, ou seja, nem todas as suas instâncias podem ser exatamente iguais. Isso se deve a ruídos na imagem, variações no ponto de vista e iluminação, entre outros. [10], [11]

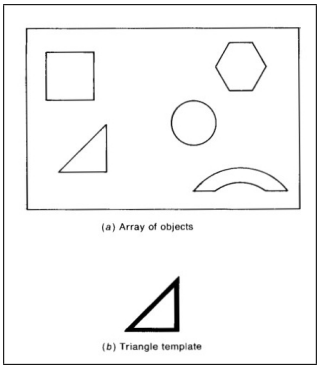

Figura 2: Exemplo de *Template Matching* retirado de [10]. a)Imagem a ser percorrida; b)modelo de objeto.

Segundo o que foi descrito em [10] e [11], uma maneira de encontrar correspondências utilizando um modelo é encontrar a diferença entre o modelo e um campo da imagem $D(x, y)$, assim se essa diferença for menor do que um valor estabelecido L, o objeto correspondente foi identificado. Em outras palavras, para uma diferença

$$D(x, y) = \sum_j \sum_k [F(j, k) - T(j - x, k - y)]^2 \quad (1)$$

em que $F(j, k)$ representa a imagem a ser percorrida e $T(j, k)$ o modelo desejado, existe correspondência se

$$D(x, y) < L \quad (2)$$

3.2. Feature Matching

Para realizar a correspondência por características ou *Feature Matching*, deve-se gerar um conjunto de características para cada uma das imagens a serem comparadas. Um método que pode ser utilizado para isso é o *Scale Invariant Feature Transform (SIFT)*, que transforma dados de imagem em coordenadas invariantes de escala em relação a características locais, gerando um grande número de recursos que cobrem a imagem, independente de escalas e locais [1], [12], [13].

Este método realiza as mesmas operações em múltiplas resoluções e em seguida combina recursos no mesmo nível. Para realizar a correspondência, compara-se individualmente cada vetor de característica para encontrar candidatos correspondentes com uma mesma distância euclidiana, por exemplo. Ou seja, o descritor é obtido por meio de gradientes de imagem, magnitude e orientação, sendo formado por um conjunto de histogramas de gradientes de imagem que devem ser normalizados. [1], [12], [13].

Inicialmente o método deve identificar locais e escalas que podem ser atribuídos com diferentes vistas a um mesmo objeto, o espaço de escala de uma imagem $L(x, y, \sigma)$ pode ser definido como

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \quad (3)$$

em que $G(x, y, \sigma)$ representa um *Gaussiano* de escala variável e $I(x, y)$ a imagem de entrada, considerando que $*$ representa uma operação de convolução. Para localizar *keypoints* no espaço de escala $D(x, y, \sigma)$, pode-se calcular a diferença entre duas escalas separadas por um fator k, ou seja uma escala é k vezes a outra [1].

$$D(x, y, \sigma) = (G(x, y, k_0) - G(x, y, k_0)) * I(x, y) \quad (4)$$

$$D(x, y, \sigma) = L(x, y, k_0) - L(x, y, k_0) \quad (5)$$

Na Figura 3, é apresentado um exemplo de Espaço de Escala e Diferença de *Gaussianas*. De forma incremental, realiza-se a convolução da imagem de entrada com Gaussianas, produzindo imagens separadas por k no espaço de escala, criando uma pirâmide de imagens. Como observa-se, as imagens em azul representam a diferença de Gaussianas, que é calculada a partir da subtração de imagens adjacentes. Quando uma oitava é completada, ou seja, o número de imagens com mesma dimensão, a imagem de Gauss é reamostrada, considerando que ela possui o dobro do valor inicial de σ. Assim, mudando de uma oitava para outra, as imagens são reduzidas pela metade. Os máximos e mínimos locais de $D(x, y, \sigma)$ são detectados comparando cada ponto de amostragem, seus oito vizinhos na imagem atual e nove vizinhos na escala acima e abaixo, para verificar se é maior ou menor que todos os vizinhos. [1].

Depois de detectar o espaço de escala e a diferença das *Gaussianas*, os *keypoints* com baixo contraste e muito próximos das bordas devem ser localizados e, consequentemente, eliminados.

O próximo passo consiste em atribuir uma orientação a cada *keypoint* com base em propriedades da imagem local. Considerando a escala do *keypoint*, seleciona-se a imagem suavizada com *Gauss* na escala mais próxima, permitindo que os cálculos não possuam variações de escala. Para que sejam determinadas as orientações, deve-se calcular os gradientes $m(x, y)$ e a orientação $\theta(x, y)$

$$m(x, y) = \sqrt{(L(x + 1, y)L(x1, y))^2 + (L(x, y + 1)L(x, y - 1))^2} \quad (6)$$

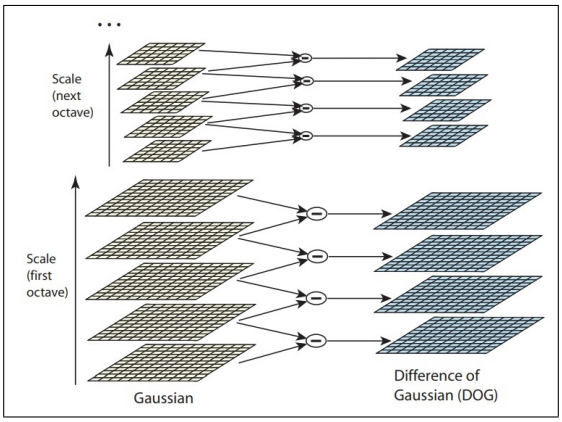

Figura 3: Espaço de escala e diferença de *Gaussianas*. Exemplo de retirado de [1]

$$\theta(x,y) = \frac{tan^{-1}(L(x,y+1)L(x,y-1))}{(L(x+1,y)L(x-1,y))} \tag{7}$$

A partir das orientações de gradiente em cada região ao redor do *keypoint*, um histograma de orientação de 36 posições é obtido (Figura 4). Como existem 360 graus de orientação, as posições do histograma são representadas em intervalos de 10 a 10 graus. As direções de cada gradiente local correspondem aos picos no histograma de orientação, sendo que o pico mais alto no histograma é detectado e, se existir outro pico local que esteja dentro de 80% do pico mais alto, cria-e um ponto chave com essa orientação, podendo existir vários pontos-chave criados no mesmo local e escala, mas com orientações diferentes [1].

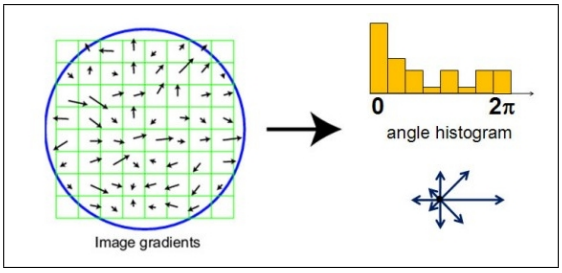

Figura 4: Histograma de Orientação. Exemplo de retirado de [12]

Por fim, para cada *keypoint* é criado um descritor com base nas suas informações extraídas. Com isso, pode-se calcular o descritor com base na magnitude e gradiente em torno da localização de cada *keypoint* em relação a escala.

A invariância de orientação é obtida rotacionando as coordenadas do descritor e as orientações do gradiente em relação à orientação do ponto principal. E em seguida, uma função de ponderação gaussiana é utilizada para atribuir um peso à magnitude de cada *keypoint*, para evitar mudanças repentinas no descritor, além de desconiderar gradientes que estão longe do centro do descritor. Dessa forma, para cada

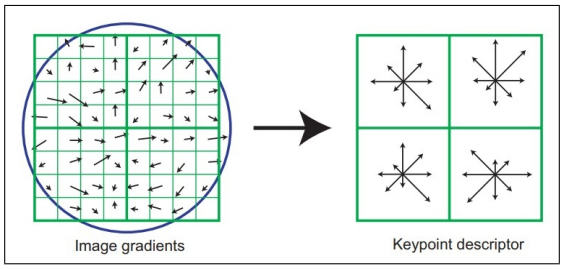

Figura 5: Descritor de *keypoint*. Exemplo de retirado de [1]

keypoint é obtido um descritor formado por um vetor contendo os valores do histograma de orientação. Esse vetor de características deve ser normalizado, com objetivo de reduzir efeitos de possíveis mudanças de iluminação. Como pode ser observado na Figura 5 cada descritor é representado por uma matriz $16x16$, em que cada sub-região $4x4$ representa os histogramas de cada amostra ao redor do *keypoint*, cada seta representa a soma das magnitudes dos gradientes próximos a cada região [1].

Com a extração de descritores e localização de *keypoints* em cada imagem, deve-se estabelecer correspondências entre elas.

Uma estratégia simples para encontrar combinações entre descritores é identificar o seu vizinho mais próximo utilizanda a Distância Euclidiana. Neste método define-se um *threshold*, assim, as correspondências devem estar dentre deste limite.

Quando temos um algoritmo que detecta um grande número de falsos positivos, deve-se diminuir o valor do *threshold*, assim como esse valor deve ser aumentado, caso sejam identificados muitos falsos negativos. Visto que, com interferências de fundo ou oclusões que podem ocorrer nas imagens, um *threshold* global pode não funcionar bem, não encontrando as correspondências de forma correta. Uma medida que pode auxiliar neste problema é comparar a distância do vizinho mais próximo ao do segundo vizinho mais próximo. Essa medida vem do princípio de que correspondências corretas precisam ter o vizinho mais próximo significativamente mais próximo do que a correspondência falsa mais próxima [1], [12].

A Figura 6 apresenta um exemplo de correspondências em imagens utilizando o *SIFT* e a estratégia de encontrar os vizinhos mais próximos, utilizando *OpenCv*.

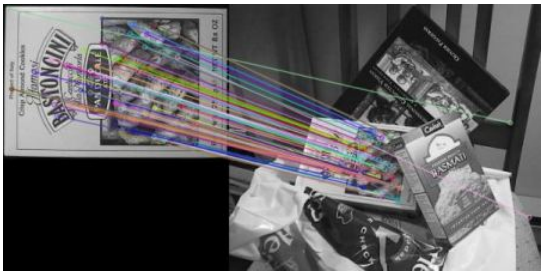

Figura 6: Descritor *SIFT*

4. Conclusão

Este trabalho abordou técnicas para encontrar correspondências em imagens digitais. Para uma técnica que localiza correspondências ser executada com sucesso depende de vários fatores geométricos e fotográficos discretos na imagem, tais como iluminação, profundidade e cores presentes na cena. Detectar estes fatores, seja por meio de *feature matching* ou *template matching*, é igualmente complexo e demanda implementações precisas que frequentemente refletem em custo computacional elevado.

Para atingir os objetivos da Correspondência, que são inversamente proporcionais - a precisão na detecção e o custo reduzido - houve progresso, mas é necessário que se progrida ainda mais, visto que a área vem sendo cada vez mais importante no século XXI, podendo ser útil em várias aplicações da visão computacional.

Referências

[1] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[2] W. Yang, D. Yi, Z. Lei, J. Sang, and S. Z. Li, "2d–3d face matching using cca," in *Automatic Face & Gesture Recognition, 2008. FG'08. 8th IEEE International Conference on*. IEEE, 2008, pp. 1–6.

[3] S. Satoh, "Comparative evaluation of face sequence matching for content-based video access," in *Automatic Face and Gesture Recognition, 2000. Proceedings. Fourth IEEE International Conference on*. IEEE, 2000, pp. 163–168.

[4] U. Park, S. Pankanti, and A. K. Jain, "Fingerprint verification using sift features," in *Biometric Technology for Human Identification V*, vol. 6944. International Society for Optics and Photonics, 2008, p. 69440K.

[5] C.-B. Yu, H.-F. Qin, Y.-Z. Cui, and X.-Q. Hu, "Finger-vein image recognition combining modified hausdorff distance with minutiae feature matching," *Interdisciplinary Sciences: Computational Life Sciences*, vol. 1, no. 4, pp. 280–289, 2009.

[6] Y. Lee, T. Hara, H. Fujita, S. Itoh, and T. Ishigaki, "Automated detection of pulmonary nodules in helical ct images based on an improved template-matching technique," *IEEE Transactions on medical imaging*, vol. 20, no. 7, pp. 595–604, 2001.

[7] S. Pereira and T. Pun, "Robust template matching for affine resistant image watermarks," *IEEE transactions on image Processing*, vol. 9, no. 6, pp. 1123–1129, 2000.

[8] A. K. Jain, Y. Zhong, and S. Lakshmanan, "Object matching using deformable templates," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 18, no. 3, pp. 267–278, 1996.

[9] F. Schaffalitzky and A. Zisserman, "Automated location matching in movies," *Computer Vision and Image Understanding*, vol. 92, no. 2-3, pp. 236–264, 2003.

[10] W. K. Pratt, *Digital image processing: PIKS Scientific inside*. Wiley-interscience Hoboken, New Jersey, 2007, vol. 4.

[11] R. Brunelli, *Template matching techniques in computer vision: theory and practice*. John Wiley & Sons, 2009.

[12] R. Szelis ki, *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[13] D. A. Forsyth and J. Ponce, "A modern approach," *Computer vision: a modern approach*, pp. 88–101, 2003.