

Projeto 4 de Princípios de Visão Computacional

Gabriel Martins de Miranda

130111350

Universidade de Brasília

Email:gabrielmirandat@hotmail.com

Resumo—A presente demonstração faz uso do programa *Find-Object* criado por Mathieu Labb. Nele é possível combinar vários detectores e descritores usados para reconhecer e acompanhar um objeto numa cena. Os objetos usados para as comparações entre os diferentes métodos foram uma dama da carta de baralho e um controle remoto para DVD player. Os detectores/descritores usados foram: SIFT/SIFT , SURF/SURF , FAST/BRIEF e ORB/ORB.

I. INTRODUÇÃO

A primeira coisa que tem-se que ter em mente é a diferença entre os termos *detector* e *descritor*. *detector* se refere ao método usado para se detectar as features. *descritor* se refere ao método usado para casar as features encontradas de uma imagem em outra.

Vamos introduzir agora uma breve explicação dos métodos utilizados:

1) SIFT/SIFT

SIFT - Scale-Invariant Feature Transform

Sift é um detector e descritor. Ele usa a invariância à rotação do detector de cantos de Harris e corrige a não-invariância à escala utilizando diferença da gaussiana (DoG) para vários tamanhos de escala. Após isto, os keypoints encontrados são refinados e são eliminados qualquer keypoint de baixo contraste e de borda, e o que sobra são pontos de interesse mais robustos. Logo em seguida, uma orientação é atribuída a cada keypoint para que se conquiste invariância à rotação na imagem. Uma vizinhança dependente de escala, além da magnitude do gradiente e direção são calculados na região. Um histograma de orientações com peso é criado. Ao final desta etapa são criados keypoints com mesma localização e escala, mas direções diferentes. Isto contribui para a estabilidade do matching. Para o descritor do keypoint, uma vizinhança 16x16 é tomada e divida em 16 sub-blocos 4x4. Para cada sub-bloco, 8 orientações de histograma são criados. Logo, temos um total de 128 valores de histograma disponíveis. Isto é representado por um vetor para formar o descritor do keypoint. Em adição a isto, várias medidas são tomadas para se obter robustez contra mudanças de iluminação, rotação, etc. Ao final, os keypoints entre duas imagens são casados identificando-se seus vizinhos mais próximos. Porém, pode ocorrer de o segundo casamento mais próximo para um mesmo ponto ser bem próximo do primeiro. Neste caso, a relação entre os dois é tomada, e se for maior que 0,8, são ambos rejeitados. Isto elimina perto de 90% de falsos matches enquanto descarta apenas 5% de matches corretos.

Devido à grande quantidade de cálculos utilizados, SIFT é considerado devagar em relação aos métodos que se seguem.

2) SURF/SURF

SURF - Speeded-Up Robust Features

Como o nome sugere, é uma versão acelerada do SIFT. No SIFT, Lowe aproximou o laplaciano da gaussiana (LoG) pela diferença da gaussiana (DoG) para encontrar invariância escala-espaco. SURF aproxima o Log com filtro de caixa. A convolução com o filtro de caixa pode ser facilmente calculada com a ajuda de imagens integrais. E isto pode ser feito paralelamente para diferentes escalas. SURF também depende do determinante da matriz Hessiana para invariância á escala e localização. Para assinalar orientação, SURF usa respostas wavelet nas direções horizontal e vertical para uma vizinhança de tamanho 6. Pesos gaussianos adequados também são aplicados. A orientação dominante é estimada calculando-se a soma de todas as respostas dentro de uma janela de orientação deslizante de 60°. Para várias aplicações, a invariância à rotação não é necessária, então não se precisa achar esta orientação, o que torna o processo mais rápido. Para o descritor, SURF usa resposta wavelet nas direções horizontal e vertical (novamente, o uso da imagem integral faz as coisas ficarem mais fáceis). Uma vizinhança de tamanho 20 é tomada em volta do keypoint. Ela é divida em sub-regiões 4x4. Para cada sub-região, respostas wavelet vertical e horizontal são tomadas e um vetor é formado, de dimensão 64. Quanto menor a dimensão, maior a velocidade dos cálculos e matching, porém diminui a distinção das features.

Há uma versão de dimensão 128, mais robusta, que não adiciona tanta complexidade computacional. Outra melhora importante é o uso do sinal do laplaciano (traço da matriz hessiana) para destacar o ponto de interesse. Não adiciona custo pois já é calculado na detecção. Ela distingue bolhas claras em fundos escuros da situação inversa. No estágio de matching, apenas compararmos features se tiverem o mesmo tipo de contraste. Isto permite velocidade sem reduzir a performance do descritor. Análises indicam que o SURF é três vezes mais rápido comparado ao SIFT enquanto que possui performance similar.

SURF é bom manipulando imagens com borrado e rotação, mas não bom com mudanças no ponto de vista e de iluminação.

3)FAST/BRIEF

FAST - Features from Accelerated Segment Test

É importante destacar que FAST é um algoritmo apenas de detecção, usado para detecção de cantos. Vimos muitos detectores de features, mas para aplicações em tempo real, eles não são rápidos o bastante. Um exemplo da necessidade de velocidade é em SLAM mobile robot, que tem recursos computacionais limitados. Os passos utilizados pelo algoritmo são:

- Encontrar um ponto p de intensidade I_p .
- Escolher um limiar t .
- Fazer um círculo de 16 pixels em volta do pixel p . p será canto se existem n pixels do círculo contíguo mais claros que $I_p + t$ ou mais escuros que $I_p - t$. n por padrão foi escolhido como 12.
- Teste para excluir grande número de não-cantos. Um grande problema do teste é que várias features são detectadasumas adjacentes às outras. Dos 4 pontos considerados no teste, os 3 primeiros são endereçados com uma abordagem de aprendizado de máquina enquanto o último usa supressão de não-máximos.

Estudos indicam que FAST é diversas vezes mais rápido que qualquer outro detector de bordas.

BRIEF - Binary Robust Independent Elementary Features Aqui novamente deve-se dizer que BRIEF é um método descritor, e não detector. SIFT usa vetores para descritores de tamanho 128, o que toma basicamente 512 bytes de memória por vetor. SURF similarmente usa um de 64, com mínimo de 256 bytes. Criar um vetor assim para milhares de features toma um monte de memória que não é viável com recursos limitados e sistemas embarcados. Quanto maior a memória, maior o tempo gasto para matching. Toda esta dimensão pode ser comprimida usando vários métodos, como PCA, LDA, etc. Um deles, o LSH (Locality Sensitive Hashing) é usado para converter estes descritores SIFT de floats para números binários usando hashing. Essas strings binárias são usadas para casar features usando distância de Hamming. Isso dá muita velocidade pois para encontrar a distância de Hamming é só aplicar uma XOR e contagem de bits, coisas rápida nas CPUs modernas. Porém, primeiro precisamos encontrar os descritores, só assim podemos aplicar hashing , o que não soluciona o problema de memória. BRIEF chega neste momento. Ele nos dá um atalho para encontrar as strings binárias diretamente sem encontrar os descritores. Ele toma pedaços da imagem borrados e seleciona um set de pares localizados. Daí algumas comparações de intensidade de pixels são feitas nestes pares. Assim são construídas as strings de bits. Por padrão este vetor possui tamanho 256 e é representado por 32 bytes. Então, a partir dele, você pode calcular a distância de Hamming para casar estes descritores. Como foi dito anteriormente, BRIEF é um feature descriptor, ele não provê métodos para encontrar features. Logo, deve-se usar qualquer outro feature detector, como SIFT, SURF, etc. O paper recomenda usar o detector CenSurE (conhecido como STAR no OpenCV), que casa melhor com o BRIEF que o SIFT. BRIEF é o mais rápido descritor, provê alto grau

de matching desde que não haja grandes rotações no plano.

4)ORB/ORB

ORB - Oriented FAST and rotated BRIEF

Criado no OpenCV Labs. É uma alternativa ao SIFT e ao SURF em termos de custo computacional, performance de matching e principalmente as patentes. (Ele é free, ao contrário dos outros dois). ORB é basicamente uma fusão do detector FAST com o descritor BRIEF com várias modificações para melhorar a performance. Primeiro se usa o FAST para encontrar keypoints, depois Harris para encontrar os top N pontos entre eles. Também usa pirâmide para encontrar features- multiescalados. O problema é que FAST não calcula as orientações, o que não é invariante à rotação. Então os autores modificaram para obter esta característica. Isto calcula a intensidade ponderada do centroid do pedaço cujo canto está localizado no centro. A direção do vetor que vai do canto até o centroid nos dá a orientação. A invariância à rotação também é melhorada. Para os descritores, ORB usa o descritor BRIEF. Mas já vimos que o BRIEF é pobre com rotação. ORB então "direciona" BRIEF de acordo com a orientação dos keypoints. ORB contrói uma look up table de padrões pré- computados do BRIEF. Contando que a orientação do keypoint seja consistente, o set correto de pontos do BRIEF com rotação é usado para computar o descritor. BRIEF tem uma característica importante que cada feature do bit tem larga variância e média perto de 0.5. Porém, uma vez que está orientado com a direção do keypoint, ele perde essa propriedade e se torna mais distribuído. Grande variância faz uma feature mais discriminativa, desde que responde diferentemente às entradas. Outra propriedade desejada é ter os testes não correlacionados, já que cada teste vai contribuir para o resultado. Para resolver isto, ORB procura entre todas as possibilidades de testes binários para encontrar aqueles que tem tanto grande variância e média perto de 0.5, assim como não sendo correlacionados. O resultado é chamado rBRIEF. O paper diz que ORB é bem mais rápido que SURF e SIFT e que o descritor ORB funciona melhor que SURF.

II. METODOLOGIA

Snaptshots foram tirados para cada situação envolvendo objeto sendo utilizado e detector/descritor utilizado, o que nos dá 8 situações possíveis. As imagens abordaram movimentação, distância, iluminação, rotação, oclusão do objeto, etc. Resultados muito interessantes foram obtidos e foi possível comparar e confirmar algumas qualidades e defeitos de cada método. Para a aquisição dos snaptshots, colocou-se o objeto de frente com a câmera e logo em seguida foi tirado um print. Salvou-se a imagem e o processo foi repetido diversas vezes.

III. RESULTADOS

Quinze imagens aleatórias foram consideradas para comparação, sendo que abordam todas as características citadas acima.

Como as imagens foram tiradas em horários diferentes, houve grandes mudanças na iluminação, o que inutilizou alguns dos métodos utilizados. Desta forma, teve-se que escolher uma nova imagem base nas iluminações atuais à fase de teste para que se pudesse usar estes métodos.



Fig. 1: Imagem base 1 - carta de baralho. Tirada pela própria webcam.



Fig. 2: Nova imagem base 1 - carta de baralho. Necessário devido á inutilização de alguns métodos. Tirada da câmera do celular.



Fig. 3: Imagem base 2 - controle remoto. Tirada da webcam.



Fig. 4: Imagem base 2 - controle remoto. Mesmo motivo da imagem base 1. Tirada da câmera do celular.

SIFT

1) CARTA

- PERTO (10 cm) Da amostra escolhida, SIFT identificou em todas elas. Situação de rotação, oclusão, iluminação e inclinação também foram identificados.
- LONGE (30 cm) Aqui também SIFT mostrou grande eficiência e reconheceu todos os casos.

2) CONTROLE

- PERTO (10 cm) Da amostra escolhida, SIFT identificou em 9/10 casos.
- LONGE (30 cm) Aqui SIFT identificou 8/10 casos.

SIFT reconheceu muito bem a maioria dos casos. É importante lembrar SIFT não apresentou problemas com iluminação, o que nos mostra que tem robustez contra ela. Das amostras não identificadas ambas estavam ligeiramente inclinadas.

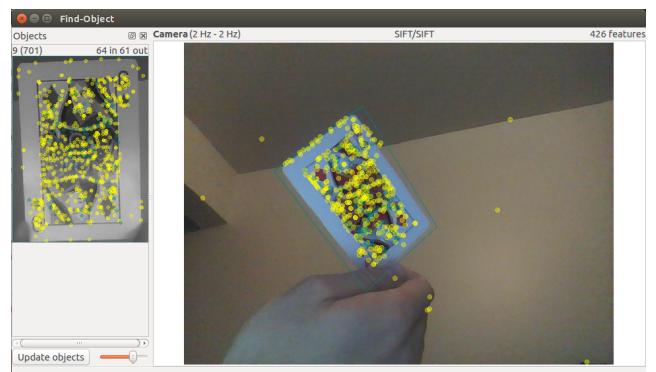


Fig. 5: Amostra SIFT carta perto.

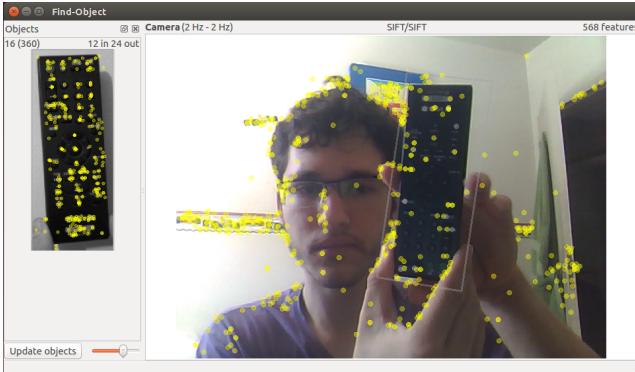


Fig. 6: Amostra SIFT controle longe.

SURF

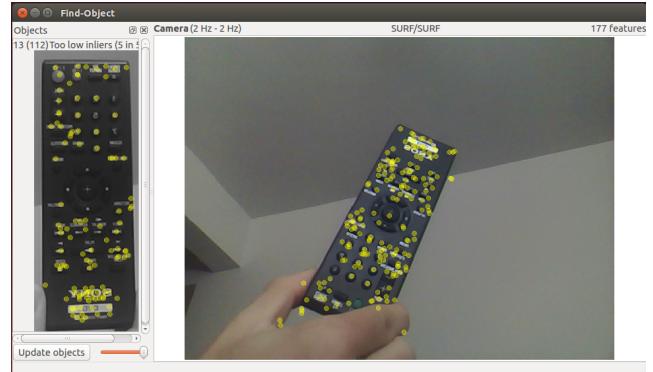


Fig. 8: Amostra SURF controle perto.

FAST/BRIEF

1) CARTA

- PERTO (10 cm) SURF encontrou 7/10 amostras. Mostrou-se bom em relação à rotação. Não foi obtido êxito em situações onde houve inclinação.
- LONGE (30 cm) Êxito em 4/10 amostras. SURF não foi eficiente em relação a distânciamento da câmera.

2) CONTROLE

- PERTO (10 cm) Reconheceu 6/10 amostras. Aqui novamente o problema maior foi em relação à inclinação.
- LONGE (30 cm) Aqui SURF reconheceu só em um dos casos.

SURF não se mostrou eficiente em casos onde houve inclinação, distância e iluminação. Foi necessário que se utilizasse a outra imagem base para realizar a comparação com as amostras.

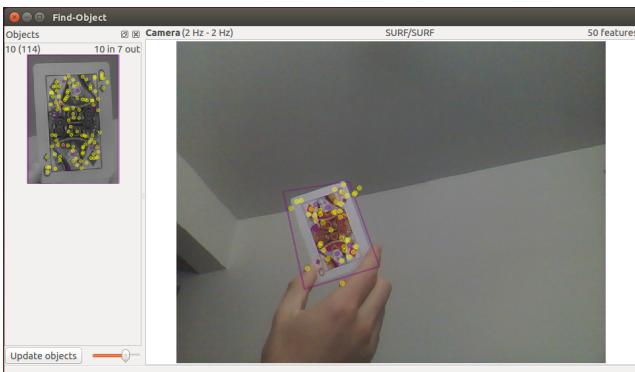


Fig. 7: Amostra SURF carta longe.

1) CARTA

- PERTO (10 cm) Encontrou 5/10 casos. Rotação e inclinação foram praticamente ignorados. Não houve bom reconhecimento nestes casos. Alguns ruídos também ocorreram.
- LONGE (30 cm) Reconheceu 2/10 casos. Não se mostrou muito eficiente à uma distância maior.

2) CONTROLE

- PERTO (10 cm) Encontrou em 5/10 casos. Aqui rotações pequenas foram encontradas, porém as maiores e inclinações foram ignoradas.
- LONGE (30 cm) Aqui não reconheceu nenhum caso.

FAST/BRIEF não se mostrou eficiente à longas distâncias e a rotações e inclinações. Além disto, foi-se necessário usar outra imagem base devido à iluminação. Desta forma, não houve eficiência contra iluminação.

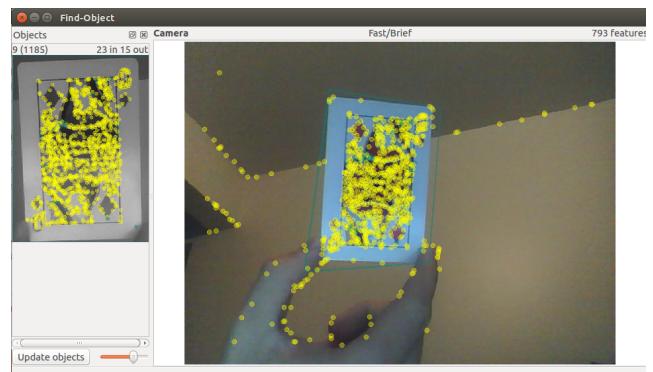


Fig. 9: Amostra FAST/BRIEF carta perto.

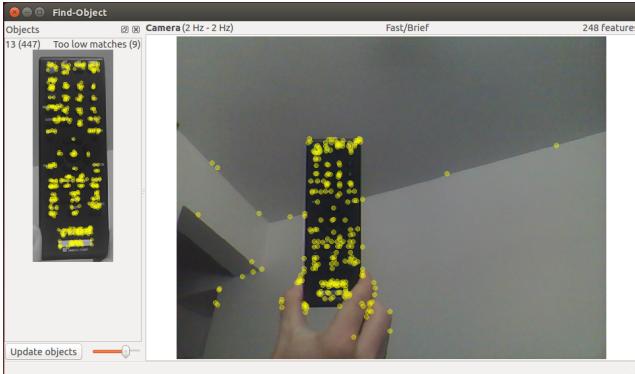


Fig. 10: Amostra FAST/BRIEF controle longe.

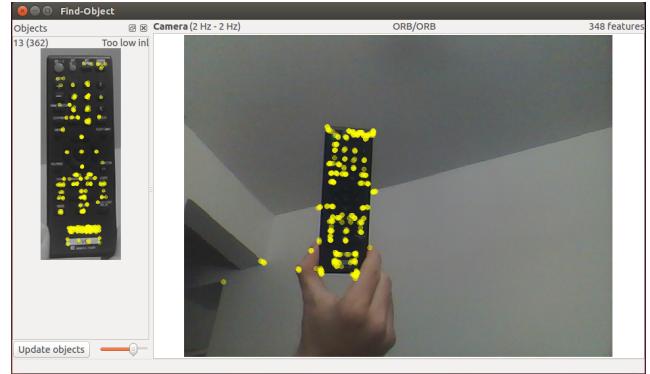


Fig. 12: Amostra ORB controle longe.

ORB

1) CARTA

- PERTO (10 cm) ORB encontrou em 9/10 casos. Assim como o SIFT, mostrou grande eficiência em situação de rotação, oclusão, iluminação e inclinação.
- LONGE (30 cm) ORB encontrou em todos os casos.

2) CONTROLE

- PERTO (10 cm) ORB reconheceu 9/10 casos. Mostrou-se bem eficiente às situações mencionadas acima. O caso não identificado estava com inclinação considerável. Também houve ruído em um deles.
- LONGE (30 cm) ORB reconheceu 6/10. Houve dificuldade para distâncias consideráveis. Houve muito ruído também naqueles que foram reconhecidos.

ORB reconheceu grande parte das amostras, tendo nestas condições obtido desempenho similar ao SIFT. Teve alguns problemas apenas na detecção do segundo objeto quando este estava mais distante. Em termos de velocidade, pareceu mais rápido que o SIFT, sendo este dado estimado por observação do desempenho do computador.

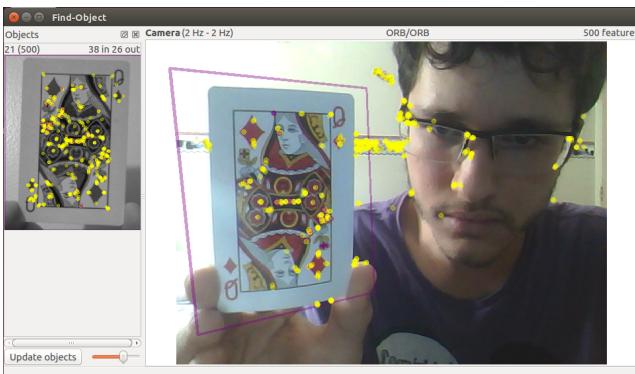


Fig. 11: Amostra ORB carta perto.

IV. CONCLUSÃO

Para as situações impostas no proposto experimento, pudemos dizer que SIFT obteve os melhores resultados, seguido de ORB, SURF e FAST/BRIEF, sendo que os dois últimos não obtiveram bons resultados, já que a variância à iluminação esteve presente em ambos. Pudemos notar também que o reconhecimento é muito dependente do objeto a ser reconhecido.

V. REFERÊNCIAS

- [1] <http://www.mathworks.com/matlabcentral/answers/166657-difference-between-feature-detection-extraction-descriptor-selection-and-matching>
- [2] <http://stackoverflow.com/questions/6832933/difference-between-feature-detection-and-descriptor-extraction>
- [3] <https://computervisionblog.wordpress.com/2012/02/10/the-most-cited-papers-in-computer-vision/>
- [4] https://www.willowgarage.com/sites/default/files/orb_final.pdf
- [5] http://web.eecs.umich.edu/~silvio/teaching/EECS598/lectures/lecture10_1.pdf
- [6] <http://www.cs.berkeley.edu/~malik/cs294/lowe-ijcv04.pdf>
- [7] <http://www.vision.ee.ethz.ch/~surf/eccv06.pdf>
- [8] http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/AV1011/AV1FeaturefromAcceleratedSegmentTest.pdf
- [9] <https://www.robots.ox.ac.uk/~vgg/rg/papers/CalonderLSF10.pdf>
- [10] <http://computer-vision-talks.com/articles/2011-01-04-comparison-of-the-opencv-feature-detection-algorithms/>
- [11] http://opencv-python-tutorial.readthedocs.org/en/latest/py_tutorials/py_feature2d/py_sift_intro/py_sift_intro.html
- [12] http://opencv-python-tutorial.readthedocs.org/en/latest/py_tutorials/py_feature2d/py_surf_intro/py_surf_intro.html
- [13] http://opencv-python-tutorial.readthedocs.org/en/latest/py_tutorials/py_feature2d/py_fast/py_fast.html
- [14] http://opencv-python-tutorial.readthedocs.org/en/latest/py_tutorials/py_feature2d/py_brief/py_brief.html