

Relatório trabalho 3*

*Introdução ao Processamento de Imagens, 2021/1, Prof. Bruno Macchiavello

1st Lucas de Almeida Bandeira Macedo
Departamento de Ciéncia da Computação (CiC)
Universidade de Brasília (UnB)
Brasília, Brasil
lucasabmacedo@hotmail.com

Resumo—Este é o relatório para o projeto final da disciplina Introdução ao Processamento de Imagens, em que será coberta a solução encontrada da implementação de um Jogo da Velha [1] em Realidade Aumentada [2], incluindo sua base teórica e os resultados obtidos pelo algoritmo desenvolvido.

I. INTRODUÇÃO

Este projeto foi realizado com o objetivo de estudar o conteúdo ministrado nas aulas de Processamento de Imagens, além estimular a busca de novos conhecimentos independentemente.

O projeto se resume em implementar o jogo "Jogo da Velha", em realidade aumentada, sendo que um dos jogadores é humano, jogando em um papel filmado, e o outro jogador é a máquina, lendo as informações do papel e fazendo suas próprias jogadas, exibindo elas em realidade aumentada.

O relatório será dividido em três partes, daqui para frente: Seção II - Metodologia, onde será explicado o funcionamento do artigo escolhido, assim como uma leve base teórica para cada uma das técnicas utilizadas; Seção III - Resultados, em que analisaremos os resultados adquiridos pelos procedimentos descritos na Metodologia; e Seção IV - Conclusão, onde será feita uma análise geral dos resultados, fechando o relatório.

Tanto a Metodologia quanto os resultados serão divididos em três grupos, representando os passos da solução:

- 1) "Homografia", referente ao primeiro passo da solução, em que extrairemos a imagem do tabuleiro no vídeo em tempo real, através de uma imagem base gerada anteriormente;
- 2) "Sub-Imagens", onde extrairemos as informações de cada um dos 9 sub-campos de um tabuleiro de jogo da velha, para detectar jogadas do player humano e permitir que a máquina faça também suas ações;
- 3) "Homografia Reversa", o último passo, em que retornaremos as informações obtidas no passo anterior para o tabuleiro original.

II. METODOLOGIA

O projeto foi realizado utilizando da linguagem de programação Python [3], com as bibliotecas Numpy [5] e OpenCV [4].

A biblioteca do OpenCV é uma biblioteca de processamento de imagens e visão computacional. Através dela, conseguimos ler imagens para a memória como matrizes, fazer transformações e alterações, para atingir determinados objetivos.

A biblioteca Numpy age como uma biblioteca auxiliar, pois algumas estruturas de dados da biblioteca OpenCV são implementadas por ela, e requerem o uso de funções próprias para determinados fins.

Agora, entrando a fundo na metodologia necessária para cada etapa do projeto:

A. O cálculo da Homografia

O primeiro passo necessário para a execução da solução é a calibragem. É necessário que o usuário (que no escopo do projeto, é um jogador) coloque o campo do Jogo da Velha enquadrado em uma determinada área da filmagem, e em seguida armazene uma foto. Esta imagem gerada será utilizada pelo resto do projeto pois, com ela, conseguimos detectar de fato onde está o tabuleiro na filmagem em tempo real.

Essa detecção funciona encontrando uma homografia entre a imagem de calibração e a filmagem, ou seja, o programa deve tentar encontrar a imagem gerada na filmagem em tempo real. Para isso, foi utilizada a técnica ORB [6] (Oriented FAST and Rotated BRIEF) que, como o nome indica, é um algoritmo que usa as técnicas FAST [7] e BRIEF [8].

1) *ORB*: A grosso modo, FAST detectará *keypoints* na imagem, ou seja, pontos chaves para a caracterização da imagem. Esses pontos são cantos, que são detectados por diferença no nível de brilho dos pixels ao redor do pixel analisado. Já o BRIEF criará descritores para a imagem a partir dos keypoints gerados pelo FAST, que são strings binárias geradas a partir de uma comparação entre dois pixels randômicos ao redor de determinado keypoint. Esses descritores, diferentemente dos keypoints, ajudam a distinguir forma, cores e textura em uma imagem, dando a possibilidade de encontrar a correspondência entre, por exemplo, duas imagens de um mesmo objeto tiradas de ângulos diferentes.

Esse procedimento (FAST e BRIEF) é feito tanto para a imagem base da comparação (no nosso caso, na imagem da calibragem) quanto para a imagem comparada (frame do vídeo ao vivo). Então, com os descritores e pontos chaves de

ambas imagens em mãos, podemos caminhar na direção da homografia.

O próximo passo é encontrar as correspondências dos descritores entre as duas imagens, e faremos isso por um *matcher* de força bruta. Para cada descritor da primeira imagem, checaremos se ele corresponde a algum descritor da segunda imagem. Para exemplificar este passo, a figura 1 mostra uma simulação do resultado até aqui.

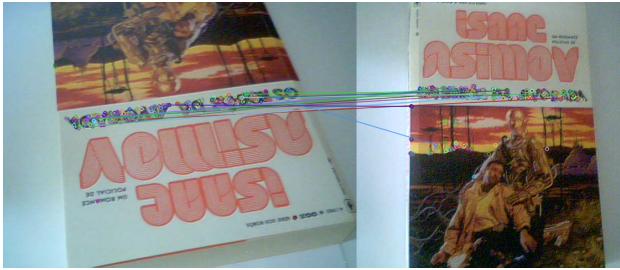


Fig. 1. Para cada correspondência, é desenhada uma linha ligando os dois pontos.

Com isso, é possível extraír muitos pontos de correspondência, mas nem todos são ótimos. Pelo próprio exemplo da figura 1 podemos ver que algumas linhas não estão associadas aos pontos corretos. Por isso, é necessário filtrar essas correspondências: extrairemos apenas as 15% das melhores correspondências, seguindo o critério de similaridade presente entre cada correspondência dos descritores. Com as correspondências selecionadas, podemos pegar seus keypoints associados e finalmente teremos os pontos para calcular a homografia.

2) *Homografia*: Com dois grupos de pontos, cada um ligados entre si pela correspondência calculada anteriormente, conseguimos calcular a homografia.

A homografia é uma matriz que permite que os pontos de um primeiro grupo sejam transformados em pontos do segundo grupo. Ou seja, é uma matriz de transformação para sair da imagem vista em tempo real para a imagem base da calibragão. Aplicar, por exemplo, a homografia na imagem da esquerda da figura 1, gerará uma mudança de perspectiva que a aproximará da imagem da direita. Podemos ver isso na imagem 2.



Fig. 2. Esquerda: imagem filmada em tempo real. Direita: aplicação da homografia na imagem da esquerda, alterando a orientação do livro para a da imagem de calibração.

B. Sub-Imagens

O segundo passo a ser realizado, após o cálculo da homografia, é a descoberta das sub-imagens do tabuleiro de jogo da velha, e computar seu conteúdo. Essas sub-imagens correspondem a cada uns dos 9 campos do jogo, onde se pode desenhar um X (xis) ou um círculo.

1) *Extração de campos*: Felizmente, essa identificação é fácil de ser feita, uma vez que calibrarmos o tabuleiro no início da execução. Essa calibragão, além de trazer a destaca o tabuleiro na filmagem em tempo real, pela homografia, também extrai ele em uma orientação fixa, já que o usuário é orientado o tabuleiro alinhado com algumas linhas que são desenhadas na tela, como referência. Essas linhas são mostradas na figura 3, referente a tela de calibragem do jogo.

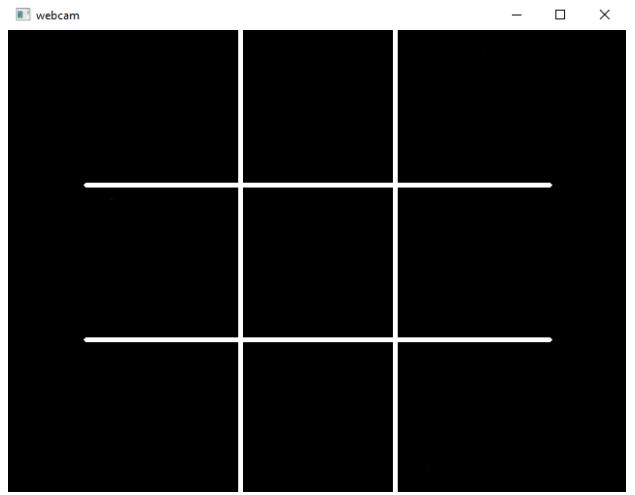


Fig. 3. Linhas de referência para o usuário calibrar o tabuleiro

Portanto, com um campo alinhado nesse sentido (em decorrência da calibragem e da homografia), para extraír uma sub-imagem para cada um dos 9 campos basta extraír uma área fixa, com lados iguais ao tamanho total de um lado do tabuleiro (que é quadrado) dividido por 3.

2) *Detecção de Jogada*: Por padrão, o usuário só pode jogar com o símbolo X (xis). Ou seja, basta identificar se, em cada uma das 9 sub-imagens geradas, existe um X que não foi computado anteriormente. Para isso usaremos a técnica de template matching.

Essa técnica se resume em passar um template pela imagem e computar o quanto similar ele é em relação a cada ponto da imagem. Foi feito, então, um template do X, igual a figura 4. Este template foi feito usando 3 "xis", levemente rotacionados entre si, para dar mais precisão para caso o usuário escreva de maneira torta. Consideraremos uma "correspondência" sempre que encontrarmos, na matriz de correspondência gerada, um valor maior que 40% de correspondência.

C. Homografia Reversa

Agora, detectando as jogadas e sua localização, podemos computar a jogada do adversário (computador). Definir onde será desenhada a jogada, no entanto, é a verdadeira dificuldade,

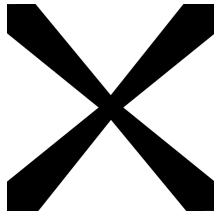


Fig. 4. Template para o X do usuário

pois devemos desenha-la no vídeo em tempo real. Simplesmente desenha-la na imagem gerada pela homografia inicial seria fácil, pois está tudo alinhado e sabemos onde todos os pontos importantes estão, como o centro de cada um dos 9 campos. Então, usaremos esses pontos da imagem gerada pela homografia e descobriremos onde eles estão no vídeo real, para desenhar corretamente.

Para isso, precisamos de uma homografia que retorne para a imagem original, então calcularemos e usaremos o inverso da homografia para isso, já que o objetivo é justamente voltar para a imagem antes da primeira homografia.

Assim, podemos escolher pontos na imagem e transformá-los de volta para seus pontos correspondentes na imagem crua da filmagem, e desenhar o necessário nessas novas coordenadas descobertas (sendo os círculos dos movimentos do adversário, ou o contorno do tabuleiro).

III. RESULTADOS

Nesta seção mostraremos e discutiremos os resultados adquiridos pelos passos descritos na Seção II.

Todos os resultados foram extraídos em uma seção de testes, para manter a uniformidade do comportamento a ser exibido. Se algum resultado for mais dependente de ambiente e fatores externos, será explicitado na análise.

A. Homografia

Primeiramente é necessário fazer a calibragem. A imagem gerada nesse passo será usada para o resto da explicação, e está presente na figura 5. Após isso, estamos prontos para a execução do jogo propriamente dito.



Fig. 5. Imagem base da calibração inicial para os testes.

Para testar a homografia, rotacionaremos a imagem em diversos ângulos e orientações, esperando adquirir uma perspectiva semelhante a da figura 5. O resultado dessa experiência corresponde a figura 6.

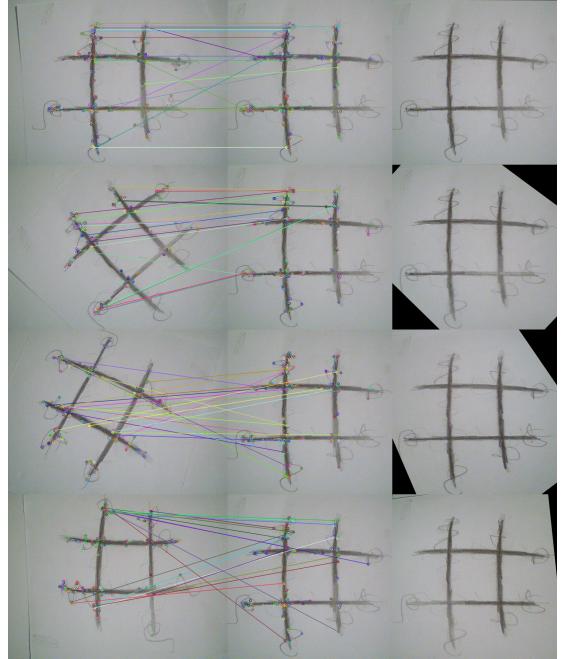


Fig. 6. 1^a coluna corresponde à imagem ao vivo. 2^a Coluna à imagem base da calibração. 3^a Coluna imagem adquirida pela homografia.

A precisão do resultado da figura 6 depende de diversos fatores: diferença de iluminação, câmera de baixa qualidade, sombras bem marcadas, tabuleiro muito claro (pouco marcado)... são todos fatores que reduzem a precisão da homografia. Por outro lado, um fator que aumenta a precisão são detalhes e ornamentações desenhadas por cima do tabuleiro. Quanto mais imperfeito for, o algoritmo fará menos falsas correspondências entre dois pontos nas duas imagens. Imagens sintéticas e perfeitamente simétricas são especialmente danosas à precisão do programa.

B. Sub-Imagens

Em seguida, extrairemos as sub-imagens referentes a cada um dos 9 campos do tabuleiro. A precisão desse resultado é completamente dependente da imagem de calibração e o quanto ela está alinhada com o grid de referência, mostrado na figura 3. Para o correto funcionamento do programa, não é preciso que a imagem base da calibração esteja perfeita em relação à grade, é suficiente que tenha espaço suficiente para um X e que não haja traços semelhantes a X's, para evitar que o template matching (Seção II-B2) faça falsas correspondências. A imagem extraída para cada um dos 9 campos, separados por linhas azuis, está presente na figura 7.

Precisamos, agora, detectar ou não a presença de um X. A precisão desse passo também depende diretamente do usuário. Se um X foi mal desenhado, demasiadamente pequeno

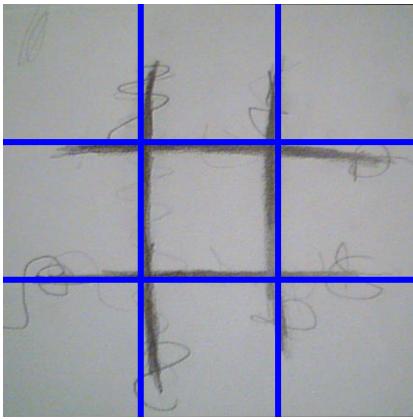


Fig. 7. Cada um dos 9 campos é uma imagem separada.

ou torto, ou pouco definido (muito claro), ser que ele não corresponda ao template. A figura 8 representa os os X que corresponderam ao template, a figura 9 representa os X que não responderam.

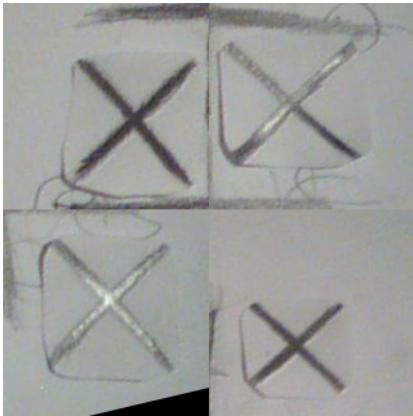


Fig. 8. Campos que corresponderam ao template.

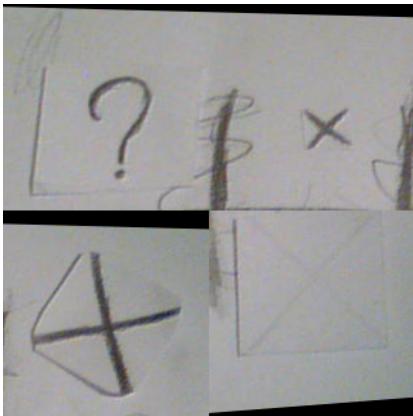


Fig. 9. Campos que não corresponderam ao template.

C. Homografia Reversa

Finalmente, usando a homografia reversa, os dados adquiridos nos passos passados e o estado atual do tabuleiro do jogo, podemos mostrar para o usuário o resultado de nosso processamento, desenhando na filmagem crua o que for necessário. Duas coisas serão desenhadas com auxílio da homografia reversa: o contorno do tabuleiro e os círculos, quando houver. A figura 10 demonstra o passo a passo de uma "partida".

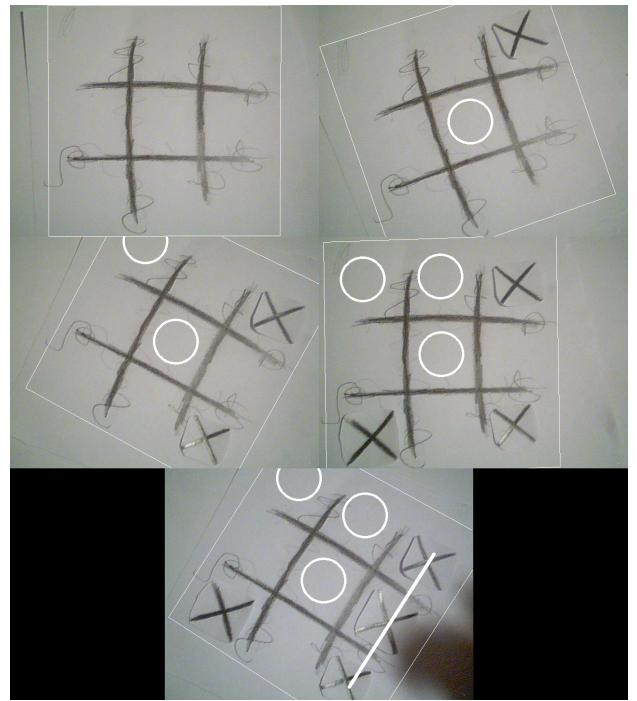


Fig. 10. Partida completa de jogo da velha, com 5 rodadas, resultando na vitória do player humano. Leves rotações são realizadas entre os rounds para fins de demonstração.

IV. CONCLUSÕES

Este projeto final foi, de fato, um grande desafio. A maioria das técnicas aqui utilizadas não foram ensinadas em sala de aula. Mas a surpresa veio exatamente nesse ponto: foi aprender essas técnicas e entendê-las. Isso apenas explicita o aprendizado adquirido ao longo do semestre: não foi um aprendizado superficial em que as técnicas são decoradas e aplicadas, como uma caixa preta. Foi um aprendizado das raízes do processamento de imagens. A base de conhecimento adquirida no decorrer do semestre se mostrou sólida. Por consequência disso, o resultado do projeto foi extremamente satisfatório, criando a vontade de explorar um pouco mais essa área de estudos, e aperfeiçoar o produto que foi produzido aqui.

Esse trabalho também serviu para mostrar o poder de algoritmos de *feature matching*, como o ORB. A "realidade aumentada" se tornou muito famosa com o nascimento do jogo Pokemon GO [9], e está em constante expansão, sendo sempre retratada em ficções científicas como tecnologia do futuro... embora esteja mais próxima que parece. Extrair informações

do ambiente, como em alguns carros modernos com direção automática, também é um ótimo exemplo de todo o potencial que algoritmos semelhantes podem atingir.

Um futuro onde seus óculos têm um ambiente de Realidade aumentada para mostrar as horas, identificar rostos em tempo real e ampliar textos começa em pequenos projetos, como esse. O futuro da Internet das Coisas nasce de tecnologias como essas aqui exploradas.

REFERENCES

- [1] Jogo da Velha, https://pt.wikipedia.org/wiki/Jogo_da_velha.
- [2] Realidade Aumentada, https://pt.wikipedia.org/wiki/Realidade_aumentada.
- [3] Python, <https://python.org/>.
- [4] Opencv-Python, <https://pypi.org/project/opencv-python/>.
- [5] Numpy, <https://numpy.org/>.
- [6] E. Rublee, V. Rabaud, K. Konolige and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," 2011 International Conference on Computer Vision, 2011, pp. 2564-2571, doi: 10.1109/ICCV.2011.6126544.
- [7] E. Rosten and T. Drummond. Machine learning for highspeed corner detection. In European Conference on Computer Vision, volume 1, 2006.
- [8] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. Brief: Binary robust independent elementary features. In In European Conference on Computer Vision, 2010.
- [9] Pokemon GO, https://pokemongolive.com/pt_br/.
- [10] Internet das Coisas, <https://www.gov.br/mcom/pt-br/noticias/2021/marco/internet-das-coisas-um-passeio-pelo-futuro-que-ja-e-real-no-dia-a-dia-das-pessoas>.