Rainbow-Po

Rafael Macito, Gabriel Borges, Bruno Fonseca

May 5, 2015

Abstract

Resumão das parada em português e inglês, deixar para o final.

1 Introdução

O objetivo deste projeto é fazer um jogo de jokenpo onde o jogador irá jogar contra a máquina usando suas próprias mãos. Para isso serão utilizados conceitos de Computação Visual com a biblioteca OpenCV[2] e a linguagem C[1].

No caso deste projeto, o jokenpo escolhido foi um criado pela série de TV The Big Bang Theory[6], que utiliza, como o original, pedra, papel e tesoura, com a adição de dois novos elementos: Lagarto e Spok, como na imagem a seguir.

IMAGEM DO JOKENPO VAI AQUI

2 Ferramentas

A biblioteca utilizada para tratar as imagens capturadas pela câmera é o OpenCV[2]. Porém suas funções não serão utilizadas diretamente, sendo chamada através de uma segunda biblioteca criado pelo Marcelo Hashimoto[4].

A biblioteca Allegro[3] está sendo usada para criação do jogo e suas mecânicas.

E a linguagem de programação usada no projeto é C[1].

3 Métodos

3.1 Análise

Para construção do algoritmo, levou-se em consideração três pontos focais distintos, sendo eles: Precisão, Velocidade e Jogabilidade.

Encontrar um equilíbrio entre esses três pontos foi essencial para que o algoritmo conseguisse ser preciso o suficiente para identificar a mão do jogador e suas respectivas jogadas, rápido o suficiente para não perder muitos quadros, o que poderia gerar uma baixa na precisão por analisar menos quadros por segundos, e por fim a jogabilidade, para não tirar muito conforto do jogador.

3.2 Soluções

O processo de desenvolvimento pôde ser separado em duas partes, a captura do movimento da mão para contagem do balançar da mão, mais focado em velocidade no processamento, para sempre acompanhar a mão e perder o mínimo de movimentos possível, e a captura da jogada selecionada pelo jogador, esse sim tendo que ser muito mais preciso.

Para encontrar um ponto intermediário entre os três fatores, foi construído um algoritmo que foca somente na imagem da mão do jogador, obtida na calibração, para diminuir o vetor de busca da mão e otimizar o tempo de processamento, mantendo a precisão.

Para tornar tanto a calibragem como o processamento da imagem mais viável, uma "luva colorida" azul com a ponta dos dedos magenta foi usada.

4 Algoritmos Base

4.1 Detectação de Movimento

4.2 RGB para HSV

4.3 Disjoint-Set

4.4 Componente Conexo

4.5 Filtro de Erosão e Dilatação

5 Calibragem

6 Rastreamento do Movimento da Mão

Após os parâmetros serem calibrados, será necessário determinar o delimitador do movimento da mão, que será o ponto médio entre a altura máxima e a mínima que o jogador balançar a mão. Até poderia usar o meio da tela como padrão, porém isso será um problema caso a pessoa não consiga subir ou descer a mão até esse ponto, ou até consiga porém com um pouco de desconforto, sem falar que esse ponto pode variar conforme a distância entre o jogador e a câmera muda.

O rastreador nada mais é do que a distância entre o primeiro e o ultimo pixel pertencentes a mão na horizontal e vertical, mais uma constante para quando o jogador mover a mão, tendo quatro variáveis, norte, sul, leste e oeste.

O processo de obtenção e redefinição do rastreador é bem simples, para não deixar o processo lento. A imagem obtida pela câmera é primeiro transformada de RGB para HSV, depois é verificado se matiz e a saturação de cada pixel está na "range" definida na hora da calibragem, se ele estiver, o pixel é "pintado" de branco, caso contrário, de preto.

imagem.

Depois que a imagem tiver sido transformada em uma imagem binária (uma imagem em preto e branco), é utilizado um filtro de "fechamento", nesse filtro é processado primeiro uma erosão e logo em seguida uma dilatação, eliminando o

ruído sem alterar a imagem original, pois os pixels da mão que foram perdidos no processo de erosão serão recuperados com a dilatação.

imagem.

Agora com o ruído tendo sido eliminado, a imagem que sobrou é varrida de cima para baixo, de baixo para cima, da esquerda para direita e da direita para esquerda. Quando um pixel branco é encontrado, dependendo da ordem da varredura, a variável do rastreador será definida como o ponto que o pixel se encontra na matriz mais uma constante definida no programa.

imagem.

Esse processo será repetido durante todo o programa, e sempre que o rastreador terminar de ser atualizado, será verificado se o centro do rastreador (que deve ser próximo ao centro da mão) ultrapassou o delimitador, se sim, somar 1 a um contador até que chegue a 3, e caso a mão do jogador esteja descendo, uma flag do rastreador deve ser setada como verdadeiro, para controlar quando a mão do jogador está subindo ou descendo. Caso o rastreador perca a mão, pois o jogador tirou do alcance da câmera, uma recalibragem deve ser efetuada.

7 Analise da Jogada

Após o jogador balançar a mão três vezes, o algoritmo continuará atualizando o rastreador até que o movimento pare, para isso será usado o algoritmo de detecção de movimento.

Só quando o movimento parar, será executado o algoritmo que analisa a jogada, esse algoritmo necessita ser muito mais preciso que o anterior, pois qualquer alteração na imagem leva a uma analisa errada e o computador pode interpretar como uma jogada diferente do que o jogador realmente jogou.

O início do processo da analise é similar ao do rastreador, primeiro transforma a imagem de RBG para HSV, para obtenção da matiz e saturação da imagem, depois a imagem é analisa pixel por pixel verificando se a matiz e a saturação estão na range determinada pela calibragem, se sim, pinta o pixel de branco, se não, de preto.

imagem.

Depois da obtenção da imagem binária, um filtro de abertura é aplicado, que é o processo inverso do fechamento, sendo que primeiro se aplica uma dilatação e em seguida uma erosão, nesse caso se perde muito menos ruído, porém tem uma porcentagem menor de perda de pixels dos dedos, que dependendo da distância entre o jogador e a câmera, pode ser um problema. Agora com o filtro aplicado na imagem, é executado um algoritmo de componentes conexas.

imagem.

Porém o foco do algoritmo criado nesse projeto é um pouco diferente de um de connected components labeling, por exemplo, onde o foco é obter no final uma estrutura que contém todos os pixels semelhantes (uma lista ligada por exemplo). No caso, as únicas coisas que importam são a quantidade de componentes na tela e sua massa, então uma estrutura contendo todos os pixels referentes aquele conjunto foi substituída por apenas uma variável chamada massa, que é a quantidade de pixels referente a um conjunto.

Após o algoritmo ter reconhecido todos os conjuntos e contado suas massas, nem todos são válidos, justamente pelo filtro não ter retirado todo o ruído da imagem para preservar os dedos, e será necessário elimina-los. Para isso,

pode ser aplicada uma consistência pela massa, aonde os conjuntos com massas menores que a massa mínima definida na calibragem são zerados, e então passa-se a considerar apenas os conjuntos com massa maior que zero.

Agora sim, após a consistência, pode-se fazer uma contagem de quantos conjuntos sobraram na imagem, é importante também salvar quais índices do grafo estão os conjuntos que não estão com a massa zerada, para facilitar um futuro acesso.

Com tudo isso feito, sobra analisar a quantidade de conjuntos que sobraram. Caso ela seja um, a jogada é pedra, pois apenas o dedão aparece na imagem. Caso seja 2, pode ser lagarto ou papel, caso seja papel, os quatro dedos estarão juntos e farão parte do mesmo conjunto, isso pode ser identificado verificando a massa dos conjuntos, caso a massa de um conjunto seja maior que pelo menos 2 vezes a massa máxima, então é papel. Caso o número de conjuntos seja três, caímos no mesmo dilema acima, mas agora podendo ser tesoura ou spock, no caso do spock, a massa de um dos conjuntos será pelo menos duas vezes maior que a massa mínima, pois dois dedos estarão juntos. Caso tenha cinco conjuntos, só pode ser papel. E por fim, caso não seja nenhum dos anteriores, é uma jogada inválida.

8 Resultados

Resultados obtidos dos algoritmos implementados no projeto, positivos e negativos.

9 Conclusão

References

[1] C, linguagem de programação, versão C99.

[2] OpenCV, Biblioteca de Computação Visual, versão 2.4.10, 2014.

[3] Allegro, biblioteca livre para criação de jogos, versão 5.0.10, 2013.

[4] Marcelo Hashimoto.

[5] Janken-pon, jogo recreativo criado no japão, muito antigo.

[6] Jokenpo do BBT, criado na série de TV The Big Bang Theory. link do video: