Projeto Demonstrativo 6 - Rastreamento de Objeto

Natalia Oliveira Borges
natioliveira97@hotmail.com
Matrícula:
16/0015863
Lívia Fonseca
liviagcf@gmail.com
Matrícula:
16/0034078

Departamento de Ciência da Comptutação
Universidade de Brasília
Campus Darcy Ribeiro, Asa Norte
Brasília-DF, CEP 70910-900, Brazil,

[0]

Abstract

Esse trabalho consiste em detecção de bola utilizando o método Viola Jones (Haar Cascade) e o método LBP como rastreador do bola de futebol aplicado para futebol de robôs humanoides. Como a bola é preta e branca, foi utilizada uma contagem de pixels pretos e brancos para validação dos candidatos retornados pelo método. Foi realizado o treinamento do método e a aplicação em vídeos.

1 Introdução

Rastreamento de objetos em imagens é uma das funções mais úteis de visão computacional e já é utilizada em várias aplicações práticas mundo a fora. Uma das aplicações mais importantes é o rastreamento de movimento dos olhos para auxiliar pessoas com deficiência a formar palavaras e se comunicar.

Nosso objetivo é rastrear uma bola de futebol em um jogo de futebol de robôs humanoides. Pretendemos futuramente aplicar esses conhecimentos de forma prática em competições de robóticas futuras.

1.1 Haar Cascade

O classificador Haar Cascade [1] é um método rápido de detecção de objeto por aprendizado de máquina proposto por Paul Viola and Michael Jones.

Esse algoritmo precisa de um banco de imagens positivas e negativas do objeto de interesse. No nosso caso, imagens de bolas e imagens de não bolas.

O treinamento consiste, primeiramente de extração de features das imagens de treinamento. Para extrair features usam-se kernels específicos para bordas, linhas, cantos, centros, como mostrados na figura 1

© 2018. The copyright of this document resides with its authors.
It may be distributed unchanged freely in print or electronic forms.

[0] Ambos os membros contribuiram de igualmente para a elaboração do código e relatório

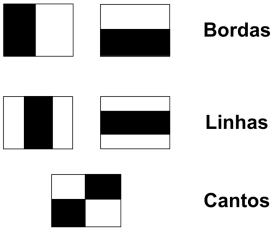


Figure 1: Exemplos de kernels usados para detecção de features

Calcular todas essas features exige muito processamento, para otimizar esse processo são utilizadas imagens integrais. Essa imagens trabalham com a soma dos valores dos pixels, reduzindo a extração de características de um pixels para operações entre quatro valores.

De todas as features obtidas, apenas algumas serão efetivamente usadas para detectar o objeto. No processo de treinamento todas as features são testadas em todas as imagens e para cada uma é estabelecido um peso. As features que melhor classificam o objeto desejado e não classificam outros objetos possuem um peso maior na detecção.

No processo de treinamento, imagens erroneamente classificadas possuem uma maior influência na redistribuição dos pesos na próxima iteração. Para cada iteração esses pesos vão sendo recalculados até que os parâmetros de toletância desejados sejam alcançados [2].

No final do processo, temos várias features que sozinhas não são muito efetivas para detectar o objeto, mas juntas, com a soma ponderada de seus pesos, conseguem descrever muito bem e relativamente rápido esse objeto, tornando possível a detecção e rastreamento em tempo real.

1.2 Local Binary Paterns

Local Binary Paterns é um descritor de textura que pode ser usado para detecção de features. O LBP não usa a matriz de coocorrência para calcular textura, mas sim um cálculo local realizado pixel a pixel[3].

Para cada pixel é determinada uma janela ao redor. O pixel central é comparado com todos os pixels da janela. Se ele for maior que o pixel a ser comparado o valor no descritor desse pixel é setado como 1. Já se o pixel central for menor, o valor do descritor, nessa posição, será 0. Para cada pixel, resultado é um vetor de binários que descreve a textura local.

Calculando histogramas de ocorrência dos padrões encontrados pelo LBP é possível identificar features e usá-la para detecção de objetos.

2 Metodologia

Esse trabalho foi feito em Linux Xubuntu versão 18.04 e em Linux Ubuntu versão16.04, ambos usando Opencv versão 3.2.0. O código foi feito python3.5.

2.1 Obtenção das imagens de treinamento

Possuíamos alguns vídeos de treinamento. Nesses vídeos há frames da bola em várias posições, orientações e condições de iluminação. Fizemos então um programa que corta

pedaços de frames e os classifica como bola e não bola.

Para utilizar essa ferramenta, devemos fazer um retângulo em volta do pedaço que devemos cortar, pressionando o mouse no vértice superior esquerdo e soltando no vértice inferior direito. A nova imagem terá largura $x_2 - x_1$ e altura $y_2 - Y_1$. Em seguida devemos classificar o corte em positivo e negativo, clicando em 'p' e 'n', respectivamente.

Esse programa também salva o nome da imagem em arquivos texto contendo as informações que serão necessárias para o treinamento.

2.2 Obtenção do ground truth das posições da bola

Criamos um programa que escreve em um arquivo a posição da bola em cada frame do vídeo para comparar com o resultado final.

Para utilizá-lo deve-se clicar na posição do centro da bola e apertar 'p' caso haja bola no frame ou apenas apertar qualquer outra tecla para passar um frame que não há bola.

Para cada vídeo geramos o gabarito da posição da bola para futura análise.

2.3 Treinamento dos detectores Cascade

Para realizar o treinamento, utilizamos as imagens obtidas a partir dos vídeos e os arquivos texto indicando quais imagens representavam bola e não-bola. Utilizamos a função de treinamento que a openCV disponibiliza para treinamento de algoritmos Cascades.

Um dos parâmetros desse classificador indica se queremos usar os descritores Haar Cascade ou LBP. Aplicamos o treinamento em vídeos e observamos a mudança dos parâmetros de treinamento nos resultados finais variando o número de iterações no treinamento.

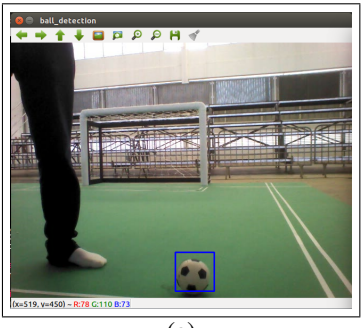
2.4 Contagem de pixels brancos e pretos

O método Viola Jones e LBP nos retorna possíveis candidatos a bola, as vezes com muitos falsos positivos. Como a bola é preta e branca, para filtrar esses candidatos, utilizamos a contagem de pixels brancos e pretos no interior da área do quadrilátero que contém o candidato. Para isso, para cada frame, fizemos uma máscara de branco e uma de preto e fizemos a interseção dela com uma imagem de mesmo tamanho que continha apenas um retângulo branco preechido na posição do candidato. Nas imagens resultantes, fizemos a contagem dos pixels diferentes de zero. Em seguida, essa quantidade foi normalizada pela área e, se o valor resultante fosse maior que um limiar estabelecido, o canditado era classificado como bola.

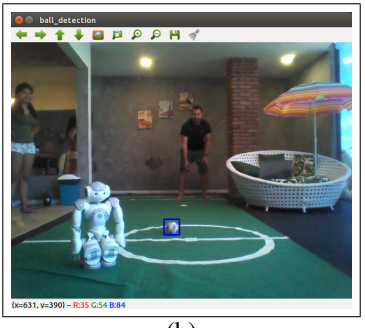
3 Resultados e Análise

Coletamos imagens de 3 vídeos (Vídeo1, Vídeo2 e Vídeo3) e aplicamos o treinamento em um outro vídeo(Vídeo4), para a anãlise dos resultados geramos 187 imagens positivas e 187 imagens negativas. Para todos os vídeos geramos gabaritos.

Nos dois métodos conseguimos identificar a bola em muitos frames do vídeo. Porém, enfrentamos o problema de falso positivos, melhoramos o método aplicando a contagem de pixel mas ainda houve muitos frames com mais de uma bola identificada.

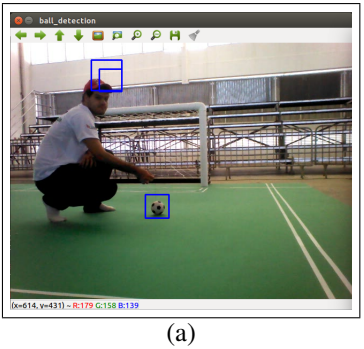


(a)

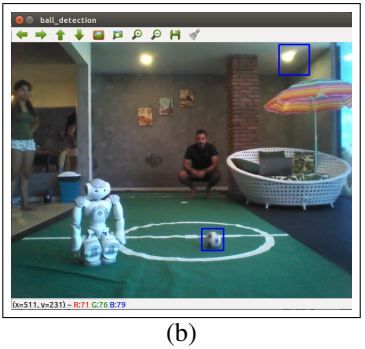


(b)

Figure 2: Frames em que a bola foi identificada corretamente



(a)



(b)

Figure 3: Frames em que houveram falsos positivos

3.1 Método Haar Cascade

Table 1: Acurácia para o rastreamento de bola por número de iterações no método Haar Cascade

Iterações	Vídeo1	Vídeo2	Vídeo3	Vídeo4(Validação)
3	88.5%	75,0%	29,1%	99,0%
5	95.3%	57,4%	20,0%	95,6%
7	81,1%	58,4%	20,3%	80,6%
9	72,63%	54,3%	15,1%	63,17%
11	70,0%	53,8%	10,5%	56,2%
13	70,9%	56,9%	10,5%	44,4%

Table 2: Número máximo de falsos positivos em um frame por número de iterações no método Haar Cascade

Iterações	Vídeo1	Vídeo2	Vídeo3	Vídeo4(Validação)
3	54	73	60	25
5	25	32	32	9
7	10	12	12	4
9	3	5	6	3
11	2	4	4	2
13	3	4	2	3

Vimos que com um treinamento mais 'relaxado' o método identificou a bola muitas fezes, porem possui um número muito grande de falsos positivos, logo, não seria muito útil para ser utilizado como rastreador sem uma boa filtragem. Quando o treinamento ficou mais rigoroso, houveram poucos falsos positivos, mas a bola também não foi identificada muitas vezes, principalmente quando estava mais longe da câmera.

3.2 Método Local Binary Paterns

Table 3: Acurácia para o rastreamento de bola por número de iterações no método Local Binary Paterns

Iterações	Vídeo1	Vídeo2	Vídeo3	Vídeo4(Validação)
7	75.3%	66,0%	15,7%	91,74%
9	53.4%	62.9%	23.25%	92,1%
11	70,0%	52,3%	19,8%	79,6%
13	38,9%	37.5%	10,4%	53.4%

Table 4: Número máximo de falsos positivos em um frame por número de iterações no método Local binary Paterns

Iterações	Vídeo1	Vídeo2	Vídeo3	Vídeo4(Validação)
7	42	60	56	133
9	14	31	24	82
11	2	16	12	63
13	2	6	3	17

O método LBP também apresentou o comportamento de diminuir a acurácia e os falsos positivos com o aumento do número de iterações. Mas aprensentou um desempenho pior que o método Haar Cascade.

4 Discussão e Conclusões

O método Viola Jones foi eficaz para encontrar a bola, visto que a identificou na em grande parte dos frames dos vídeos usados para teste. Contudo, o método retornou muitos falsos positivos. Isso pode ter sido fruto das amostras de treinamento que não foram em um número tão grande, pois foi gerada "na mão" por nós. Para corrigir o fato do método retornar falsos positivos, aplicamos contagem de pixels brancos e pretos. O resultado foi melhor, o que

mostra que essa abordagem também é válida, mas não foi totalmente eficaz e o programa continuou retornando falsos positivos.

Para melhorar esse método, podemos gerar e realizar o treinamento com mais amostras e possivelmente aplicar outro método para selecionar os candidatos, como um filtro de Kalman, visto que existe uma continuidade na posição da bola ao longo do vídeo.

Como interesse pessoal do grupo, esse projeto é uma aplicação direta de problemas enfrentados, pois participamos de uma equipe de futebol de robôs. E um dos nossos desafios é a identificação e o ratreamento da bola no campo.

References

[1] OpenCV Contribuitors. Face detection using haar cascades, 2018. https://docs.opencv.org/3.4/d7/d8b/tutorial_py_face_detection.html[Online; accessed 29-November-2018].

[2] OpenCV Contribuitors. Cascade classifier training, 2018. https://docs.opencv.org/3.4/dc/d88/tutorial_traincascade.html[Online; accessed 29-November-2018].

[3] Adrian Rosebrock. Local binary patterns with python opencv, December 7, 2015. https://www.pyimagesearch.com/2015/12/07/local-binary-patterns-with-python-opencv/[Online; accessed 29-November-2018].