Projeto Demonstrativo 06 - Rastreamento visual

Realizado em Junho de 2018

Hugo Luís Andrade Silva
Departamento de Ciência da Computação
Universidade de Brasília
Brasília, Brasil
hugosilva664@aluno.unb.br

Lukas Lorenz de Andrade
Departamento de Ciência da Computação
Universidade de Brasília
Brasília, Brasil
lukaslorenzdeandrade@gmail.com

***Abstract*—Projeto visando aplicação de diferentes métodos de rastreamento de objetos, como o Boosting, MIL, KCF e TLD e Medianflow a fim de se contrapor os resultados destes com e sem filtro de Kalman. Será aplicado no dataset [12] disponibilizado, onde se tem frames de dois veículos em movimento, o primeiro com 945 e o segundo com 9928. A eficiência dos métodos se dará pelo índice de Jaccard e robustez de cada um quanto a detecção do alvo.**

***Index Terms*—opencv, computervision, imageprocessing, tracking**

I. Introdução

O cenário de reconhecimento de padrões, assim como de pessoas e objetos, tem sido um campo visado nos últimos anos, de forma que tem sido o foco de conferências de visão computacional, como a *Computer Vision and Pattern Recognition*. Para tanto, tem sido desenvolvido distintos sistemas de rastreamento de objetos, utilizando-se redes neurais como a FlowNet [1] e a FlowNet 2.0 [2].

No entanto, este projeto cobrirá métodos mais simples (Boosting, MIL, KCF e TLD e Medianflow - conforme o artigo [14]) de forma a se avaliar os métodos pelas métricas da robustez e da acurácia. Assim, tais algoritmos serão explicitados nas subsessões seguintes.

A. Boosting -

Esse método é baseado em Adaboost, proposto em [7] e cuja aplicação nos detectores de objetos que utilizam Viola-Jones [13] é bastante popular devido a sua alta eficiência. O algoritmo consiste basicamente num cascateamento de filtros de Haar, que, por sua vez, são compostos por adições e subtrações de regiões retangulares da imagem que estiver sendo filtrada, como na figura 1. O motivo da alta eficiência no uso de tal método se deve ao conceito de imagem integral, em que cada elemento dessa imagem integral corresponde à soma dos valores dos pixels à esquerda e acima do elemento em questão. Tal representação permite que o cálculo da soma de elementos dentro de determinado retângulo seja computada a partir dos valores dos quatro vértices desses retângulos na imagem integral.

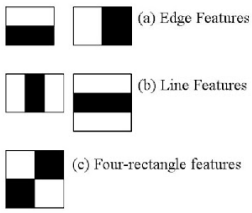

Fig. 1: Exemplos de filtros de Haar típicos, em que regiões que são subtraídas são representadas em preto e regiões a serem somadas são representadas em branco

Os classificadores feitos dessa maneira são cascateados de forma que as classificações individuais são combinadas a fim de obter uma precisão maior. Nesse contexto, os erros dos classificadores mais simples são enfatizados nos classificadores seguintes de forma que seu peso na perda a ser minimizada é aumentado. Tal procedimento é ilustrado na figura 2, em que os limiares vertical e horizontal indicam decision boundaries de classificadores simples e o tamanho dos símbolos indica o peso empregado. Na prática, esse detector não é tão preciso, até porque é o mais antigo dentre os avaliados nesse trabalho.

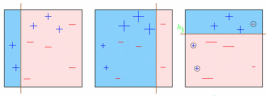

Fig. 2: Representação de boosting

B. MIL - Multiple Instance Learning

Nesse classificador, proposto em [6] e cuja página oficial pode ser encontrada em [4], a ideia é fazer as atualizações da classificação baseadas em patches dos objetos, de forma que, dentro de um patch de imagens contendo o objeto, pelo menos uma das imagens irá apresentá-lo com uma centralização próxima do ideal. A imagem 3 ilustra tal processo. O principal problema desse filtro é a ineficência, que leva a sequência de tracking com FPS baixo.

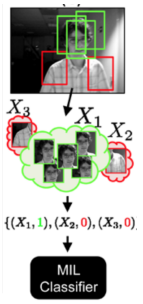

Fig. 3: Representação de MIL

C. KCF - Kernelized Correlation Filters

Proposto em [8], esse filtro combina as ideias do Boosting e do MIL, mas faz uso de propriedades matemáticas que o fazem ser eficiente e preciso. A ideia é fazer uso do fato de que os patches usados para diferenciar o objeto a ser detectado do fundo (como na figura 3) contêm muita informação redundante. Com uso de transformadas de Fourier, é feita uma diagonalização da matriz de dados de forma a tornar simples as computações. Além de rápido, esse tracker produz resultados relativamente precisos.

D. TLD - Tracking, learn and detect

Proposta em [10], a ideia é separar a tarefa em três etapas: a primeira consiste em rastrear o objeto frame a frame; a segunda consiste em localizar todas as aparências que o objeto já apresentou e corrigir o tracker caso necessário e a terceira estima os erros passados e tenta usar a informação para evitar cometer os mesmos erros no futuro. Além disso, a aprendizagem é feita com uma combinação de experts chamada P-N learning, em que P se especializa em encontrar as detecções que foram perdidas e N se especializa na detecção de alarmes falsos. Na prática, esse detector apresenta uma grande quantidade de falsos positivos, embora capaz de se recuperar de oclusões.

E. Medianflow

Proposto em [9], esse classificador faz uso de duas sequências de frames: o forward pass, que é a sequência normal de execução do vídeo e o backward pass, que faz o rastreamento usando a sequência inversa. Essa combinação leva a uma melhora do resultado em relação a abordagens tradicionais. Na prática, esse filtro e o KFC são os que obtêm resultados relativamente precisos com uma taxa alta de FPS.

F. Filtro de Kalman

O filtro de Kalman foi proposto em [11] e, além de muito usado na área de rastreamento e navegação robótica, também tem aplicações em outras áreas, como econometria e processamento de sinais. A ideia, ilustrada na figura 4, consiste em utilizar as medições já encontradas para estimar a posição futura do objeto. Quando obtida uma nova medição, seus valores são comparados com o previsto e os devidos ajustes de parâmetros são feitos. Nesse projeto, o Kalman é usado em conjunto com os outros filtros, o que significa que, para cada um dos filtros aqui apresentados, serão estudadas as versões com e sem Kalman. A teoria indica que o fato de as medições passadas estarem sendo levadas em consideração nesse filtro produz resultados mais robustos.

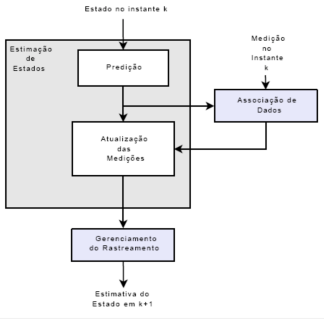

Fig. 4: Esquemático simplificado do filtro de Kalman

II. Materiais e Metodologia

O objetivo consiste em rastrear os alvos (carros) durante uma sequência de frames contida no *dataset* Kalal [12] - como no artigo [14]. Primeiramente, aplica-se os métodos no *dataset* obtendo os valores de robustez e acurácia para cada modelo. Em seguida, aplicou-se um filtro de Kalman na saída de cada método e contrastou-se com os resultados obtidos.

A. Sem Kalman

Para se alternar entre os métodos utilizados fez-se necessário apenas mudar o argumento do inicializador do rastreamento, conforme porposto nos tutoriais LearnOpencv e [5]. Dessa forma, preparou-se os dados de leitura conforme a base de dados chamada de 0 e 1 da Kalal, onde esta é a do primeiro veículo com 945 frames e aquela a com 9928.

Após se ter as imagens preparadas para uso, lê-se o *ground truth* (GT), que consiste em coordenadas da bounding box do carro a fim de se comparar o modelo previsto com as coordenadas retornadas pelo tracker. Dada a possibilidade do rastreador se perder ou haver oclusão da janela de contorno do elemento rastreado, caso em que se tem como NaN as coordenadas do GT, há a reinicialização do rastreador com as coordenadas do GT assim que disponíveis, ou seja, assim que as coordenadas deixarem de ser NaN.

Após o término da execução da sequência em questão, é calculada a roubustez de acordo com a equação 1, em que F é o número de falhas e N é o número de frames válidos na sequência. Vale ressaltar que, para o cálculo de N, foram contabilizados apenas os frames com BB, ou seja, frames que tiveram NaN como GT foram desconsiderados. Além disso, o contador de falhas F só foi incrementado em duas situações:

- Quando a operação de rastrear novo frame do tracker retornava código de falha em frames válidos
- Quando a interseção entre a bounding box do tracker e do GT era zero

Isso significa que falhas nos frames com NaN (frames inválidos) não foram contabilizadas, afinal, o tracker não teria o que acertar. Além disso, nos casos em que há oclusão e

o GT apresenta-se como NaN, o tracker continua rodando tendo como objetivo estimar a posição aproximada do objeto através de sua direção e velocidade de movimento anteriores. Quando as coordenadas de BB ficam disponíveis novamente, só é incrementado o contador de falhas se uma das condições acima tiver sido atingida, ou seja, caso o tracker seja robusto à oclusão e ainda possua BB com overlap com o GT quando a oclusão chegar ao fim, não é registrada falha.

$$rob = e^{-S\frac{F}{N}} \tag{1}$$

Nos casos em que há coordenadas de BB tanto do tracker quando do GT (não houve falha de rastreamento e nem oclusão), são calculadas as áreas da interseção e união entre as BB para se obter o Jaccard, sendo que o Jaccard médio é reportado ao fim das sequências de vídeo. Além disso, foi registrado também o FPS médio de cada run, a fim de se verificar as velocidades dos diferentes métodos.

Por fim, aplicou-se os métodos no dataset [12] da Kalal-Volkswagen, gerando as métricas a fim de comparar os métodos de forma mais gernérica.

B. Com Kalman

Dado que o filtro de Kalman é aplicado no resultado obtido de cada método, os passos anteriores são realizados da mesma forma, com a diferença de que na etapa de inicialização do rastreador. Nesta, também se inicializa o filtro de Kalman e, após se ter as coordenadas da bb, aplica-se o filtro nas direções 'x' e 'y', separadamente - como no tutorial Filterpy.

III. Resultados

Para o rastreador KFC-Kalman, obteve-se a figura 7, de modo que se tem o exemplo de quando o rastreador acerta (no *dataset* 0) e erra (no *dataset* 1). A *bounding box - bb* vermelha corresponde ao GT e a azul ao predito. Percebe-se pela figura 7 que as bb do GT não correspondem de fato aos limites do veículo rastreado.

Além disso, os gráficos 5 e 6 representam a distribuição dos dados obtidos nos resultados dos algoritmos quanto a robustes e o índice de Jaccard.

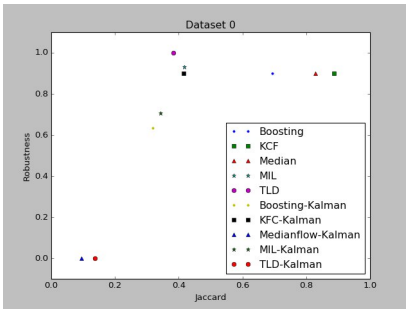

Fig. 5: Gráfico dos resultados com e sem filtro de kalman para o dataset 0, referentes as tabelas geradas nas subsessões III-A e III-B.

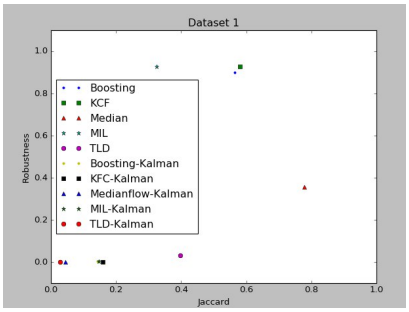

Fig. 6: Gráfico dos resultados com e sem filtro de kalman para o dataset 1, referentes as tabelas geradas nas subsessões III-A e III-B.

(a) Dataset 0

(b) Dataset 1

(c) Dataset 1 - Reinicialização do método

Fig. 7: Exemplo e contra exemplo de acerto para os *datasets* 0 e 1, respectivamente, com o desenho da bb e do previsto para o método KFC com o filtro de Kalman. Além disso, na figura (c), tem-se um expoente da reinicialização do rastreador, de modo que este é inicializado com os valores do gt.

A. Sem o filtro de Kalman

A partir da execução dos algoritmos abordados neste trabalho, foram obtidos os resultados descritos nas tabelas I e II.

TABLE I: Resultados obtidos com os 5 rastreadores no dataset 0.

$Classificador$	Jcc_{media}	Jcc_{Std}	rob	FPS
Boosting	0.693412	0.199753	0.900639	61.509815
KCF	0.887354	0.067973	0.965718	136.421570
MedianFlow	0.828978	0.143858	0.900639	547.498657
MIL	0.416822	0.309185	0.932611	24.867821
TLD	0.382347	0.181916	1.000000	27.415567

TABLE II: Resultados obtidos com os 5 rastreadores no dataset 1.

$Classificador$	Jcc_{media}	Jcc_{Std}	rob	FPS
Boosting	0.564843	0.290225	0.901292	100.394814
KCF	0.581136	0.290075	0.926619	764.085388
MedianFlow	0.777721	0.237549	0.356175	595.018494
MIL	0.324462	0.192935	0.926619	23.238115
TLD	0.397417	0.252297	0.031190	30.535599

No segundo dataset, onde se tem 9928 frames, devido ao número de frames ser quase 10 vezes mais que do anterior, o número de falhas aumenta, diminuindo a precisão relativa dos métodos, de modo geral. Além disso, a segunda sequência trata de uma perseguição policial, em que o veículo rastreado constantemente ultrapassa os limites da tela e é recapturado posteriormente, o que dificulta o tracking, já que vai além dos casos normais de oclusão em que o objeto é encoberto com algum obstáculo.

TABLE III: Resultados obtidos com os 5 rastreadores no dataset Volkswagen.

$Classificador$	Jcc_{media}	Jcc_{Std}	rob	FPS
Boosting	0.526834	0.281459	0.517159	72.409859
KCF	0.753961	0.151730	0.055983	416.653992
MedianFlow	0.066386	0.205666	0.000000	363.869263
MIL	0.258059	0.136861	0.932370	14.518142
TLD	0.089191	0.202578	0.000001	21.348278

B. Com o filtro de Kalman

A partir da aplicação do filtro de Kalman nos métodos Boosting, KCF, Medianflow, MIL e TLD obteve-se os resultados das tabelas IV e V.

TABLE IV: Resultados obtidos com os 5 rastreadores no dataset 0.

$Classificador$	Jcc_{media}	Jcc_{Std}	rob	FPS
Boosting	0.317974	0.165519	0.635408	66.020493
KCF	0.415214	0.184834	0.247747	64.572540
MedianFlow	0.093893	0.137867	0.000002	835.276917
MIL	0.342144	0.159674	0.705508	15.667500
TLD	0.137459	0.158183	0.000006	141.138672

TABLE V: Resultados obtidos com os 5 rastreadores no dataset 1.

$Classificador$	Jcc_{media}	Jcc_{Std}	rob	FPS
Boosting	0.142367	0.167683	0.001664	104.747284
KCF	0.159086	0.198663	0.000046	256.178864
MedianFlow	0.044651	0.104130	0.000001	703.368652
MIL	0.147397	0.164493	0.003282	15.480242
TLD	0.028832	0.081362	0.000001	70.765167

IV. Análise de resultados

A partir das tabelas I,II,IV e V, percebe-se que o efeito do filtro de Kalman na saída do método piora os resultados uma vez que as coordenadas são dispersas em torno do valor de referência, o resultado do método analisado. Além disso, o seu desempenho, em termos de custo computacional, depende diretamente do hardware disponível, ou seja, da CPU.

Além disso, por meio das tabelas citadas anteriormente, tem-se que o rastreador KCF obteve o melhor desempenho nesta base de dados (maior Jaccard), com o menor desvio padrão. Quanto ao tempo de execução (FPS), tem-se que o método Medianflow obteve o melhor desempenho, de forma geral, conforme as tabelas nas sessões III-A e III-B, e com um Jaccard próximo ao do KFC.

No entanto, os piores métodos para estas bases de dados foi o MIL e o TLD. Principlamente quanto ao desempenho FPS, o MIL obteve o menor índice e o menor jaccard dentre todos os métodos, no geral.

Por fim, uma possíbilidade de melhoras para os resultados com kalman, seria a variação dos parâmetros de inicialização, de forma que esta não fosse randômica, mas sim com os pontos do gt - explorando mais as potencialidades do filtro.

References

[1] Flownet 2.0: Evolution of optical flow estimation with deep networks.

[2] Flownet: Learning optical flow with convolutional networks.

[3] Hugo Silva; Lukas Andrade. Pd06. https://github.com/hsilva664/PVC_Proj6. Accessed: 2018-06-12.

[4] Boris Babenko; Ming-Hsuan Yang; Serge Belongie. Mil project page. http://vision.ucsd.edu/~bbabenko/new/project_miltrack.shtml. Accessed: 2018-06-12.

[5] Opencv dev team. Opencv documentation. https://docs.opencv.org/. Accessed: 2018-05-01.

[6] Thomas G Dietterich, Richard H Lathrop, and Tomás Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial intelligence*, 89(1-2):31–71, 1997.

[7] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[8] João F Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):583–596, 2015.

[9] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. Forward-backward error: Automatic detection of tracking failures. In *Pattern recognition (ICPR), 2010 20th international conference on*, pages 2756–2759. IEEE, 2010.

[10] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. Tracking-learning-detection. *IEEE transactions on pattern analysis and machine intelligence*, 34(7):1409–1422, 2012.

[11] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

[12] R. Fergus L. Fei-Fei and P. Perona. Tracking dataset kalal. Dataset available at http://cmp.felk.cvut.cz/~vojirtom/dataset/tv77/.

[13] Paul Viola and Michael J Jones. Robust real-time face detection. *International journal of computer vision*, 57(2):137–154, 2004.

[14] Lukas Čehovin, Aleš Leonardis, and Matej Kristan. Visual object tracking performance measures revisited. *arXiv preprint arXiv:1502.05803*, 2016.