Contagem de vagas em estacionamentos

Daniel Marcos Botelho Barbosa
17/0052427
DanielM.B.Barbosa@hotmail.com

Gabriel Filipe Botelho Barbosa
12/0050935
gabrielfbbarbosa@gmail.com

Abstract

Este projeto se caracteriza como uma pesquisa sobre um método de contagem de vagas na área da visão computacional sem a utilização de inteligência artificial.

Várias das atividades no campo de visão computacional se apoiam em atividades menores. Com isso, é bastante notório que o processo de contagem de vagas em um estacionamento tem uma grande abrangência no campo de estudos, não apenas da visão computacional. Técnicas como análise de texturas, detecção de gradiente, processamento morfológico (limiarização e binarização, dilatação e erosão), extração de características e clusterização estão bem presentes neste trabalho.

As tarefas realizadas envolvem desde abrir uma imagem simples do tipo `BGR` *até as mais diversas técnicas para o desenvolvimento de uma solução da problemática. Para isso, a liguagem utilizada foi C++ com o padrão C++ 11 e a versão 3.2.0 do OpenCV.*

1. Objetivos

O objetivo principal o desenvolvimento desse projeto é chegar à uma solução sobre a precisão e o resultado do processo utilizado para a contagem das vagas. Além disso, a atividade como um todo objetiva a exploração dos conceitos aprendidos ao longo do curso de Princípios de Visão Computacional e ferramentas disponíveis na biblioteca em questão, OpenCV[4]. Com isso, tornar afim delas.

2. Introdução

Para se obter esse resultado da contagem, foi realizado um procedimento complexo desde detecção de gradiente até criação de algorítmos para determinar a similaridade de duas retas. O procedimento desta metodologia foi o seguinte:

Etapa 1 Abre uma imagem de estacionamento;

Etapa 2 Calcula as matrizes GLCM's;

Etapa 3 Aplica o algorítmo de detecção *Sobel Detector*[6];

Etapa 4 A partir de Sobel, aplica *Hough Line Transform*;

Etapa 5 A partir de Sobel, calcula a Transformada Probabilística de Hough.

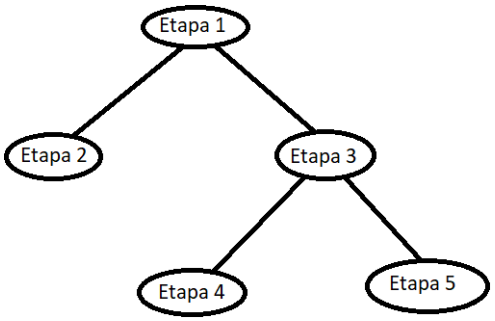

Figure 1. Diagrama em árvore da ordem e disposição das etapas realizadas.

2.1. Imagem

Os padrões de orientação e de pixelagem são tratados diferentemente pelo OpenCV.

2.1.1 Padrão RGB

Tanto imagens como vídeos são armazenados da mesma forma. A classe `cv::Mat` pode armazenar a matriz de pixels de uma imagem, bem como os frames de um vídeo.

Cada pixel tem seus canais armazenados, por padrão, na ordem Vermelho (*Red*), Verde (*Green*) e Azul (*Blue*) (`RGB`). No entanto, a biblioteca em questão utiliza um padrão diferente, definido como `BGR`, invertendo a posição do valor do pixel azul com o vermelho.

É interessante converter a imagem do tipo `BGR` para `gray scale` porque o algorítmo necessita. Trabalhando com níveis de cinza só existe uma única informação a ser extraída: a intensidade do pixel, de 0 a 255. Todavia, ao trabalhar com imagem colorida, as informações triplicam. Três canais não relacionados aumentam o nível de complexidade por não terem uma ordenação linear que seja fácil de colocar em uma matriz.

2.1.2 Coordenadas

O padrão de referências cartesianas computacionais é com origem no extremo noroeste da tela. Com eixo x crescendo para a direita e eixo y crescendo para baixo. No entanto, o padrão adotado pela biblioteca é situada com origem no extremo norte-oeste da tela, porém com eixo x crescendo para baixo e eixo y crescendo para a direita.

2.1.3 Acesso dos dados

Para realizar o acesso dos dados matriciais da classe `cv::Mat`, utiliza-se, recomendavelmente, pela documentação[1], o método `cv::Mat::at<type T> (cv::Point(j, i))`. Contudo, esse acesso garante o percorrimento para cada índice de acesso em toda a matriz. Com isso, o desempenho é comprometido. Para a retificação desse problema, pode-se utilizar o recurso de ponteiros. A biblioteca disponibiliza um tipo de dados unsigned char chamado `uchar * cv::Mat::data` em que aponta para o primeiro canal do primeiro pixel de uma matriz `cv::Mat`. No entanto, isso é para o caso de imagem colorida, como estamos trabalhando com níveis de cinza, não há preocupação com isso.

Desse modo, como a imagem, neste ponto, está em níveis de cinza, o retorno desse método `at<unsigned char>(j, i)`, em cada pixel (i, j), será um valor de 0 a 255 indicando a intensidade de pixel para cada pixel analisado. Então é só comparar com os pixels da vizinhança.

3. Trabalhos relevantes

4. Metodologia proposta

análise de texturas, detecção de gradiente, processamento morfológico (binarização e limiarização), extração de características e clusterização

4.1. Análise de texturas

As medidas de textura de co-ocorrência de nível de cinza têm sido a força de trabalho da textura da imagem desde que foram propostas por Haralick[8] na década de 1970.

Uma matriz de co-ocorrência, ou distribuição de co-ocorrência,[7] é uma matriz que é definida sobre uma imagem para ser a distribuição de valores de pixel co-ocorrentes (valores de tons de cinza ou cores) a um dado deslocamento. Esta matriz aproxima a distribuição de probabilidade conjunta de um par de pixels.

Ela representa a relação entre distância e relação de angulação espacial sobre uma sub-relação de imagem de uma região específica e de tamanho específico. Ou seja, com essa matriz de coocorrência é possível detectar, de certa forma, a textura de objetos capturados em uma imagem. Neste projeto terão quatro matrizes finais que serão obtidas pelas direções 0º, 45º, 90º e 135º, em ambos sentidos.

4.2. Detecção de gradiente

O pré-processamento da imagem foi realizado com o algorítmo de Sobel.

Sobel O detector de bordas de Sobelfootnotesize [6] se basea em algumas alterações e adaptações do operador matemático Laplaciano[2] para um espaço dimensional discreto e bidimensional[3]. O efeito desse operador no espectro matemático é o mesmo quando aplicado numa imagem, ou seja, ele ameniza variações de baixa frequência e atenua variações de alta frequência.[5]

Ele realiza esse procedimento realizando uma convolução nos eixos da imagem usando um *kernel* direcional e realizando a média dos resultados. Ele é, geralmente, quadrado com tamanho ímpar, comumente 3, onde a soma de todos os elementos é 0.

-1	0	1
-2	0	2
-1	0	1

Figure 2. Operador Sobel horizontal para imagens bidimensionais

Além disso, se considerarmos o *kernel* utilizado para computador o gradiente horizontal (Figure 2), ou seja, na coordenada x, todos os elementos da coluna central será 0. Todos os *kernels* não são nada além de uma rotação ao redor do pixel central de algum outro.

4.3. Processamento morfológico

4.3.1 Limiarização e Binarização

A binarização realizada após as detecções serem finalizadas foi trivial, mas merece ser rapidamente explicada pois o valor final não foi 0 ou 1. Para que fosse possível se visualizar a binarização, ela foi escalada de forma que 1 seja 255 mas que não possua nenhum outro número entre 255 e 0. Ou seja, todo pixel maior que um *threshold* determinado seria setado como 255, e 0 caso contrário.

4.3.2 Dilatação e Erosão

Filtros morfológicos sxploram as propriedades geométricas dos sinais (níveis de cinza da imagem). Para filtros morfológicos, as máscaras são denominadas elementos estruturantes e apresentam valores 0 ou 1 na matriz que correspondem ao pixel considerado. Os filtros morfológicos básicos são o filtro da mediana, erosão e dilatação. No entanto, utilizou-se apenas a dilatação e a erosão.

Dilatação provoca efeitos de dilatação das partes escuras da imagem (baixos níveis de cinza), gerando imagens mais claras.

Erosão provoca efeitos de erosão das partes claras da imagem (altos níveis de cinza), gerando imagens mais escuras.

4.4. Extração de características

4.4.1 Hough

A transformação de Hough é uma técnica que pode ser usada para isolar características de uma determinada forma dentro de uma imagem. Pelo fato de necessitar que as características desejadas sejam especificadas em alguma forma paramétrica, a transformação Hough clássica é mais comumente usada para a detecção de curvas regulares, como linhas, círculos, elipses, etc. A transformada Hough generalizada pode ser empregada em aplicações onde uma descrição analítica simples de uma ou mais características não é possível. Apesar das suas restrições de domínio, a transformada clássica de Hough contém muitas aplicações.

A vantagem principal da transformada de Hough é que é uma técnica que não é afetada por espaços entre as características obtidas por detectores de borda e é relativamente invariante ao ruído nas imagens. A ideia de se utilizar esta transformada é para encontrar as linhas que dividem cada vaga.

4.4.2 *Probabilistic* Hough *Transform*

Na transformada de Hough, percebe-se que mesmo para uma linha com dois argumentos é preciso muito processamento. A Transformada probabilística de Hough é uma otimização da transformada clássica de Hough. Não leva todos os pontos em consideração, em vez disso, toma apenas um subconjunto aleatório de pontos que é suficiente para a detecção da linha. Apenas temos que definir o limiar.

4.5. Agrupamento

Clusterização é uma técnica que é usada para particionar elementos em um conjunto de dados, de modo que elementos semelhantes sejam atribuídos ao mesmo cluster, enquanto elementos com propriedades diferentes são atribuídos a diferentes clusters. É utilizado para executar uma pesquisa eficiente de elementos em um conjunto de dados. O agrupamento é particularmente eficaz em dados multidimensionais que, de outra forma, podem ser difíceis de organizar de maneira efetiva.

Uma das primeiras técnicas de agrupamento na literatura é o método de *clustering* K-means. Nesta técnica, o agrupamento é baseado na identificação de elementos K no conjunto de dados que podem ser usados para criar uma representação inicial de clusters. Esses elementos K formam as sementes do cluster. Os elementos restantes no conjunto de dados são então atribuídos a um desses clusters. Mesmo que o método pareça direto, sofre o fato de que pode não ser fácil identificar claramente os elementos K iniciais ou as sementes para os clusters.

K-means foi utilizado para que as linhas retas ou segmentos de retas semelhantes geradas pelas transformadas Hough sejam agrupadas. Com isso, a contagem tornou-se algo já possível de se enxergar.

5. Resultados

6. Conclusões

References

[1] Devdocs. *http://devdocs.io/*.

[2] Laplace operator. *http://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/laplace_operator/laplace_operator.html*.

[3] Laplacian edge detector. https://de.mathworks.com/matlabcentral/answers/uploaded_files/51086/Edge%20detection%20-%20Laplace%20edge%20detection%20algorythm.pdf. Accessed: 20/06/2017 - 21/06/2017.

[4] Opencv 3.2.0 documentation. *http://docs.opencv.org/3.2.0/*.

[5] Sobel derivatives. *http://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/sobel_derivatives/sobel_derivatives.html*.

[6] Sobel edge detector. http://homepages.inf.ed.ac.uk/rbf/HIPR2/sobel.htm. Accessed: 20/06/2017 - 21/06/2017.

[7] M. Hall-Beyer. The glcm tutorial. *http://www.fp.ucalgary.ca/mhallbey/tutorial.htm*, Fevereiro, 2007.

[8] R. M. Haralick, K. Shanmugam, and I. Dinstein. Textural features for image classification. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-3(6):610–621, Dezembro, 1973.