Contagem de veículos e medição de velocidade a partir de vídeos

UFMG
Visão Computacional
1º semestre 2011

Aluno: Alex Benfica
Prof.: Mário Montenegro

Tópicos desta apresentação

Tempo previsto: 20'

- Motivação
- Outros trabalhos na área
- Escopo
- Metodologia
- Conclusões e resultados
- Melhorias e trabalhos futuros

Contagem de veículos e medição de velocidade a partir de vídeos

Motivação

Fiscalização

450 mil novos veículos por mês no Brasil
(Fenabrave)

Trânsito + complicado

Estatísticas para otimização

Controle adaptativo de semáforos

Fluxo e utilização das vias em tempo real: obras, interrupções programadas

Controle de tráfego inteligente, GPS

Outras motivações

- Medição de velocidade
  - Fiscalização
  - Sinalização adequada
- Contagem de veículos
  - Detecção de tráfego crescente em vias residenciais
  - Fluxo medido em tempo real e detecção de congestionamentos

Contagem de veículos e medição de velocidade a partir de vídeos

Outros trabalhos na área

Revisão bibliográfica disponível para consulta

Objetivos de outros trabalhos

- Contagem
- Velocidade
- Contra mão
- Veículos parados
- Identificação de placa
- Categoria
- Atravessar rua
- Direção assistida e alertas
- Direção autônoma
- Deteção de pista e faixas
- Vigilância
- Monitorar cruzamentos

Abordagens de outros trabalhos

- Estimativa do *ground plane*
- Detecção do modelo 3D do veículo
- Detecção e subtração do *background*
- Uso de 2 câmeras
- Câmeras de CCTV (menor custo)
- Câmeras: *low frame rate* (*storage*)
- Detecção do tipo de veículo
- Funcionamento em cruzamentos, esquinas ou bifuracações
- Extração de contornos
- Filtro de *Kalman*
- Com ou sem calibração
- Utilização de cores
- Utilização de *features*
- Quinas e rigidez

Limitações de outros trabalhos

- Oclusão parcial
- Oclusão total temporária
- Luzes do veículo, contínuas e piscantes
- Iluminação noturna da via
- Sombras dos veículos (segmentação)
- Degradação de performance com a distância entre os veículos e a câmera
- Chuva, neblina, poeira, fumaça
- Sombras incidentes sobre a pista

Resultados de outros trabalhos

- **Comparação difícil**
  - Limitações variadas
  - Várias formas de aquisição de video
  - Sem base padrão para testes sistemáticos de cada caso
- Resultados específicos
  - Dependentes da cena
  - Objetivos diferentes

Números

Em **boas condições de iluminação***, há trabalhos com

- 90 a 95% de acerto na contagem
- 90 a 93% de acerto na medição da velocidade

* como quantificar?

Contagem de veículos e medição de velocidade a partir de vídeos

Escopo

Escopo

- 1 faixa e 1 sentido
- Veículo deve caber no *frame*
- 30 a 50 frames para identificação do *background*.
- Video com ~30 fps
- 1 veículo de cada vez no *frame*

- Parâmetros manuais
  - Tamanho máximo e mínimo de veículos
  - Limiares relativos a sombras
- Parâmetros dependem da posição da câmera e resolução do vídeo
- Cálculo de velocidade **ainda** não implementado

Contagem de veículos e medição de velocidade a partir de vídeos

Metodologia

Linguagens, bibliotecas, sistemas operacionais

- Python com wrapper para Open CV
- Open CV
- Ubuntu 11.04
- Numpy

Tudo o que foi utilizado é OpenSource e multiplataforma (Windows e Linux, ao menos)

Aquisição dos vídeos

- Câmera Cybershot DSC-W180.
- AWB desligado
- Vídeo 320x240
- 30 fps, RGB
- Angulo de ~45º da câmera em relação à faixa
- Distância ~9m até o centro da pista

Detecção do *background*

- Amostra 50 frames sem veículos (~2s de vídeo)
- Filtro de mediana em cada frame
- **Média entre todos os frames é o background**
- É feito em cores, em cada canal

Método de processamento

Cada *frame* é lido e processado individualmente

O ***background*** e o ***frame* anterior** estão sempre disponíveis para comparação

Processamento em cada frame

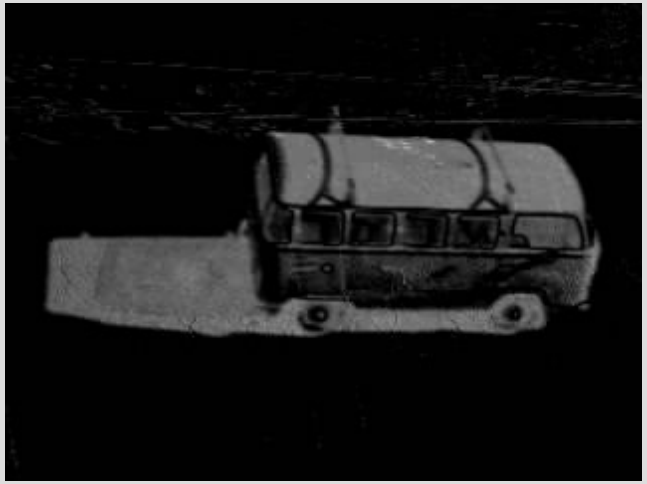
- Cada processamento abaixo gera uma **imagem binária em memória**:

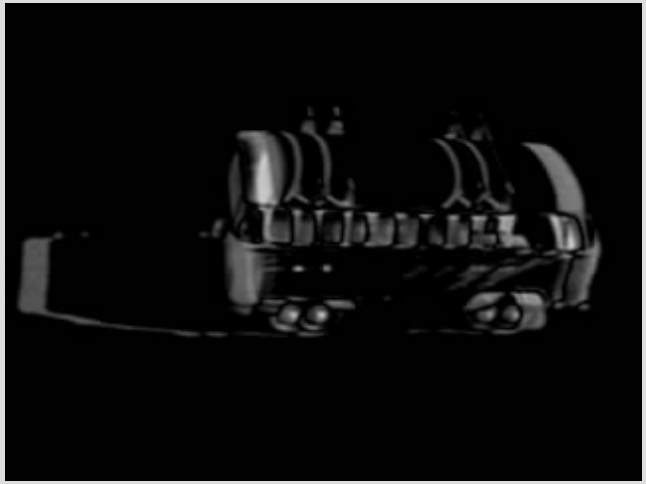
  1 - Detecção de sombras
  2 - Diferenças em relação ao *background*
  3 - Diferenças em relação ao *frame* anterior

  Cada um é explicado nos slides seguintes

Detecção de sombras

Sombra atrapalha pois aparece nas diferenças





Como dectar a sombra

Espaços de cor HSV – Em cada pixel...
(Hue, Saturation, Value) – Relações e limiares

HUE

- **Ih** = *frame*
- **Bh** = *background*

SATURATION

- **Is** = *frame*
- **Bs** = *background*

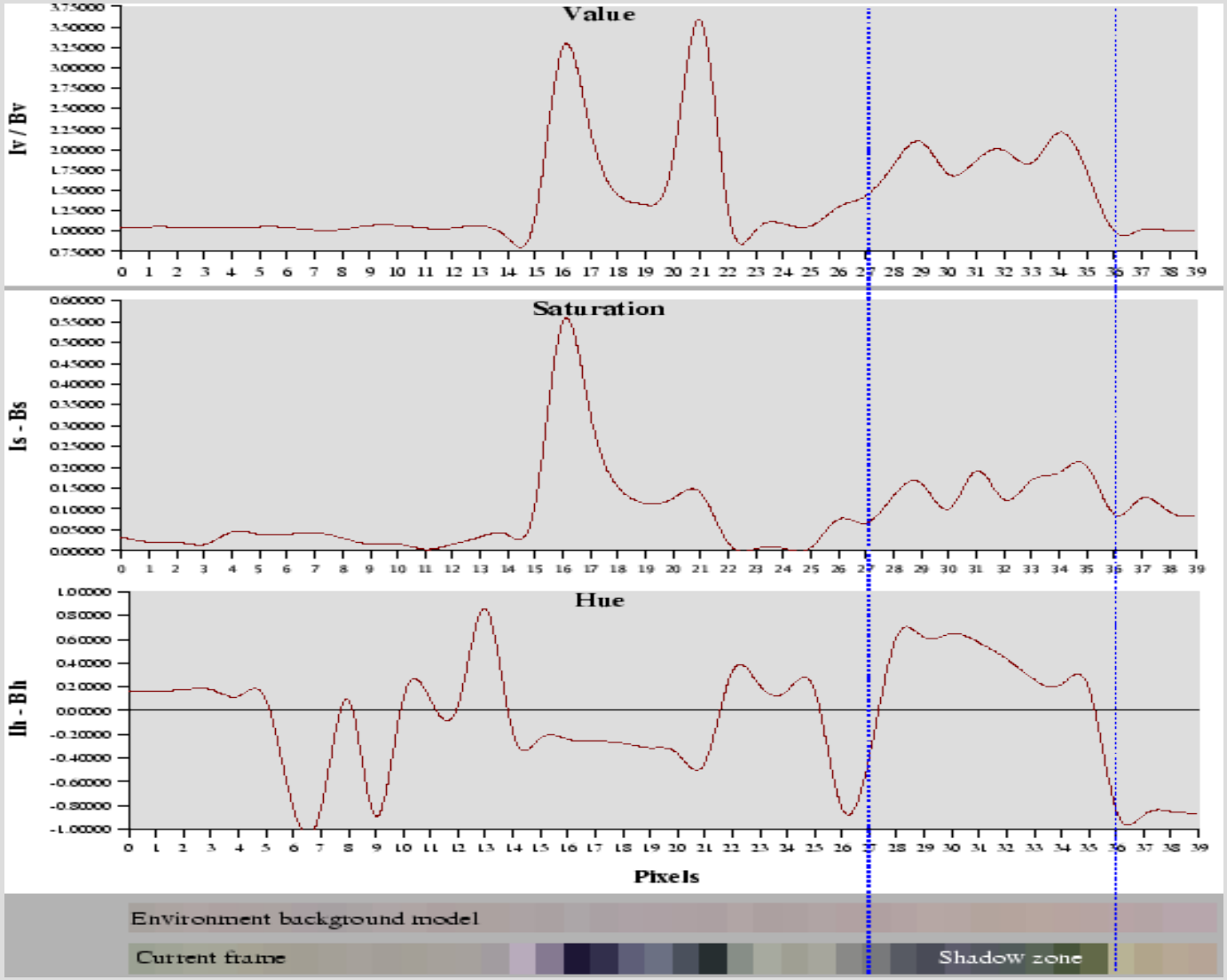
VALUE

- **Iv** = *frame*
- **Bv** = *background*

- Relações
  - Iv / Bv
  - Is – Bs
  - Ih – Bh
- Limiares (gráfico)
  - alfa
  - beta
  - tH
  - tS

Como detectar a sombra

Sombra entre as linhas azuis: usar limiares



Limiares sobre as relações HSV

SPk(x,y) = Shadow Point Mask(x,y)
SPk(x,y) = 1 → pontos verdes na imagem

$$SP_k(x,y) = \begin{cases} 1 & \text{if } \alpha \leq \frac{I_k^V(x,y)}{B_k^V(x,y)} \leq \beta \\ & \wedge\, I_k^S(x,y) - B_k^S(x,y) \leq \tau_S \\ & \wedge\, \left| I_k^H(x,y) - B_k^H(x,y) \right| \leq \tau_H \\ 0 & \text{otherwise} \end{cases}$$

SPK

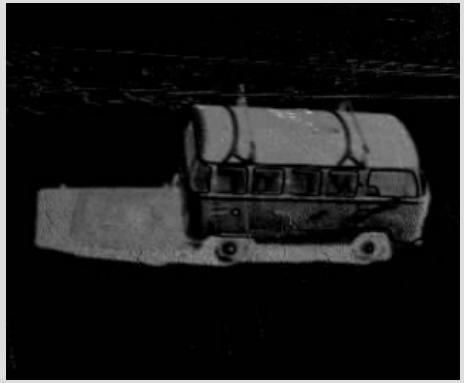
A escolha dos limiares varia com a iluminação da cena!

Diferença frame e *background*

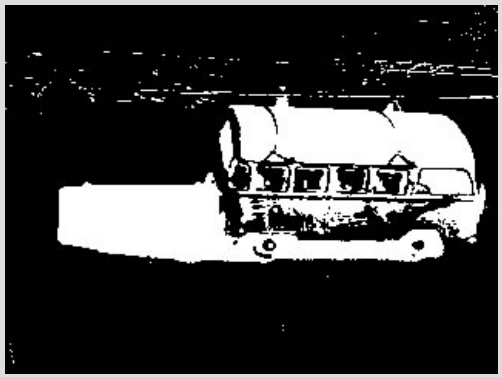
Ambos convertidos para cinza

- Diferença absoluta


-

=


- Limiarização:
  threshold = 40



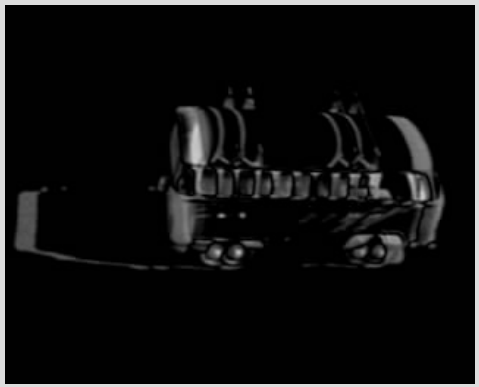
= DIF_BG

Diferença do frame anterior

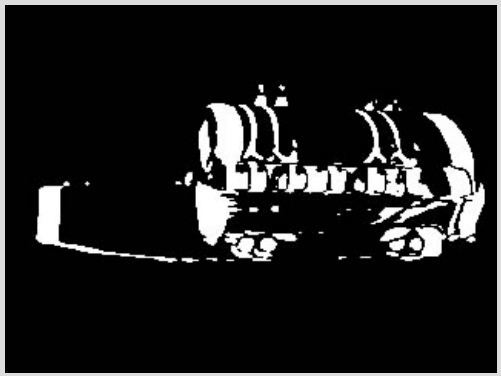
Ambos convertidos para cinza

- Diferença absoluta


-

=


- Limiarização:
  threshold = 40

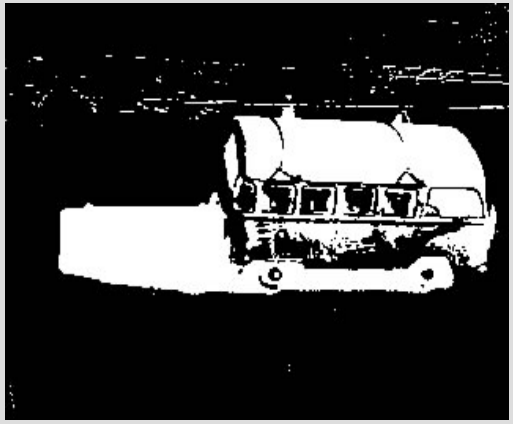

= DIF_FA

Eliminando sombras da imagem

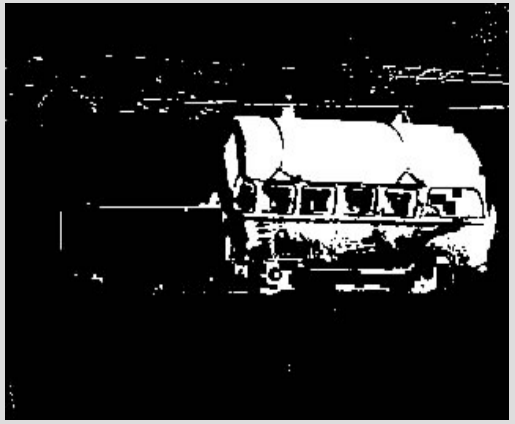
- **SPK: Dilate** 2x (Fechar buracos e pegar bordas) + **Not** (inverte). = IMG_A
- Faz um **AND** de IMG_A com DIF_BG

- Resta a diferença do fundo sem a sombra, mas ainda com ruídos (DIF_SS)

IMG_A



DIF_BG

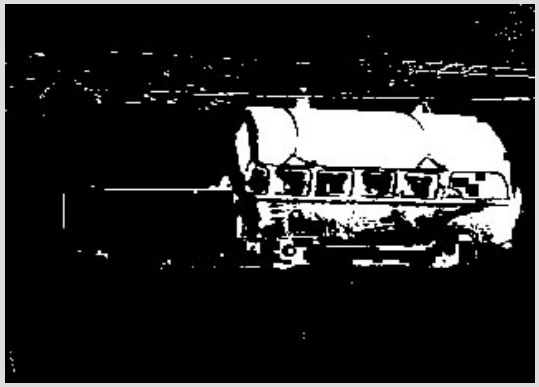


DIF_SS

Transformações morfológicas

Eliminando ruídos

- Em DIF_SS:
  - Fitro de mediana, Erode2x, Dilate3x = IMG_C
  - AND de C com DIF_SS = IMG_D
  - **Dilate2x** em DIF_FA = IMG_E
  - AND de IMG_D com IMG_E = DIF_FINAL



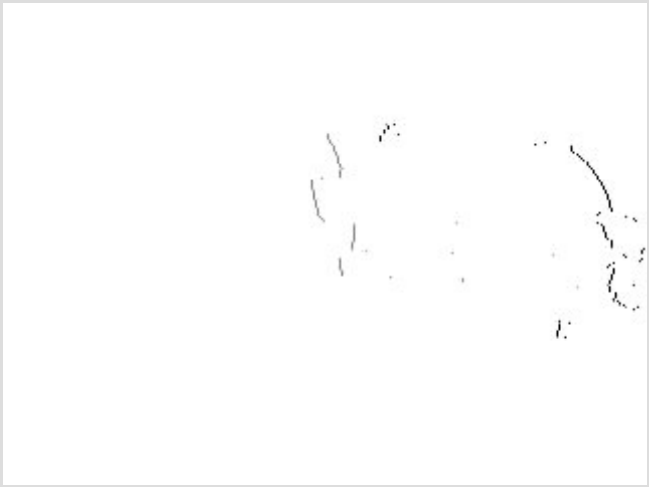
DIF_SS

DIF_FA

DIF_FINAL

Linhas iniciais e finais

- Varre a imagem DIF_FINAL e gera IMG_P (do meio)
  - da esquerda para a direita: gerar linha verde
  - da direita para a esquerda: gerar linha vermelha



Distância entre linhas

Pontos máximos e mínimos

distanciaLinhas = abs(xlinhaFinal - xlinhaInicial

minI = menor x dos pontos de IMG_P que geraram a linha inicial

maxF = maior x dos pontos de IMG_P que geraram a linha final

Existe veículo?

```
588
589 # Se as linhas estão a distância maior que um veículo e não estão próximas às bordas
590 if (distanciaLinhas > distMinimaExisteVeiculo and  ((minI > 20) and (maxF < 300)) ):
591     # se NÃO tinha veículo e agora tem...
592     if temVeiculo == False:
593         # conta +1 veículo.
594         nVeiculos += 1
595     # agora tem veículo.
596     temVeiculo = True
597
598 # se as linhas estão próximas ou na bordas...
599 else:
600     # se a distancias entre as linhas é pequena...
601     if ((distanciaLinhas < distMinimaNaoExisteVeiculo)
602         # OU as duas linhas estão próximas ao lado esquerdo...
603         or ((minI < 80) and (maxF < 80))
604         # OU as duas linhas estão próximas ao lado direito...
605         or ((minI > 240) and (maxF > 240))
606         # OU estão MUITO DISTANTES...
607         or distanciaLinhas > 280 ):
608         # não tem veículo (prepara para contar outro)
609         temVeiculo = False
610
611
```

Resultados obtidos

Nos casos de testes, considerando o escopo, **mais de 80% de acerto**, com ajuste manual de parâmetros.

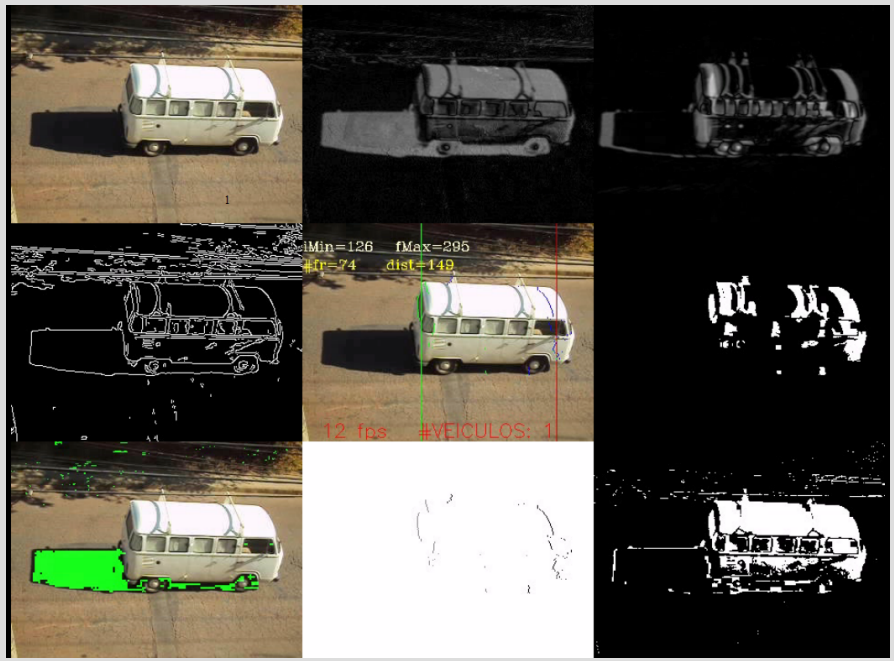
Robusto a **pequenas** variações de iluminação.

Falsas contagens quando o veículo para em frente à câmera e recomeça o movimento.

Detecção de sombras é sensível a grandes mudanças de luminosidade

Legenda do vídeo

(1) Original
(2) DIF_BG
(3) DIF_FA
(4) Canny (tentativa)
(5) RESULTADO
(6) DIF_FINAL
(7) Sombra destacada
(8) IMG_P
(9) DIF_SS



Trabalhos futuros

- Usar movimento das linhas para velocidade
- Recalcular background em intervalos regulares
- Definir limiares em tempo de execução
- Detectar mais de um veículo no frame
- Contar em mais de uma faixa