CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA
FUNDAÇÃO DE ENSINO "EURÍPIDES SOARES DA ROCHA"
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Estudo Comparativo de Algoritmos de Extração de Fundo

VICTOR UBIRACY BORBA

Marília, 2014

CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA
FUNDAÇÃO DE ENSINO "EURÍPIDES SOARES DA ROCHA"
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Estudo Comparativo de Algoritmos de Extração de Fundo

Monografia apresentada ao Centro Universitário Eurípides de Marília como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciência da Computação.
Orientador: Prof. Ms. Rodolfo Barros Chiaramonte.

Marília, 2014

CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

TRABALHO DE CONCLUSÃO DE CURSO – AVALIAÇÃO FINAL

Victor Ubiracy Borba

Estudo Comparativo de Algoritmos de Extração de Fundo

Banca examinadora da monografia apresentada ao Curso de Bacharelado em Ciência da Computação do UNIVEM/F.E.E.S.R., para obtenção do Título de Bacharel em Ciência da Computação.

Nota: 8 (oito)

Orientador: Rodolfo Barros Chiaramonte

1°. Examinador: Paulo Augusto Nardi

2°. Examinador: Fábio Dacêncio Pereira

Marília, 02 de dezembro de 2014.

Dedico este trabalho à minha família, amigos e a todos que me ajudaram e me apoiaram durante este trabalho.

AGRADECIMENTOS

A Deus, primeiramente, pela vida e por ter me guiado nessa caminhada.

A minha família que sempre acreditou em mim e me deu forças para concluir esta conquista.

Aos meus amigos que sempre estiveram ao meu lado me cobrando e ajudando a atingir meus objetivos.

A meu orientador pela dedicação, paciência e auxilio prestados a mim.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Siglas

CCD - *Charge Coupled Device*

CIE - Comissão Internacional sobre iluminação

CMY - Modelo de cor composto pelas cores: *ciano, magenta, yellow*

CRT - *Cathodic Ray Tube*

dB - *Decibel*

DIP - *Digital Image Processing*

EGMM - *Enhanced Gaussian Mixture Model*

HSI - Modelo de cor composto pelas cores: *hue, saturation, intensity*

LCD - *Liquid Cristal Display*

LED - *Light Emitting Diode*

MoG - Mistura de Gaussianas

NTSC - *National Television Standards Committee*

PAL - *Phase Alternating Line*

PDI – Processamento Digital de Imagens

PSNR - *Peak signal to noise ratio*

RAM - *Random Access Memory*

RGB – Modelo de cor composto pelas cores: *red, green, blue*

RMA - Robôs Móveis Autônomos

SCV - Sinal Composto de Vídeo

SDRAM - *Synchronous dynamic random access memory*

VIBE – *Visual Background Extrator*

RESUMO

O trabalho em questão visa realizar uma comparação entre métodos de subtração de fundo, afim de servir como base para estudo na identificação de regiões navegáveis para veículos (robôs) autônomos. A abordagem proposta irá comparar algoritmos de extração de fundo que identificam possíveis objetos em movimento encontrados no ambiente. A extração de fundo é utilizada em diversos sistemas de visão computacional, pois permite que o sistema foque em um objeto em movimento e possa resolver de forma automática o que fazer com relação a tal objeto, desta forma, a extração de fundo é uma peça crucial para a navegação autônoma de veículos.

Palavras-chave: Extração de fundo, Processamento Digital de Imagens (PDI), Objetos em movimento, Percepção de ambiente, Veículos autônomos, Visão Computacional.

ABSTRACT

This work aims to conduct a comparison of background subtraction methods in order to serve as a base for study in the identification of navigable regions for autonomous vehicles (robots). The proposed approach will compare algorithms for background subtraction for identifying possible moving objects found in the environment. The background extraction is used in many computer vision systems, as it allows the system to focus on a moving object and can automatically resolve what to do about that object, thereby background subtraction is a crucial piece to autonomous navigation of vehicles.

Key-Words: Extracting background, Digital Image Processing (DIP), Objects in motion, Autonomous vehicles, Computer Vision.

1. INTRODUÇÃO

SegundoWolf et. al. (2009), atualmente as atenções se voltam para robôs móveis, capazes de se deslocar em qualquer ambiente e nas mais variadas condições de terreno. Os Robôs Móveis Autônomos (RMAs) possuem como características a capacidade de locomoção e operação de modo semiautônomo ou completamente autônomo. Neste ponto, um maior nível de autonomia é alcançado à medida que o robô passa a integrar outros aspectos como capacidade de percepção (leitura do ambiente), capacidade de agir (deslocamento no ambiente), robustez e inteligência (capacidade de interpretar o ambiente e executar tarefas de acordo com as situações vivenciadas). O uso dessas técnicas é chamado de "Controle Robótico Inteligente" que tem por característica tentar modelar o comportamento emergente do ser vivo a ser modelado.

Ainda de acordo com Wolf et. al. (2009), o desenvolvimento de robótica móvel requer conhecimento de diversas áreas, como da Engenharia Mecânica, Engenharia Elétrica, Engenharia da Computação e das diversas áreas dentro da Computação. A Computação possui um papel muito importante como área que provê o suporte de conhecimentos e técnicas que irão permitir dotar robôs móveis de sistemas de controle mais robustos, seguros, autônomos e inteligentes.

Dentro da Computação será aplicada uma área chamada Visão Computacional, na qual proverá estrutura para tratamentos gráficos em imagem e vídeo. Para introduzir visão computacional a este trabalho, será utilizada a biblioteca OpenCV, que dará suporte ao método de extração de fundo, captura de vídeo e tratamento de ruídos em vídeo, assim como diversas outras funcionalidades que podem auxiliar na capacidade de percepção de um RMAs, então melhorando o nível de automação dos mesmos, sendo possível identificar objetos em movimento e realizar um tratamento de colisão.

1.1. MOTIVAÇÕES

Os veículos autônomos precisam de métodos para reconhecer quais são as regiões navegáveis e identificar possíveis objetos de colisão, sendo assim, este reconhecimento deve ser feito em tempo real, portanto, para que isso seja possível, é ideal que este tratamento seja feito em hardware ao invés de software. Por esse motivo é necessário um melhor estudo sobre as formas de extração de fundo, na qual irão auxiliar na detecção de objetos em movimento,

para que futuramente possam ser implementadas em hardware funções mais eficientes de extração de fundo e reconhecimento de objetos em movimento.

1.2. OBJETIVOS

O objetivo geral do projeto é realizar a comparação entre o algoritmo de extração de fundo EGMM (*Enhanced Gaussian Mixture Model*), desenvolvido por Zivkovic et.al (2006), com o algoritmo utilizado no método Vibe (*Visual Background Extractor*), desenvolvido por Barnich e Droogenbroeck (2009), para que possa nortear futuras implementações e auxiliar a detecção de objetos em movimento e regiões navegáveis para RMAs.

1.3. ORGANIZAÇÃO DO TRABALHO

Este trabalho está organizado em cinco capítulos. No Capítulo 2 é realizada uma revisão bibliográfica sobre visão computacional, computação gráfica, processamento de imagens e algoritmos de extração de fundo, bem como sobre os tópicos teóricos e matemáticos. O Capítulo 3 contém breve síntese de um trabalho correlato ao trabalho proposto. No Capítulo 4 estão descritos os resultados obtidos e a discussão sobre os mesmos. E, no Capítulo 5 apresentam-se as conclusões do trabalho e as propostas de trabalhos futuros.

2. VISÃO COMPUTACIONAL

Para entender o que são sistemas baseados em visão, primeiro é necessário compreender a definição de visão computacional, Marr (1982) afirma que “visão é o processo que produz, a partir de imagens do mundo externo, uma descrição que é útil ao usuário e que não é repleta de informações irrelevantes”.

Trivedi e Rosenfeld (1989) também dão uma definição tão conhecida quanto Marr, na qual afirmam que “visão computacional é a disciplina que investiga as questões computacionais e algorítmicas associadas à aquisição, ao processamento e à compreensão de imagens”.

A seguir serão relatados elementos que fazem parte de sistemas de visão computacional, assim como conceitos relacionados à visão computacional.

2.1. Elementos De Sistemas De Visão Computacional

A maioria dos sistemas de visão computacional possuem os seguintes elementos, que serão citados nos próximos tópicos da monografia: aquisição de imagem, armazenamento, processamento e interfaces de comunicação, na qual grande parte das variações ocorre em sua localização quanto à estrutura. Câmeras inteligentes possuem todos os elementos em um único dispositivo, enquanto os sistemas de visão baseados em computadores possuem os elementos de aquisição separados dos demais elementos.

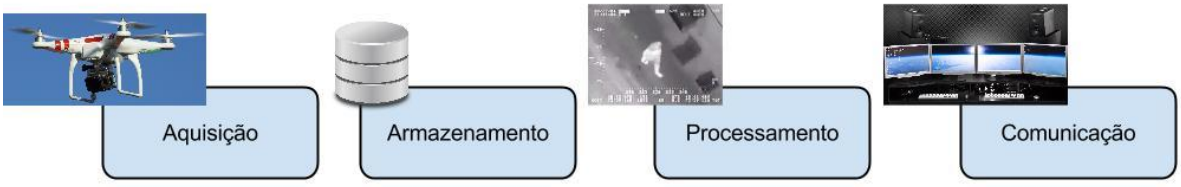


Figura 1 - Elementos da visão computacional

2.1.1. AQUISIÇÃO DA IMAGEM

A aquisição da imagem é feita por um dispositivo sensível à banda do espectro eletromagnético, na qual converte a energia captada em um sinal elétrico e este dispositivo pode ter saída na forma analógica, ou também, utilizando um converso, pode ter saída já em formato digital.

Segundo Ogê e Hugo (1999), “o primeiro passo na conversão de uma cena real

tridimensional em uma imagem eletrônica é a redução de dimensionalidade“.

> [...] o dispositivo de aquisição de imagens mais utilizado atualmente é a câmera CCD (*Charge Coupled Device*). Ela consiste de uma matriz de células semicondutoras fotossensíveis, que atuam como capacitores, armazenando carga elétrica proporcional à energia luminosa incidente. O sinal elétrico produzido é condicionado por circuitos eletrônicos especializados, produzindo à saída um Sinal Composto de Vídeo (SCV) analógico e monocromático. (OGÊ E HUGO, 1999)

Ainda segundo Ogê e Hugo (1999), para a aquisição de imagens coloridas utilizando CCDs é necessário um conjunto de prismas e filtros de cor encarregados de decompor a imagem colorida em suas componentes R, G e B, cada qual capturada por um CCD independente. Os sinais elétricos capturados, correspondentes a cada componente, são combinados posteriormente conforme o padrão de cor utilizado, como por exemplo NTSC (*National Television Standards Committee*) ou PAL (*Phase Alternating Line*).

Uma câmera CCD monocromática simples consiste basicamente de um conjunto de lentes que focalizarão a imagem sobre a área fotossensível do CCD, o sensor CCD e seus circuitos complementares.

2.1.2. ARMAZENAMENTO

Consiste em armazenar as imagens para que possam ser processadas posteriormente. Como as imagens já foram geradas em formato digital, pelo dispositivo de aquisição, elas podem ser armazenadas em memórias da mesma forma como qualquer outro tipo de dado digital. Os meios mais comuns de armazenamento atualmente são memórias tipo SDRAM, memória Flash, discos rígidos, mídias ópticas entre outros.

2.1.3. PROCESSAMENTO

O processamento de imagens trabalha com operações expressas em forma de algoritmos. Na maior parte das vezes pode ser implementado em software e informação resultante do processamento pode ser uma imagem modificada, uma saída digital binária, ou outros valores quaisquer programados no software ou hardware.

A informação resultante do processamento pode ser uma imagem modificada, uma saída digital binária, ou outros valores quaisquer.

2.1.4. INTERFACES DE COMUNICAÇÃO

Segundo Schneider (2007), as interfaces de comunicação em sistemas de visão computacional envolvem a troca de informações entre dispositivos, podendo ser transmissão das imagens do dispositivo de aquisição para o de processamento, do dispositivo de processamento para dispositivos de armazenamento ou exibição das imagens, ou do dispositivo de processamento para outros dispositivos externos ao sistema.

Imagens digitais normalmente precisam de uma grande capacidade de transmissão em seus meios, pois é transmitida uma grande quantidade de informações. Por mais que existam várias técnicas de compactação, a grande quantidade de dados a serem transmitidos tende a crescer cada vez mais, pois os sensores tendem cada vez mais a possuir uma resolução maior.

Ainda segundo Schneider (2007), as imagens são exibidas por dispositivos que convertem sinais elétricos em imagens, que podem ser aparelhos de televisão ou monitores, monocromáticos ou coloridos, do tipo CRT, LCD, plasma ou LED. Ainda pode-se incluir neste grupo os projetores de imagens e os dispositivos de impressão.

2.2. MODELOS DE IMAGENS

A representação de imagens digitais, para que possam ser utilizadas computacionalmente requerem modelos matemáticos adequados.

O termo imagem refere-se a função de intensidade luminosa bidimensional, denotada por f(x,y), em que o valor ou amplitude de f nas coordenadas espaciais (x, y) dá a intensidade (ou brilho) da imagem naquele ponto (GONZALES e WOODS, 1992).

Basicamente a natureza de f(x,y) pode ser expressa por dois componentes: a intensidade da luz incidindo sobre cena observada e a intensidade de luz refletida pelos objetos na cena. A isso se da o nome de iluminância e refletância, e são representadas da seguinte forma: f(x,y) = i(x, y).r(x, y) (1), na qual: 0 < i(x, y) < ¥ e 0 < r(x, y) < 1.

Os limites acima indicam que a refletância é limitada entre 0 (absorção total) e 1 (reflexão total). A natureza da iluminância é determinada pela fonte da luz e da refletância pelas características físicas dos objetos.

Pixel é o nome que se da à representação matricial de f(x,y). O valor de um pixel é

definido de acordo com um sistema de cores ou modelo de cores, no caso de uma imagem monocromática o pixel representa um nível de cinza, em uma imagem colorida ele representa uma cor, que é definida por um modelo de cor. No próximo item, será descrito como as cores são geradas, quais suas funções e alguns modelos de cores mais utilizados.

2.3. FUNDAMENTOS DE CORES

A cor exerce as ações de impressionar, expressar, e construir. A cor é recebida por nossa retina e passada para o cérebro como uma emoção, portanto ela pode ter significado próprio, possuir valor de símbolo, construir uma linguagem e transmitir uma ideia ou texto.

A cor, na computação gráfica, mostra varias utilidades, como melhora na legibilidade da informação, gerar imagens realistas, indicar mecanismos de segurança, focar a atenção do observador, passar emoções, dentre outras coisas. Mais a diante, serão descritos métodos de segmentação de imagens e estes métodos podem utilizar modelos de cores para realizar a segmentação de imagens coloridas.

O estudo sobre a origem das cores originou-se em 1666, quando, Isaac Newton descobriu que um feixe de luz solar ao atravessar um prisma de vidro, na saída é obtido um feixe de luz com um espectro contínuo de cores variando do violeta ao vermelho.

Na figura 2 podemos observar o espectro de cores, que pode ser dividido em seis regiões principais: violeta, azul, verde, amarelo, laranja e vermelho. As cores misturam-se gradativamente, de forma que não existe um limite abrupto entre elas.

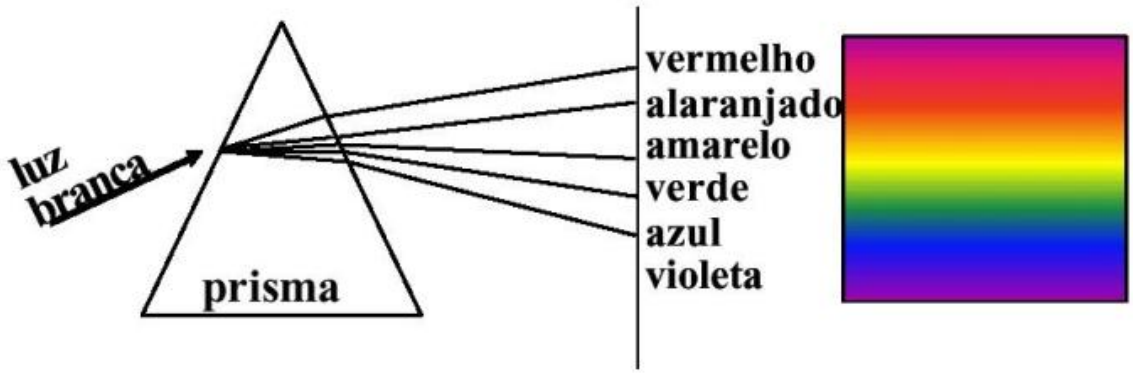


Figura 2 - Prisma de decomposição da luz (GONZALES E WOODS, 1992)

Pode-se ver o espectro eletromagnético e o posicionamento da luz visível dentro do espectro na figura 3. O comprimento de onda, nesta figura, é representado pelo eixo l (m), e o eixo f (Hertz) representa a frequência das cores. A tabela 1 apresenta o comprimento de onda

das cores do espectro visível classificado por cor.

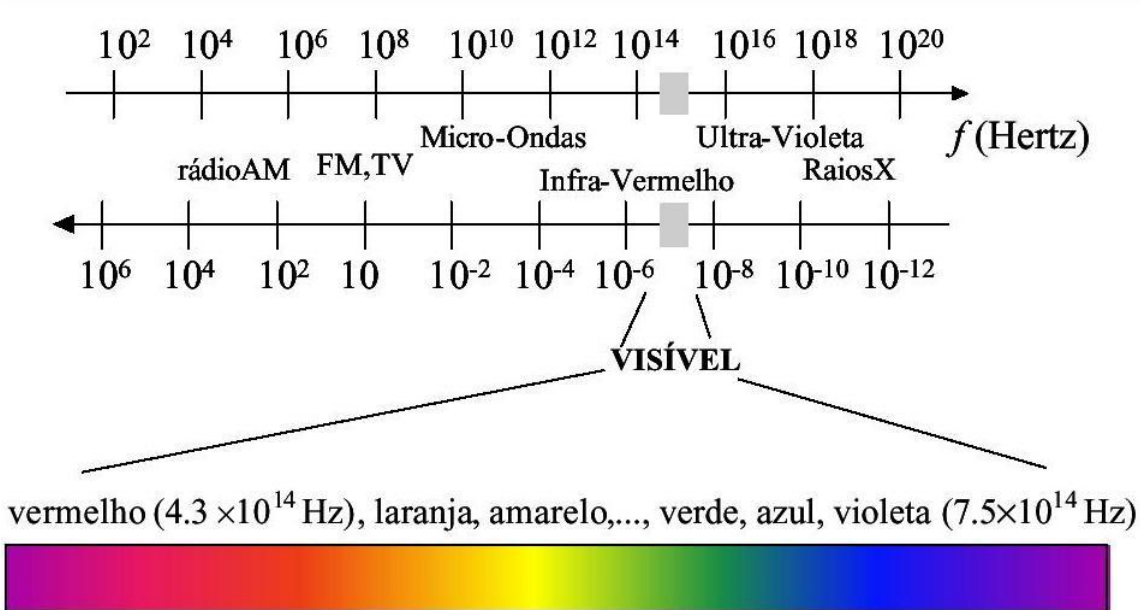


Figura 3 - Espectro eletromagnético (GONZALES E WOODS, 1992)

Tabela 1 - Comprimento de onda (GATTASS, 2007) Cor l (nm)

Violeta	380-440
Azul	440-490
Verde	490-565
Amarelo	565-590
Laranja	590-630
Vermelho	630-780

Para o propósito de padronização, a Comissão Internacional sobre iluminação (CIE), designou em 1931 os comprimentos de onda das três cores primárias: azul=438,5nm, verde=546,1nm e vermelho=700nm.

A partir das cores primárias podem-se obter as cores secundárias: magenta (vermelho e azul), ciano (verde e azul) e amarelo (vermelho e verde), como ilustra a figura 4.

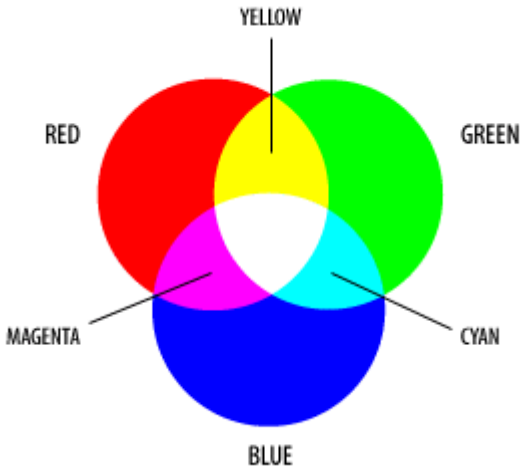


Figura 4 - Mistura de Cores Primárias Aditivas (GATTASS, 2007)

Para um melhor entendimento sobre cores, é necessário caracterizá-las. Portanto as cores possuem os seguintes atributos:

a) Luminosidade ou Intensidade (nível de clareza ou escuridão): é o atributo da percepção visual onde uma área parece emitir mais ou menos luz.

b) Tonalidade ou Matiz: é a propriedade onde uma cor é percebida como pura, seja ela verde, azul, vermelho, amarelo, etc. Os tons brancos, pretos e cinza puros não possuem tonalidade e saturação.

c) Saturação: é a propriedade que indica o grau de pureza da cor, quanto maior o grau, mais vívida será a cor, ou seja, saturada. A saturação indica o grau de como a cor está diluída pelo branco.

GROB (1989) define as cores em sinais de vídeo como:

Luminância: mostra como a cor surgirá numa reprodução monocromática. Indica a quantidade de luz que é percebida pelos olhos.

Crominância: é o termo usado para combinar o matiz de cor e a saturação. Inclui assim as informações de cores sem considerar a luminância. Este termo também pode ser chamado de croma ou também cromaticidade.

Contraste: é o termo usado para definir a diferença de intensidade entre as partes claras e escuras da imagem.

Para este trabalho, o estudo de cores é importante, pois os métodos de extração de fundo utilizam os atributos citados acima para realizar uma subtração de fundo mais eficiente, levando em consideração as sombras geradas por objetos de acordo com a luminosidade

incidente, a tonalidade de cor dos objetos entre outros aspectos que podem ser encontrados em uma imagem ou vídeo.

2.3.1. MODELOS DE CORES

Existem formatos padrões que especificam as cores e estes são chamados de modelos de cores. Entre os modelos de cores mais comuns estão: RGB (*red, green, blue*), CMY (*ciano, magenta, yellow*), YCrCb, HSI (*hue, saturation, intensity*).

2.3.1.1. RGB

É composto por três valores, que representam cada qual uma cor, vermelho (*red*), verde (*green*) e azul (*blue*). Desta forma, qualquer cor pode ser definida em função de suas três cores primárias. É o modelo de cor mais utilizado para monitores de vídeo e câmeras. Na figura 5 é mostrado o espaço de cores RGB, definido como um quadrado, cujos vértices definem as cores primárias e secundárias.

Considerando que intensidade é o nível de luz (claro ou escuro) em um pixel, no modelo RGB essa intensidade então será dada pelo módulo da distância da origem ao pixel considerado. Podemos observar isso na figura 5, pois o menor módulo representa a cor preta e o maior módulo representa a cor branca. Portanto, o matiz ou a cor será definido pela posição angular do vetor do pixel em relação aos eixos das cores fundamentais Vermelho, Verde e Azul.

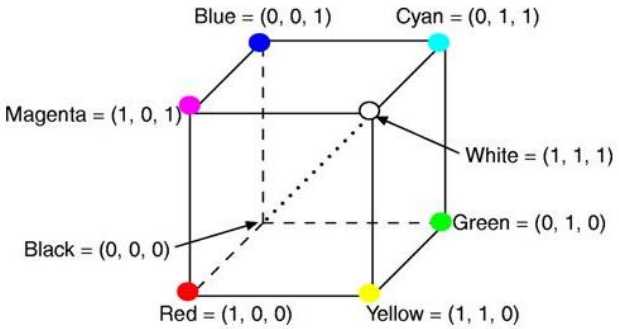


Figura 5 - Modelo RGB (GATTASS, 2007)

2.3.1.2. RGB NORMALIZADO

Para se obter o RGB normalizado, divide-se os componentes R, G e B pela soma total de seus valores, conforme observamos na figura 6:

$$r = \frac{R}{R+G+B}$$

$$g = \frac{G}{R+G+B}$$

$$b = \frac{B}{R+G+B}$$

Figura 6 - Expressões do RGB normalizado

2.3.1.3. HSI

HSI é um modelo de cor que a representa em termos de matiz (H), saturação (S) e intensidade (I). Existem também autores que nomeiam este modelo como HSL (*Hue, Saturation and Lightness*). Matiz conforme já citado anteriormente, descreve a cor pura, a saturação descreve a diluição da cor pura com o branco e intensidade dá a luminosidade da cor. Na figura 7 pode-se ver uma representação do modelo.

Como podemos observar na figura 7, o modelo HSI tem o formato de dois cones unidos pela base, o valor do Matiz (Hue) define a posição angular de uma cor, podendo estar entre 0° e 360º em relação à linha de referência posicionada entre as bases dos cones. O valor da saturação (Saturation) é dado pelo comprimento do vetor até o eixo que une as duas pontas dos cones. E o valor da intensidade (Lightness) é dado como a projeção do vetor sobre este eixo.

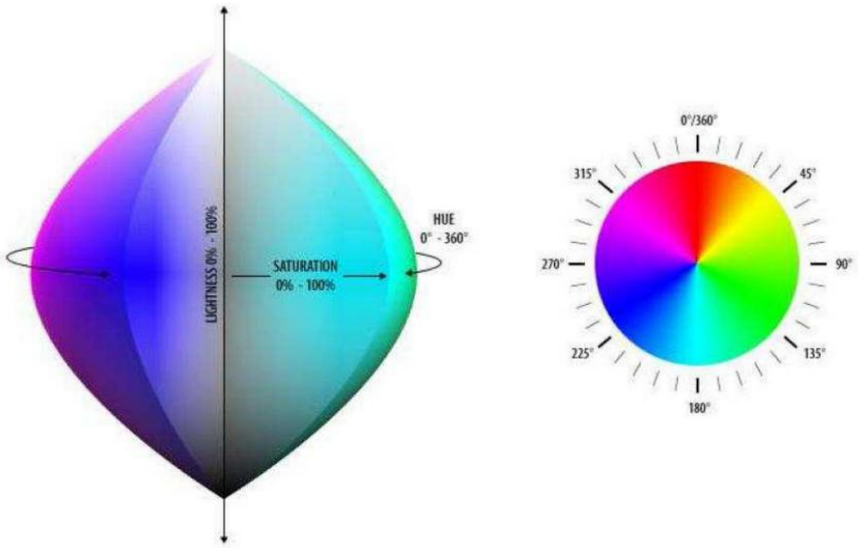


Figura 7 - Modelo HSI (GATTASS, 2007)

2.3.2. FILTRO BAYER

O padrão do filtro de Bayer é composto por 50% de foto sensores verde, 25% de vermelho e 25% de azul, como pode ser observado na figura 8. A cor verde é mais utilizada, pois o olho humano possui maior poder de resolução com esta cor, portanto o filtro tende a imitar o olho humano (WIKIPEDIA, 2007).

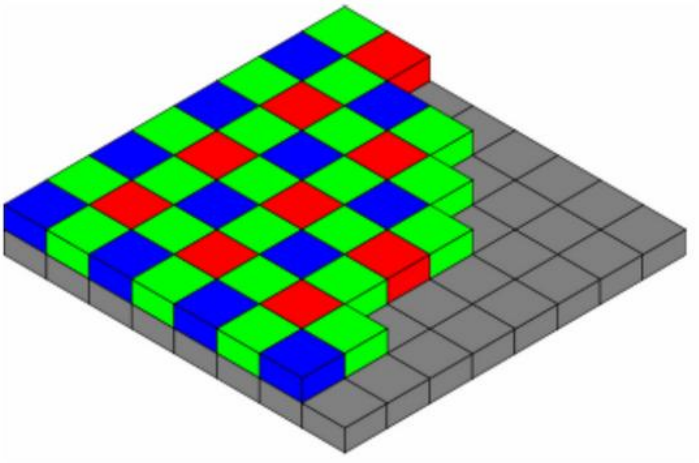


Figura 8 - Mosaico do filtro Bayer (WIKIPEDIA, 2007)

2.4. PADRÕES DE VÍDEO

Para entender um pouco melhor como são formados e como são percorridos os vídeos, deve-se saber que todos os padrões de vídeo dividem uma imagem em linhas horizontais,

chamadas de raster, iniciando com a linha na parte superior e terminando na parte inferior. As linhas são percorridas da esquerda para a direita em forma de pixels. Cada varredura completa do vídeo forma um frame (campo, quadro ou field).

2.5. TÉCNICAS DE SEGMENTAÇÃO

Neste item será realizado um estudo sobre os métodos e algoritmos utilizados para a identificação e detecção de objetos móveis em uma sequência de imagens ou vídeo, na qual auxilia o reconhecimento de regiões navegáveis. Normalmente, estas cenas estão sob a influência de mudanças na iluminação e sujeitas à presença de sombras ou até mesmo a mudanças no próprio fundo da imagem.

Segundo Schneider (2007), existem diversas técnicas de segmentação, no entanto, as mais comuns fazem parte do método chamado de Subtração de Fundo. Basicamente, a subtração do fundo consiste em subtrair a imagem atual de outra usada como referência, ou seja, uma imagem contendo somente o fundo da cena e construída a partir de uma sequência de imagens com variações de iluminação durante um período de treinamento. A subtração de fundo é utilizada para capturar apenas objetos não estáticos (em movimento) e novos objetos em uma cena, que é um passo fundamental, mas não único para detectar regiões navegáveis.

O segredo para a subtração de fundo está na modelagem do *background* (fundo). Cheung e Kamath (2004) classificaram as técnicas de modelagem de fundo, em recursivas e não recursivas.

2.5.1. TÉCNICAS NÃO RECURSIVAS

De acordo com Schneider (2007), as técnicas não recursivas consistem em armazenar uma quantidade de imagens em um buffer e estimar o Fundo baseado na variação de cada pixel por tempo. Dentre elas encontramos as seguintes técnicas:

Diferenciação de Frame: É a técnica mais simples, que utiliza o frame do instante t-1 como modelo de fundo para o frame do instante t.

Filtro da Média: O Fundo é estimado baseado na média de cada pixel dentro do buffer. Ou seja, é feito a média de todas as imagens amostradas.

Filtro linear preditivo: Toyama et al. (1999) estimaram o fundo aplicando um filtro linear preditivo em cada pixel dentro do buffer.

2.5.2. TÉCNICAS RECURSIVAS

As técnicas recursivas não mantêm um buffer para estimar o fundo. Ao invés disso atualizam recursivamente o modelo de fundo através de cada frame de entrada. Comparado ás técnicas não-recursivas, as recursivas não requerem muita capacidade de memória, pois o modelo de fundo é atualizado a cada cena, porém um erro na modelagem, pode permanecer por um longo período de tempo alterando os resultados.

Dentro destas técnicas tem-se:

Aproximação do filtro da média: Técnica apresentada por McFarlane e Schopfeld (1995) que apresentaram um filtro recursivo simples para estimar a média. Através desta técnica a estimação da média é incrementada em um se o pixel de entrada é maior que o estimado, ou decrementado de um, se for menor que o estimado. Com isso somente uma imagem de referência precisa ser armazenada na memória.

Filtro de Kalman (1960): É um conjunto de equações matemáticas que constitui um processo recursivo eficiente de estimação, uma vez que o erro quadrático é minimizado. Através da observação de uma variável observável, outra variável, não observável pode ser estimada eficientemente.

Podem ser estimados os estados passados, o estado presente e mesmo previstos os estados futuros. O filtro de Kalman é um procedimento aplicável quando os modelos estão escritos sob a forma espaço-estado. Além disso, o filtro de Kalman permite a estimação dos parâmetros desconhecidos do modelo através da maximização da verossimilhança via decomposição do erro de previsão.

Mistura de Gaussianas (MoG): Diferente do Filtro de Kalman que rastreia a evolução de uma única Gaussiana, o método MoG rasteia múltiplas distribuições Gaussianas simultaneamente. MoG desfrutou tremenda popularidade desde que foi proposto pela primeira vez para modelagem de fundo por Friedman e Russel (1997). Semelhante aos modelos não paramétricos descritos no item 2.5.1, MoG mantém uma função de densidade para cada pixel.

Assim, é capaz de lidar com distribuições de fundo multimodais. Por outro lado, desde que MoG é paramétrico, os parâmetros modelo podem ser adaptativamente atualizados sem manter um grande buffer de frames de vídeo, assim não é necessário uma grande quantidade de memória para armazenar o modelo de fundo.

2.6. ALGORITMOS E TÉCNICAS DE EXTRAÇÃO DE FUNDO

2.6.1. ALGORITMO DE HORPRASERT

O algoritmo descrito a seguir foi desenvolvido por Horprasert, et al. (1999). Ele basicamente descreve uma extração de fundo na qual é realizada uma modelagem do fundo para ser usada como base e após isso extrair o objeto baseado na diferença entre as cores medidas através de distorções de brilho (*brightness*) e cromaticidade (*chromaticity*).

Na figura 9 é ilustrado o modelo computacional de cor proposto neste algoritmo dentro do espaço tridimensional RGB, modelo de cor que pode-se observar no item de modelos de cores. Levando em consideração o pixel i ilustrado, define-se $E_i = [E_R(i), E_G(i), E_B(i)]$ como sendo o vetor que representa a cor esperada do pixel (imagem de referência). A linha OE_i, que passa pela origem e através do ponto E_i, é chamada de linha de cor. O ponto representado por $I_i = [I_R(i), I_G(i), I_B(i)]$ é o vetor da cor RGB na imagem atual, da qual se quer subtrair o fundo. Neste modelo basicamente se quer medir a distorção existente entre I_i e E_i. Isto é possível medindo a distorção de brilho e de cor, no valor RGB do pixel.

A distorção de brilho (α_i) é o valor escalar que projeta a cor atual I_i na linha de cor de modo que ela se aproxime o máximo possível do valor de referência E_i.

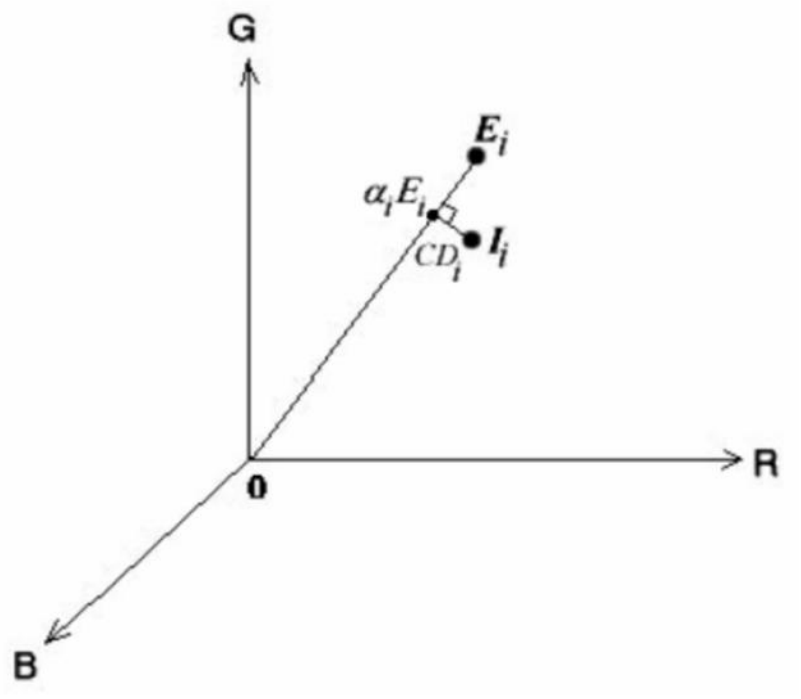


Figura 9 - Modelo de cor Horprasert

Truyenque (2005, p.30) explica bem o conceito de distorção de brilho e cor:

> A distorção de brilho é um valor que fica dentro da linha de cor, ou seja, é um valor escalar que posiciona a cor atual ao longo da linha de cor. Essa distorção indica que houve apenas uma mudança no brilho, ou seja, na iluminação da imagem e não uma mudança significativa de cor. Isto pode ser usado para identificar sombras baseando-se no modelo de referência, pois as sombras em uma mesma cor apresentam brilho menor que a imagem original. Já a distorção de cor, indica o grau de afastamento de uma cor de sua linha de cor, pois assim é possível identificar se o pixel que está sendo analisado faz parte ou não do objeto do modelo de referência. (TRUYENQUE, 2005)

No algoritmo de Horprasert cada pixel é modelado como sendo uma 4-tupla (I_i, s_i, a_i, b_i), onde E_i é o valor esperado da cor, s_i é o desvio padrão, a_i é a distorção média de brilho e b_i é a distorção média de cor. Neste algoritmo, E_i é considerado como sendo a imagem média das imagens de treino e s_i como sendo o desvio médio padrão presente neste conjunto de imagens.

Segundo Hoprasert, et al. (1999), câmeras tipicamente têm diferentes sensibilidades para diferentes cores. Assim, para manter os pesos de balaços nas três bandas de cores (R, G, B), os valores dos pixels precisam ser normalizados por pesos e isso é feito através do desvio padrão. Desse modo as respostas dos sensores RGB são similares para a mesma variação de intensidade de cor.

Após normalizar os valores dos pixels, estes podem ser classificados, na qual Hoprasert, et al. (1999), adotaram o seguinte critério de classificação dos pixels de uma imagem:

- Fundo original (B) se a distorção de brilho e a distorção de cor são similares aos valores correspondentes na imagem de referência.
- Fundo com baixa iluminação ou sombra (S) se a distorção de cor é similar, mas a distorção de brilho é bem menor do que aquela na imagem de referência.
- Fundo com alta iluminação (H) se existe uma distorção de cor similar, mas uma distorção de brilho maior do que aquela na imagem de referência.
- Objeto em movimento (F) se a distorção de cor é diferente aos valores

esperados no treinamento, caso em que o pixel é considerado como sendo parte do objeto em movimento.

Segundo Schneider (2007), o algoritmo de Horprasert pode ser sintetizado assim:

Processos de treino:

Passo 1: Captura de um número de imagens de treino, a partir das quais calcula-se uma imagem média e outra representando o desvio padrão.

Passo 2: Para cada uma das imagens adquiridas calcula-se as imagens da distorção de brilho.

Passo 3: Para cada uma das imagens adquiridas calcula-se as imagens da distorção de cor.

Passo 4: Calcula-se a média da distorção de brilho utilizando-se todas as imagens do passo 2.

Passo 5: Calcula-se a média da distorção de cor utilizando-se todas as imagens do passo 3.

No final do processo de treino são obtidas 4 imagens: a imagem média, a imagem do desvio padrão, a imagem média da distorção de brilho e a imagem média da distorção de cor.

Processos de subtração:

Passo 6: Calcula-se a imagem da distorção de brilho, utilizando agora a imagem atual.

Passo 7: Calcula-se a imagem da distorção de cor, utilizando agora a imagem atual.

Passo 8: Normaliza-se a imagem da distorção de brilho.

Passo 9: Normaliza-se a imagem da distorção de cor.

Passo 10: Efetua-se a classificação dos pixels utilizando os limiares escolhidos.

2.6.2. ALGORITMO DE CHEUNG

O algoritmo implementado por Cheung, et al. (2000), consiste basicamente em verificar o ângulo entre os pixels da imagem de referência e a imagem atual.

Segundo Cheung, et al. (2000), a imagem de referência é obtida através da média das imagens de aprendizado. Considerando que no espaço RGB uma variação de ângulo entre os vetores formados por dois pixels define uma variação de cor, este algoritmo permite definir um limiar de variação de ângulo aceitável, ou mais precisamente a variação de cor aceitável.

O algoritmo funciona da seguinte forma: Se existe uma variação grande de cor, em módulo, acima de um limiar T_U, o algoritmo considera que a mudança neste pixel é

consideravelmente alta, e assim considera o pixel como sendo parte do objeto. Se não existe uma mudança considerável na cor, então pode ser que os valores de cor nos pixels não tenham sofrido grandes mudanças. Deste modo, se a variação de cor é menor do que um outro limiar T_L o pixel é considerado como sendo parte do fundo. Já para os pixels que apresentam variações de cor acima do T_L e abaixo do T_U o ângulo deve ser medido. Para estes pixels, se o angulo for maior que um limiar T_C, o pixel é considerado como sendo parte do objeto.

Segundo Schneider (2007), o algoritmo de Cheung pode ser sintetizado assim:

Passo 1: Calcula-se a imagem média entre as imagens durante o aprendizado. Esta imagem será a imagem de referência.

Passo 2: Para cada pixel, calcula-se a diferença de cor da referência com a imagem atual.

Passo 3: Verifica-se se o pixel faz parte do objeto ou não de acordo com o T_L.

Passo 4: Calcula-se o ângulo entre os pixels.

Passo 5: Verifica-se novamente se o pixel faz parte do objeto ou não, porém agora de acordo com o T_C.

2.6.3. ALGORITMO DE KIM OU CODEBOOK

O algoritmo de Kim, et al. (2005) tem como principais características uma modelagem adaptativa e compacta que pode capturar movimentos estruturais no fundo da imagem durante um longo período de tempo e com limitações de memória, treinamento sem restrição objetos móveis sobre o fundo, já no período inicial do treinamento, além de haver uma modelagem e treinamento em níveis, permitindo haver múltiplos níveis de fundo.

Este algoritmo constrói para cada pixel um Codebook (Tabela de codewords), formado por um ou mais codewoords, que são conjuntos de dados representativos para identificar os pixels do fundo.

Seja X uma estrutura de treinamento consistindo de N vetores RGB: $X = \{ x_1, x_2, x_3, ... x_N\}$. Seja $C = \{c_1, c_2, c_3, \ldots c_L\}$ o *codebook* de um pixel constituído por L *codewords* . Cada Pixel possui um *codebook* diferente baseado na varição das suas amostras. Cada *codeword* c_i, $i = 1 \ldots L$ consiste de um vetor $v\ i = (R_i, G_i, B_i)$ e uma 6-tupla $aux\ I = (I_{imin}, I_{imax}, f_i, \lambda_i, p_i, q_i)$, onde:

(I_{imin}), (I_{imax}): Brilho mínimo e máximo de cada pixel.

(f_i): Freqüência com o qual o *codeword* aparece.

(λ_i): Maior intervalo no período de treinamento em que o *codeword* não aparece.

(p_i), (q_i): Primeiro e ultimo tempo de acesso em que o *codeword* ocorre.

Durante o período de treinamento, cada valor x_t amostrado no tempo t, é comparado com o *codebook* atual para determinar com qual *codeword* c_m ele se equivale. (m é o índice do *codeword* equivalente). Para determinar qual *codeword* que melhor equivale é adotada uma medida de distorção de cor, abaixo, na figura 10, pode-se visualizar o modelo de cor utilizado por Kim et al. (2005).

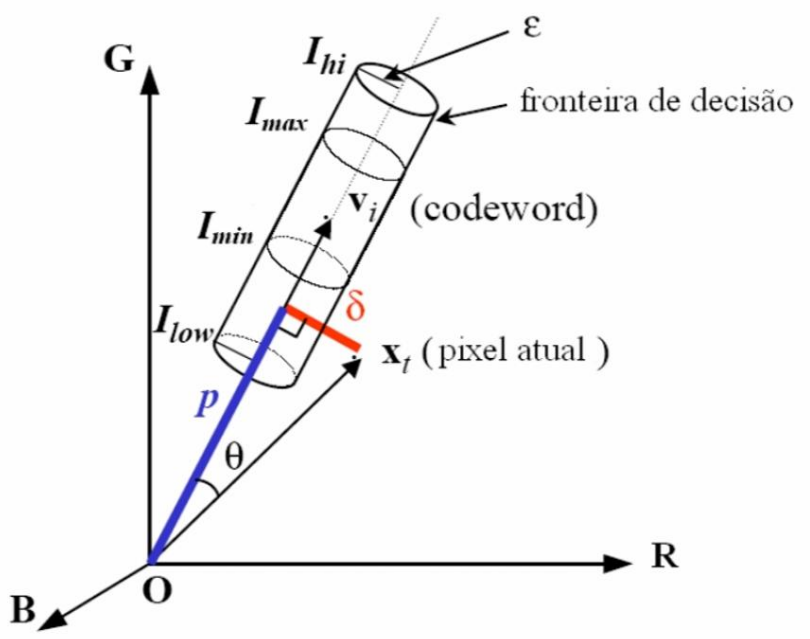


Figura 10 - Modelo de cor Codebook

2.6.4. VIBE

O algoritmo de Vibe (*Visual Background Extractor*), Barnich e Droogenbroeck (2009), é um poderoso método para extração fundo que inova em termos de precisão e reduz a carga computacional, na qual, a principal novidade prende-se com a utilização de uma política aleatória para selecionar valores para construção de uma estimativa baseada em amostras do fundo.

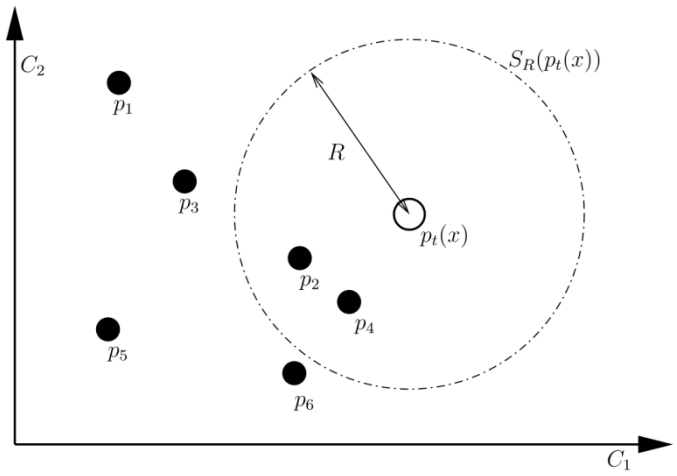


Figura 11 - Ao classificar pt (x), pode-se contar o número de amostras contidas na esfera de raio R em torno pt (x).

Pt(x) onde t significa tempo e x o valor do pixel, após ser construído o algoritmo classifica um valor de pixel como sendo um plano de fundo ou primeiro plano de fundo, dependendo da maneira que ele está estimado no PDF (*Probability Density Function*). Esta abordagem tem como principal desvantagem a avaliação do formato do PDF por ser um processo global, as camadas de fora armazenadas no pixel modelo poderá alterar a forma do PDF apesar de poder ter os valores no espaço sendo policromados distantes de Pt(x).

Ainda de acordo com Barnich e Droogenbroeck (2009), o Vibe não impõe no espaço policromático um valor que limita a vizinhança local, sendo assim, utiliza um conjunto de valores da amostra como sendo o pixel modelo. Compara-se o valor Pt(x) com os valores mais próximos dele, e é definida assim, uma esfera [$S_R(Pt(x))$], esta terá raio (R) e centro em Pt(x), conforme observa-se na figura 11.

O pixel será fundo se a cardinalidade da intersecção entre o seu domínio e o conjunto de amostras encontrar-se acima de determinado limite $\#_{min}$ para #{$S_R(Pt(x))\cap${p1, p2, ..., Pn}}. Não sendo necessária a adaptação dos dois parâmetros (Raio e Cardinalidade Mínima) na subtração de fundo para realizar a localização dos pixels da imagem.

O autor de Vibe ainda descreve que o modelo deve ser atualizado constantemente com intuito de lidar com os novos objetos que aparecem e de conseguir resultados mais precisos com o passar do tempo. Com ele x_t pode ser comparado diretamente com as amostras, por tal motivo estas devem ser mantidas no modelo pelo tempo que for de grande relevância.

Os métodos de subtração mais antigos usam modelos com política de Entrada e Saída para realizar sua atualização. Para tratar os eventos das cenas de fundo, alguns autores

escolhem por colocar uma grande quantidade de amostras de pixels, enquanto, outros incorporam dois submodelos temporais para realizar o tratamento correto das mudanças rápidas e lentas.

Enquanto no Vibe, ocorre o aumento da relevância da estimativa e permite o uso de menos amostras, quando isso ocorre realizando a escolha aleatória dos pixels da amostra que serão atualizadas ao realizar atualização de um pixel modelo, quando a amostra é descartada o novo valor substituirá o valor descartado.

Na matemática, existe a probabilidade que a amostra do pixel possua um tempo t_0 e também será encontrado o próximo tempo t_1 é ((n-1)/n)^(t_1- t_0), portanto define-se que $P(t_0, t_1) = e^{(-\ln(n/n-1)(t_1-t_0))}$.

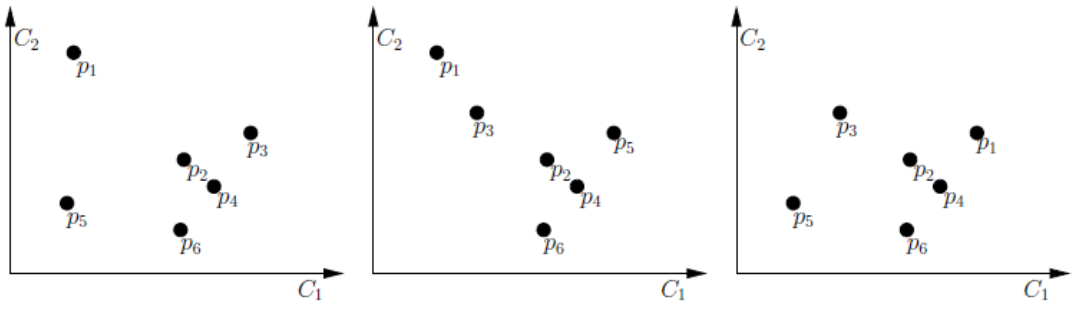


Figura 12 – Três modelos igualmente possíveis após a atualização do modelo apresentado na Figura 1.

Para garantir a coerência espacial de todo o modelo de imagem e lidar com situações práticas, como pequenos movimentos de câmera, são adotadas técnicas na qual são escolhidos valores aleatoriamente e acontece atualização do modelo pixel na vizinhança do pixel atual.

Denota-se a vizinhança espacial de um pixel x e seu valor como NG(x) e P(x), respectivamente. Supondo que para atualizar o conjunto de amostras de x por meio da inserção P(x), P(x) será usado para atualizar o conjunto de amostras da vizinhança do pixel (NG(x)), que será escolhido aleatoriamente.

Por contar muitas amostras, podem acabar, inserindo, aleatoriamente informações irrelevantes, mas isso não irá afetar a precisão do modelo, pelo fato dessa inserção ser bloqueada, por conta da necessidade de corresponder com um valor observado antes de essas informações serem propagadas. Essa abordagem de consistência espacial apresenta semelhanças com outros modelos, mas se as restrições não forem cumpridas em alguns casos como de alta iluminação, realiza por fora outras técnicas.

Mesmo que o algoritmo passe pela mesma imagem, não acontecerá de ter o mesmo

resultado, por conta da estratégia utilizada de um processo aleatório.

2.7. ESTUDO DO OPENCV

O OpenCV é uma biblioteca de visão computacional de código aberto, que inclui centenas de algoritmo. O OpenCV inclui diversos módulos, na qual dentre eles, podem-se citar como mais relevantes os seguintes módulos:

Core - um módulo compacto para definição de estruturas de dados básicas, incluindo uma matriz multidimensional (Mat) e funções básicas usadas por todos os outros módulos.

Imgproc - um módulo de processamento de imagem, que inclui filtragem linear e filtragem de imagens não lineares, transformações geométricas de imagem (redimensionamento, deformação de perspectiva, remapeamento genérico baseado em tabela), conversão de espaço de cores, histogramas, e assim por diante.

Video - um módulo de análise de vídeo, que inclui a estimativa de movimento, subtração de fundo, e os algoritmos de rastreamento de objeto.

Calib3d – visão múltipla básica de algoritmos geométricos, calibração de câmera única e sistema de som, objeto representam estimativa, algoritmos de correspondência estéreo, e elementos de reconstrução 3D.

Features2d - detectores de metragens salientes, descritores e descritores matchers.

Objdetect - detecção de objetos e instâncias de classes pré-definidas (por exemplo, rostos, olhos, canecas, as pessoas, os carros, e assim por diante).

Highgui - uma interface fácil de usar para captura de vídeo, imagem e codecs de vídeo, bem como recursos de interface simples para o usuário.

Gpu - algoritmos acelerados por GPU de diferentes módulos OpenCV .

Neste trabalho serão utilizadas as seguintes funções: *“GaussianBlur”, “BackgroundSubtractorMOG2”, “erode”, “dilate”, “findContours”* e *“drawContours”*, que descritas sucintamente abaixo:

A função *GaussianBlur* está presente na biblioteca “*imgproc*” e é utilizada para suavizar uma imagem, ou seja, remover ruídos desfocando a imagem.

A função *BackgroundSubtractorMOG2* está presente na biblioteca “*video*” do OpenCV, e implementa um modelo melhorado de uma mistura Gausiana adaptada para subtração de fundo. Esta função conta com diversos atributos que permitem o controle do

algoritmo, que são: *nmixtures, backgroundRatio, varThresholdGen, fVarInit, fVarMin, fVarMax, fCT, nShadowDetection, fTau*.

A função *findContours* faz parte da biblioteca "*imgproc*" e recupera contornos da imagem binária, usando o algoritmo Suzuki. Os contornos são uma ferramenta útil para a análise de forma e detecção de objetos e reconhecimento.

A função *drawContours*, que também faz parte da biblioteca "*imgproc*", desenha contorno na imagem se a espessura for maior ou igual a zero, ou preenche a área delimitada pelos contornos se a espessura for menor que zero.

A função *erode*, que também pertence a biblioteca "*imgproc*", corrói a imagem da fonte utilizando um elemento estruturante especificado que determina a forma de uma vizinhança do pixel durante o qual o mínimo é feita: DST(x,y) = min src(x+x',y+y'), tal que (x',y')≠0.

A função *dilate*, também pertencente a biblioteca "*imgproc*, dilata a imagem da fonte utilizando o elemento estruturante especificado que determina a forma de uma vizinhança do pixel ao longo do qual é feita a máxima: DST(x, y) = max src(x + x ', y + y ') , tal que (x',y')≠0.

As funções *erode* e *dilate* suportam o modo enlace. Na qual dilatação e a erosão podem ser aplicadas várias vezes (varias iterações). No caso de imagens multicanais, cada canal é processado de forma independente.

O estudo destas funções citadas é importante para entender como elas funcionam, pois posteriormente estas funções serão utilizadas em conjunto para construção do método de extração de fundo utilizado para comparação neste trabalho.

3. TRABALHO CORRELATO

Barnich e Droogenbroeck (2009), propõe uma comparação entre o Vibe e MoG utilizado pelo OpenCV, que faz uso do algoritmo de EGMM, na qual o seu trabalho será descrito a seguir:

Inicialização de Modelo do Vibe

Embora o modelo possa facilmente recuperar-se a partir de qualquer tipo de inicialização, por exemplo, escolhendo um conjunto de valores aleatórios, é conveniente, para obter uma estimativa precisa de fundo o mais rapidamente possível. Os autores do Vibe são capazes de segmentar as sequências de vídeo a partir do segundo quadro, o primeiro quadro a ser usado para inicializar o modelo. Uma vez que nenhuma informação temporal está disponível antes do segundo quadro, teriam que preencher os modelos de pixel com valores encontrados na vizinhança espacial de cada pixel. Mais precisamente, enchê-los com os valores retirados aleatoriamente em sua vizinhança no primeiro quadro. Esta estratégia provou ser um sucesso: a estimativa de fundo é válida a partir do segundo quadro. Segundo Barnich e Droogenbroeck (2009), único inconveniente é que a presença de um objeto em movimento no primeiro quadro irá introduzir um objeto fantasma que tem a desaparecer com o tempo.

Comparação com o modelo de mistura gaussiana

O EGMM (*Enhanced Gaussian Mixture Model*), que é um modelo aprimorado da mistura Gaussiana, Zivkovic et.al (2006) é representativo como um estado da arte do método paramétrico determinístico para a subtração de fundo. De acordo com Barnich e Droogenbroeck (2009), uma comparação visual dos resultados obtidos usando o EGMM e ViBe leva à conclusão de que, embora sendo diferente, ambas as técnicas de segmentação tem precisão equivalente para sequências internas. A maior taxa de falsos positivos do algoritmo EGMM é equilibrada pela maior taxa de falsos negativos do Vibe.

Câmeras externas levam diferentes observações com imagens que são muitas vezes de pior qualidade. Eles sofrem de tremido de câmera causado pelo vento, maior nível de movimento de fundo e compressão de imagem de alta para as questões de redução de largura de banda. Uma das sequências de teste foi realizada em um dia chuvoso e ventoso, com uma taxa de compressão forte. Segundo Barnich e Droogenbroeck (2009), enquanto ViBe

conseguiu manter resultados honrosos, o algoritmo EGMM claramente sofreu mais com estas condições difíceis. O ruído parece impedir o algoritmo EGMM de estimar corretamente os parâmetros de seu modelo.

Robustez contra ruídos

Para poder comparar objetivamente EGMM e vibração em ambientes ruidosos, Barnich e Droogenbroeck (2009) produziram um terreno de verdade de uma sequência ao ar livre usando uma tela azul como processo e adicionaram sal e pimenta como ruído para a sequência verdadeira terrestre, foi calculada a precisão e a retirada para vários níveis de ruído.

Eles produziram imagens ruidosas com a adição de um ruído aleatório distribuído uniformemente sobre todos os pixels. O ruído foi introduzido para produzir um PSNR variando de 16 [dB] para 51 [dB].

Os resultados mostrados na Figura 13 mostram a eficiência contra ruído do método Vibe, enquanto o método de EGMM no decorrer dos testes foi perdendo o contorno dos objetos devido ao ruído inserido. No entanto, podemos olhar com mais precisão os gráficos presentes na Figura 14. Os gráficos, segundo o autor de Vibe, exibem claramente que o mesmo resiste a importantes níveis de ruído aditivo, mesmo para PSNR tão baixo quanto 30 [dB]. Por outro lado, a precisão do algoritmo EGMM decai muito rapidamente, mesmo em pequenas quantidades de ruído adicionado.

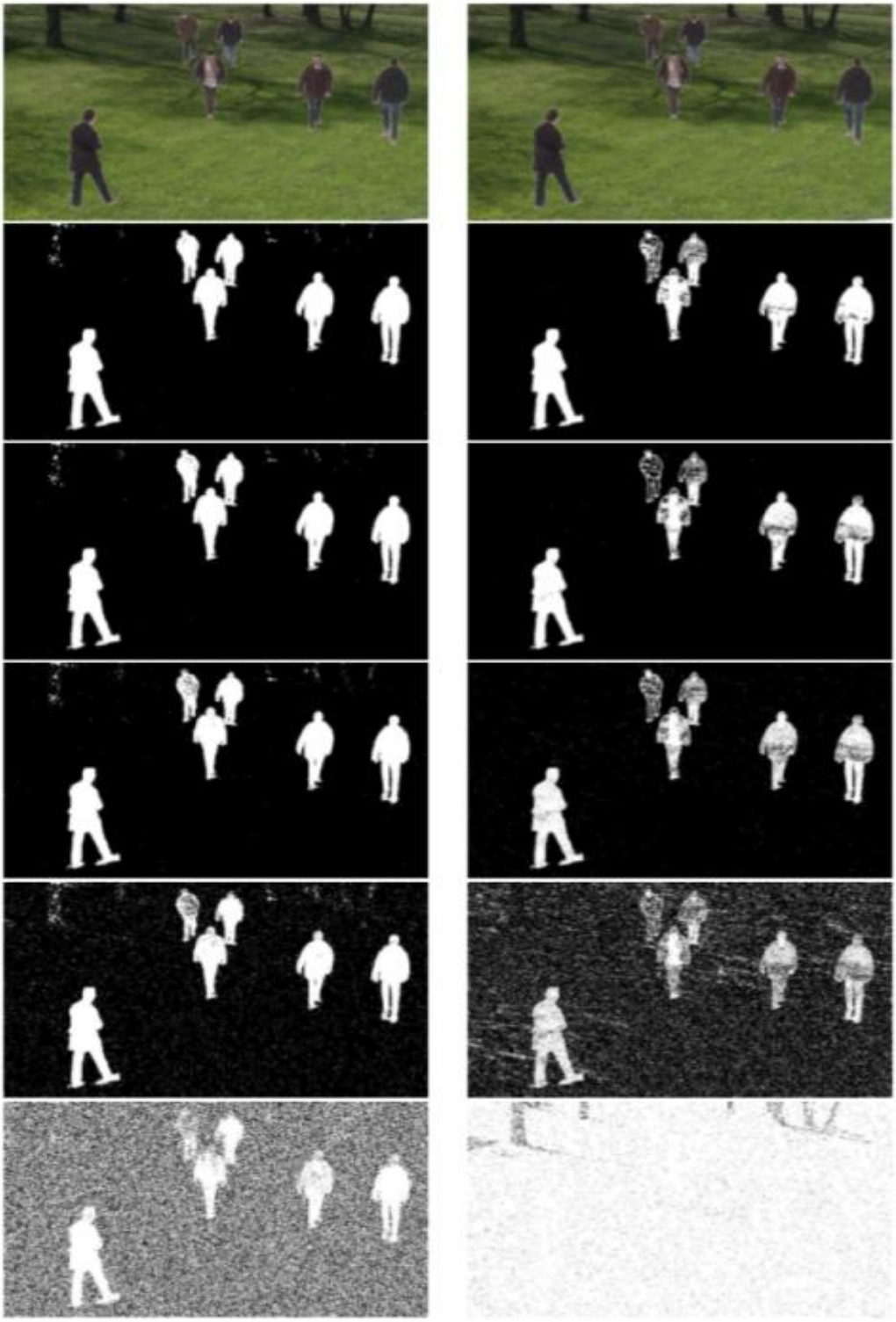


Figura 13 - Resultados da ViBe (à esquerda) e de EGMM (à direita) para PSNR de 51 [dB], 39 [dB], 30 [dB], 25 [dB], e 19 [dB].

Na Figura 14, as linhas do gráfico devem ser observadas da esquerda para a direita, lembrando que quanto maior o PSNR, menor é o ruído da imagem, então, pode-se observar no

gráfico a qualidade do Vibe (linha azul) em comparação ao EGMM (linha vermelha) com relação ao ruído nos testes, na qual inicia-se com bastante ruído e aos poucos consegue-se observar que o Vibe se destaca a medida que o ruído fica menor.

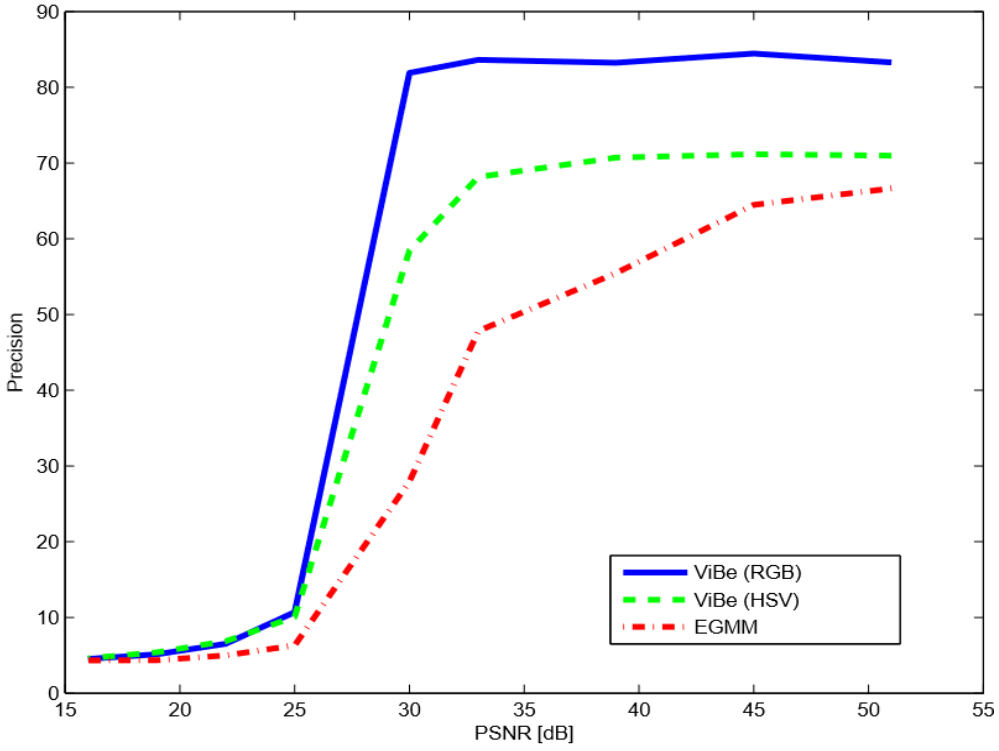


Figura 14 - Gráfico de precisão com curvas do ViBe e o EGMM para vários PSNR's

Nesta comparação apresentada por Barnich e Droogenbroeck (2009), podemos observar uma relação de qualidade, na qual ele compara a qualidade dos resultados obtidos com seu algoritmo e os resultados obtidos com o método de EGMM, no entanto ele em nenhum momento exibe resultados em termos de desempenho computacional.

No resultados descritos a seguir, além dos resultados de qualidade, poderá se constatar resultados com relação ao desempenho dos algoritmos citados nessa comparação de Barnich e Droogenbroeck (2009), e pode-se afirmar mais a frente o que foi dito nesse trabalho correlato e mais resultados que poderão ser analisados futuramente.

4. RESULTADOS E DISCUSSÕES

Primeiramente foi realizado um estudo sobre as técnicas de extração de fundo, assim como uma pesquisa sobre trabalhos correlatos que foram muito relevantes no que deveria ser analisado nesta comparação e de extrema importância para uma base solida sobre o tema pesquisado.

Para obter resultados neste trabalho, foi necessário implementar um método de extração de fundo para realizar a comparação com o método Vibe descrito nos trabalhos correlatos. Nesta implementação foram utilizadas funções do próprio OpenCV que são elas: *"GaussianBlur", "BackgroundSubtractorMOG2", "erode", "dilate", "findContours"* e *"drawContours"*, que juntamente destacam um objeto em movimento, essa implementação será nomeada como "OpenCV".

Neste trabalho será utilizado para comparação com o OpenCV, uma implementação conhecida como Vibe, na qual é implementado um método de extração de fundo utilizando funções do OpenCV para captura de vídeo, suavização de imagens e exibição de resultados, porém o método de extração de fundo não é composto pela função "*BackgroundSubtractorMOG2*" disponibilizada pela biblioteca OpenCV e sim uma construção própria de Barnich e Droogenbroeck (2009).

Mais a frente, no item 4.3, poderá se observar os resultados da comparação entre o Vibe e a a implementação utilizando o EGMM, que serão demonstrados da seguinte maneira: comparação de frames por segundo em três diferentes resoluções de vídeo (360p, 720p "HD" e 1080p "FullHD") e em duas máquinas diferentes.

4.1. ESTRUTURAS PARA TESTE

Para teste foram utilizadas duas máquinas diferentes, na qual a Máquina 1 é composta por um notebook Acer Aspire 5738, com 3Gb de memória RAM, processador Intel Core 2 Duo de 2.0Ghz em uma arquitetura de 32 bits utilizando um sistema operacional Ubuntu 14.04.

A Máquina 2 é composta por um computador Dell OptiPlex 3020, com 5Gb de memória RAM, processador Intel Core I5 de 3.2Ghz em uma arquitetura de 64 bits utilizando um sistema operacional Ubuntu 14.04.

Para testes foi utilizado um vídeo que pode ser encontrado no Youtube acessando o

seguinte link: “https://www.youtube.com/watch?v=b0MpBPPT0pw”, e foi utilizado para download do vídeo em diferentes resoluções o seguinte site: “http://pt.savefrom.net/”.

Após a configuração do ambiente e o download do vídeo para teste, foram implementadas funções de métrica nas duas aplicações, a fim de calcular a quantidade de frames por segundo processados e gravar essas informações em um arquivo para uma analise futura.

4.2. CÓDIGOS TESTADOS

Para realizar um teste efetivo de desempenho, é necessário realizar alterações nos códigos para ser possível coletar os dados sobre a quantidade de frames processados por segundo. Para que seja possível coletar tais informações, foi necessário adicionar funções de métrica no código das duas aplicações, na qual o apêndice 1, mostra como a métrica é feita, o código demonstrado no apêndice 1 foi implementado pelo autor deste trabalho, utilizando somente funções do OpenCV e será o primeiro código a ser testado. No código do Vibe foi adicionada a mesma métrica utilizada no OpenCV demonstrado no apêndice 1.

4.3. RESULTADOS DOS TESTES

Após estruturar o ambiente e modificar os códigos para coletar os dados, foram realizadas 10 repetições do teste em cada situação, testando as duas aplicações em duas máquinas diferentes e em três resoluções de vídeo diferentes, conforme citado anteriormente.

Com os testes pode-se observar na tabela 2, os dados que foram extraídos e mais adiante poderá ser observado os gráficos de comparação dos testes.

Tabela 2 - Dados extraídos dos testes

OpenCV			**Vibe**		
Máq.x Res.	Média	Desvio Padrão	Máq.x Res.	Média	Desvio Padrão
Máq.1 - 360p	19,84865629	1,010431585	Máq.1 - 360p	27,48	3,274827933
Máq.1 - 720p	5,934317343	0,452430041	Máq.1 - 720p	7,785786802	0,979360696
Máq.1 - 1080p	3,189914866	0,449131847	Máq.1 - 1080p	4,059496568	0,507392894
Máq.2 - 360p	81,16564417	5,640750983	Máq.2 - 360p	127,99	18,36195837
Máq.2 - 720p	21,71495327	1,640381813	Máq.2 - 720p	32,65737052	5,447397127
Máq.2 - 1080p	10,29387187	1,003820601	Máq.2 - 1080p	15,33870968	2,546498699

Após a obtenção desta tabela, foi possível realizar a comparação do desempenho das duas aplicações, conforme pode-se acompanhar adiante.

No gráfico apresentado na Figura 15, observa-se os resultados da comparação com base no método do OpenCV, na qual se pode observar uma grande diferença de desempenho com relação ao tempo nas duas máquinas, porém a qualidade da extração continuou a mesma.

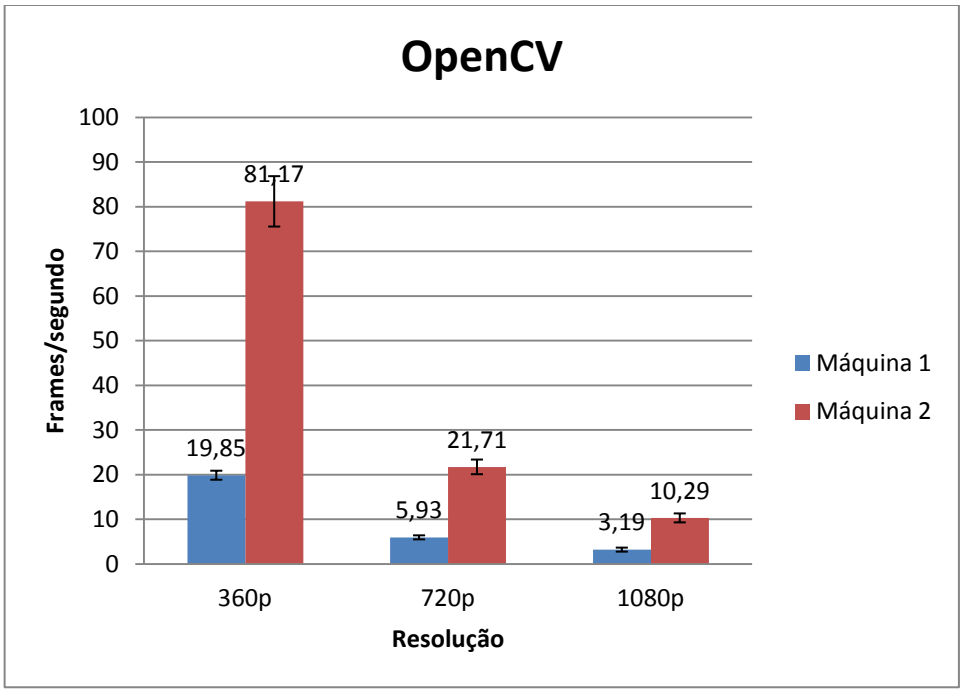


Figura 15 - Gráfico referente aos testes do OpenCV

No gráfico apresentado na Figura 16, observa-se os resultados dos testes feitos no método Vibe, na qual também é possível perceber uma grande diferença de desempenho com relação ao tempo nas duas máquinas e a qualidade da extração também continuou a mesma. Vale ressaltar que o desvio padrão, utilizado para representar a variação do processamento de frames por segundo, do Vibe é maior devido à maneira que o mesmo foi implementado, ou seja, devido ao mesmo realizar um pré-processamento no primeiro frame, deixando o processamento deste frame muito mais lento, os demais frames são processados mais rapidamente, conseguindo realizar a leitura de uma quantidade muito superior de frames que a primeira implementação, porém a implementação está sujeita a ficar com ruído no resultado, pois caso o vídeo inicie com objetos em movimento, o primeiro frame ficara como padrão de fundo, gerando assim “fantasmas” que atrapalham na detecção dos frames seguintes. Para um resultado satisfatório com Vibe, é necessário que se obtenha imagens do fundo sem nenhum

objeto, para que primeiramente seja realizado um treinamento e somente após isso fazer a leitura dos objetos em movimento.

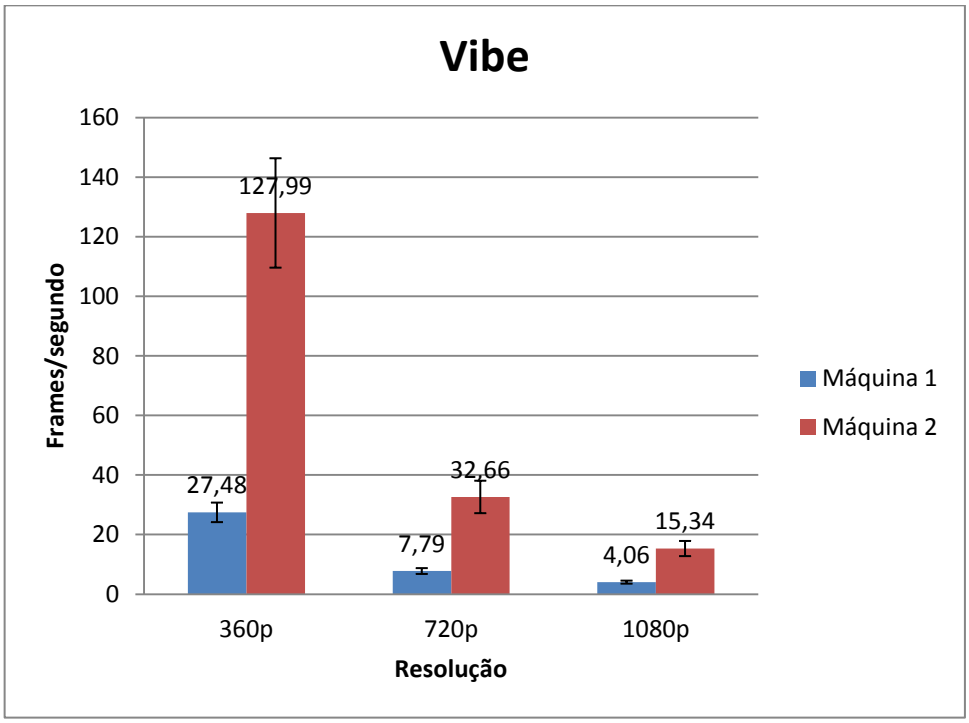


Figura 16 - Gráfico referente aos teste do Vibe

Observando os gráficos gerados anteriormente, nota-se resultados muito parecidos de desempenho sobre as máquinas, portanto é necessário colocar as implementações lado a lado para entender melhor a diferença entre elas, pois em gráficos distintos fica difícil a visualização entre a diferença dos métodos de extração de fundo apresentados. Na Figura 17, será apresentado o gráfico com a comparação das implementações, onde poderá ser observada uma grande diferença entre elas.

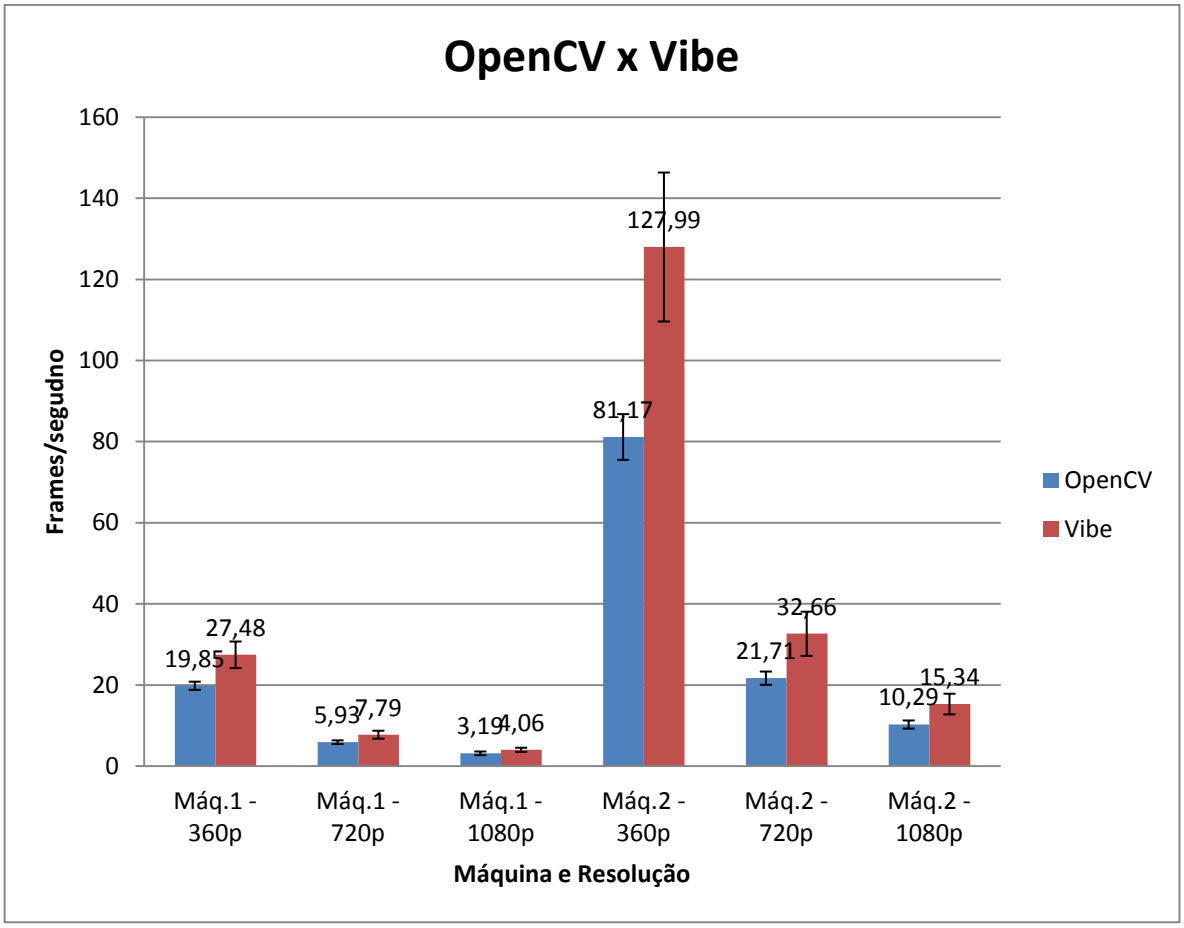
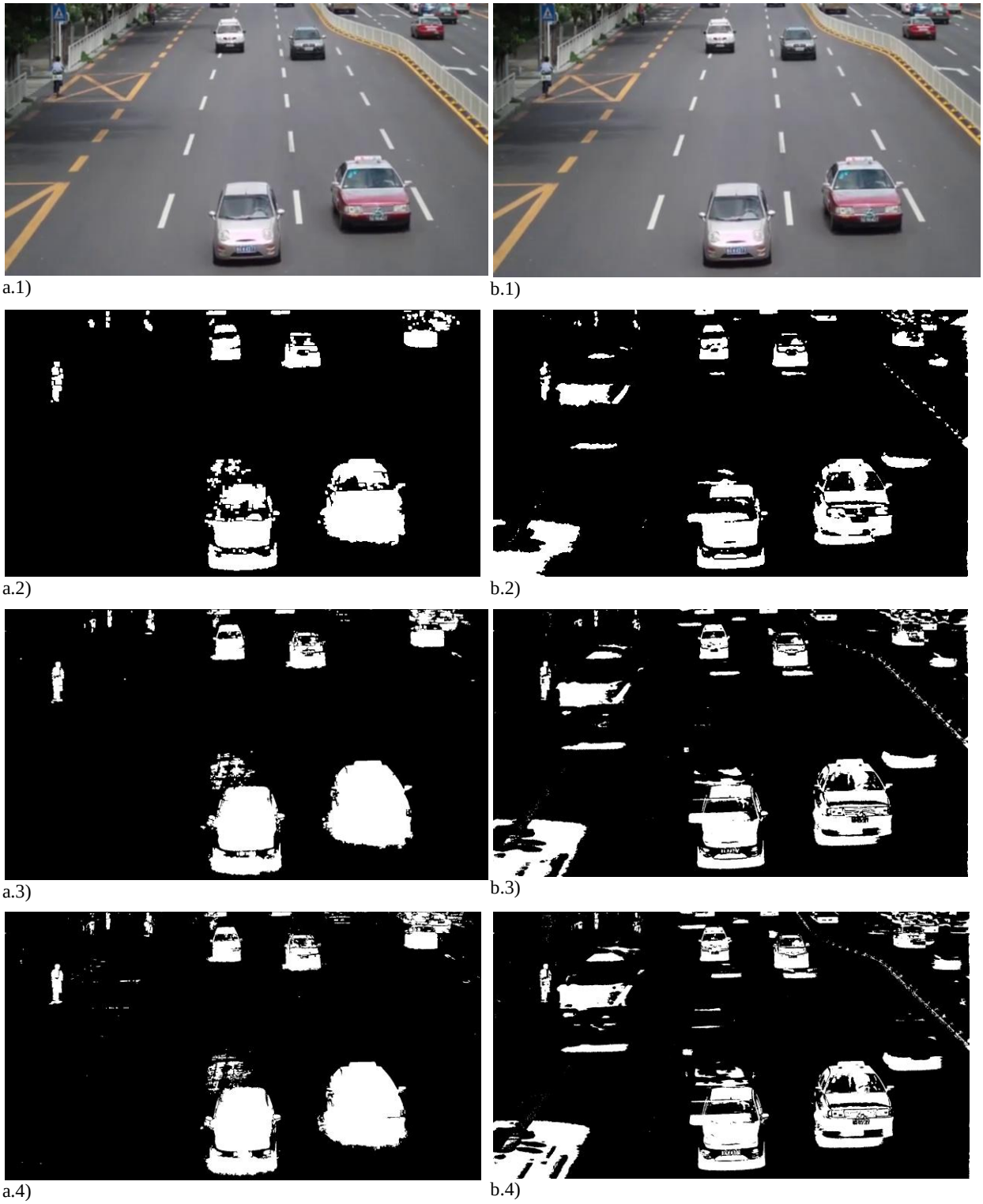


Figura 17 - Gráfico da comparação entre as aplicações

Neste gráfico nota-se que quanto maior a resolução menor é a diferença entre elas no resultado, portanto concluí-se que existe uma tendência de igualdade em termos de desempenho entre as duas implementações, no entanto, o desvio padrão do Vibe é sempre substancialmente maior que o do OpenCV, logo nota-se que, para que o Vibe possa ter um desvio padrão menor, seria necessário um novo processamento do fundo a cada frame, como é feito pelo OpenCV, pois assim, a implementação iria atualizar a cada instante quais são os objetos do fundo e quais estão em movimento, assim não deixando a possibilidade de existir "fantasmas" como ocorre atualmente.

Entretanto, essa alteração no Vibe teria um elevado custo em processamento, o que faria seu desempenho cair de forma considerável, tornando-o menos atrativo para utilização em soluções de detecção de objetos em movimento. Por esse motivo, é preferível fazer um treinamento correto no primeiro frame do Vibe, para que não existam "fantasmas" no resultado final.

A seguir será mostrado o resultado da extração de fundo que foi realizada nos testes para coleta de dados, na qual será possível observar os “fantasmas” citados neste capitulo assim como uma pequena diferença no tratamento de ruídos no vídeo.



a.1) b.1)

a.2) b.2)

a.3) b.3)

a.4) b.4)

Figura 18 - Resultados das implementações sendo “a” da Implementação 1 e “b” do Vibe (a.1 e b.1 são imagens do vídeo original em 360p, a.2 e b.2 resultado na resolução 360p, a.3 e b.3 resultado em 720p e, a.4 e b.4 resultado em 1080p)

Na Figura 18, temos do lado direito os resultados do teste com o Vibe, denominados na figura como “b”, pode-se perceber que na implementação do Vibe existe menos ruído que no OpenCV, porém além dos objetos em movimento, existem “fantasmas”, que são objetos que foram detectados no primeiro frame do vídeo. Do lado esquerdo, temos os resultados dos testes realizados com o OpenCV, na qual existe mais ruído que no resultado do Vibe, porém nota-se que não existem “fantasmas” nesta implementação.

Durante os testes, foi possível perceber que em ambas implementações quanto maior a resolução da imagem, mais ruídos eram visíveis, pois quanto mais detalhes visíveis na imagem, mais objetos poderão variar no fundo da imagem.

5. CONCLUSÕES

Ao longo deste trabalho, foram apresentadas análises elaboradas para testar a qualidade e o desempenho de dois métodos de extração de fundo, o método do OpenCV, utilizando EGMM e um método conhecido como Vibe, na qual foram discutidos e apresentados resultados nos capítulos anteriores.

O método utilizado na Implementação 1, que faz uso puramente da funções do OpenCV, é baseado em EGMM e possui alguns problemas com ruído, mas nada que afete de forma considerável na detecção de objetos em movimento. Possui um bom desempenho, porém perde para o Vibe na quantidade de frames processados por segundo, pois não reaproveita o fundo como modelo e sim faz um novo treinamento do fundo.

O método conhecido como Vibe, tem vantagem em termos de desempenho, já que cria um modelo para utilizar como fundo fixo. Desde que seja realizado um treinamento correto do modelo de fundo, não haverá um efeito ruim em termos de qualidade de extração, pois caso contrário, aparecerão “fantasmas” que atrapalharam no reconhecimento de objetos móveis.

A comparação mostrou que ainda existem vários aspectos a serem melhorados nos métodos de subtração de fundo, tanto em qualidade, removendo ruídos, quanto em desempenho em altas resoluções, pois a tendência do mercado é que cada vez mais câmeras tenham alta resolução e hoje este tratamento de detecção de objeto em movimento ainda é um problema em vídeos de alta definição, pois exige um elevado desempenho de processamento para que se possa obter um resultado satisfatório.

Conclui-se que ambos os métodos são bons, porém cada um com suas características, ficando a critério do usuário utilizar o método que melhor se adequar para sua situação. O uso destas técnicas para RMA ainda necessita de um processamento feito por um software, ou seja, demanda-se de um tempo de resposta tardio, portanto este estudo serviu para fomentar interesse em pesquisas de implementação em hardware de métodos de extração de fundo.

Ainda, sugere-se como trabalhos futuros a implementação de novas técnicas de extração de fundo, baseando-se nos algoritmos criados por Hoprasert, et al. (1999) ou Cheung, et al. (2000), para tentar resolver os problemas encontrados nesta comparação, a fim de colaborar com a comunidade cientifica, disponibilizando códigos abertos e sem nenhum custo de distribuição, para que novos pesquisadores possam utilizar e melhorar os métodos de extração já conhecidos.

REFERÊNCIAS BIBLIOGRÁFICAS

Análise de Movimento e Rastreamento de Objeto. API OpenCV. Disponível em: <http://docs.opencv.org/modules/video/doc/motion_analysis_and_object_tracking.html#backgroundsubtractormog2-backgroundsubtractormog2>.

Analise Estrutural e descritores de forma. API OpenCV. Disponível em: <http://docs.opencv.org/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html>.

BARNICH, O. and DROOGENBROECK, M. V., **ViBe: a powerful random technique to estimate the background in video sequences**. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2009)*, pages 945-948, April 2009.

CHEUNG, K., KANADE, M., BOUGUET, J. and HOLLER, M**., A Real Time System for Robust 3D Voxel Reconstruction of Human Motions**, Proceedings of the 2000 IEEE Conference on Computer Vision and Pattern Recognition (CVPR '00), Vol. 2, June, 2000, pp. 714 – 720.

CHEUNG, S. C. and KAMATH, C., **Robust techniques for background subtraction in urban traffic video. Video Communications and Image Processing**, Volume 5308, pp 881-892, SPIE Electronic Imaging, San Jose, January 2004.

ELGAMMAL, A., HARWOOD, D., DAVIS, L., **Non-parametric model for background subtraction**. 6th European Conference on Computer Vision. Dublin, Ireland, June/July 2000.

Filtragem de imagem. API OpenCV. Disponível em: <http://docs.opencv.org/modules/imgproc/doc/filtering.html>.

FRIEDMAN, N. and RUSSEL, S., **Image Segmentation in Video Sequences: A Probabilistic Approach.** Proc. Thirteenth Conf. on Uncertainty in Artificial Intelligence (UAI 97), 1997.

GATTASS, M., **Introdução às cores**. Disponível em http://www.tecgraf.puc-rio.br/~mgattass/cg/pdf/.

GONZALEZ, R. C., WOODS, R. E. **Digital Image Processing**. New York: Addison-Wesley Publishing Company, 1992.

GROB, B., **Televisão e sistemas de vídeo. 5 ed**., Rio de Janeiro: Editora Guanabara, 1989.

HOPRASERT, T., HARWOOD, D., **A Robust Background Subtraction and Shadow Detection.** In the proceedings of the fourth Asian Conference on Computer Vision, 2000.

HORPRASERT, T., HARWOOD, D. and DAVIS, L., **A Statistical Approach for Real-Time Robust Background Subtraction and Shadow Detection**. Proceedings IEEE ICCV'99 FRAME-RATE Workshop, Greece, September 1999.

Introdução ao OpenCV. API OpenCV. Disponível em:<http://docs.opencv.org/modules/core/doc/intro.html#api-concepts>.

KALMAN, R. E., **A New Approach to Linear Filtering and Prediction Problems, Transactions of the ASME** - Journal of Basic Engineering, Vol. 82: pp. 35-45, 1960.

KIM, K., HORPRASERT, T., HARWOOD, D. and DAVIS, L., **Real-time Foreground–background segmentation using codebook model**. Real-Time Imaging 11, pp. 172–185, 2005.

MARR, D., **Vision: a Computational Investigation into the Human representation and Processing of Visual Information**. New York: Freeman, 1982.

MASSIMO PICCARDI. **Background Subtraction Techniques: A Review Computer Vision Group**, Faculty of Information Technology University of Technology, Sydney (UTS), Australia.

McFARLANE, N. and SCHOPFIED, C., **Segmentation and tracking of piglets in images**, Machine Vision and Applications 8(3), pp. 187-193, 1995.

MARQUES FILHO, Ogê; VIEIRA NETO, Hugo. **Processamento Digital de Imagens**, Rio de Janeiro: Brasport, 1999. ISBN 8574520098.

SCHNEIDER, M. R., **Sistema de segurança e proteção baseado em visão computacional**. Curitiba. UTFPR, 2007.

Suzuki, S. and Abe, K., **Topological Structural Analysis of Digitized Binary Images by Border Following**. CVGIP 30 1, pp 32-46 (1985).

TOYAMA, K., KRUMM, J., BRUMITT, B., and MEYERS, B., **Wallower: Principles and practice of background maintenance**,ICCV (1), pp. 255-261, 1999.

TRIVEDI, M.M., ROSENFELD, A., **On making computers "See"**. IEEE transactions on systems, man and cybernetics, v.19, n.6, p.1333-6, Nov. 1989.

TRUYENQUE, M. A. Q. **Uma Aplicação de Visão Computacional que Utiliza Gestos da Mão para Interagir com o Computador**, Dissertação de Mestrado, PUC-Rio, Março de 2005.

WIKIPEDIA, **Bayer Filter**. Disponível em: <http://en.wikipedia.org/wiki/Bayer_filter>.

WOLF, D. F.; SIMOÊS, E. V.; OSÓRIO, F. S.; TRINDADE JUNIOR, O. **Robótica Móvel Inteligente: da simulação às aplicações no mundo real**, Minicurso, Jornada de Atualização em Informática, Bento Gonçalves/RS, 2009.

Z.Zivkovic, F. van der Heijden, **Efficient Adaptive Density Estimapion per Image Pixel for the Task of Background Subtraction**, Pattern Recognition Letters, vol. 27, no. 7, pages 773-780, 2006.

Z.Zivkovic, F.van der Heijden, **Recursive unsupervised learning of finite mixture models**,

IEEE Trans. on Pattern Analysis and Machine Intelligence, vol.26, no.5, pages 651-656, 2004.

Z.Zivkovic, **Improved adaptive Gausian mixture model for background subtraction**, International Conference Pattern Recognition, UK, August, 2004.

APÊNDICE

1) Código da implementação do método de extração de fundo utilizando somente funções do OpenCV, no trabalho chamado de “OpenCV”:

```
#include <opencv2/opencv.hpp>
#include <vector>
#include <opencv2/core/core.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <iostream>
#include <cstdio>
#include <ctime>

int main(int argc, char* argv[])
{
	FILE *arq; //DECLARACAO DO ARQUIVO PARA GUARDAR INFORMACOES
	arq = fopen("teste1080p.txt", "a+");  // CRIA UM ARQUIVO TEXTO PARA GRAVA-
CAO
	const char * file = (argc >= 2)? argv[1]:"video.mp4"; //CASO NAO INFORME NOME
DE VIDEO ELE BUSCAR POR ESSE NOME

	clock_t initialTime = 0; //
	clock_t finalTime = 0;   // DECLARACAO DE VARIAVEIS PARA CALCULO DE
FRAMES POR SEGUNDO
	double segPerFrame = 0.0;//
	int i=0,f=0;		//
  cv::Mat frame;//
  cv::Mat back; //DECLARACAO DAS MATRIZES PARA SEREM PROCESSADAS
  cv::Mat fore; //

  //cv::VideoCapture cap(0); // ABRE WEBCAM
  cv::VideoCapture cap(file); // ABRE ARQUIVO DE VIDEO

  cv::BackgroundSubtractorMOG2 bg; //
  bg.nmixtures = 3;                                // DECLARACAO DA FUNCAO DE EXTRA-
CAO DE FUNDO E CONFIGURACAO DE PARAMETROS
  bg.bShadowDetection = true;                      //
  std::vector<std::vector<cv::Point> > contours; // DECLARACAO DA ARRAY DE
CONTORNOS
	cv::namedWindow("Frame",0);	//
	cv::namedWindow("Background",0); // DECLARACAO DA JANELAS ONDE SERAO
EXIBIDOS OS RESULTADOS
	cv::namedWindow("Foreground",0); //

	for(i=1;;i++) //LACO PARA CAPTURA DE FRAMES
	{
		cap >> frame; // CAPTURA DE FRAME
		initialTime = clock(); // ATRIBUI O TEMPO INICIAL
```

```
                GaussianBlur( frame, frame, cv::Size(3,3), 0, 0, cv::BORDER_DEFAULT ); // LIMPEZA DE RUIDOS
                bg.operator ()(frame,fore,-1);// CRIAR UM FOREGROUND A PARTIR DO FRAME UTILIZANDO MOG2
                bg.getBackgroundImage(back);// SUBTRACAO DE FUNDO COM MOG2
                cv::erode(fore,fore,cv::Mat());//    EROSAO E DILATACAO AFIM DE REMOVER PIXEIS DESNECESSARIOS E DEIXAR A IMAGEM UNIFORME
                cv::dilate(fore,fore,cv::Mat());//

        cv::findContours(fore,contours,CV_RETR_EXTERNAL,CV_CHAIN_APPROX_NONE);// CRIANDO ARRAY DE CONTORNOS PARA MARCAR
                cv::drawContours(fore,contours,-1,cv::Scalar(255,255,255),-1);// MARCANDO CONTORNOS DE ACORDO COM O ARRAY GERADO ANTERIORMENTE

                finalTime = clock(); // ATRIBUI O TEMPO FINAL
                segPerFrame += ((float)(finalTime - initialTime))/CLOCKS_PER_SEC; // CALCULO DE TEMPO A CADA FRAME
                if(segPerFrame >= 1.0){       // CODICAO PARA QUANDO O TEMPO ATINGIR 1 SEGUNDO
                        segPerFrame --;                 // SUBTRAI UM SEGUNDO DO TEMPO
                        fprintf(arq,"10\t%d\n",i); // IMPRIME EM UM ARQUIVO A QUANTIDADE DE FRAMES POR SEGUNDO
                        i=1;                            // VOLTA O CONTADOR PARA 1
                }
                cv::imshow("Frame",frame);    // MOSTRA O VIDEO ORIGINAL
                cv::imshow("Background",back);// MOSTRA O FUNDO EXTRAIDO
                cv::imshow("Foreground",fore);// MOSTRA OS OBJETOS EM MOVIMENTO

                if(cv::waitKey(30) >= 0) break; // CASO PRESSIONE ALGUMA TECLA ELE ABORTA A EXECUCAO
        }
        fclose(arq); // FECHA O ARQUIVO
  return 0; // ENCERRA A APLICACAO
}
```