unesp UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
CAMPUS DE GUARATINGUETÁ

ROBERTO OTSUBO GARCIA

SISTEMA DE MONITORAÇÃO DE BLOQUEIOS TIPO PORTA DE VIDRO

Guaratinguetá
2011

UNESP
Faculdade de Engenharia do Campus de Guaratinguetá

Guaratinguetá
2011

ROBERTO OTSUBO GARCIA

SISTEMA DE MONITORAÇÃO DE BLOQUEIOS TIPO PORTA DE VIDRO

Trabalho de Graduação apresentado ao Conselho de Curso de Graduação em Engenharia Elétrica da Faculdade de Engenharia do Campus de Guaratinguetá, Universidade Estadual Paulista, como parte dos requisitos para obtenção do diploma de Graduação em Engenharia Elétrica.

Orientador: Prof. Dr. Leonardo Mesquita

Guaratinguetá
2011

Garcia, Roberto Otsubo
G216s Sistema de monitoração de bloqueios tipo porta de vidro / Roberto Otsubo Garcia – Guaratinguetá : [s.n], 2011.
76 f : il.
Bibliografia: f. 48

Trabalho de Graduação em Engenharia Elétrica – Universidade Estadual Paulista, Faculdade de Engenharia de Guaratinguetá, 2011.
Orientador: Prof. Dr. Leonardo Mesquita

1. Visão por computador 2. Pessoas – rastreamento automático
I. Título

CDU 007.52

unesp UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
CAMPUS DE GUARATINGUETÁ

SISTEMA DE MONITORAÇÃO DE BLOQUEIOS TIPO PORTA DE VIDRO

ROBERTO OTSUBO GARCIA

ESTE TRABALHO DE GRADUAÇÃO FOI JULGADO ADEQUADO COMO PARTE DO REQUISITO PARA OBTENÇÃO DO DIPLOMA DE **"GRADUADO EM ENGENHARIA ELÉTRICA"**

APROVADO EM SUA FORMA FINAL PELO CONSELHO DE CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Prof. Dr. SAMUEL EUZÉDICE DE LUCENA
Coordenador

BANCA EXAMINADORA:

Prof. Dr. LEONARDO MESQUITA
Orientador/UNESP-FEG

Prof. Dr. SAMUEL EUZÉDICE DE LUCENA
UNESP-FEG

Prof. MSc. FERNANDO RIBEIRO FILADELFO
UNESP-FEG

Novembro de 2011

DADOS CURRICULARES

ROBERTO OTSUBO GARCIA

NASCIMENTO	03.04.1987 – SÃO PAULO / SP
FILIAÇÃO	Marco Antonio Garcia Luisa Otsubo
2002/2004	Ensino Médio Colégio Agostiniano Mendel
2006/2011	Curso de Graduação em Engenharia Elétrica Faculdade de Engenharia do Campus de Guaratinguetá Universidade Estadual Paulista

AGRADECIMENTOS

Agradeço primeiramente à minha família, em especial aos meus pais *Marco e Luisa* que fizeram possível esta oportunidade, que sempre me incentivaram e auxiliaram nos momentos de dificuldade e nunca mediram esforços para que eu tivesse uma boa educação.

aos meus amigos de faculdade, que nos momentos de alegria e de dificuldades estiveram ao meu lado, sempre me apoiando e incentivando. Em especial aos meus colegas: *Diego, Denis, Leandro, Anderson, Rodrigo, Rafael, André, Saulo, Luan e João Pedro*.

aos meus amigos *Lemos, Gustavo, Alan e Matheus*, que me acompanharam durante todo curso não só me ajudando nas horas de estudo, mas também me auxiliando em todos momentos difícieis que se passaram.

aos docentes e funcionários da faculdade, que me passaram muito de seu conhecimento técnico e de vivência, e me fizeram sentir orgulho de estudar nessa instituição.

“O impossível existe até que alguém duvide dele e prove o contrário”

Albert Einsten

GARCIA, R. O. **Sistema de monitoração de bloqueios tipo porta de vidro.** 2011. Trabalho de Graduação (Graduação em Engenharia Elétrica) – Faculdade de Engenharia do Campus de Guaratinguetá, Universidade Estadual Paulista, Guaratinguetá, 2011.

RESUMO

Sistemas automatizados de monitoramento tem sido motivação de inúmeros estudos, devido as inúmeras possibilidades de aplicação. Neste trabalho foi desenvolvido um sistema de rastreamento e contagem de pessoas bem como detecção de atividades suspeitas. Este modelo rastreia objetos que passam pelo campo de visão de uma câmera e utiliza algoritmos para classificar as atividades de cada pessoa, e baseando-se nessas informações, detecta eventos de risco. Este trabalho inclui uma revisão de diversas técnicas já desenvolvidas, e tem como objetivo o desenvolvimento de um sistema robusto e com baixo custo computacional para ser utilizado em bloqueios tipo porta de vidro de estações de trem com o intuito de detectar fraudes.

PALAVRAS-CHAVE: Visão computacional**.** Sistema de monitoramento automático. Rastreamento de pessoas. Contagem de pessoas. Filtro de kalman.

GARCIA, R. O. **Automatic Surveillance System For Glass Door Barrier.** 2011. Graduate Work (Graduate in Electrical Engineering) – Faculdade de Engenharia do Campus de Guaratinguetá, Universidade Estadual Paulista, Guaratinguetá, 2011.

ABSTRACT

Automatic video surveillance system has been a frequent topic of research due to the large number of promising applications. In this research, we developed a tracking and counting people system, as well as suspicious activities detector. The model tracks individual objects as they pass through the field of vision of the camera using vision algorithms to classify the activities of each person, and according to this features, detect dangerous situations. This dissertation includes a review of several techniques trying to develop a robust and low computacional costs system to be used in glass door barrier turnstiles avoiding fraud.

KEYWORDS: Computer vision**.** Automatic surveillence system. Tracking people. Counting people. Kalman filter.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

RGB	-	Formato de cores com componentes Vermelho(*Red*), Verde(*Green*) e Azul(*Blue*)
HSV	-	Formato de cores baseado em Matiz (*hue*), Saturação(*Saturation*) e Valor(*Value*)
Blob	-	Estrutura contendo dados binários (*Binary Large Object*)
AMF	-	Filtro de detecção de plano de fundo (*Approximated Median Filter*).
MoG	-	Filtro de detecção de plano de fundo baseado em misturasa gaussianas (*Mixture of Gaussians*).
Hmm	-	*Hidden Markov Models* – Cadeias ocultas de Markov

SUMÁRIO

1 INTRODUÇÃO

Atualmente as tecnologias que simulam a visão humana despertam um grande interesse mundial devido às inúmeras soluções que podem oferecer como: vigilância automática, controle de acesso, controle de tráfego de carros, interação homem-máquina. Em virtude deste interesse a comunidade científica tem evidado esforços, e, consequentemente apresentado resultados significativos apesar das dificuldades em segmentar e analisar imagens digitais.

Dentre os segmentos de visão computacional a área de vigilância automatizada se destaca diante da crescente necessidade de implementação de sistemas de vigilância que otimizem o monitoramento de pessoas e diminuam a quantidade de mão-de-obra utilizada para estes serviços.

A utilização de sistemas de monitoramento automático possibilita, mediante a análise simultânea de diversas câmeras, o emprego da atenção humana somente quando necessário.

Esse trabalho tem como objetivo desenvolver um sistema capaz de auxiliar na detecção de fraudes em bloqueios tipo porta de vidro, uma vez que constatada deficiência no sistema de sensores infra-vermelho. Para tanto, o sistema de vigilância automática que é desenvolvido nesta monografia terá como primeiro objetivo auxiliar os bloqueios e alertar quando houver fraudes, consequentemente, esse sistema gera a possibilidade da criação de um dispositivo para contagem de pessoas, implementando um sistema de baixo custo em relação à tecnologia empregada atualmente.

O segundo objetivo deste trabalho, em complementação ao primeiro ponto apresentado, é desenvolver algoritmos de detecção de objetos abandonados e de tumultos, prevendo situações que possam oferecer riscos ou danos aos usuários dos trens.

Para o desenvolvimento desta proposta foram empregadas diversas técnicas apresentadas em sistemas de rastreamento (*Tracking People*), contagem de pessoas (*Counting People*) e Reconhecimento de Atividades Humanas (*Human Activity Recognition*).

Portanto este trabalho tem um caráter de revisão bibliográfica e, concomitantemente, visa encontrar técnicas que viabilizem a criação de um sistema robusto e de baixo custo a ser utilizado em ambientes específicos, como no caso em questão em estações de trem.

A primeria etapa deste projeto consiste na detecção de pessoas em um ambiente restrito. Por se tratar de local específico a técnica utilizada para localização de pessoas emprega uma segmentação de plano de fundo, uma vez que inexistem objetos em

movimento além das pessoas, evitando assim grande esforço computacional gerado para detecção de faces.

Na segunda etapa são rastreados os objetos da cena mediante a implementação do filtro de Kalman que torna possível a detecção de oclusões - uma das maiores barreiras no sistema de monitoração utilizando apenas uma câmera. Além deste filtro ser robusto, ele apresenta a vantagem de exigir menos esforço computacional comparado com técnicas de detecção de características (forma, tamanho, segmentação de cores) como Sift(*Scale Invariant Feature Transform*) e *Mean Shift* discutidos neste trabalho.

Ao final é desenvolvido um algoritmo para classificar as ações que um objeto pode apresentar numa cena. E a partir destes dados será possível detectar eventos como objetos abandonados, brigas ou tumultos. Também será desenvolvido um algoritmo para contagem de pessoas que cruzam um determinado local, possibilitando, assim, a detecção de fraude nos bloqueios.

1.1 Introdução ao processamento de imagens

O processamento digital de imagens é uma área de conhecimento que tem objetivo a manipulação e análise de imagens digitais por um *hardware* visando a extração de informações de interesse específicos e também o aperfeiçoamento das imagens para a visão humana.

A utilização de técnicas de processamento digital está cada vez mais presente em nosso cotidiano. Principalmente devido à redução de custo dos equipamentos de vídeo e do aumento significativo do poder de processamento digital nas últimas décadas.

Uma das primeiras aplicações da área de processamento de imagens foi aprimoramento da qualidade de impressão de imagens digitalizadas transmitidas por um cabo submarino entre Londres e Nova Iorque. Porém, essa área só teve um grande impulso com o início do programa espacial Norte Americano, quando imagens transmitidas da lua por uma sonda Ranger eram processadas por um computador para corrigir vários tipos de distorção inerentes à câmera de TV acoplada à sonda.

Atualmente, a área de processamento de imagens já apresenta aplicações em quase todos os ramos da atividade humana. Na medicina, com processamento de imagens utilizado para interpretar dados gerados por raio-x; na biologia, com capacidade de processar automaticamente imagens geradas a partir de microscópios, com alto grau de precisão e

repetibilidade; na análise de dados geográficos de imagens capturadas com satélite; na visão artificial em robótica e automação industrial e inúmeras outras áreas como astronomia, segurança, publicidade e direito.

1.1.1 Sistemas de processamento de imagens

Os elementos presentes em um sistema de processamento de imagens são mostrados na figura 1, os quais abrangem as principais operações que se podem efetuar sobre uma imagem: aquisição, processamento e exibição.

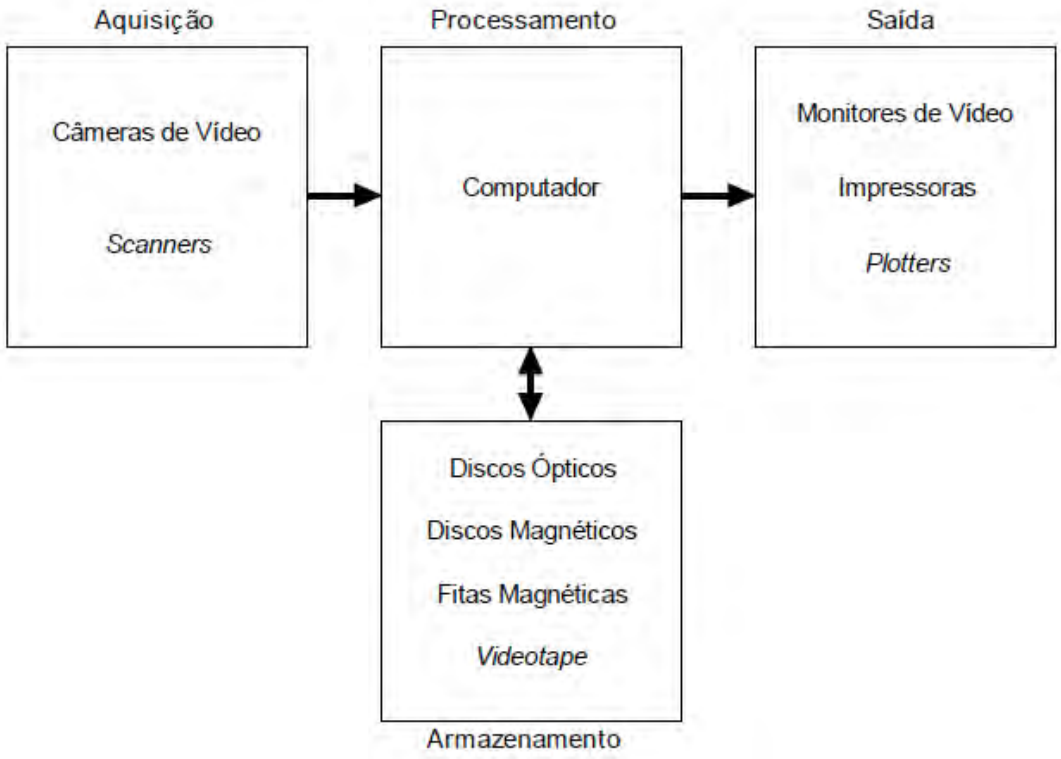


Figura 1 – Sistema básico de processamento de imagens

A primeira etapa consiste em converter as informações analógicas adquiridas para uma representação adequada para o processamento digital, que é dividida em duas partes: o dispositivo físico, que converte uma faixa do espectro eletromagnético em um sinal elétrico para o digitalizador converter esse sinal analógico em uma informação digital, representada por “0” e “1”.

A etapa de armazenamento é um grande desafio para o projeto de um sistema de processamento digital de imagens, devido à grande quantidade de espaço para o mesmo.

O processamento da imagem consiste em desenvolver um software/hardware para detectar e interpretar as imagens que variam de acordo com a aplicação que está sendo desenvolvida.

E finalmente, a parte de exibição, na qual ocorre a interação homem-máquina do sistema ou o armazenamento dos dados.

1.2 Sistema de visão computacional

Podemos definir a visão computacional como um conjunto de métodos e técnicas para interpretar imagens através de métodos computacionais. No ponto de vista de Siebel (2005) os sistemas de visão computacional tem como objetivo criar aos computadores uma capacidade de percepção semelhante à humana.

A estrutura de um sistema de visão computacional varia de acordo com o tipo de aplicação. Porém algumas funções típicas são encontradas em vários sistemas de visão computacional, conforme sistema a ser observado na figura 2.

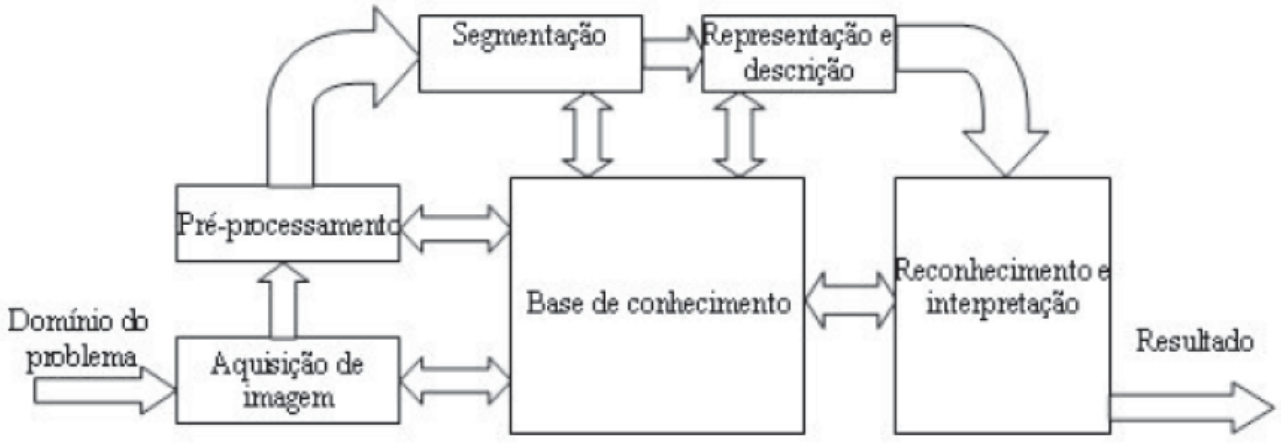


Figura 2 – Estrutura típica encontrada em sistemas de visão computacional.Fonte: Geisler(2006)

Na primeira etapa do sistema ocorre a aquisição da imagem, normalmente captadas por sensores CMOS e CCD. Dependendo do tipo do sensor, o resultado pode variar entre uma imagem bidimensional, uma cena tridimensional ou ainda uma sequência de imagens. Os valores dos pixels geralmente indicam a intensidade da luz em uma ou várias faixas de cor (o

que forma imagens em tom de cinza ou coloridas), mas também podem indicar valores físicos como profundidade e absorção ou reflexão das ondas eletromagnéticas.

A segunda etapa, chamada de pré-processamento. Para adequar o formato de vídeo antes de um método de visão computacional ser aplicado em uma imagem para extrair informação. Exemplos incluem remapeamento (para assegurar o sistema de coordenadas), redução de ruídos (para assegurar que as informações são verdadeiras) e aumento de contraste (para assegurar que as informações relevantes serão detectadas). A segmentação da imagem inclue a seleção de regiões de interesse específicos e segmentação de uma ou mais regiões que contém um objeto de interesse.

Finalmente, na etapa de reconhecimento e interpretação, é neste ponto a entrada é geralmente um conjunto pequeno de dados. O processo posterior inclui a verificação da satisfação dos dados, a estimativa de parâmetros sobre a imagem e a classificação dos objetos detectados em diferentes categorias.

1.2 Técnicas em processamento de imagens

Neste capítulo serão apresentadas algumas das técnicas básicas utilizadas em processamento de imagens que possibilitaram o desenvolvimento de um sistema de vigilância automática.

1.2.1 Técnicas em processamento de imagens

A morfologia matemática é utilizada em processamento de imagens para extração de componentes de uma imagem que sejam úteis para representação e descrição de formas de uma região, como fronteiras, esqueletos e fechos convexos.

A linguagem matemática é a teoria de conjuntos. Os conjuntos em morfologia representam objetos de uma imagem. Em imagens binárias, os conjuntos em questão são membros do espaço bidimensional de números inteiros Z^2.

Inicialmente é necessário compreender algumas definições de conjuntos para entender as operações básicas da morfologia:

Sendo *A* e *B* conjuntos em Z^2, cujos componentes são $a=(a_1,a_2)$ *e* $b=(b_1,b_2)$. A translação de *A* por $x=(x_1,x_2)$,denotada $(A)_x$, é definida como:

$$(A)_x = \{c|c = a + x, para\ a \in A\} \quad (1)$$

Reflexão de *B*, Denotada $\hat{B}$, é definida como:

$$\hat{B} = \{x|x = -b, para\ b\ \in B\} \quad (2)$$

O complemento do conjunto *A* é:

$$A^c = \{x|x \notin A\} \quad (3)$$

Diferença entre *A e B* é:

$$A - B = \{x|x \in A, x \notin B\} = A \cap B^c \quad (4)$$

- Dilatação

Sejam *A* e *B* conjuntos do espaço Z^2 e seja $\emptyset$ o conjunto à vazio. A dilatação de *A* por *B*,mostrada na figura 3 e denotada como $A \oplus B$, é:

$$A \oplus B = \{x|\left(\hat{B}\right)_x \cap A \neq \emptyset\} \quad (5)$$

Ou seja, o processo de dilatação obtém-se da reflexão de *B* em torno da sua origem, seguido da translação dessa reflexão por *x*. A dilatação é então o conjunto de todos os deslocamentos *x* tais que *A* sobreponham-se em pelo menos um elemento não-nulo.

$$A \oplus B = \{x|[\left(\hat{B}\right)_x \cap A] \sqsubseteq A\} \quad (6)$$

O conjunto *B* é normalmente denominado elemento estruturante.

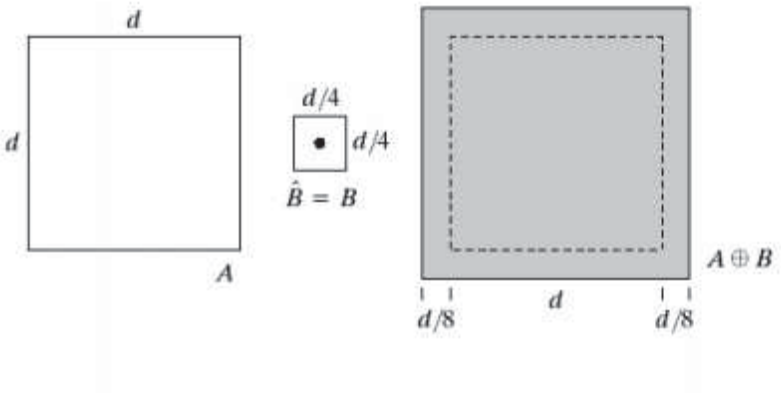


Figura 3 – Dilatação do elemento A pelo elemento B.

- Erosão

Sejam *A* e *B* conjuntos no espaço Z^2. A erosão de *A* por *B*,mostrada na figura *4* e denotada como $A \ominus B$, é:

$$A \ominus B = \{x(B)_x \subseteq A\} \tag{7}$$

A erosão de *A* por *B* é o conjunto de todos os pontos *c* tais que *B*, quando transladado por *x* fique contido em *A*. A dilatação e erosão são operações duais em relação à complementação e reflexão de conjuntos,ou seja:

$$(A \ominus B)^c = A^c \oplus \hat{B}) \tag{8}$$

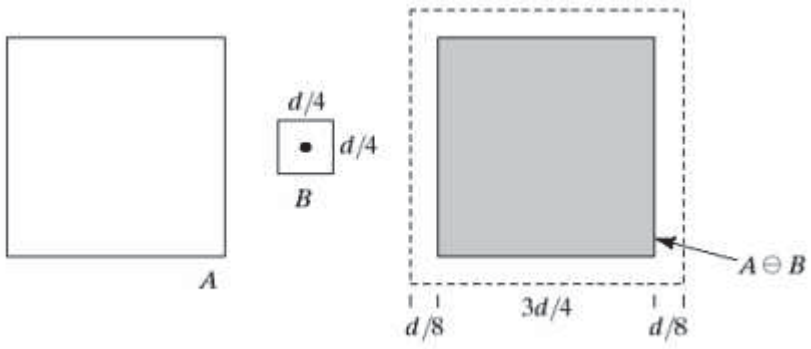


Figura 4 – Erosão do A pelo elemento B.

- Abertura e Fechamento

A abertura em geral suaviza o contorno de uma imagem, quebra istmos e estreitos e elimina proeminências delgadas. O fechamento, por sua vez, funde pequenas quebras e alarga golfos estreitos, elimina pequenos orifícios e preenche orifícios no contorno.

A abertura de *A* por um elemento estruturante *B*,denotado $A \circ B$ e mostrado na figura 5, é dado por:

$$A \circ B = (A \ominus B) \oplus B \tag{9}$$

Ou seja, a abertura de *A* por *B* é simplesmente a erosão de *A* por *B* seguida de uma dilatação do resultado por *B*.

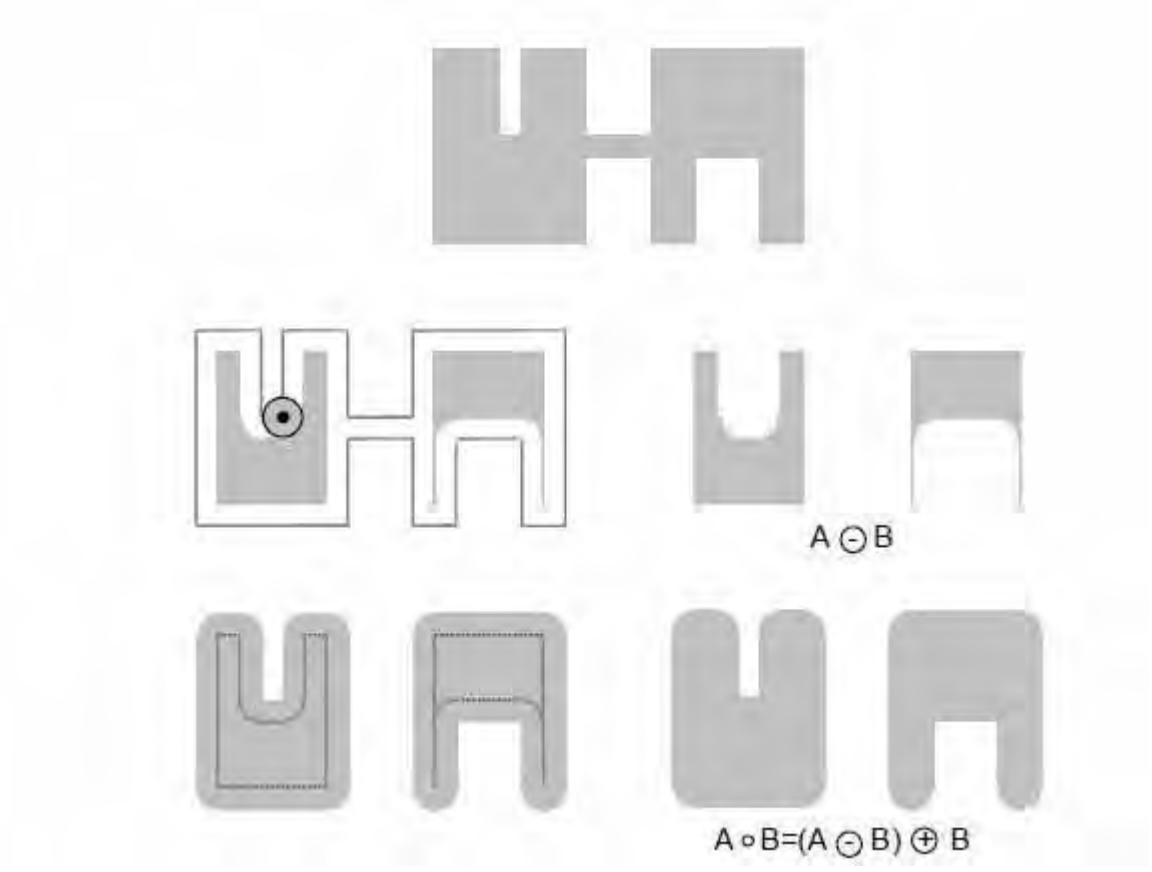


Figura 5 – Abertura do elemento A utilizando o elemento B.

O fechamento do conjunto *A* por *B*, denotado $A \cdot B$ e mostrado na figura 6, é dado por:

$$A \cdot B = (A \oplus B) \ominus B \tag{10}$$

Ou seja, é a dilatação de A por *B* seguida da erosão do mesmo por *B*.

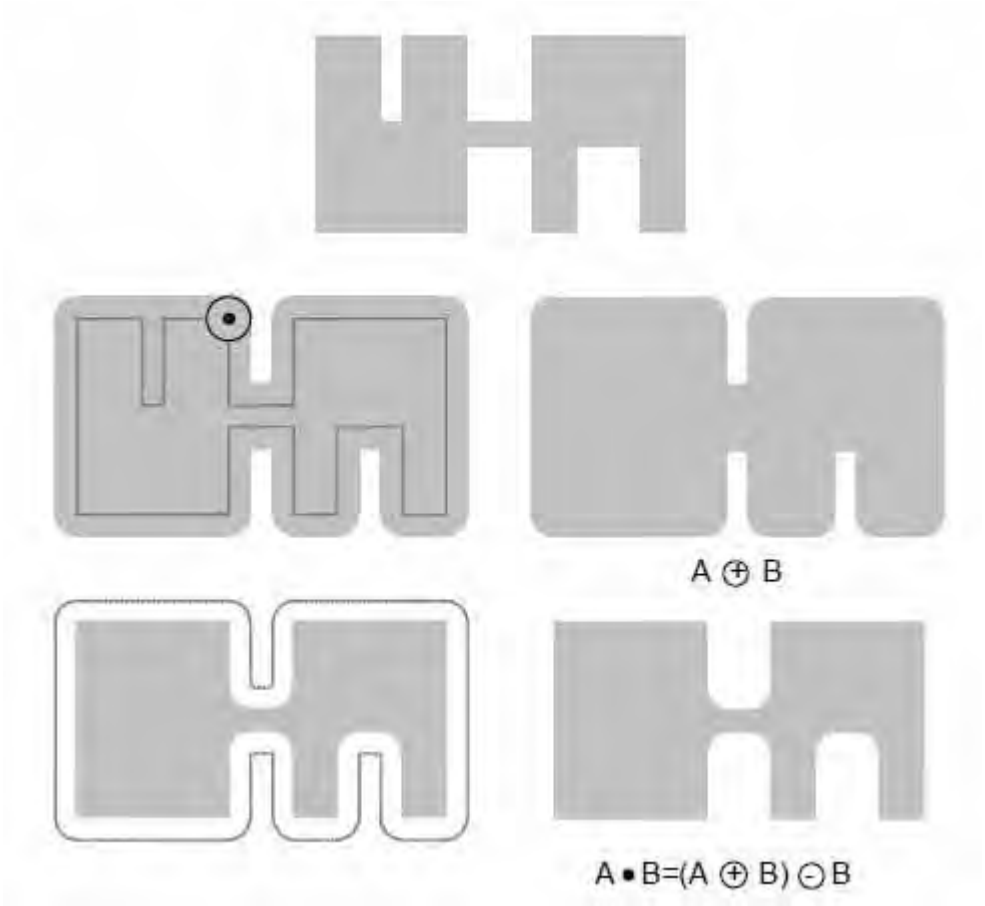


Figura 6 – Fechamento do elemento A utilizando o elemento B.

1.2.2 Limiarização

A técnica de limiarização (*thresholding*) consiste em separar regiões quando esta representa duas classes, o fundo e o objeto. Por produzir uma imagem binária como resultado essa técnica é também conhecida como binarização. Matematicamente a operação pode ser descrita de forma que uma imagem de entrada *I(x,y)* de *N* níveis de cinza produz uma saída em que valores maiores que *T* serão representados por "1" e valores de pixel menores que um serão "0":

$$l(x,y) = \begin{cases} 1, I(x,y) \leq T \\ 0, I(x,y) > T \end{cases} \quad (11)$$

O caso mais simples de limiarização é quando há um histograma bimodal. Nessa situação o histograma da imagem apresenta dois picos separados por um vale entre eles, mostrado na figura 7.

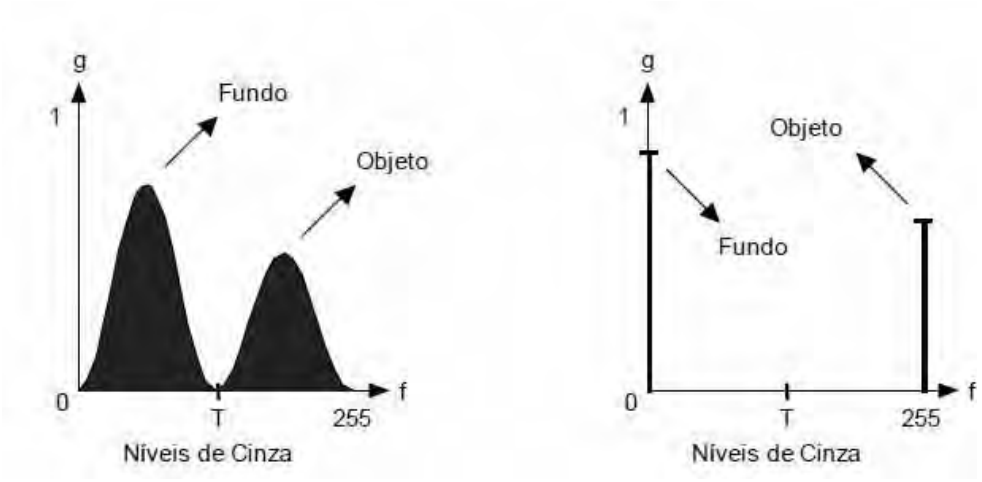


Figura 7 – Limiarização quando há um histograma bimodal.

A localização do vale entre os picos de histogramas bimodais geralmente não é muito simples devido à presença de ruídos nas imagens – próximo do vale podem existir diversos mínimos locais, não sendo trivial encontrar automaticamente o melhor limiar.

Para resolver este problema, é freqüentemente utilizado um filtro para suavizar a imagem antes de proceder com a limiarização, por exemplo, pode ser utilizado um filtro de médias ou algoritmos de limirização mais complexos como o método de Otsu.

1.2.3 Processamento de imagens coloridas

Existem diversos tipos de modelos de cores que são utilizadas no processamento de imagens. A seguir são descritos os espaços de cores mais conhecidos.

- RGB (*Red Green Blue*)

Esse sistema de cores é representado por três componentes: Vermelho (*Red*), Verde (*Green*) e Azul (*Blue*).Podemos representar esse modelo utilizando um cubo tridimensional onde com componentes de cada cor em cada eixo mostrada na figura 8.

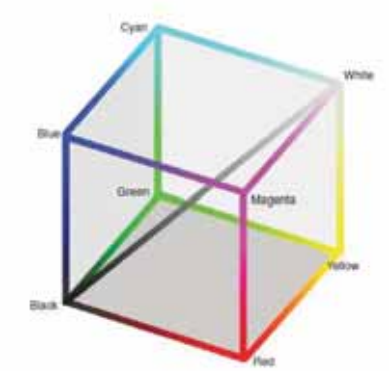


Figura 8 – Representação do modelo de cores RBG.

De acordo com Schnack (2005), este modelo simplifica o design de sistemas gráficos, porém não é o ideal para todas as aplicações. Isto se deve ao fato de os componentes vermelho, verde e azul serem amplamente correlacionados, o que dificulta a execução de certos algoritmos de processamento. Diversas técnicas de processamento, como a equalização do histograma trabalha apenas na informação da intensidade de imagem, portanto são mais facilmente implementados utilizando um modelo que já contenha essa informação como, por exemplo, o HSB.

- Modelo HSV (*Hue Saturation Value*)

Ao contrário do RGB, o espaço HSV (Matiz Saturação e brilho) é perceptualmente uniforme. Neste espaço, as cores são formadas também por 3 valores:

- Matiz - Verifica o tipo de cor, podendo ser vermelho, amarelo e azul. Atinge valores de 0 a 360, mas algumas aplicações, esse valor é normalizado de 0 a 100%.

- Saturação – Também chamado de pureza. Quanto menor seu valor, a imagem aparecerá com mais tons de cinza. Quanto maior esse valor a imagem será mais “pura”.

- Brilho – Define o brilho da cor.

O espaço de cores HSV trabalha com a representação de um cone, no qual o ângulo *H* (*Hue*) com em relação ao eixo horizontal representa a matiz da cor desejada. A distância perpendicular do centro até a borda determina a saturação e a distância vertical, determina a luminosidade *V*. A representação é mostrada na figura 9.

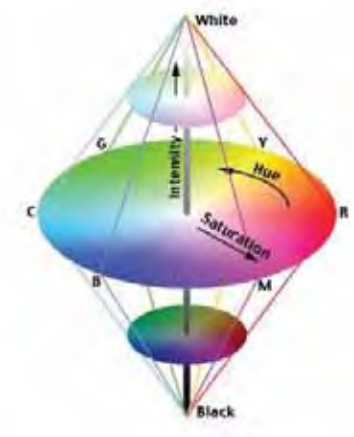


Figura 9 – Representação do modelo de cores HSV, relacionando os elementos RBG.

- Conversão do sistema RBG para HSV

Conforme descrito por Begun (2006), a conversão do modelo RGB para HSV é realizada a partir das seguintes equações (assumindo MAX como o valor máximo da tripla (RBG)e MIN como o menor valor):

$$H = \begin{cases} 60 \times \frac{G-B}{MAX-MIN} + 0, & se\ MAx = R\ e\ G \geq B \\ 60 \times \frac{G-B}{MAX-MIN} + 360, & se\ MAx = R\ e\ G \geq B \\ 60 \times \frac{B-R}{MAX-MIN} + 120, & se\ MAx = G \\ 60 \times \frac{R-G}{MAX-MIN} + 240, & se\ MAx = B \end{cases} \quad (12)$$

$$S = \frac{MAX-MIN}{MAX} \quad (13)$$

$$B = Max \quad (14)$$

2 MÓDULOS DE UM SISTEMA DE MONITORAMENTO

Geralmente, sistemas de vigilância automática são divididos em 3 módulos que envolvem 3 áreas citadas por Aggaward and Cai (1999): Detecção de movimento, rastreamento de pessoas e classificação.

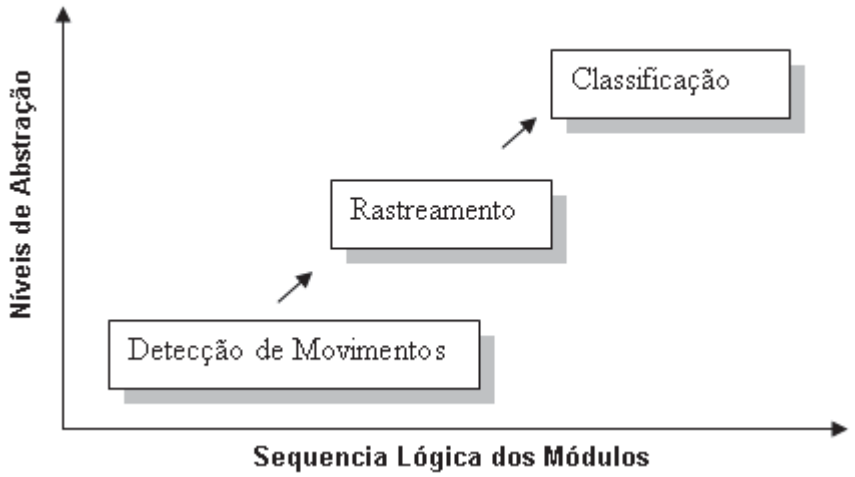


Figura 10 – Módulos de um sistema de monitoramento.

O primeiro módulo consiste na segmentação da imagem a fim de encontrar os objetos em movimento da cena, em outras palavras, significa separar o *background* (Plano de Fundo) do *foreground* (Objetos de interesse). A partir do *foreground* obtido é necessário fazer uma limiarização da imagem a fim de obter uma imagem binarizada para simplificar a análise dos objetos encontrados.

O módulo de rastreamento tem a importância de estabelecer as relações temporais entre os objetos alvo e a sequência de frames, isto é, identificar e localizar os objetos na transição de um quadro para o outro.

Uma vez identificado os objetos na cena, a última etapa consiste em classificar as ações que ocorrem no sistema. Neste sistema as atividades a serem determinadas para classificar cada objeto são mostradas na tabela abaixo:

Tabela 1 – Detecção de eventos.

Ação	Descrição	Saída
Contagem de pessoas	contar o número de pessoas que passam em um determinado local pré-estabelecido	Número de pessoas
Objeto Abandonado	detectar se um objeto foi abandonado	Alarme : Objeto Abandonado
Alerta	detectar se houver uma confusão na cena, bem como brigas, atos de vandalismo e tumultos	Alarme : Alerta

A seguir são mostrados os principais métodos estudados para implementação de cada módulo.

2.1 Segmentação

2.1.1 Subtração de Plano de Fundo

Identificar objetos que se movem em um vídeo é uma tarefa fundamental na maioria dos sistemas de Visão Computacional. Existem muitos desafios para conseguir separar os objetos em movimento do cenário. Inicialmente, o algoritmo deve detectar a variação de iluminação do ambiente.Outra etapa, na maioria dos casos, é detectar objetos de fundo não-estático, como exemplo, arbustos, árvores, sombras e chuva. E por fim, ele deve detectar objetos que em certo momento ficam imóveis, como exemplo pessoas que permaneçam imóveis durante um período de tempo.

Já foram desenvolvidos diversos algoritmos para subtração de plano de fundo, que geralmente são separados em 2 grupos distintos, os algoritmos recursivos e não recursivos.

- Técnicas não-recursivas.

As técnicas não-recursivas utilizam um sistema de armazenamento de dados de um número determinado de quadros, assim é possível estimar o plano de fundo baseando-se na variação de cada pixel durante um intervalo pré-determinado. A desvantagem dessa técnica é que ela utiliza um grande espaço na memória.

A técnica ***Median Filter*** é um dos algoritmos não-recursivos mais utilizados para determinação de plano de fundo. O plano de fundo é identificado como uma média de cada

pixel armazenado no *buffer*. Neste algoritmo é pressuposto que o fundo esteja mais do que metade dos quadros armazenados.

O Modelo **Não-paramétrico** (*Non-parametric Model*) é um filtro desenvolvido por Elgammal(2002) para estimar a função densidade de probabilidade de cada pixel utilizando uma estimativa de densidade em Kernel. Este filtro permitiu que o modelo de fundo se adaptasse mais rapidamente às mudanças no atual cenário de fundo.

. Apesar de apresentar bons resultados esta técnica possui um custo computacional muito grande.

- Técnicas recursivas.

As técnicas recursivas não mantêm um *buffer* para estimar o plano de fundo. Ele atualiza um modelo simples de plano de fundo baseando-se em cada quadro, portanto quadros que tem um passado distante não possuem efeito algum sobre a imagem atual de plano de fundo. Comparando-se com as técnicas não-recursivas, esse algoritmo não necessita de um grande armazenamento de dados, porém qualquer erro do modelo de plano de fundo pode durar por um longo período de tempo. Alguns modelos de filtros recursivos são descritos a seguir.

A técnica recursiva mais simples para modelar o plano de fundo é chamada de **Subtração de Plano de Fundo (*Background Subtraction*),** que consiste em utilizar uma imagem estática do ambiente observada no momento em que não haja objetos se movimento pelo mesmo. A partir deste ponto, de cada quadro de vídeo subtrai-se a imagem estática. Este modelo é denominado não-adaptativo, pois não é tolerante a possíveis mudanças no plano de fundo. É dado por:

$$|I_t(i,j) - B_t(i,j)| > \tau \tag{15}$$

Em que $I_t(i,j)$ é o quadro corrente, $B_t(i,j)$ é a imagem correspondente ao modelo de plano de fundo e t é uma constante definida experimentalmente.

Outra técnica também muito simples é a de **Modelos de diferença Temporal *(Frame Differencing*)**, que utiliza a diferença entre os quadros consecutivos comparado com um nível determinado de limiar para determinar o plano de fundo:

$$|frame_i - frame_{i-1}| > T_s \tag{16}$$

A principal desvantagem desse filtro é que há uma dificuldade de identificar os pixels quando há um interior largo, portanto é capaz somente de identificar as bordas. Outra desvantagem é o fato que se o objeto fica estático durante o período de um quadro ele não é mais detectado.

Devido ao sucesso do filtro não-recursivo *Median Filter*, McFarlane e Scholfield (1995) propuseram ***Approximated Median Filter*** para estimar um filtro mediano recursivo. Essa técnica geralmente é muito utilizada para monitoramento de tráfego. O filtro funciona da seguinte maneira: Se um pixel do quadro de entrada possuir um valor maior que o atual pixel de *background*, o valor do *background* será acrescentado em 1. Caso contrário, ele será decrementado de 1.

O ***filtro de Kalman*** é muito utilizado quando se deseja identificar planos de fundo que possuem ruído Gaussiano. Existem muitas variações deste filtro, a mais simples utiliza somente a variação de luminosidade, outras utilizam a intensidade à variação temporal e a variação espacial. Este filtro será discutido posteriormente.

O filtro ***Mixture os Gaussians (MoG)*** assume que as observações de um pixel da imagem pode ser modelado por uma mistura de *k* gaussianas (*k* é um valor normalmente de 3 a 5). Dessa forma é possível lidar com múltiplos modelos de fundo, podendo lidar com objetos de fundo não estático.

Similar ao filtro não-parametrico, descrito anteriormente, MoG mantém uma função densidade para cada pixel. Neste modelo proposto por Stauffer e Grimson, (1999), o valor de cada pixel representa a soma das integrais da função refletância das superfícies presentes na cena, ao longo de todo o espectro das fontes de iluminação (Salvador; Cavallaro; Ebrahimi, 2004). Desse jeito o modelo de fundo é construído dinamicamente.

Neste modelo, os valores de cada pixel (por exemplo, escalares para valores em cinza e vetores para imagens coloridas) durante o tempo é considerado como “pixel de processo” e a história recente de cada pixel $\{X_1, ..., X_t\}$, é modelado como a mistura de *K* distribuições gaussianas. A probabilidade de observação do pixel em questão vem de:

$$P(X_t) = \sum_{i=1}^{K} \varpi_{i,t} * \eta(X_t, \mu_{i,t}, \Sigma_{i,t}) \qquad (17)$$

onde $W_{i,t}$ é uma estimativa do peso da i-ésima Gaussiana$(G_{i,t})$ na mistura no tempo *t*, $u_{i,t}$ é o valor médio de $G_{i,t}$ e $E_{i,t}$ é a matriz de covariância de $G_{i,t}$ e *n* é a função densidade de probabilidade Gaussiana:

$$\eta(X_t, \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(X_t-\mu_t)^T\Sigma^{-1}(X_t-\mu_t)} \quad (18)$$

quanto maior o valor de *K*, maior será o custo computacional utilizado.

2.1.2 Remoção de Sombras

Um dos maiores desafios da separação do plano de fundo é a detecção de sombras. A presença de sombra causa problemas como distorção de formas dos objetos, união de objetos alvo e perda de objetos.

A dificuldade da remoção de sombras é causada pois essas regiões geralmente são identificadas como objetos (*foreground)* pelas técnicas de subtração de fundo e possuem o movimento semelhante aos objetos que devem ser identificados e analisados.

Cucchiara et al. (2001) desenvolveu um método de remoção de sombras utilizando o espaço cromático em que as componentes de iluminação e cromaticidade estão separadas (HSV). Pode-se notar que áreas sombreadas possuem os componentes H e S quase que inalterados dos planos de fundo enquanto o valor de V deve ser menor na área sombreada.

Isso ocorre, pois sombras são causadas por uma obstrução parcial ou total da luz incidente em uma superfície devido ao objeto que intercepta a luz. Este princípio pode se determinar a seguinte equação 19.

$$SP_k(x,y) = \begin{cases} 1 & se\ \alpha \leq \frac{I_k^V(x,y)}{B_k^V(x,y)} \leq \beta\ e \\ \left|B_k^S(x,y) - I_k^S(x,y) \leq \tau_S\right| e\ |B_k^H(x,y) - I_k^H(x,y) \leq \tau_H| & \\ 0 & caso\ contrário \end{cases} \quad (19)$$

Sendo *H, V e S* as componentes do espaço de cores, *I e B* são o quadro de e β são limiares entre 0 e 1 que definem quão sensível à sombra será a função $SP_k(x,y)$. Caso 1 o pixel é considerado sombra, caso contrário será classificado como não sombra.

2.1.3 Regiões conectadas

Após identificar as regiões de plano de fundo e os objetos da cena, deve-se detectar quais regiões estarão conectadas. Para isso serão atribuídos identificadores que são chamados *Blobs*. Esses objetos serão identificados com um quadrado de área mínima com especificações de posição e dimensão. Posteriormente, são computadas características que permitem a identificação desses objetos alvo em quadros posteriores. Na figura 11 está identificado o objeto alvo através de seu *Blob*.

Figura 11 – Objeto alvo identificado a partir de seu *blob.*

2.1.4 Ruídos

Mesmo após a remoção de sombras, a imagem resultante ainda pode conter falsos positivos que são gerados por motivos de ruído. Geralmente esse ruído é incorporado ao sistema devido ao processo de aquisição de imagens, bem como as falhas de segmentação do próprio processo.

Falsos positivos provenientes de ruído gaussiano geralmente podem ser removidos simplesmente utilizando operações primitivas de processsamento de imagens como filtros de mediana, erosão e dilatação (Gonzalez and Woods, 2002).

Na maioria dos casos, os ruídos geram falsos positivos de tamanho bem inferior aos objetos alvo, portanto uma solução simples seria eliminar as regiões conectadas de pequenas proporções. Essa operação pode ser feita após a tarefa de detecção dos *Blobs.*

Um outro problema que geralmente ocorre é quando regiões como pés e braços aparecem desconectadas após o processo de segmentação. Siebel (2003) aplicou um algoritmo de junção das regiões muito próximas para contornar esse problema, porém além de ser um

processo com um custo computacional um pouco elevado, não obteve muito sucesso. Dedeoglu (2004) aplicou com êxito sucessivas operações de dilatação e erosão para eliminar falhas de segmentação devido a ruídos, fechar regiões abertas e unir partes separadas.

Apesar da variedade de estratégias adotadas para minimizar ruído de segmentação, a utilização das mesmas não trará, necessariamente, bons resultados em qualquer situação. A eliminação de regiões pequenas, por exemplo, depende da definição de um limiar para exclusão de tais regiões. Um certo limiar poderá eliminar grande parte dos falsos positivos de imagens de ambiente interno, mas poderá também eliminar objetos alvo verdadeiros em cenas de ambientes externos, em que a tomada da cena geralmente abrange uma área maior. Algoritmos de junção, por sua vez podem também atuar negativamente, pois podem realizar a junção de partes de regiões diferentes.

2.1.5 Rastreamento

Como já citado anteriormente o rastreamento consiste em observar um objeto através do tempo, ou seja, nos quadros consecutivos. As técnicas mais utilizadas podem ser divididas em dois grupos. O primeiro consiste em rastrear algum objeto utilizando suas características. Citaremos três métodos neste trabalho *Sift (Scale Invariant Feature Transform*), *Mean Shift* e método Amer. Porém estes métodos não apresentam uma boa eficiência em se tratando de custo computacional, além de ser uma técnica falha visto que quando algo se move numa imagem ele tende a mudar seu formato e aspecto, como exemplo uma pessoa que vira de costas para câmera.

O outro grupo rastreia um objeto através do seu movimento é muito mais utilizado por ter um custo computacional mais baixo.Entre essas, o método mais utilizado é baseado no filtro de Kalman, discutido em detalhes a seguir.

- Método de Amer

Apresentado por Amer (2005) essa técnica consiste no rastreamento de objetos baseando-se no casamento de características de tamanho, forma e deslocamento. As características utilizadas estão detalhadas na tabela 1. Este método consiste em 3 etapas: Extração de características, casamento de características e monitoramento de características. A comparação entre as caracteríscas normalmente é feita utilizando equações de restrição, distância e similaridade.

Tabela 2 – Características utilizadas por Amer(2005).

Nome	**Definição**	**Descrição**
Altura	$\omega = y_{max} - y_{min}$	y_{max}:maior valor de y assumindo por um pixel do objeto; y_{min}: menor valor de y assumido por um pixel do objeto.
Largura	$h = x_{max} - x_{min}$	x_{max}:maior valor de x assumido por um pixel do objeto; x_{min}: menor valor de x assumido por um pixel do objeto
Área	a	Número de pixels do objeto
Perímetro	p	Número de pixels da borda do objeto
Retângulo mínimo	$MBB = (x_{min}, x_{max}, y_{min}, y_{max})$	Idem descrições de altura e largura
Relação de extensão	$e = \frac{h}{w}\ se\ h < w\ ou\ e = \frac{h}{w}\ se\ w < h$	*h:* altura; *w:* largura
Compactação	$c = \frac{a}{hw}$	*A:* área; *h:* altura; *w:* largura
Irregularidade	$r = \frac{p^2}{4\pi a}$	*P:* perímetro; *a:* área

Seguindo o mesmo raciocínio, outras técnicas foram apresentadas por outros autores como Dedeoglu(2004) e humphreys(2004).

- SIFT (*Scale-Invariant Feature Transform*)

Publicada por David Lowe(1999), SIFT é um método em visão computacional que permite extrair uma coleção de caracterícticas locais de uma imagem. Estas características são invariantes à alteração de escala, translação, rotação, parcialmente robusto a mudanças de luminosidade e robusto a distorções geométricas locais.

Os pontos chaves dos objetos em questão inicialmente são extraídos de uma imagem de referência e armazenados em um banco de dados. Um objeto é reconhecido em uma imagem comparando-se cada característica da imagem em questão e achando um candidato que tenha relações com as características do banco de dados baseando-se em distâncias euclidianas de cada vetor.

Localizações de pontos chave são definidos através de máximos e mínimos do resultado de uma função de diferenciação gaussiana.

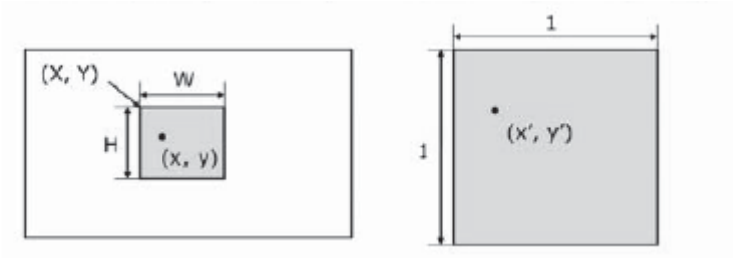


Figura 12 – Translação das coordenadas de um pixel.

As coordenadas das características Sift são transformadas em uma posição relativa da característica a ser utilizada para uma média da localização e tamanho do objeto selecionado. Como mostrado na figura 12, o retângulo é redimensionado para ficar com comprimento igual a 1. A relação entre a coordenada *(x',y')* e a coordenada do pixel *(x,y)* pode ser escrita como:

(20)

(21)

- *Mean Shift*

Introduzido por Fukanaga e Hostetler (1975), *Mean Shift* é um método não-paramétrico de procura. Incorporado para rastreamento por Comaniciu e Meer (2003) é um método que otimiza uma similaridade medida entre um modelo alvo e o candidato entre dois quadros consecutivos.

Primeiramente uma característica espacial deve ser definida para caracterizar o alvo. Essa característica pode ser o espaço de cores, que geralmente é utilizado RBG.

O objeto modelo é representado por uma função densidade de probabilidade. O modelo do objeto é chamado de objeto alvo e o candidato modelo e comparado no quadro consecutivo.

- Filtro de Kalman

Criado por Rudolf Kalman em 1960 este filtro consiste em coletar medições ao longo do tempo, que geralmente são contaminadas com incertezas e ruídos e gerar resultados que tendam a se aproximar dos valores reais, ou seja, este filtro permite produzir estimativas do estado de um sistema de forma a minimizar o quadrado da média do erro [Welch, 1965].

Trata-se da solução ótima para o seguimento, caso sejam satisfeitas as seguintes condições: O ruído deve ter uma distribuição Gaussiana de parâmetros conhecidos e a transição de estados deve ser representada pelo modelo de sistemas lineares.

O filtro de Kalman produz estimativas dos valores reais de grandezas medidas e valores associados predizendo um valor, estimando a incerteza do valor predito e calculando uma média ponderada entre o valor predito e o valor medido. O peso maior é dado ao valor de menor incerteza. As estimativas geradas pelo método tendem a estar mais próxima dos valores reais que as medidas originais, pois a média ponderada apresenta uma melhor estimativa de incerteza que ambos os valores utilizados no seu cálculo.

A sua habilidade para incorporar os efeitos de erros e sua estrutura computacional fez com que o filtro de Kalman tivesse um amplo campo de aplicações, especialmente no que se refere à análise de trajetórias em visão computacional.

O Filtro de Kalman se destina ao problema geral de tentar estimar o estado de um tempo discreto em um processo controlado que é governado pela equação diferencial estocástica.

Este modelo estima que o estado real encontrado *k* é obtido através do estado anterior *(k-1).*

$$x_k = F_k x_{k-1} + B_k u_k + w_k \tag{22}$$

Onde F_k é o modelo de transição de estado, B_k é o modelo de entradas de controle e w_k é o ruído do processo, assumindo como sendo uma amostra de uma distribuição normal multivariada de média zero e covariância Q_k, onde

$$w_k \sim N(0, Q_k) \tag{23}$$

No tempo *k*, uma medição z_k do estado real x_k é feita de acordo com:

$$z_k = H_k x_k + v_k \tag{24}$$

Onde, H_k é o modelo de observação, v_k é o ruído da observação:

$$v_k \sim N(0, R_k) \tag{25}$$

Na prática, o processo de covariância do ruído Q_k e as medidas de covariância do ruído, matriz R_k podem mudar, em cada medição ou temporalmente, contudo assume-se que são constantes.

O estimador do vetor de estado no instante *k+1*, construído no instante *k*, será representada na forma *x(k+1| k),* cuja distribuição, por ser gaussiana, é definida por sua média e matriz de covariância de dados, de acordo com Anderson e Moore (1979), e pelas equações 26 e 27.

Este estimador é calculado através da transformação linear, dada pela equação 26, da distribuição do vetor de estado *x(k),* das entradas *u(k)* e do ruído do processo *v(k).* Este primeiro passo do filtro de Kalman é conhecido como predição, pois é construído um estimador que prevê a distribuição do vetor de estado um passo a frente.

$$\hat{x}(k+1|k) = F(k)\hat{x}(k|k) + G(k)u(k) \tag{26}$$

$$P(k+1|k) = F(k)P(k|k)F(k)^T + V(k) \tag{27}$$

Onde *(k+1|k)* é o instante *k+1* construído no instante *k*, $\hat{x}$ é o vetor de médias da distribuição *x, P* é a matriz de covariâncias da distribuição *x e V* é a matriz covariância da distribuição *v*.

No instante *k* a observação esperada do vetor de estado do instante *k=1* é dada pelo estimador *z(k+1|k).* Este estimador é obtido através da transformação linear dada pela eq. 22. O estimador é uma transformação linear de uma distribuição gaussiana, logo é também uma gaussiana.

No instante *k+1* é obtida, através dos sensores uma observação do vetor de estado, *x(k+1),* observação representada por *z(k+1).* Este é gaussiano, por ser uma combinação linear de gaussianas, que de acordo com Anderson e Moore(1979) é definido por seu vetor de médias e matriz de covariância, respectivamente:

$$\Delta\hat{z}(k+1) = z(k+1) - H(k+1)\hat{x}(k+1|k) \quad (28)$$

$$S(k+1) = H(k+1)P(k+1|k)H(k+1)^T + W(k+1) \quad (29)$$

Onde $\Delta\hat{z}$ é o vetor de médias da distribuição Δz, z é o vetor de valores observados na medição e W é a matriz de covariância da distribuição w.

A existência da nova informação, gerada pela observação z(k+1|k), permite que o estimador do vetor de estado x(k+1) seja atualizado. Essa atualização é feita através da adição de um termo de correção ao estimador x(k+1|k). O termo de correção é dado por um ganho K vezes a diferença das observações, Δz(k+1). O ganho K pode ser visto como um peso que leva em conta a relação entre a acurácia da estimativa *x(k+1|k)* e o ruído da medida. Quanto maior o ganho *K*, maior é a possibilidade que o observado está mais próximo do correto que a estimativa propagada. Este passo é conhecido como atualização, pois o estimador para o vetor estado *x(k+1),* feito no instante *k*, é atualizado no instante *k+1*. O novo estimador *(k+1|k+1)* é gaussiano e descrito por seu vetor de médias e matriz de covariância, respectivamente:

$$\hat{x}(k+1|k+1) = \hat{x}(k+1|k) + K(k+1)\Delta\hat{z}(k+1) \quad (30)$$

$$P(k+1|k+1) = P(k+1|k) - K(k+1)H(k+1)P(k+1|k) \quad (31)$$

Onde (k+1|k+1) é o instante k+1 construído no instante k+1 e K é a matriz de ganhos.

O ganho K que minimiza o erro quadrático na estimação do estado é também conhecido como o ganho ótimo de Kalman.

$$K(k+1) = P(k+1|k)H(k+1)^T P(k+1)^{-1} \quad (32)$$

As equações (28) a (30) são as equações do filtro Kalman, que fornecem um estimador ótimo para o vetor de estado, quando a hipótese de que os ruídos são gaussianos é mantida.

As equações se mantêm válidas para o caso onde as distribuições *x(0), v(t), w(t)* não forem conjuntamente gaussianas. Entretanto, dependendo da distribuição de *v(t) e w(t),* os dois primeiros momentos podem não ser suficientes para descrever a distribuição de probabilidade do vetor de estados *x(k).*

A hipótese de que os ruídos do processo e da medida serem brancos pode ser relaxada. Considerando estes processos correlacionados, os mesmos podem ser modelados como um sistema linear de dimensão finita, onde a entrada é um ruído branco. Então, o vetor de estado do filtro de Kalman será composto pelos sinais originais do modelo e pelos do modelo de ruído.

O filtro de Kalman estima um processo usando uma forma de controle por realimentação: o filtro estima o estado do processo em dado instante e obtém então o *feedback* na forma de medidas (ruidosas). Assim, as equações para o filtro de Kalman caem em dois grupos: atualização das equações de tempo e a atualização das equações de medição. As equações de atualização do tempo são responsáveis para projetar com antecedência (no tempo) as estimativas da covariância do estado atual e do erro, para obter as estimativas anteriores para a próxima etapa. As equações de atualização da medida são responsáveis pelo *feedback*, isto é incorpora-se uma nova medida na estimativa anterior para obter posterior com uma estimativa melhorada.

As equações de atualização do tempo podem também ser pensadas como equações de predição, quando as equações de atualização da medida podem ser pensadas como equações de correção. Certamente o algoritmo final de estimação assemelha-se àquele de um algoritmo do preditor-corretor para resolver problemas numéricos como mostrado abaixo na figura 13.

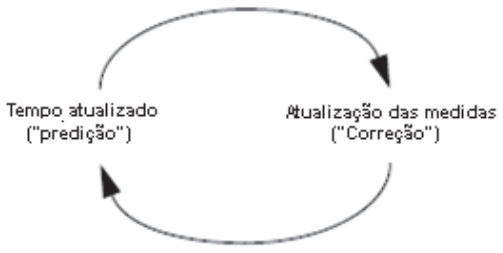


Figura 13 – Ciclo de transição do filtro Kalman.

2.2 Classificação

Existem inúmeras técnicas utilizadas na classificação das ações que ocorrem numa seqüência de quadros, geralmente encontradas na área denominada *Human Activity Recognition* (Reconhecimento de atividades humanas).

A etapa de detecções de ações é uma área ainda imatura, apesar ser estudas a um tempo relativamente grande. A prova disso é a existência de inúmeras técnicas e estudos com diferentes abordagens sobre o assunto sem chegar a um consenso a respeito do assunto.

- Técnicas clássicas

As técnicas mais bem sucedidas de reconhecimento de ações foram desenvolvidas por Polana e Nelson(1994) e Yamat(1992) utilizando técnicas de reconhecimento de padrões utilizando as Cadeias Escondidas de Markov.

Esta técnica é utilizada com sucesso em problemas como reconhecimento de voz e reconhecimento de escritas, encontrando-se, também em estudos que abrangem diversas áreas para utilização das Hmm(*Hidden Markov Models*- Cadeias ocultas de Markov).

As Cadeias de Markov consistem em uma técnica estocástica baseada nos estudos de Andrey Markov. A propriedade de markov pode ser interpretada do seguinte modo: a probabilidade condicional de qualquer estado futuro, conhecido os estados do presente e do passado, é independente dos estados do passado, ou seja, para predizer o futuro só depende do presente.

Em outras palavras, as cadeias de Markov são uma série finita de estados em que a transição de um estado para o próximo depende somente do estado atual.

As cadeias escondidas de Markov são processos em que nem todos os estados são conhecidos. Mas pode-se dizer que cada estado corresponde a uma observável no sistema. Então podemos dizer que Hmm são usados na modelagem de processos Markovianos que geram observáveis de forma indireta, em função da transição de estados das cadeias de Markov.

Matematicamente, as Hmm é uma tripla $\lambda = \{A, B, \pi\}$. Onde A, b,π são matrizes que representam a probabilidade de transição entre estados (A), probabilidades dos símbolos de saída (B) e a matriz inicial de probabilidades (π).A figura 14 mostra a representacao das Hmms.

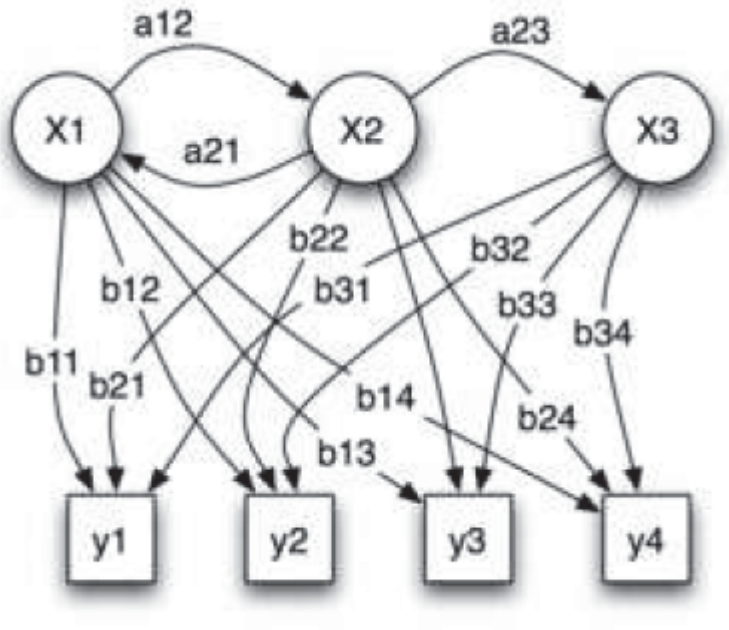


Figura 14 – Representação da transição de estados de uma Hmm.

Para detecção e classificações das ações utilizando Hmm á necessário a fase de treinamento das Hmm. Alguns autores utillizaram vetores de características para determinar as ações como no caso de Yamato et al. (1992). Também encontram-se estudos em que foi implementado um pseudo algoritmo 2d para realização da etapa de treinamento e classificacação. Em outros casos são utilizadas transformadas discreta de cossenos antes de alimentar o sistema.

- Simples Heurísticas

A técnica de simples heurísticas basea-se na observação dos acontecimentos afim de determinar uma solução simples e direta para o problema em questão. Neste trabalho foram utilizadas características espaciais e temporais para definir a situação de cada objeto em cena. Primeiramente, foram definidas características intuitivas para distinguir diferentes atividades na cena, conforme demonstra tabela 3.

Tabela 3 – Características atribuídas a cada atividade de um blob.

Id	Atividade	Descrição
1	Inativo	Objeto estático na cena. Não possui velocidade e não há movimento no objeto.
2	Parado	Uma pessoa parada na cena, porém que possui pequenos movimentos
3	Andando	Apresenta pouco movimento e uma velocidade baixa.
4	Correndo	Apresenta uma maior velocidade de deslocamento.
5	Lutanto	Uma baixa velocidade com movimentos bruscos.

A primeira característica a ser determinada, portanto, é a velocidade de cada blob. A velocidade é determina pela derivada discreta da posição de cada blob dado pelo seu centróide.

$$\vec{p}(t) = (X(t), Y(t)) \tag{33}$$

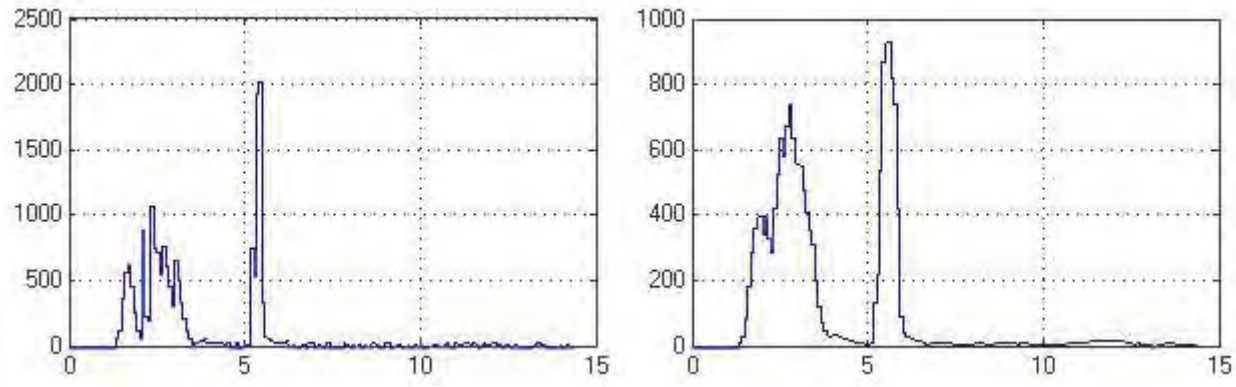
$$\vec{v}(t) = \frac{d\vec{p}}{dt} = (v_x(t), v_y(t) \cong (X(t) - X(t-1), Y(t) - Y(t-1)) \tag{34}$$

$$v(t) = \|\vec{v}(t)\| = \sqrt{v_x^2(t) + v_y^2(t)} \tag{35}$$

Onde, *X(t)* é a posição horizontal e *Y(t)* a vertical no instante *t*, *p(t)* é o vetor posição,*v(t)* é a velocidade. Devido à derivada discreta ser de dois pontos consecutivos ela possui ruídos, para solucionar este problema um filtro de médias simples foi introduzido. Os resultados podem ser observados no gráfico1 onde utilizou-se T=5.

$$\overline{v_t}(t) = \frac{1}{T}\sum_{i=t-T+1}^{t} v(i) \tag{36}$$

Onde T é o número total de quadros em que os vetores velocidades serão somados. A figura 15 mostra o módulo da velocidade de um determinado blob durante o tempo.



(a) (b)

Figura 15 – Velocidade de um Blob durante um período de tempo. (a) sem o filtro de médias, (b) utilizando um filtro de médias

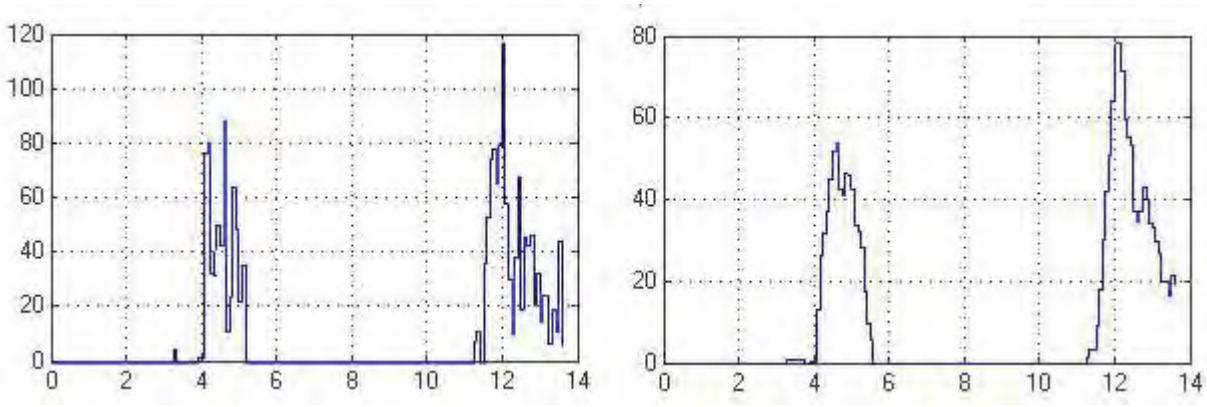
O outro grupo de características a ser determinado é o fluxo óptico de cada objeto da cena. A fórmula a seguir mostra como foi definido o fluxo óptico de cada elemento da imagem.

$$\vec{F}(t,x,y) = (f_x(t,x,y), f_y(t,x,y), (x,y)\epsilon\, P(t)) \tag{37}$$

Novamente foi utilizado um filtro de médias para reduzir os ruídos incorporado ao sistema.

$$\vec{F}(t)_1 = \frac{1}{N}\sum_{(x,y)\epsilon P(t)} \vec{F}(t,x,y) \tag{38}$$

A figura 16 mostra a somatória de todos vetores de fluxo óptico de um determinado Blob.



(a) (b)

Figura 16 – Módulo da somatória dos vetores de fluxo óptico de um blob durante um período de tempo.(a)Sem filtro de médias, (b)utlizando um filtro de médias.

3 PROPOSTA DE IMPLEMENTAÇÃO E ANÁLISE DE DADOS

Após a revisão dos principais métodos utilizados em Visão Computacional para sistemas de vigilância, este capítulo inicialmente aborda o sistema e a base de dados utilizados. Em seguida serão expostas as técnicas utilizadas na implementação desse sistema.

3.1 Detalhes de implementação

Para facilitar a implementação do modelo, a ferramenta escolhida foi o software Simulink/Matlab. Dentro deste software é possível encontrar ferramentas e algoritmos específicos para Visão computacional e processamento de imagens, assim como também é possível encontrar uma extensa lista de ferramentas matemáticas que possibilitaram uma implementação rápida do modelo em questão.

A grande vantagem da utilização deste software é a possibilidade de simulação em tempo real, bem como a capacidade de gerar códigos em linguagens como VHDL , C-code e C++.

Devido à dificuldade de acesso aos vídeos de estações e bloqueios, a base de dados utilizadas nesse trabalho foi retirada da Internet. O projeto CAVIAR disponibiliza seus vídeos codificados no formato MPEG2 na resolução 384x288 pixels, capturados no salão de entrada do laboratório INRIA, na França. Dentre os diversos vídeos disponíveis é possível encontrar situações adversas como brigas e objetos abandonados.

3.2 Arquitetura e implementação

A figura 17 mostra a divisão dos módulos, a divisão do modelo apresentada no Simulink é mostrada no ANEXO A.

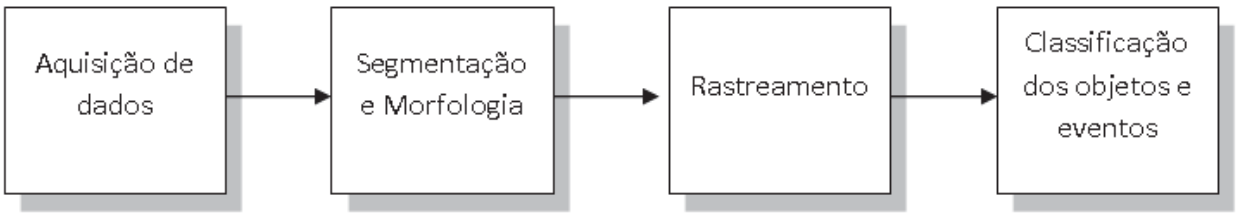


Figura 17 – Sequência de módulos utilizado na implementação deste sistema.

Na construção do sistema esses módulos foram divididos em blocos. O primeiro módulo de segmentação e morfologia é dividido em três partes: Subtração do plano de fundo; em seguida temos o bloco de rastreamento; e por fim, dois blocos constituem o módulo de classificação: Classificação de Objetos e Classificação dos eventos. A figura 18 mostra um diagrama com cada bloco e a relação de transição de dados.

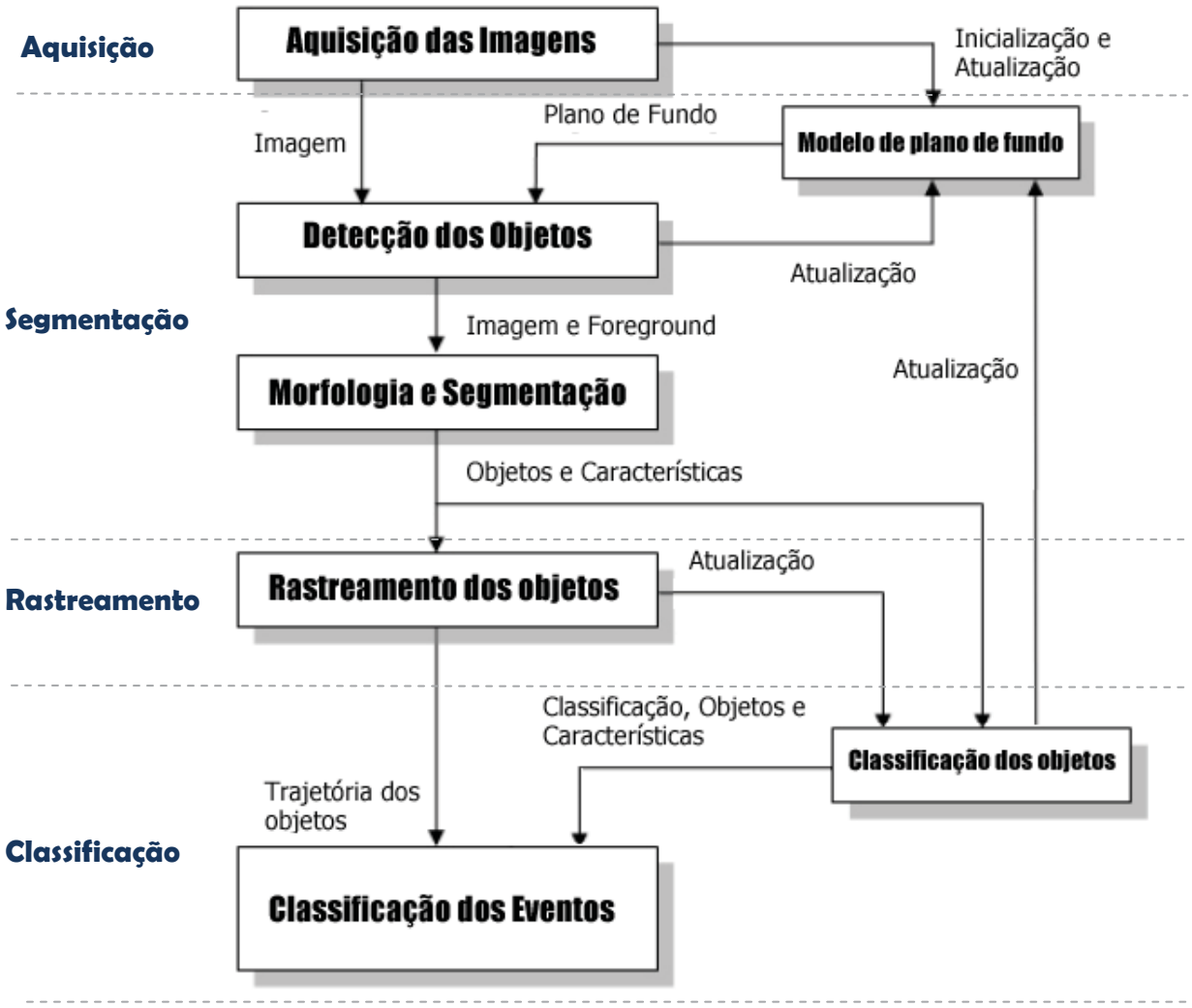


Figura 18 – Divisão em blocos do sistema com a representação de transição de dados entre cada bloco.

A seguir será detalhada a função específica de cada bloco, mostrando as técnicas utilizadas para o desenvolvimento do modelo, juntamente com a análise de dados obtidos com o modelo final.

3.2.1 Segmentação e Morfologia

A primeira etapa do processo consiste em segmentar a imagem a fim de separar os objetos de fundo (*Background*) dos objetos alvo (*Foreground*). Inicialmente foi utilizado um

filtro para converter a imagem em níveis de intensidade no formato *unsigned* 8, ou seja, a imagem foi convertida para níveis de cinza na escala de 0 a 255.

A técnica de subtração de plano de fundo escolhida foi *Approximated Median Filter*, pois por ser uma técnica recursiva e não-estatística, ela apresenta um desempenho computacional muito melhor comparado com algumas técnicas, como o *Mixture of Gaussians*. A figura 19 mostra os dois modelos de subtração de fundo.

(a) (b) (c)

Figura 19 – Comparação entre os filtros estimadores de plano de fundo Approximateds Median Filter e MoG. (a) Cena utilizada, (b) filtro MoG (c) Filtro Filtro Approximated Median Filter.

Figura 20 – Estimador de plano de fundo Approximated Median Filter. (a) Cena original, (b) Modelo do plano de fundo e (c) Foreground detectado após a etapa de segmentação.

Apesar deste filtro apresentar um melhor custo computacional, o filtro apresenta duas deficiências: a primeira é o fato de que não é possível reconhecer objetos de fundo não estáticos, entretando o sistema será desenvolvido para ambientes internos, onde não é encontrado esse tipo de objeto, fazendo-se desnecessário o uso de algoritmos mais complexos; a segunda deficiência deste filtro é que se um objeto permanece no mesmo local durante um certo período de tempo o objeto tende a se tornar parte do fundo gerando um falso negativo.

Para solucionar problema gerado por objetos que permanecem imóveis na cena, foi implementado um algoritmo de *feedback* no qual os dados obtidos no bloco de classificação selecionam os objetos que permanecem estáticos e impedem que seja atualizado no modelo de fundo.

Este módulo funciona da seguinte maneira: Inicialmente o objeto deve ser selecionado de modo a impedir que seja classificado como estático caso tenha surgido de uma separação. Isto torna-se necessário, pois caso o objeto tenha se incorcoporado ao modelo de fundo na inicialização do sistema, ou seja, o objeto permanece inalterado desde o 1 quadro e começa a se movimentar gerando um falso positivo. Em seguida o algoritmo seleciona os objetos que permanecem imóveis durante um certo período de tempo e impede que seja atualizado no modelo de plano de fundo. No ANEXO 5 é mostrado o algoritmo desenvolvido.

Após a detecção do plano de fundo foi utilizada a técnica de limiarização desenvolvida por Otsu, que consiste em determinar o fator de limirização a partir da bipartição do histograma de níveis de cinza. O histograma da imagem é tratado como uma função de densidade de probabilidade discreta.

$$p_r = \frac{n_q}{n} \quad q = 0,1,2, \ldots < l - 1 \tag{39}$$

Onde, *n* é o número total de pixels na imagem, n_q é o número de pixels com intensidade r_q e *l* é o número total dos possíveis níveis de intensidade da imagem.

Um valor *k* para a limiarização é escolhido tal que:

- C0 seja a classe de pixels com níveis entre [0,k-1] e
- C1 seja a classe de pixels com níveis entre [k,l-1]

O método Otsu escolhe *k* tal que maximize a variância inter-classes:

$$\sigma^2 = \varpi_0(\mu_0 - \mu_T)^2 + \varpi_1(\mu_1 - \mu_T)^2 \tag{40}$$

Onde,

$$\omega_0 = \sum_{q=0}^{k-1} p_q(r_q), \qquad \omega_1 = \sum_{q=k}^{k-1} p_q(r_q) \tag{41) e (42}$$

$$\mu_0 = \sum_{q=0}^{k-1} q p_q(r_q)/\omega_0, \qquad \mu_1 = \sum_{q=0}^{k-1} q p_q(r_q)/\omega_1 \tag{43) e (44}$$

$$\mu_t = \sum_{q=0}^{l-1} q p_q(r_q) \tag{45}$$

Utilizando o algoritmo de Otsu não foi necessário a implementação de técnicas para detectar sombras, pois o mesmo consegue selecionar valores de limiarização que eliminam as sombras na binarização da imagem. A desvantagem do uso deste filtro ocorre quando não há objetos se movendo na cena, pois o fator de limiarização fica muito próximo de zero. Como solução, foi determinado um valor mínimo de threshold evitando que os objetos rastreados fossem perdidos quando não houvesse objetos e movimento na cena.

A próxima etapa consiste em segmentar a imagem a fim de retirar os falsos positivos gerado por ruídos. Para eliminar os ruídos de origem gaussiana um filtro de médias é utilizado. Em seguida algumas operações morfológicas são acrescentadas, mostradas na figura 21 e 22.

Figura 21 – Sequência de operações morfológicas utilizadas(fechamento, erosão, dilatação).

Figura 22 – *Foreground* gerado após as operações morfologicas.

Por fim, a última etapa deste bloco consiste em solucionar o problema que ocorre com a separação de braços, pernas e cabeça devido às operações morfógicas. Um algoritmo mede

a distância entre os *blobs* próximos e utiliza um valor de limiarização para determinar se um *blob* deve-se juntar ao outro. Na figura 23b é possível notar uma segmentação do objeto, o algoritmo junta os blobs formando a figura 23c.

(a) (b) (c)

Figura 23 – Junção de blobs (a)Foreground após a segmentação e morfologia, (b) Blobs encontrados e (c) após a junção dos blobs próximos.

3.2.2 Rastreamento/ *Matching*

No rastreamento dos objetos em cena foi utilizado o filtro de Kalman, as equações de modelo de transição de estados são mostradas no Anexo B. Um algoritmo auxilar (*Matching*) foi utilizado para detectar os *blobs* e classificarem aqueles que apresentam junção ou separação detectando se houve oclusões em um determinado *blob*. Os eventos a serem classificados são mostrados na tabela 4 e o algoritmo de detecção de oclusões encontra-se no Anexo C:

Tabela 4 – Relação de oclusão de objetos na cena.

Id	Evento	Descrição
0	Objeto não identificado	Quando um objeto novo surge na cena e ainda não foi atribuido
1	Rastreando	O Objeto está sendo rastreado
2	Conjunta	Objeto se juntou a outro objeto e permaneceu durante um certo período de tempo
3	Dividida	O objeto se dividiu em dois objetos

Com esse algoritmo foi possível determinar todos os momentos em que houve separação ou junção nos vídeos utilizados.

Para análise de dados foram utilizados dois vídeos. Os resultados são mostrados nas tabelas abaixo.

Tabela 5 – Eventos ocorridos no vídeo 1.

Evento	Eventos ocorridos	Eventos detectados
Rastreado	2	2
Conjunta	0	0
Dividido	1	1

Tabela 6 – Eventos ocorridos no vídeo 2.

Evento	Eventos ocorridos	Eventos detectados
Rastreado	5	4
Conjunta	1	1
Dividido	1	1

O algoritmo foi capaz de detectar todos os eventos ocorridos, exceto no vídeo 2, em que um objeto permaneceu imóvel desde o primeiro quadro até o ultimo assim gerou um falso positivo, tornando-se parte do modelo de plano de fundo.

3.2.3 Classificação

A última etapa do sistema é classificar os eventos que ocorrem na cena e determinar quando houver situações de risco ou fraudes. As situações de risco que devem ser reconhecidas pelo sistema são o abandono de objetos, tulmutos e brigas. Neste trabalho serão apenas utilizados métodos de simples heurística para determinar as ações de cada objeto na cena e o algoritmo juntamente com um diagrama é mostrado no Anexo D para maior detalhamento.

Para analisar o reconhecimento das ações de cada objeto foram utilizados 2 vídeos que continham as situações adversas que deveriam ser detectadas. O primeiro vídeo foi o exemplo de um objeto ser abandonado e o segundo uma situação em que ocorre uma briga. Os objetos foram analisados quadro a quadro para determinar quando ocorre cada atividade, escolhendo-se um candidato alvo para cada vídeo como mostra a fõrmula a seguir.

$$P(x) = \frac{Número\ de\ quadros\ reconhecidos}{Número\ de\ quadros\ analisados} \quad (46)$$

Após a primeira análise na deteccção das ações, foi possível notar que o ruído na medição de velocidade de ruído influenciou negativamente, apesar do filtro de médias utilizado.

Para solucionar este problema foram utilizados os valores estimados do filtro de Kalman, assim diminuindo o ruído de origem gaussiana incorporado ao sistema. A figura 24 mostra um comparativo entre o valor real e o valor estimado produzido pelo filtro de Kalman e a figura 25 mostra um comparativo entre as velocidades determinadas a partir dos valores reais e dos valores estimados pelo filtro Kalman.

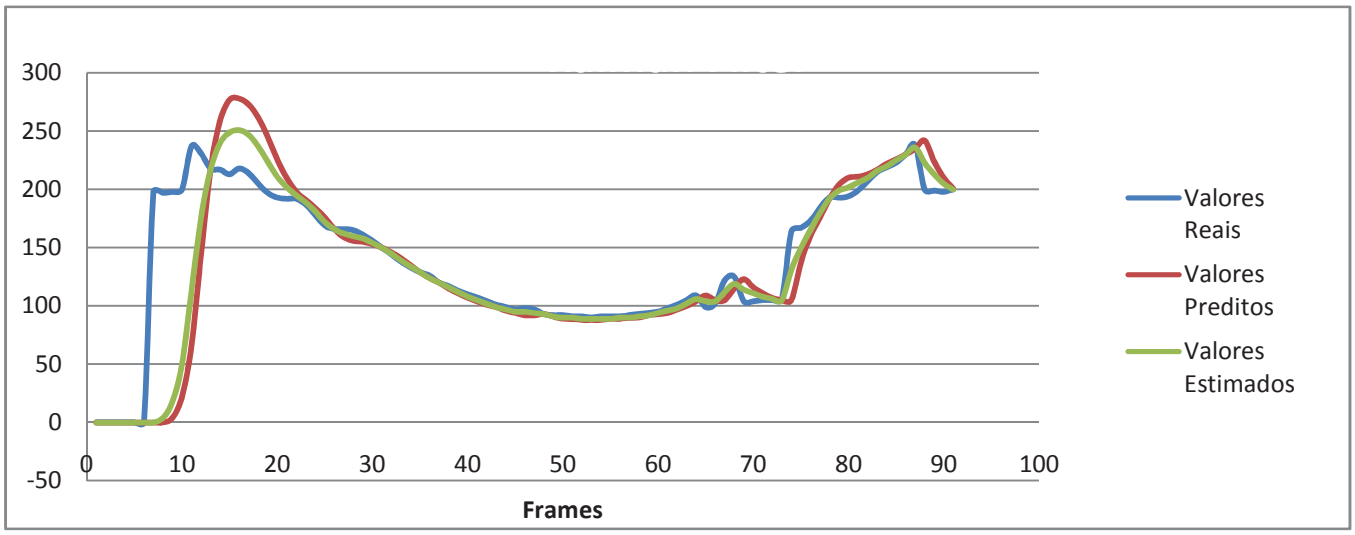


Figura 24 – Comparação entre o valor das posições de um blob após o filtro kalman.(a) Valores reais, (b)Valores estimados pelo filtro Kalman.

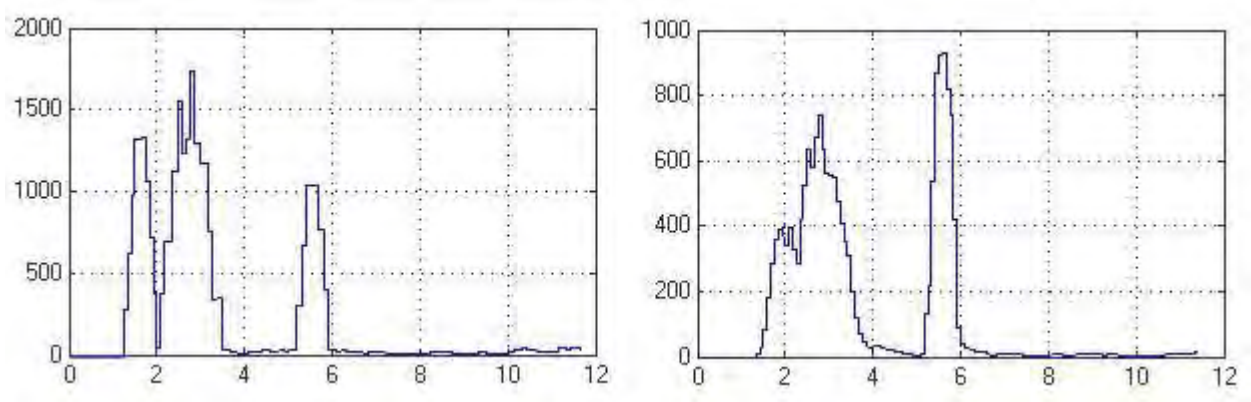


Figura 25 – Fluxo óptico de um elemento no período de 12s. (a) Valores reais, (b)Valores após a utilização de um filtro de médias.

A tabela 7 mostra a porcetagem de quadros em que ação do objeto em questão foi determinado corretamente.

Tabela 7 – Resultado da detecção de características de cada blob.

Id	Atividade	Vídeo 1- Objeto Abandonado	Vídeo 2 – Situação de Risco
1	Inativo	99%	-
2	Parado	40%	99%
3	Andando	94%	89%
4	Correndo	-	26%
5	Lutando	-	80%

Os resultados mostram que o algoritmo utilizado foi capaz identificar todas atividades, contudo devido ao acontecimento de ações que tiveram um curto perído de tempo (em torno de 10 quadros) esse algoritmo apresentou uma baixa porcentagem, que foram os casos do objeto parado no primeiro vídeo durante 1,25s (15 quadros) e no segundo vídeo uma pessoa correndo 2,5s (30 quadros) no segundo vídeo. Este problema ocorreu devido a utilização do filtro de médias. A fim de atenuar os ruídos para evitar a ocorrência de falsos positivos foi escolhido T=10 no filtro de médias, afetando negativamente quando houve transições de estados.

A próxima etapa foi identificar quando ocorre um evento na cena. No primeiro caso um objeto será considerado abandonado quando for reconhecido como separado e permanecer inativo durante um certo período de tempo. No segundo caso para que o alarme de “alerta” seja ativado podem ocorrer 3 situações distintas. A primeira é quando duas pessoas se juntam e apresentam a ação lutando, isto significa que pode estar havendo uma briga no local; A segunda situação é quando uma pessoa sozinha é classificada como lutando; a terceira situação é quando existe uma grande quantidade de pessoas correndo na cena, o que pode significar a ocorrência de tumultos.

Os dois vídeos utilizados anteriormente novamente foram utilizados para realizar os testes de detecção de situações de risco de cada cena.

Tabela 8 – Resultado da detecção de eventos.

Situações	Vídeo 1	Vídeo 2
Objeto Abandonado	Ativado	Não Ativado
Alerta	Não ativado	Ativado

A tabela 8 mostra que em ambos os casos o modelo foi capaz de detectar corretamente quando houve situações de alerta. No primeiro caso foi identificado o objeto abandonado logo depois que é separado da pessoa como mostra na figura 26.

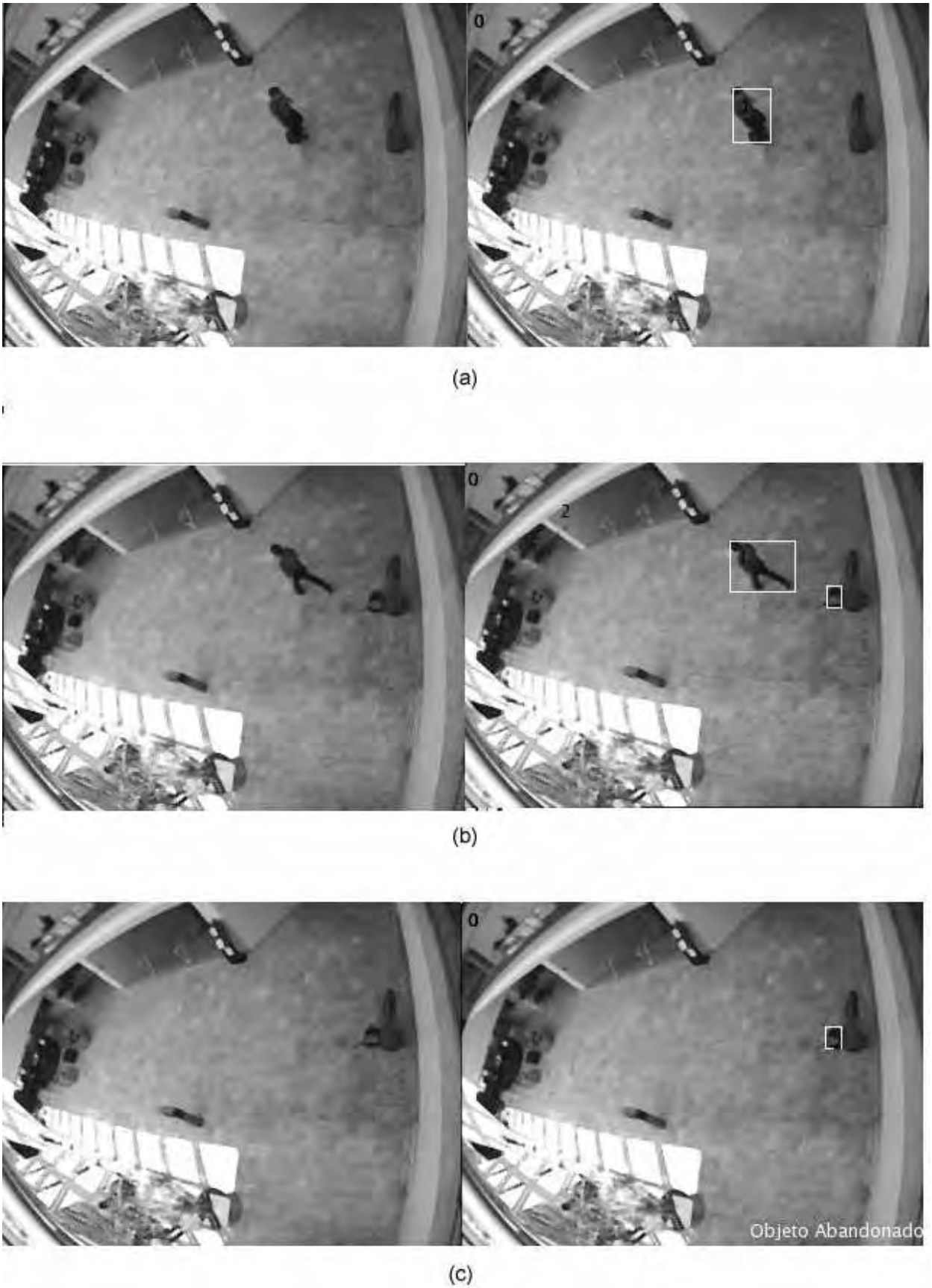


Figura 26 – Detecção do abandono de um objeto. (a)Pessoa carrega um objeto, (b) abandono do objeto e (c) alarme gerado avisando que há um objeto abandonado.

O sinal de alerta para ocorrência de brigas também foi ativado imediatamente quando se iniciou a briga simulada no vídeo 2. A figura 27 mostra o momento em que ocorre uma briga.

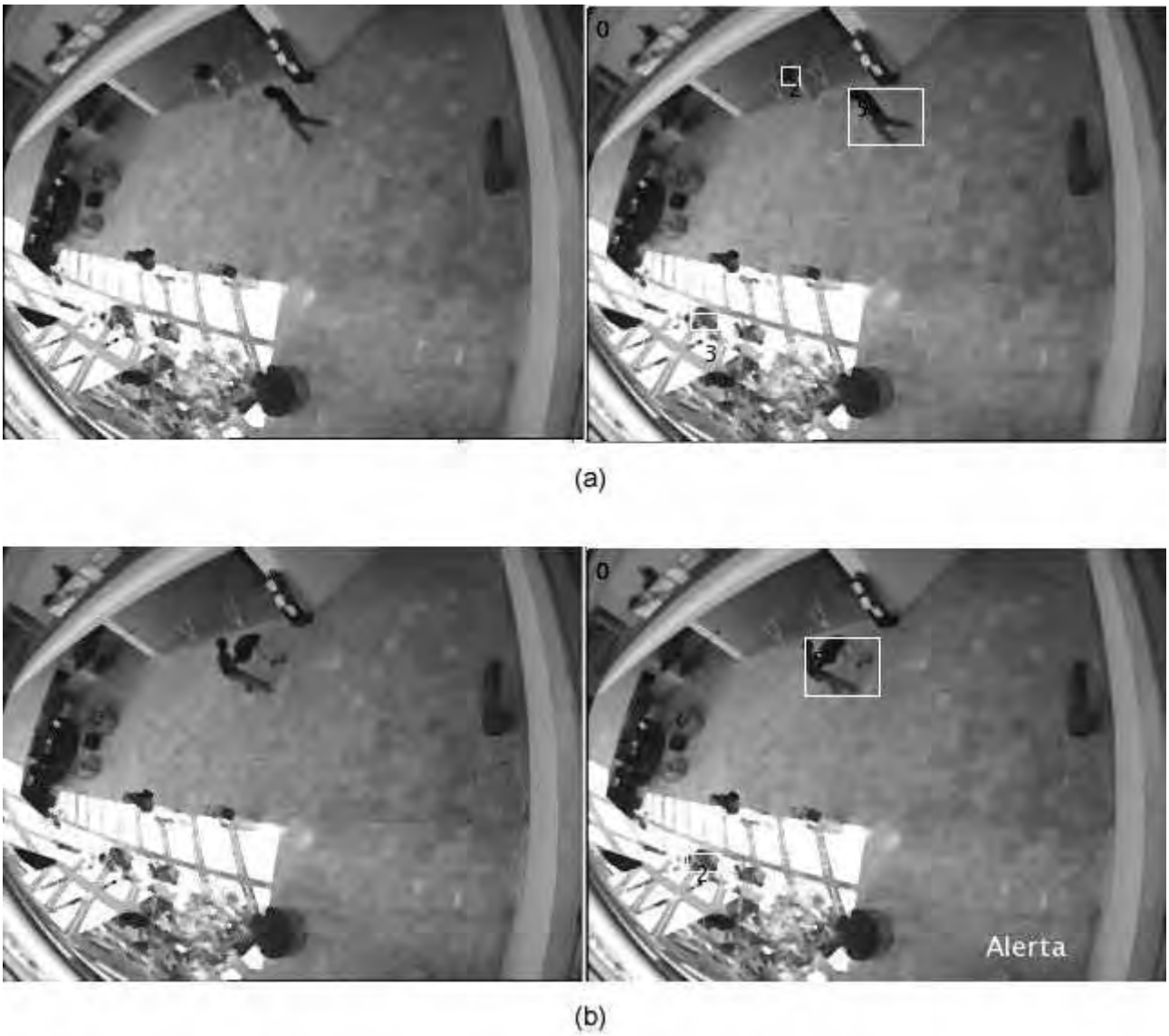


Figura 27 – Detecção da ocorrência de brigas. (a) Momento antes de começar uma briga e (b) Início da briga.

Para o sistema de contagem de pessoas foi utilizado um sistema diferente dentre os implementados por outros trabalhos.

Os trabalhos mais conhecidos utilizam uma linha virtual para contagem de pessoas detectando o vetor velocidade de cada *blob* na cena para determinar a direção do objeto, caso seja necessário contar somente as pessoas que passam num determinado sentido.

Neste trabalho adotou-se uma técnica de contagem de pessoas definindo em qual região cada blob se encontra. A Figura 28a mostra um exemplo simples em que a cena foi dividida em duas regiões. Na figura 28b a pessoa encontra-se na região 1. No instante mostrado na figura 28c a pessoa permuta para a região 2, ativando o contador que é impresso na tela.

(a) (b)

(c)

Figura 28 – Contador de pessoas. (a) Cena dividida em duas regiões, (b) momento antes da pessoa mudar de região e (c) instante após a pessoa mudar de região ativando o contador.

O algoritmo foi dividido em dois blocos: o primeiro bloco define em que região cada objeto encontra-se e logo em seguida o segundo bloco detecta quando ocorreu a transição de uma região para outra o Anexo F mostra um diagrama e os algoritmos utilizados.

A vantagem desta implementação é que assim será possível contar quantas pessoas passaram em um local específico, por exemplo, será possível delimitar as regiões de cada

bloqueio, podendo-se então determinar a quantidade exata de pessoas que passaram em cada bloqueio.

4 CONCLUSÃO

Os estudos realizados mostraram que foi possível o desenvolvimento de um sistema simples em relação ao custo computacional e ao mesmo tempo robusto quando aplicado em um ambiente restrito, no caso em questão, as estações de trem.

O modelo *Approximated Median Filter*, proposto por McFarlane e Schofield (1995), se mostrou muito eficiente comparado com técnicas mais complexas como *Mixture of Gaussians* no ambiente fechado, pois não há objetos de fundo não-estático. O algoritmo de *feedback*, que evita a atualização do modelo de plano de fundo, funcionou adequadamente, porém cabe observar que um objeto que permanece na cena no mesmo local desde o primeiro quadro pode ser considerado como elemento de fundo durante todo o tempo de execução do modelo.

Constatou-se que o filtro Kalman é uma ótima técnica utilizada para rastreamento de objetos. Assim foi possível traçar uma estratégia para detectar oclusões a partir de junções e separações de objetos em cena. O filtrotambém apresentou a vantagem de estimar o valor atual da posição de cada elemento, evitando que ruídos de origem gaussiana influenciassem a medição da velocidade e consequentemente na detecção de ações nos objetos.

Em relação à detecção de eventos, os modelos e experimentos realizados neste trabalho apresentaram uma boa solução para viabilizar a detecção de objetos abandonados e situações de risco, que se baseiam em simples heurísticas, utilizando os dados de características de velocidade, fluxo óptico, divisão e separação de objetos. Entretanto, nas detecções de eventos complexos, como andar, correr e lutar não foi possível a detecção exata de todos os quadros em que as ações ocorreram durante um período curto de tempo. Portanto, fica como proposta para trabalhos futuros a implementação de um algoritmo de classificações de ações mais avançado.Uma sugestão seria a utilização de algoritmos de inteligência artificial, como a técnica de reconhecimento utilizando as cadeias escondidas de Markov.

A técnica abordada para contagem de pessoas apresentou uma solução mais simples que as estudadas e mostrou-se como uma solução eficaz para o desenvolvimento de um sistema capaz de contar pessoas em regiões distintas de uma cena.

As soluções e abordagens encontradas neste trabalho possibilitam a continuação do mesmo a fim de aprimorar e desenvolver novos modelos que viabilizem sistemas capazes de detectar e monitorar situações mais complexas. Portanto, fica como proposta para desenvolvimento de trabalhos futuros os seguintes tópicos:

- Desenvolvimento de técnicas de reconhecimento facial;

- Investigar soluções para melhorar a detecção de ações complexas;
- Viabilizar a paralelização das tarefas realizadas pelos blocos deste sistema, com objetivo de monitorar diversos ambientes simultaneamente;
- Elaborar técnicas que utilizam imagens estéreas;
- Criação de modelos que permitam o monitoramento de outros locais, como exemplo, o tráfego de veículos;

5 REFERÊNCIAS

SIEBEL, Nils T. **Design and Implementation of people Tracking Algorithms for Visual Surveillance Applications.** Computational Vision Group, Department of Computer Science, 2003.p. 6-19

ANDRILUKA, M.; ROTH, S.; SCHIELE, B. **People-tracking-by-detection and people detection -by-tracking**. In Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on, pages 1_8, 2008.

SEGEN, J.; PINGALI, S. **A Camera-Based System for Tracking People in Real Time**, IEEE Proc. of Int. Conf. Pattern Recognition, 1996.p.4-5

SEXTON, G et al. **Advances in automated pedestrian counting**, University of Northumbria at Newcastle, England, CEM Systems. Northern Ireland.

AMER, A**. Voting-based simultaneous tracking of multiple video objects**. IEEE Transaction on Circuits and Systems for Video Technology, 2005.p. 3-7

AUVINET, E. et al. **Left-luggage detection using homographies and simple heuristics**. In Proceedings 9th IEEE International Workshop on Performance Evaluation in Tracking and Surveillance (PETS 06), 2006.p. 3-4

DEDEOGLU, Y. **Moving object detection, tracking and classification for smart video surveillance.** Master's thesis, Bilkent University, Faculty of Engineering, August 2004.

POLANA, R. , NELSON, R**., Low level recognition of human motion (or how to get your manwithout finding his body parts)**. In Proceedings of the 1994 IEEE Workshop on Motion of Non-Rigid and Articulated Objects, 1994.p. 4.

YAMATO, J., OHYA, J. , ISHII K. **Recognizing human action in time-sequential images using hidden markov model**. In Proceedings of IEEE Computer Society Conference on Computer Vision andPattern Recognition, 1992.p. 13 - 18

MCFARLANE, N. & SCHOFIELD, C. **Segmentation and tracking of piglets in images**, In Proc. of Machine Vision Applications, 1995.

STAUFFER, C. & GRIMSON, W. **Adaptive background mixture models for real-time tracking**, In Proceedings of the Int. Conf. on Computer Vision and Pattern Recognition, 1999.p. 3.

CUCCHIARA, R., et al**. Detecting Objects, Shadows and Ghosts in Video Streams by Exploiting**. Italy : D.S.I. - University of Modena and Reggio Emilia, 1996.p. 5.

ELGAMMAL, A., et al. **.Background and Foreground Modeling Using Nonparametric Kernel Density Estimation for Visual Surveillance**. Proceedings of the IEEE, 2002.p.1.

WELCH, Greg E BISHOP, Gary. **An Introduction to the Kalman Filter**. Artigo internet em: http://www.cs.unc.edu/~welch/kalman/kalman_filter/kalman.html. Acesso em abril de 2011.

HARITAOGLU, I. , HARWOOD, D., DAVIS, L. S. **W4: real-time surveillance of people and their activities**. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 2000. p. 809-830.

6 BIBLIOGRAFIA CONSULTADA

CAI, Q.; AGGARWAL, J. K. **Tracking human motion using multiple cameras** , Los Alamitos, 1996.

ROHR, K. "**Towards Model-Based Recognition of Human Movements in Image Sequences**," CVGIP: Image Understanding, 1994.

GOLDSTEIN, H. **People counting using image processing**, Technion - Israel Institute of Technology, Department of Computer Science, Advanced Programming Lab B. July 1999.

BARRON, J. L, FLEET, D.J., BEAUCHEMIN, S. S. **Performance of optical flow techniques. International Journal of Computer Vision,** 1994.

COLLINS, R., LIPTON, A. , KANADE, T., FUJIYOSHI, H., DUGGINS, D., TSIN, Y., TOLLIVER, D. ENOMOTO, N., HASEGAWA, O. **A system for video surveillance and monitoring: VSAM final report**. Robotics Institute, Carnegie Mellon University, 2000.

CHEUNG, S-C. KAMATH, C. **Robust techniques for background subtraction in urban traffic video**, In Proceedings of Electrical Imaging: Visual Communications Image Processing , 2004.

YILMAZ, A., JAVED, O., AND SHAH, M.,**Object tracking: A survey**, ACM Computing Surveys, 2006.

RIBEIRO, P. C., SANTOS-VICTOR, J.. **Human activity recognition from video: modeling, feature selection and classification architecture**. In Proceedings of HAREM 2005 - International Workshop on Human Activity Recognition and Modelling, 2005.

LOWE, D. G., **Distinctive Image Features from Scale- Invariant Keypoints**, International Journal of Computer Vision, 2004.

MOESLUND, T. B., HILTON, A. **A survey of advances in vision-based human motion capture and analysis**. Computer Vision and Image Understanding, 2006.

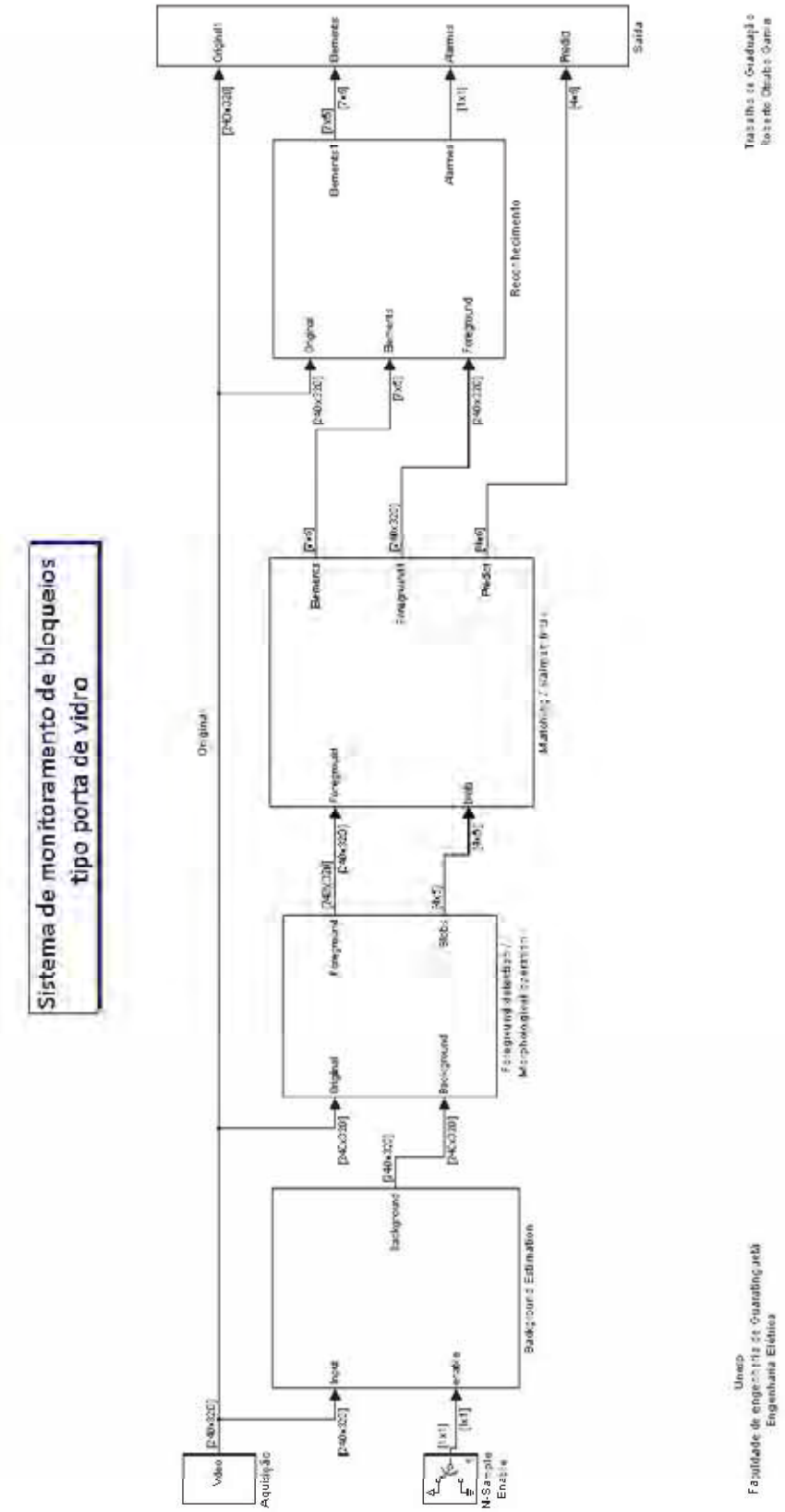
POLANA, R. , NELSON, R. **Low level recognition of human motion (or how to get your manwithout finding his body parts)**. In Proceedings of the 1994 IEEE Workshop on Motion of Non-Rigid and Articulated Objects, 1994.

RUSS, John C. **The Image Processing Handbook.** Taylor & Francis 2007.

GONZALES, Rafael C., WOODS, Richard E. **Digital Image Processing**, 2nd ed. Prentice Hall, 2001.

MARQUÊS,Ogê ,VIEIRA, Hugo. **Processamento Digital de Imagens.** Brasport, 1999.

7 ANEXO A – MODELO SIMULINK



8 ANEXO B – MATRIZES UTILIZADAS PARA O FILTRO KALMAN

A seguir encontram-se as matrizes utilizadas no modelo de transição de estados para o filtro Kalman

$$x_{k+1} = Ax_k + w_k$$

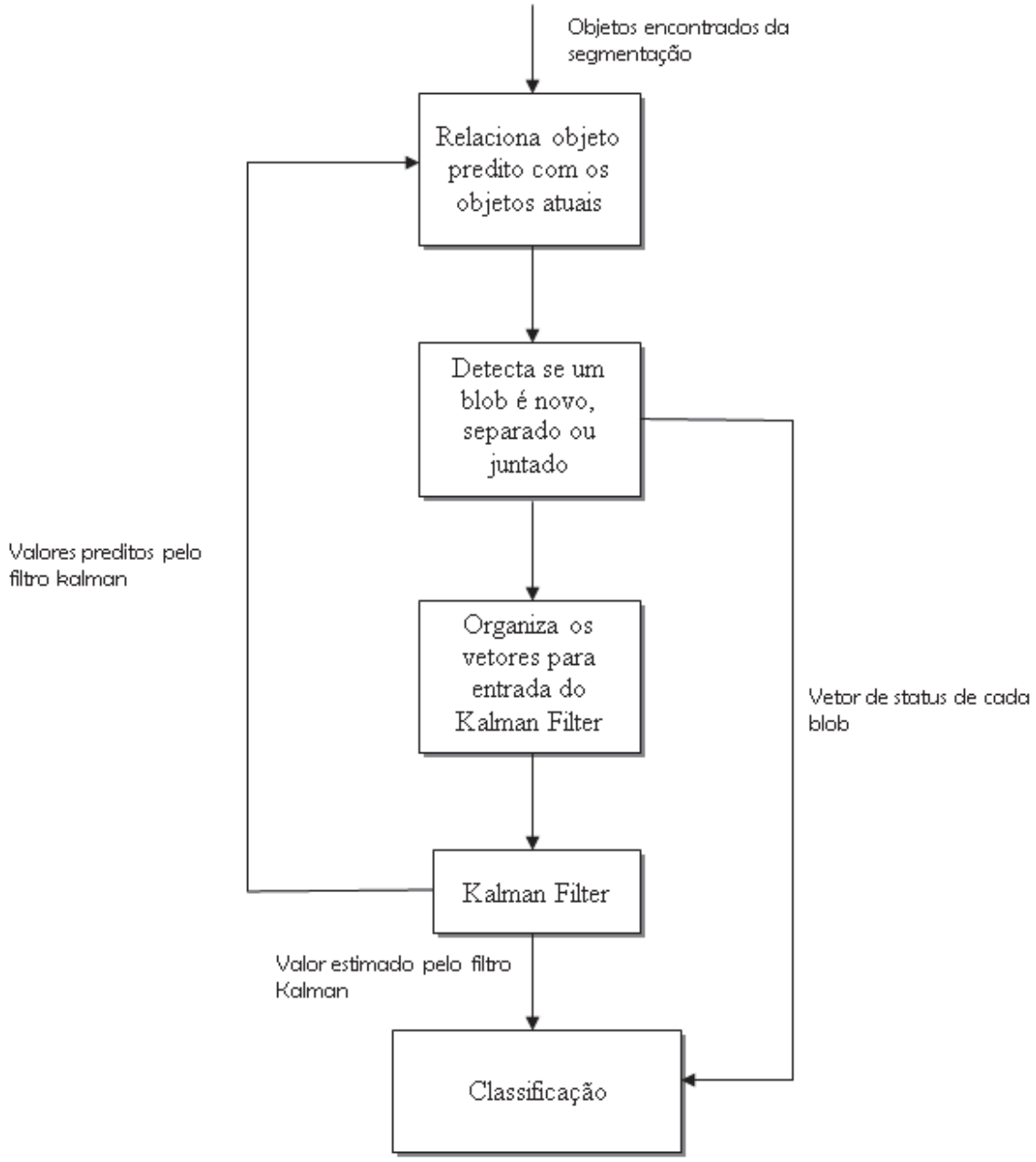
$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \Delta x_{k+1} \\ \Delta y_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_k \\ y_k \\ \Delta x_k \\ \Delta y_k \end{bmatrix} A + w_k$$

$$z_k = Hx_k + v_k$$

$$\begin{bmatrix} z_{x_k} \\ z_{y_k} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_k \\ y_k \\ \Delta x_k \\ \Delta y_k \end{bmatrix} A + v_k$$

9 ANEXO C – ALGORITMO DE JUNÇÃO E DIVISÃO DE BLOBS

As ténicas utilizadas para auxilio do filtro Kalman e o Bloco Matching são mostradas no diagrama a seguir:



O algoritmo desenvolvido é mostrado a seguir

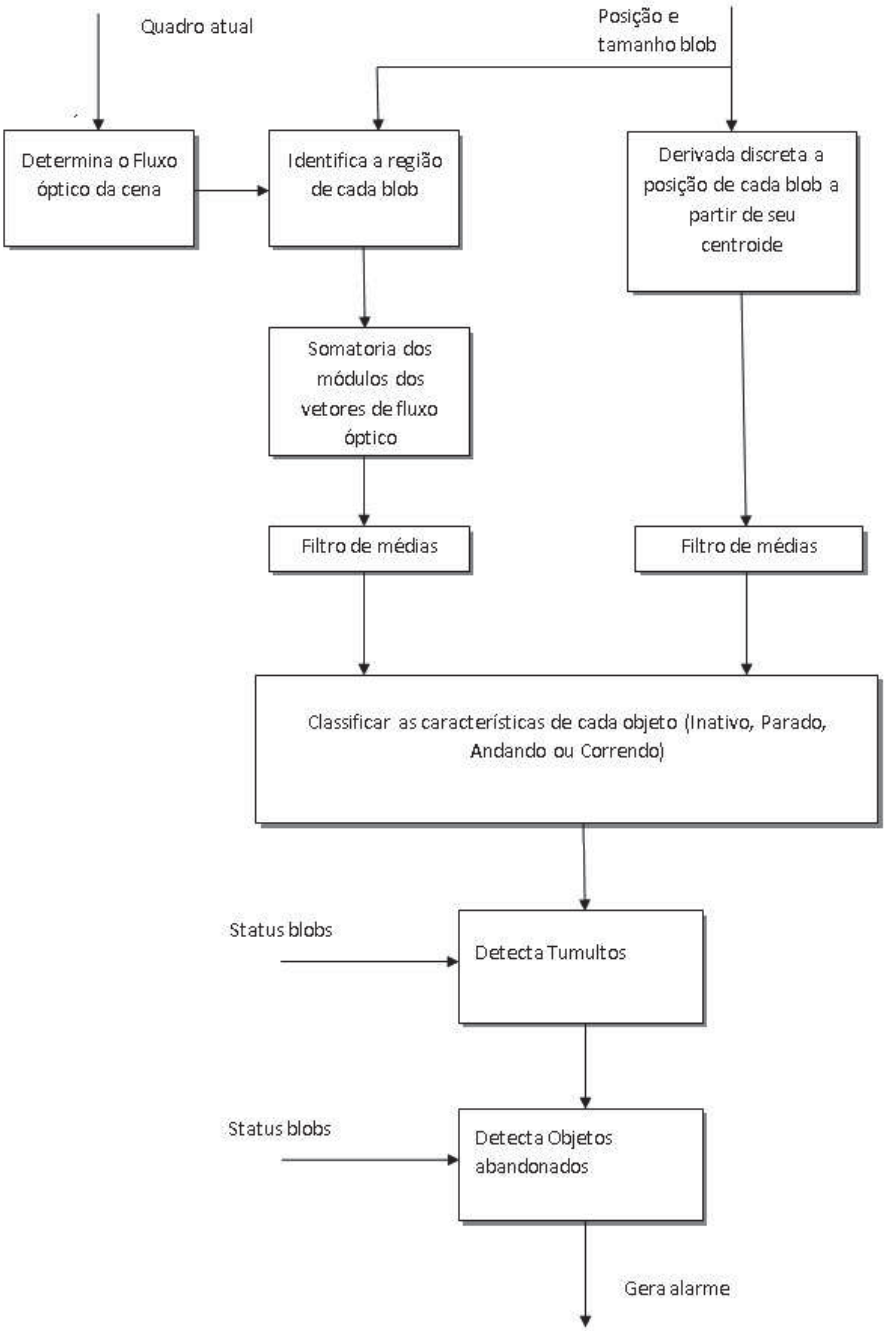
```
function blobs = matching(b_atual,b_pred,max)
%#eml
d=1;
num = 5;
thresh1 =30;
blob_ma = zeros(7,num);
e = zeros(1,5);
     for i=1:num
     %Todos blobs atuais
     if b_atual(9,i)==1;
     %checha se há blob
         tmp = 0;
          for k=1:num
          %todos blobs de predição
             if dist(b_atual(1,i),b_atual(2,i),b_pred(1,k),b_pred(2,k))< thresh1
             %compara a distancia entre b_atual e b_pred
                if (c(1,k) == 0)
               %checa se já há um blob associado

                     if tmp>=2
               %checa se mais de um blob foi associado
                      c(6,k)=2;
               %status = mergeded
                   else
                 blob_ma (1,k)= b_atual(3,i);
                 blob_ma (2,k)= b_atual(4,i);
                 blob_ma (3,k)= b_atual(5,i);
                 blob_ma (4,k)= b_atual(6,i);
                 blob_ma (5,k)= b_pred(3,k);
                 blob_ma (7,k)= b_atual(9,i);
                 tmp = tmp + 1;
                 e(1,i) = 1;
                    c(6,k)=1;
                   end
                    else
          %se tiver associado blob
                    c(6,k)=4;
          %merged
                end

             end
         end
    end
end
```

Continuação...

```
%Determina se o blob é splitted
for f=1:num
    if (e(1,f)==0)&&(b_atual(9,f)==1)
        for g=1:num
            if c(1,g)==0
                blob_ma (1,g)= b_atual(3,f);
                blob_ma (2,g)= b_atual(4,f);
                blob_ma (3,g)= b_atual(5,f);
                blob_ma (4,g)= b_atual(6,f);
                blob_ma (5,g)= max(1,d);
                d = d+1;
                blob_ma (6,g)= 2;
                blob_ma (7,g)= b_atual(9,f);
                break
            end
        end
    end
end

blobs = blob_ma;

%função utilizada para determina a distancia euclidiana

function a = dist(b,c,d,e)

a = abs(complex((b-d),(c-e)));
```

10 ANEXO D – ESTRUTURA de detecção

O diagrama a seguir mostra a estratégia adotada para determinar e classificar os eventos e ações ocorridas.



Algoritmo para detecção de objetos abandonados

```
function [alarme,fw] = det_ob(clas,fb,elements)
%#eml
num = 5;                          %Quantidade de objetos max
temp = zeros(1,num);              %Cria um vetor para armazenar o tempo de um
                                   objeto inativo
temp = fb;                        % Carrega o vetor tempo do instante t-1
tempo_obj = 50;                   % Tresholding para o num de quadros ate que objeto
                                    seja detectado como abandonado
alarme_tmp = zeros(1,num);
for i=1:num
    if Elements(3,i)==2                     %Verifica se o objeto surgiu de uma
                                              separacao
    if clas(1,i)==1                         %Verifica se o objeto permanece
                                                inativo
        temp(1,i) = temp(1,i) + 1;          %atualiza o vetor tempo
        if temp(1,i) > tempo_obj            %Verifica o periodo de tempo
            alarme_tmp(1,i) = 1;            %ativa alarme
        end
    else
        temp(1,i)= 0;

    end
    end
end

fw = temp;
alarme = s;
```

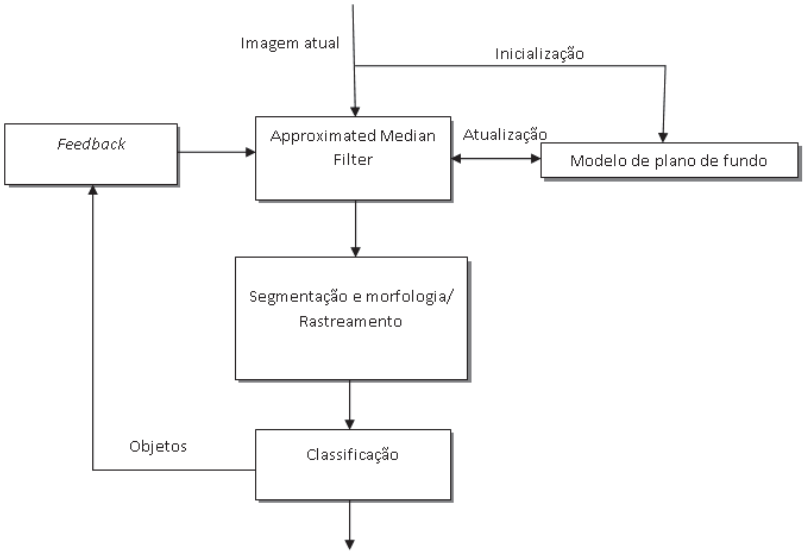
Algoritmo para detecção de situações de risco.

```
function alarme = det_ale(clas,blob)
%#eml
Alarme_tmp=0;                   %Variavel de saida :alarme
Pessoas = 0;                     %contador numero de pessoas
num = 5;                         % Numero max de objetos.
pessoas_corre = 4;               % Numero max de pessaos correndo
for i=1:num
    if (clas(1,i)==4)            % Contador para detectar a quantidade
                                   de pessoas correndo na cena
        pessoas = pessoas + 1;  % atualiza a variavel t
    end
    if clas(1,i)==5              % Checa se o status e lutando
        if blob(3,i)==2          % checa se o status do blob e merged
        alarme_tmp = 1;          % Alerta: briga
        end
        if blob(3,i)==1          % Objeto com status rastreando
        alarme_tmp = 2;                  % Alerta: vandalismo
        end
    end
end
if (pessoas == pessoas_corre)                %Checa o num de pessoas correndo
    alarme_tmp = 3;                          % Alerta: Tumulto
end

alarme = alarme_tmp;
```

Algoritmo para classificação de ações.

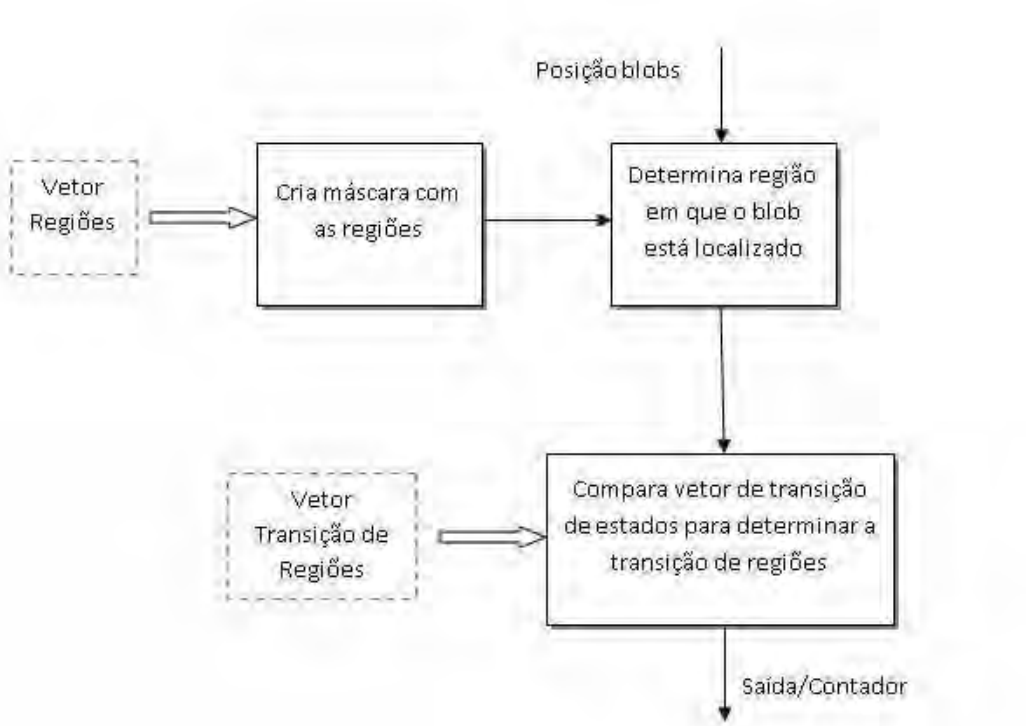
```
function y = fcn(vel,of,ena)
%#eml
num= 5;

...

s = zeros(1,num);
for i=1:num
    if ena(1,i)==1 % checa se o blob esta ativo
        % Compara os parametros para detectar a acao do blob
    if (vel(1,i)>ina_vmin)&&(vel(1,i)<ina_vmin)&&(of(1,i)>ina_ofmin)&&(of(1,i)<ina_ofmax)
            s(1,i) = 1; %blob classificado como Inativo
         end
    if (vel(1,i)>par_vmin)&&(vel(1,i)<par_vmin)&&(of(1,i)>par_ofmin)&&(of(1,i)<par_ofmax)
            s(1,i) = 2; %blob classificado como Parado
       end
    if (vel(1,i)>and_vmin)&&(vel(1,i)<and_vmin)&&(of(1,i)>and_ofmin)&&(of(1,i)<and_ofmax)
            s(1,i) = 3; %blob classificado como Andando
         end
    if (vel(1,i)>cor_vmin)&&(vel(1,i)<cor_vmin)&&(of(1,i)>cor_ofmin)&&(of(1,i)<cor_ofmax)
            s(1,i) = 4; %blob classificado como Correndo
      end
    if (vel(1,i)>lut_vmin)&&(vel(1,i)<lut_vmin)&&(of(1,i)>lut_ofmin)&&(of(1,i)<lut_ofmax)
            s(1,i) = 5; %blob classificado como Lutando
         end
    end
end

y = s;
```

11 ANEXO E – CLASSIFICAÇÃO DE OBJETOS IMÓVEIS PARA O ALGORITMO DE *FEEDBACK*



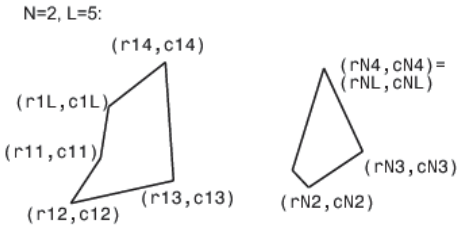

```
function [mask,fw] = feedb(img,ac,elements,fb)
%#eml
num = 5;                        %Quantidade de objetos max
si = size(img);                 %mede a resolução da imagem
ma = zeros(si(1),si(2));        %Cria matriz mascará que define o elemento de
saída
temp = fb;                         %Temporizador
lim = 10;                          %Tempo limiarização
s = zeros(1,num);
% Detecta elemento parado na cena

for i=1:num
    if (ac(1,i)==1)||(ac(1,i)==2)       %Verifica se o objeto permanece
inativo/parado
        temp(1,i) = temp(1,i) + 1;          %atualiza o vetor tempo
        if temp(1,i) > lim                 %Verifica o periodo de tempo
            s(1,i) = 1;                        %objeto parado
        end
    else
        temp(1,i)= 0;
    end
end
% Cria mascará para ser utilizada no bloco AMF
for i=1:num
    %detecta se o objeto permaneceu parado com a condição dele não ser
splitted
    if (s(1,i)==1)&&(elements(7,i)~=2)
        for j=elements(1,i):(elements(1,i)+elements(3,i))
            for k=elements(2,i):(elements(2,i)+elements(4,i))
                ma(j,k) = 1;          %cria mascara
            end
        end
    end
end
fw = temp;
```

12 ANEXO F – MÉTODO DE CONTAGEM DE PESSOAS



Para funcionamento do sistema de contagem é necessário a entrada de dois vetores. O primeiro deve conter as regiões, determinada pela matriz E com dimensões 2LxN:

$$E = \begin{bmatrix} r_{11} & \cdots & r_{N1} \\ c_{11} & \cdots & c_{N1} \\ r_{12} & \cdots & r_{N2} \\ c_{12} & \cdots & c_{N2} \\ \vdots & \cdots & \vdots \\ r_{1L} & \ddots & r_{NL} \\ c_{1L} & \cdots & c_{NL} \end{bmatrix},$$



Onde cada coluna da matriz corresponde a um poligono diferente. Se algum poligono tiver o número de lados menor que outro então deve se repetir o último ponto para completar a matriz. As regiões serão determinadas de acordo com a coluna em que o polígono ocupa na matrz de entrada.A matriz de transições de estado definida por T com dimensões 2xM,

$$T = \begin{bmatrix} x_1 & x_2 & \dots & x_M \\ y_1 & y_2 & \dots & y_M \end{bmatrix}$$

Onde M é o número de contadores, e a transição ocorre da região x para y.

Algoritmo para determinar em que região cada blob se encontra.

```
function region_blob = det_region(regioes,elements)
%#eml

num = 5;                    % determina número max de objetos
region_temp = zeros(2,num); % cria o vetor de saída contendo as regiões de
                               cada blob
for i=1:num
    if elements(1,3)==1             %detecta se o elemento esta habilitado

      %armazena o valor da região no vetor de saida

     region_temp(2,i) = regioes(elements(i,1),elements(i,2));
     region_temp(1,i) = 1;           %habilita o blob para proxima etapa
    end
end

region_blob = region_temp;
```

Algoritmo para detecção de transição entre regiões.

```
function cont = tran_region(region,T,fb,atraso)

num = 5;        %numero max de objetos
rel = size(T);  % determina o numero de transições
cont_temp = zeros(1,rel(2));  % Contadores
for i=1:num
    if region(1,i)==1   % verifica se o blob está ativo
    for k=1:rel(2)         % checa todas trans possiveis
        if (T(1,k)==region(2,i))&&(T(2,k)==atraso(1,i))
        cont_temp(1,k) = 1; % ativa o contador
        end
    end
    end
end

cont = cont_temp; % saída para o contador
```