Instituto Superior De Engenharia do Porto

Sistema de Navegação por Processamento Visual de "Edges" para Robot de Condução Autónoma

Docentes :
Eng.José Miguel Almeida
Eng.Alfredo Martins

Trabalho realizado por :
1000910-António Sérgio Silva
1990902-Ricardo Sousa

LSIS: Laboratório de Sistemas

2005/2006

Capítulo 1

Sumário

- Considerações Gerais
- Formulação do Problema
- Descrição dos Objectivos
- Projecto Runner – Robot de Condução Autónoma
  - Arquitectura de Hardware do Robot
  - Arquitectura de Software do Robot

Considerações Gerais

- Aplicações
  - Industria
  - Ambientes Hostis
  - Busca e Salvamento
  - Vigilância
  - Competições Robóticas
  - Investigação Espacial

Formulação do Problema

- Pretende-se integrar informação de fronteiras de cor na imagem, e elementos geométricos do ambiente (tais como linhas) da prova de condução autónoma do FNR no sistema de navegação do veículo autónomo

- O sistema de navegação utilizará informação disponibilizada pelo sensor de visão existente devendo fornecer informação sobre os objectos detectados, e efectuando a fusão sensorial através de informação do mundo previamente guardada e com outros sensores de navegação

Descrição dos Objectivos

- Identificar as linhas da pista e determinar, ao longo do percurso percorrido pelo robot, se estas são rectilíneas ou curvilíneas.
- Identificar os semáforos, analisar a sua forma (“X”, “↑”, “→”, “←”, “Parque”).
- Identificar a zona de parque, é um quadrado cujas arestas são brancas, com um P no meio.
- Identificar a passadeira, sendo esta reconhecível por faixas brancas e pretas na vertical e duas faixas brancas na horizontal.
- Identificar pontos únicos da pista que possam situar com precisão no “mundo” o robot, isto é, onde ele se encontra.

Descrição dos Objectivos

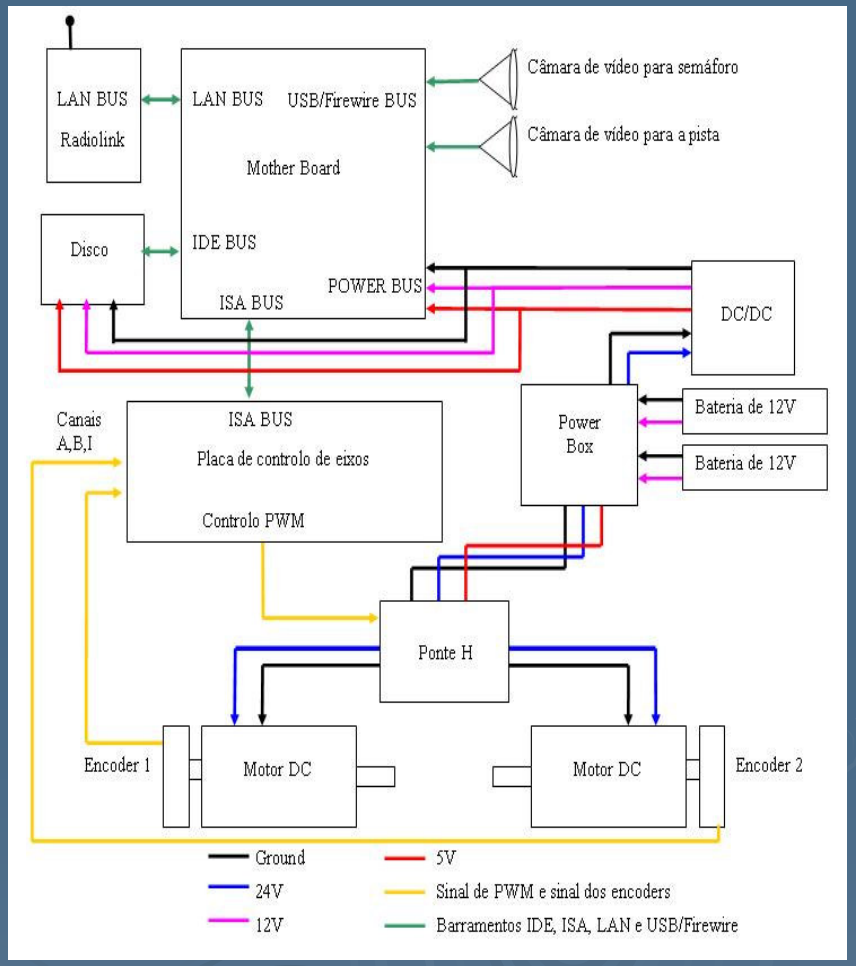
- Identificar zonas de obras, que é limitada à entrada e à saída por dois pares de cones.
- Identificar obstáculo que poderá ser colocado em qualquer local da pista.
- Identificar túnel, que será colocado num dos sectores circulares da pista.

Descrição dos Objectivos

- Identificar zonas de obras, que é limitada à entrada e à saída por dois pares de cones.
- Identificar obstáculo que poderá ser colocado em qualquer local da pista.
- Identificar túnel, que será colocado num dos sectores circulares da pista.

Arquitectura de Hardware

- Placa de controlo de Eixos
- Amplificadores de potência
- Placa de Distribuição e Protecção
- Câmara
- Computador
- Conversor DC/DC
- Placa de rede Wireless
- Motores DC

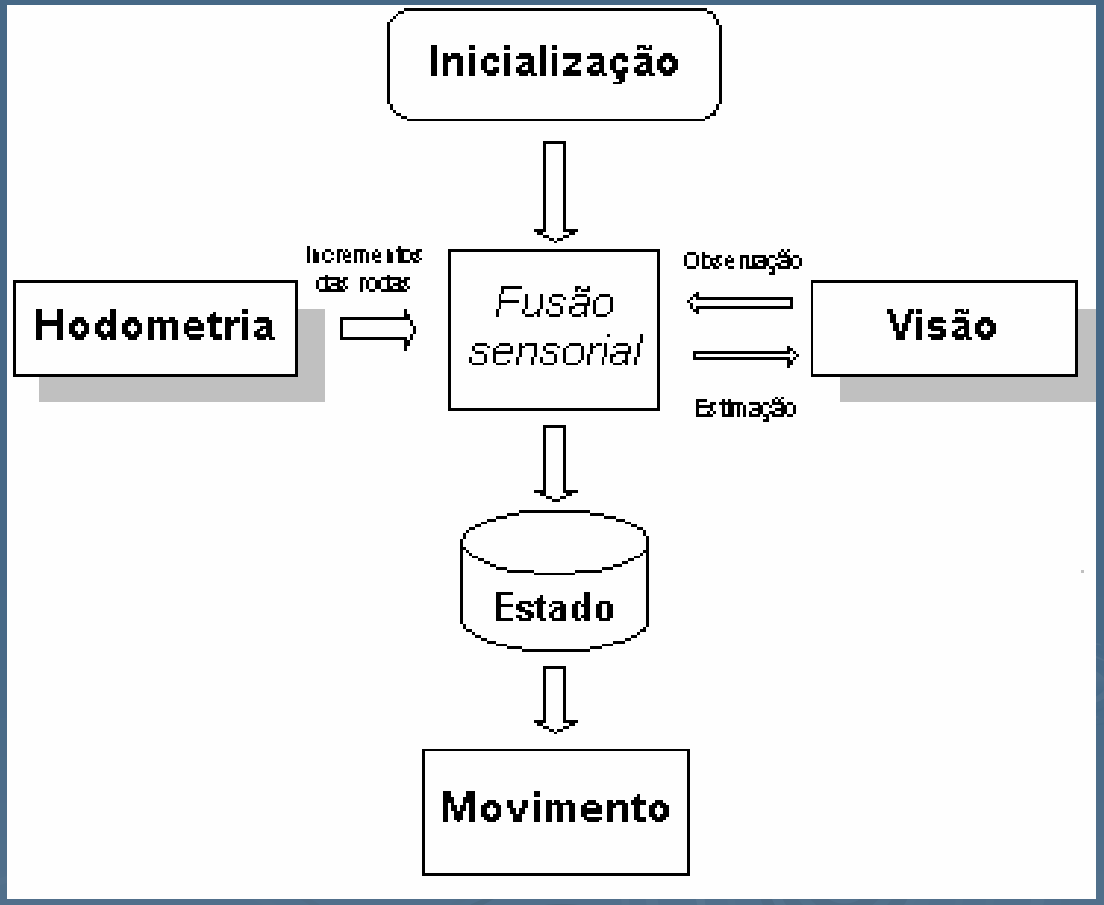


LSIS: Laboratório de Sistemas

2005/2006

Arquitectura de Software

- Hodometria
- Sistema de Visão
- Fusão sensorial



Capítulo 2

Sumário

- Edges/Blobs
- Sistemas de Coordenadas
- Sistema Operativo
- Métodos Matemáticos para detecção de rectas, curvas, pontos únicos
  - Transformada de Hough
  - Método dos Mínimos Quadrados

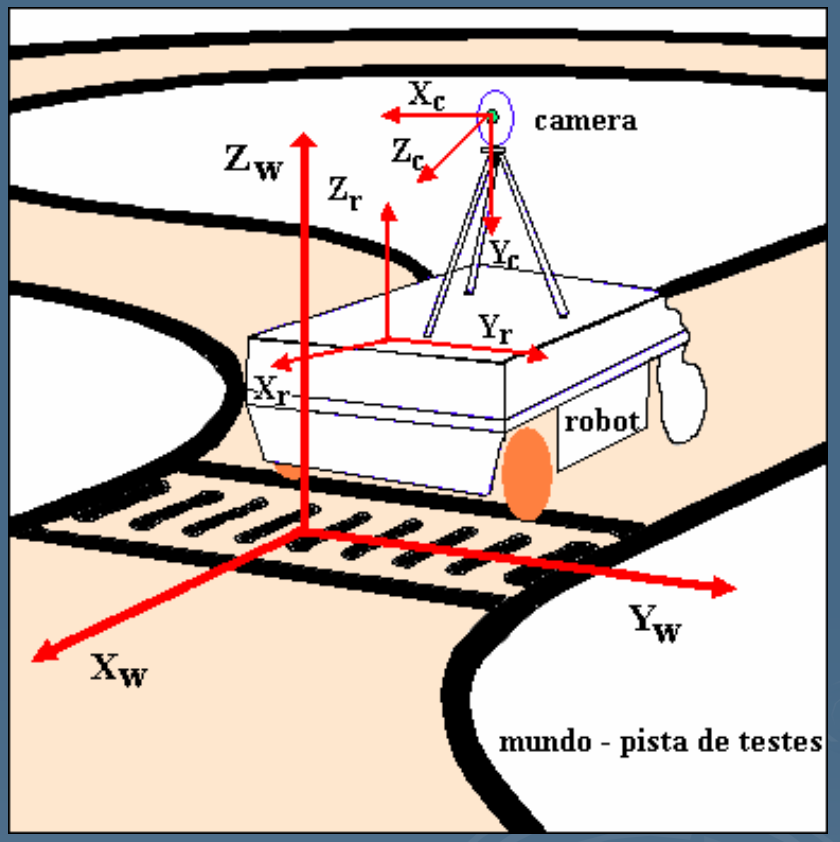
Edges/blobs

- Um “Edge” numa imagem é a fronteira entre dois valores significativamente diferentes do pixel (preto - branco ou branco - preto).

- Um “Blob” (binary large object) é geralmente um grupo de pixels da mesma cor organizados numa estrutura.

Sistemas de Coordenadas

- Sistemas de Coordenadas do Mundo
- Sistemas de Coordenadas do Robot
- Sistemas de Coordenadas da Câmara



Sistema Operativo

Linux (Distribuição Fedora core 1 e 3)

- Vantagens
  - É um sistema operativo livre.
  - Permite o acesso ao código (OPEN SOURCE).
  - É um sistema operativo actual e em constante desenvolvimento.
  - É bastante reconfigurável.
  - Suporte para o barramento USB e possibilidade de trabalhar com Vídeo.
  - Apoio a nível mundial.
  - Sistema operativo de Tempo Real (*SOTR*).

Transformada de Hough

- A transformada de Hough é um método muito usado em Visão Computacional, que permite detectar formas que sejam facilmente parametrizadas (linhas, círculos, elipses, etc) em imagens computacionais.

- Vantagens

  - Método robusto
  - Tolerante a falhas em alguns pontos da imagem
  - Muito pouco afectada pelo ruído da imagem

Transformada de Hough

Modo de funcionamento:

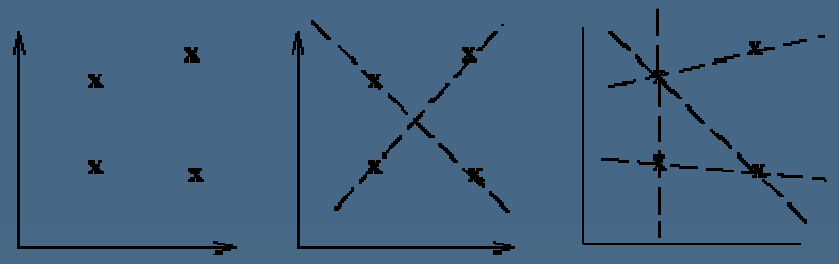
- A transformada de Hough consiste num algoritmo que procura a linha/curva que melhor se ajusta a um conjunto de pontos dado.
- A ideia motivadora por trás da transformada de Hough, para detectar linhas/curvas é que cada input, coordenada no ponto, indica a sua contribuição para uma solução consistente global, isto é, a linha/curva física que deu origem ao ponto na imagem.

Transformada de Hough

Detecção de rectas

- Consideremos o problema comum de ajustar um conjunto de segmentos de linha a um conjunto de pontos de uma imagem analisada, sendo os pontos, as coordenadas dos pixels resultantes do detector de edges.

- As imagens abaixo indicadas mostram possíveis soluções para o problema

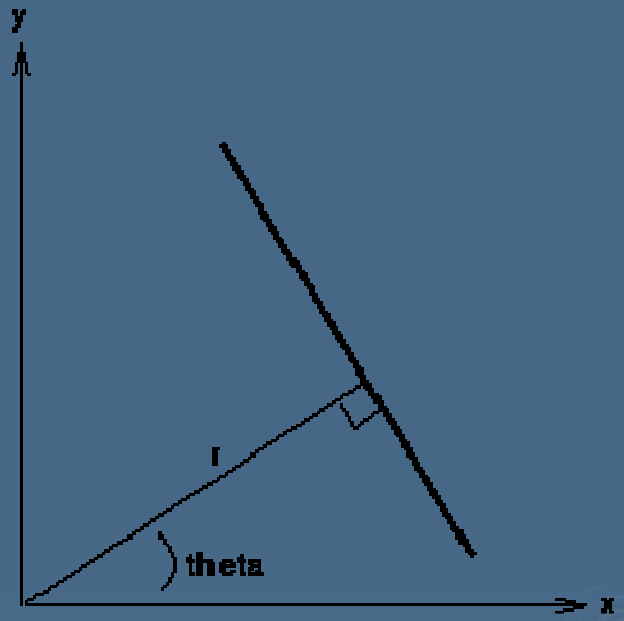


a)Coordenadas dos pontos. b e c)Possíveis segmentos de linha

Transformada de Hough

Equação Paramétrica da recta:

$$x \cos \theta + y \sin \theta = r \quad (A)$$



Transformada de Hough

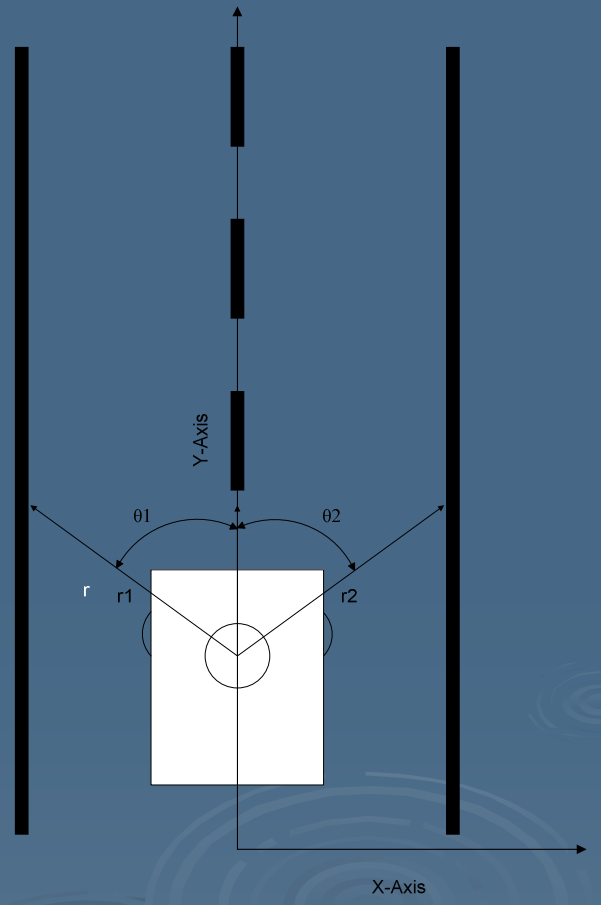
- Na análise de uma imagem, as coordenadas dos pontos, guardados nos edges (xi , yi) são conhecidos e por isso podem ser usados como constantes na equação paramétrica da linha, enquanto r e θ são as variáveis desconhecidas que procuramos.
- Para cada uma das infinitas linhas que passam por cada ponto (x, y), existe um (r , θ) associado satisfazendo a Equação (A).
- Os pontos (x , y) onde as linhas candidatas devem passar estão definidos numa imagem J(x , y).
- Cada ponto (r , θ) corresponde a uma linha com ângulo θ e distância r da origem no espaço original da imagem. Quando os pontos são visualizados no "espaço Hough", aqueles que são colineares tornam-se muito evidentes uma vez que formam curvas que se intersectam num ponto comum (r , θ)

Transformada de Hough

Linhas da Pista:

- **(θ) = 0, Linha tracejada**
- **(θ) > 0, Linha lateral direita**
- **(θ) < 0 , Linha lateral esquerda**

r, Distância do centro do robot à linha



Transformada de Hough

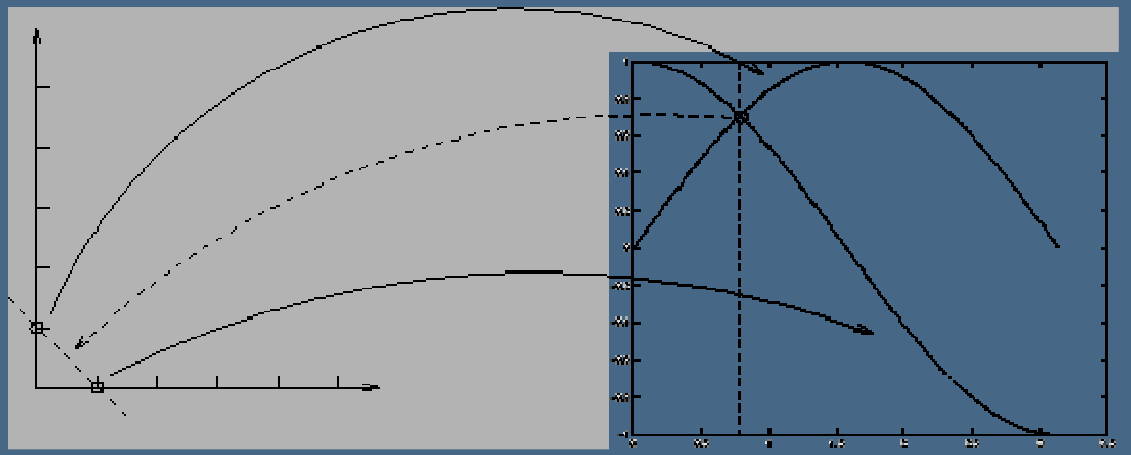
Acumulador:

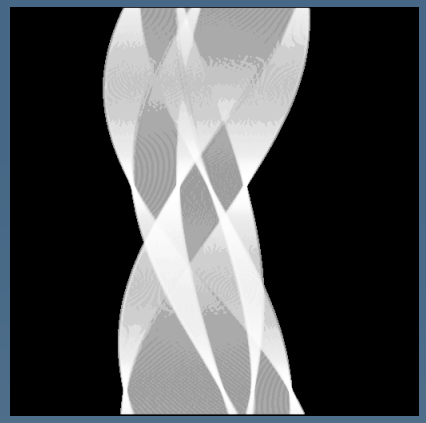
- A transformada é implementada, quantificando o “espaço Hough” em intervalos finitos ou em acumuladores
- O algoritmo usado deverá percorrer todos os pontos (xi,yi) dos edges, e projectá-los numa curva sinusoidal (r, θ) e incrementar o acumulador que pertencer a esta curva. Um valor muito elevado no vector de acumuladores representa indícios muito fortes de que existe uma determinada linha na imagem.

Transformada de Hough

Acumulador:

- O valor da função no "espaço Hough" dá a densidade ao longo da linha no espaço.





"espaço Hough"

Cada ponto na imagem da esquerda é projectado numa curva sinusoidal no acumulador, à direita, usando a equação (A). A intersecção das curvas representa a linha que conecta os pontos.

Transformada de Hough

Como determinar quais as linhas/curvas que são identificadas nos acumuladores?

- Existem vários métodos para extrair os pontos que nos interessam dos acumuladores.

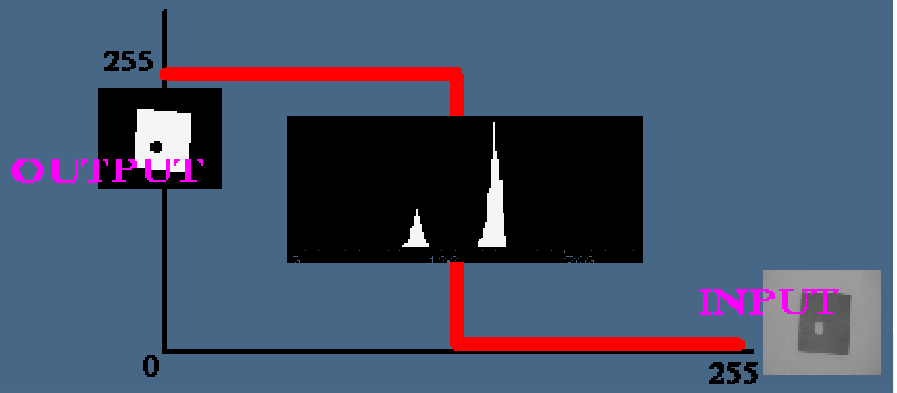
- Vamos abordar o thresholding que permite extrair somente os pontos **(r , θ)** pertencentes às linhas rectas da imagem original. Por outras palavras apenas nos interessam os acumuladores com um determinado "intensity threshold", ou máximo da matriz de acumuladores, definido por nós.

Transformada de Hough

Thresholding

- O input de uma operação de thresholding é tipicamente um grayscale ou uma imagem da cor.

- Numa única passagem, cada pixel na imagem é comparado com "intensity threshold", se a intensidade do pixel for mais elevada do que o "intensity threshold" o pixel será ajustado por exemplo ao branco na saída, se for inferior será ajustado ao preto.



Threshold

Transformada de Hough

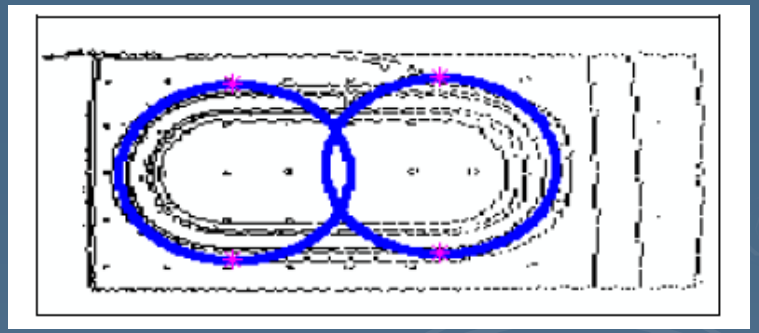
Detecção de curvas

- Uma circunferência define-se facilmente com 3 parâmetros: as coordenadas *x* e *y* do centro e o raio.

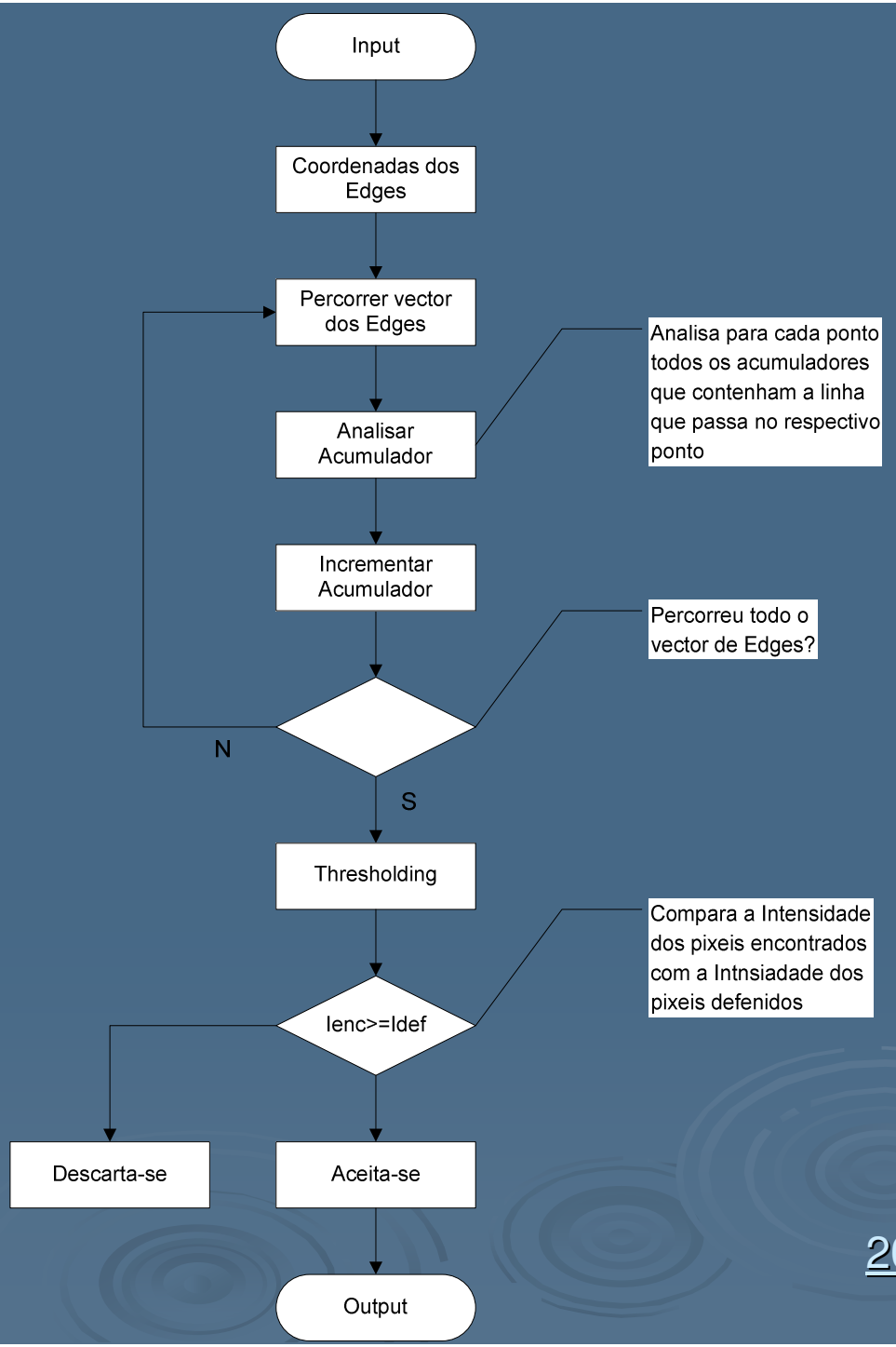
- Para calcular o espaço de acumuladores, limitamo-nos a, para cada raio, desenhar uma circunferência em torno de cada ponto **v** do espaço.

Equação Paramétrica da circunferêmcia:

$$(x - p)^2 + (y - q)^2 = r^2$$



ALGORITMO

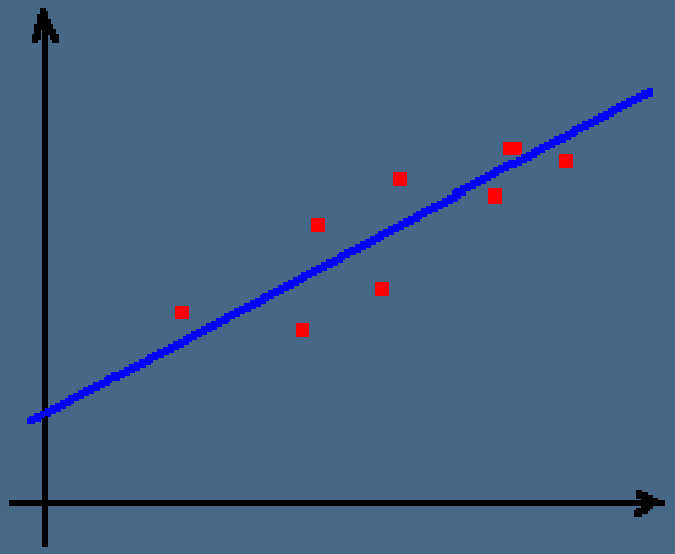


2005/2006

Método dos Mínimos Quadrados

$$S = \sum_{i=1}^{n} (y_i - f(x_i))^2.$$

- Dado um conjunto de pontos, pretendemos saber qual a recta que melhor se adapta aos valores



Regressão Linear

Perspectivas de utilização dos métodos matemáticos

- Qual o método que permitirá uma implementação mais fácil e robusta?
- Qual o método que será mais flexível, para que possa ser viável em qualquer situação?
- Qual o método todo menos pesado do ponto de vista computacional (tempo de processamento)?
- Qual o método que terá o menor erro de aproximação?
- Qual deles permite saber o ponto de transição da passagem de recta para curva?

Capítulo 3

Sumário

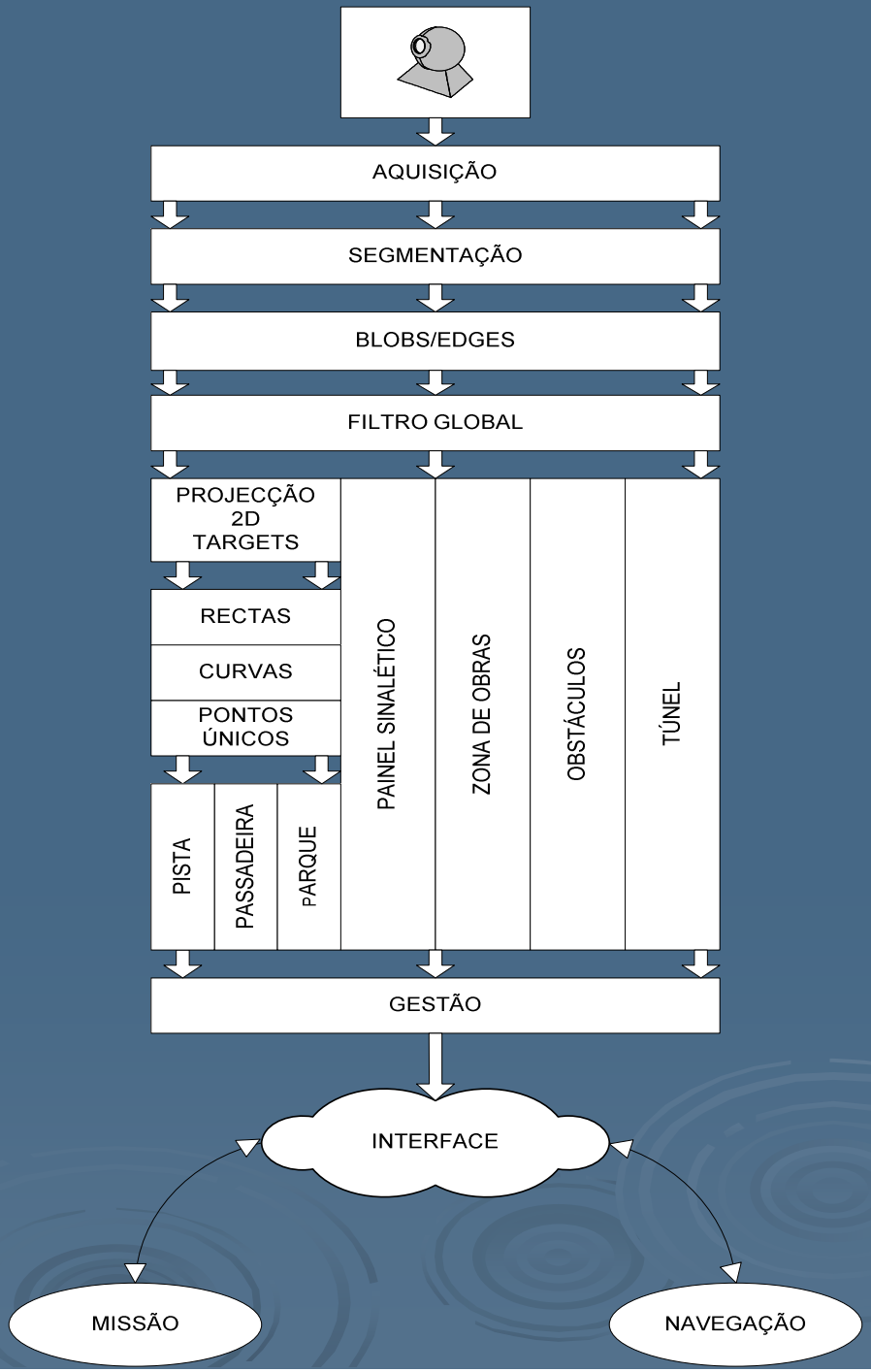
- Arquitectura do Sistema de Visão
- Blocos do Sistema de Visão e estrutura das Funções

Sistema de Visão

Arquitectura do Sistema de Visão

- Aquisição/ Processamento da imagem
- Segmentação
- Detecção de Blobs e Edges
- Filtro Global
- Detecção dos diferentes “targets”
- Módulo de Gestão

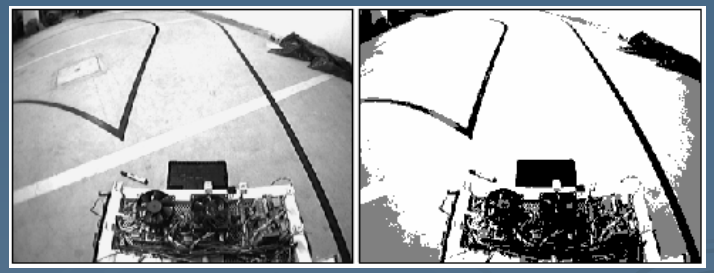
DIAGRAMA DE BLOCOS



Aquisição/Processamento

- A fase de segmentação inicia o processo de análise da imagem quanto ao seu conteúdo, daí esta fase ser muito importante no processamento da imagem, dependendo da qualidade da segmentação realizada teremos uma melhor ou pior qualidade na análise da imagem.

- Existem muitos métodos para fazer a segmentação de imagens, mas não nos interessa abordar estes métodos visto que já recebemos a imagem segmentada e com informação acerca de blobs e edges como falaremos mais à frente.



Segmentação

- O sistema de visão é constituído por dois threads, um dedicado à aquisição cíclica de imagens e outro dedicado ao processamento de imagem.
- Existe um sistema de duplo-buffer para a aquisição e o processamento de cada imagem, um dos buffers armazena a imagem adquirida pela thread de aquisição enquanto o outro vai processar essa mesma imagem.
- Este sistema é eficaz pois mal a imagem seja adquirida a thread de processamento pode rapidamente tratar os dados correspondentes à imagem libertando o outro buffer para a thread de aquisição poder armazenar uma nova imagem adquirida.

Detecção de Edges/Blobs

A detecção de edges e blobs irá fornecer informação sobre a posição dos limites dos objectos e da presença de descontinuidades.

Existem vários métodos para detecção de Edges:

- Método de Canny, Sobbel…
  - Os edges são agrupados por cor, um edge "*preto-branco*" é diferente de um edge "*branco-preto*".
  - Para transições "*preto-branco*", todos os edges estão agrupados na mesma lista, em que cada edge tem vários pontos.

Filtro Global

- O filtro vai ser constituído por um conjunto de testes que permitirão obter uma imagem mais limpa, com informação de interesse que posteriormente será analisada de forma a identificar os diversos "targets".

Vis_Fglobal
Objectivo: Limpar imagem, eliminar o que não interessa Input: Imagem, blobs, edges. Output: Nova imagem + limpa, menos distorção e ruído

Vis_Info

```
typedef struct
{
        Passadeira pas;
        Parque pq;
        Pista pt;
        Semáforo sem;
        Túnel tn;
        Zobras zb;
        Obstaculo ob;
}Vis_Info;
```

Pista, Passadeira, Parque

Vis_Pista

Objectivo:
Detectar as linhas da pista
Input:

Conjunto de linhas, detectado pelo método matemático,escolhido

Output:

linhas_pista: recta/curva
- lado direito
- lado esquerdo
- centro (tracejado)

```
struct{
   float l_esq [];
   float l_dir [];
   float l_ctr[];
}Pista;
```

Vis_Passadeira

Objectivo:
Detectar a passadeira
Input:

Conjunto de linhas, detectado pelo método matemático escolhido

Output:

linhas_verticais da passadeira:

- preto-branco
- branco-preto

```
struct{
   float pbl [n];
   float bp[n];
}Passadeira;
```

Vis_Parque

Objectivo:
Detectar o parque
Input:

Conjunto de linhas, detectado pelo método matemático escolhido

Output:

linhas_verticais do parque:

- preto-branco
- branco-preto

```
struct{
   float l_esq [];
   float l_dir [];
   float l_ctr[];
   float letra[];
}Parque;
```

Semáforos

Vis_Semaforo

Objectivo:
Identificar o semáforo

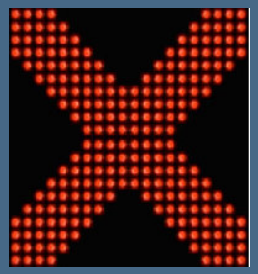
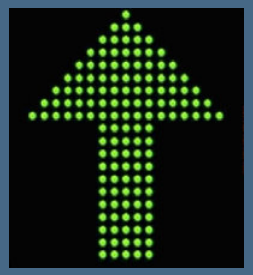
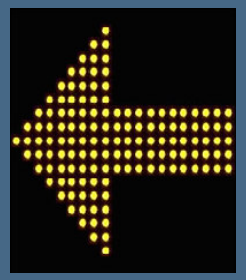
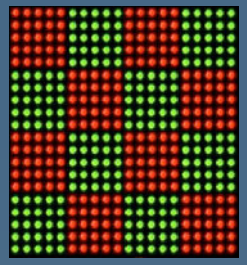
Input:

Blobs+
Conjunto de linhas/circunferências, detectado pelo método matemático escolhido

Output:

semaforo

```
struct{
   char cor[];
   char forma [];
}Semaforo;
```


Zona de Obras

Vis_Zobras

Objectivo:
Identificar a zona de obras

Input:

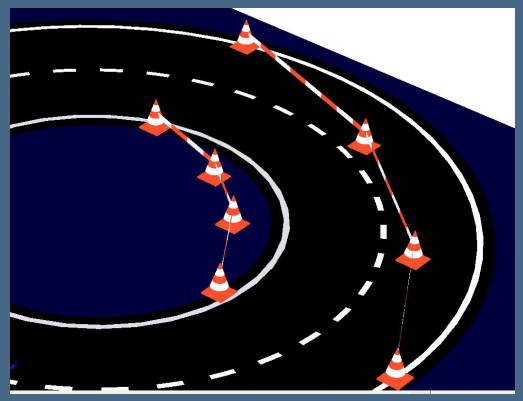
Blobs+
Conjunto de linhas, detectado pelo método matemático escolhido (projecção vertical)

Output:

linhas_verticais do :

- vermelho-branco
- branco-vermelho

```
struct{
   float vb[];
   float bv [];
}Zobras;
```



Obstáculo

Vis_Obstaculo

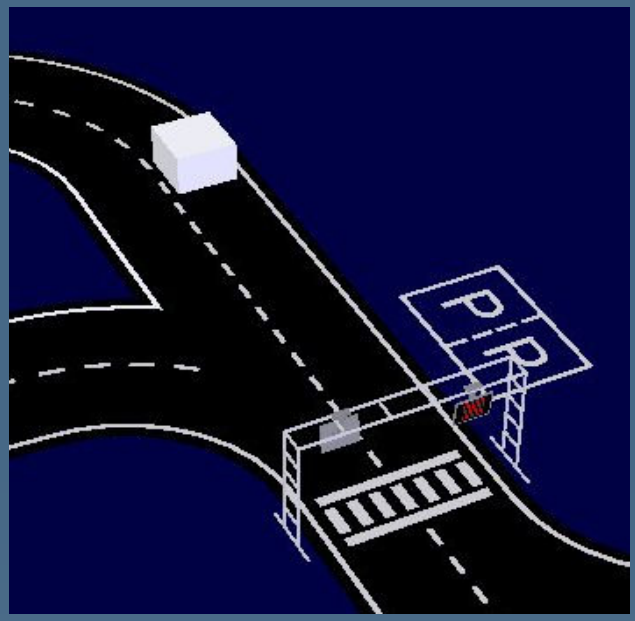
Objectivo:
Identificar o obstáculo
Input:

Blobs+
Conjunto de linhas,
detectado pelo método matemático escolhido

Output:

obstáculo

```
struct{
   char cor[];
   float pb  [];
   float bp  [];
}Obctaculo;
```



Túnel

Vis_Tunel

Objectivo:
Detectar o túnel

Input:

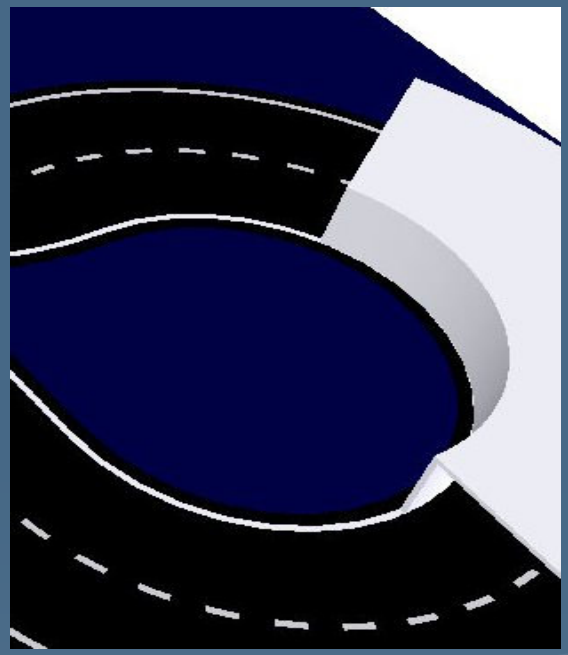
Blobs+
Conjunto de linhas,
detectado pelo método matemático escolhido
(projecção vertical)

Output:

linhas_verticais do túnel:

- preto-branco
- branco-preto

```
struct{
   float l_esq [];
   float l_dir [];
}Tunel;
```



Módulo de Gestão

- A integração de um módulo de Gestão na arquitectura do Sistema de Visão deve-se à necessidade de tornar o sistema dinâmico, haver um interface interactivo e robusto. Pretende-se dispor de um conjunto de ferramentas que permitam a troca de informação de uma forma transparente, “standard” e rápida.
  - Filas de mensagens
  - Modelo Publisher/Subscriber Ocera
  - Zona de memória partilhada + semáforos

Bibliografia

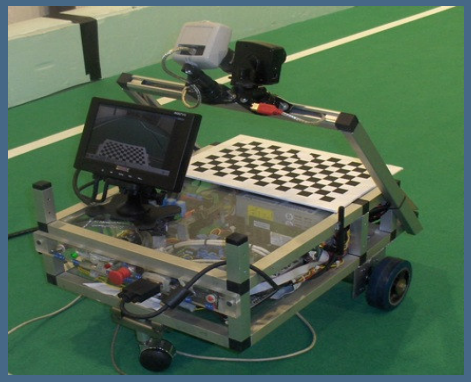
- Relatório de:

  Projecto de Final de Curso de Nuno Moreira: “Projecto Runner - Sistema de Navegação do Veiculo Autónomo do DEE”, concluído em Setembro de 2002.

- Internet:
  www.lsa.isep.ipp.pt/~aprender/pagina_lsa/runner/runner_trabalhos.html

- PDF’s:
  RegrasExpecificaçõesCA2006-v11Out2005

Fim



SPEEDRUNNER

BIGRUNNER

LSIS: Laboratório de Sistemas

2005/2006