Robot de Condução Autónoma

Sistema De Navegação Por Processamento Visual De "Edges"

Disciplina:

LSIS- Laboratório de Sistemas

Docentes:

Eng. José Miguel Almeida

Eng. Alfredo Martins

Discentes:

1000910-António Sérgio Silva

supra_s910@tvtel.pt

1990902-Ricardo Sousa

ricsousa@netcabo.pt

LSA ISEP

RUNNER

Porto, Fevereiro 2006

Índice

Página

Considerações Gerais

O estudo dos problemas associados à robótica constitui hoje em dia uma área de grande interesse e de forte aplicabilidade prática. O crescimento das suas aplicações, bem como interesse demonstrado por grupos de investigação revelam desde já cada vez uma maior importância nas sociedades actuais, seja ao nível de sistemas produtivos, serviços ou lazer.

As aplicações robóticas estão hoje desde já associadas, á substituição do homem em tarefas repetitivas, linhas de montagem, vigilância, operações em ambientes hostis etc.

Os "robots móveis" começam a ser usados em ambientes industriais para transporte de peças entre estações de fabrico, para operar em ambientes hostis, onde a presença humana e complicada e perigosa. Um robot móvel consiste numa plataforma móvel, sobre a qual é integrada de forma inteligente, percepção e acção sobre o mundo que rodeia o robot.

A utilização de sistemas de visão nos robots móveis tem sido um desafio importante para muitas equipas de investigação que desenvolvem actividade nesta área. Um dos grandes objectivos desta área e tentar implementar sistemas de visão em robots móveis que imitem as capacidade visuais dos humanos e dos restantes animais. As várias tentativas de implementação artificial de capacidades visuais, têm revelado estar-se perante um problema complexo e de resolução difícil.

A visão na área da robótica móvel é cada vez mais importante, visto que o conhecimento do meio de actuação por parte do sistema global (ex.: veículo) é fulcral para o seu controlo, onde se torna cada vez mais necessário retirar o máximo de informação sensorial. Esta informação contribuirá para a atribuição de inteligência artificial ao veículo, na medida em que este poderá "ver", "decidir" e "agir" conforme a circunstância, sem a intervenção humana.

Vários protótipos de robots móveis com navegação suportados por informação visual têm sido desenvolvidos e onde já se consegui feitos bastante importantes.

CAPÍTULO 1

O capítulo 1 é constituído pela formulação/exposição do problema, pelos objectivos do projecto e pelo estudo da arte. O objectivo principal deste é abordar os princípios fundamentais da Visão Computacional, incluindo alguns tópicos de Visão Robótica. Visão Computacional tem tido um papel cada vez mais relevante para o homem, na medida em que ela passa a colaborar em diversas tarefas essenciais, em particular no sector produtivo e em diversão. Trata-se de uma área de pesquisa que é inerentemente multidisciplinar, e onde os conhecimentos de Computação além de fundamentais, são aplicados extensivamente.

1. Formulação do problema

Sistema de navegação por processamento visual de "edges"
para robot de condução autónoma

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.

Figura 1: a) Esquerda – Robot SpeedRunner b) Direita – Robot BigRunner

"Sistema de navegação por processamento visual de "edges" para robot de condução autónoma". Decidimos chamá-lo de robot condução autónoma ao invés de veículo autónomo terrestre, porque estamos a desenvolver um Sistema de Visão que disponibilize informação "útil" para o sistema de navegação e gestor de missões dos robots "Runner", do grupo de investigação do Laboratório de Sistemas Autónomos do Isep, LSA, e também por uma questão de precisão, pois o suporte onde vamos testar o nosso trabalho são os robots "Runner".

2. Sistemas de Navegação e Visão

Sistema de Navegação

Um sistema de navegação é um sistema com uma percepção do mundo envolvente, utilizando para isso vários meios tais como GPS, Sonar, Radar, **Visão**.
Quanto mais rigoroso e evoluído for o sistema de navegação, melhores e mais precisos vão ser os dados provenientes, o que resulta numa maior confiança no sistema e consequente desempenho de quem o utiliza.
No nosso caso em particular, o problema surge quando se pretende navegar com dados provenientes de sensores (câmara mais lente), interpretá-los de modo a melhorar a compreensão do mundo e disponibilizá-los depois de devidamente "tratados" de forma a se propor a melhor acção.
Queremos com isto dizer que o nosso objectivo principal é o fornecimento de informação útil ao sistema global dos robots "Runner", relativamente à sua posição/orientação na pista de prova.
O trabalho vai se centrar na análise de imagens, deixando de parte o processamento das mesmas, isto é, não vamos realizar o estudo sobre a aquisição das imagens mas sim o seu tratamento pormenorizado.

Sistema de Visão

Dotar um robot de visão, é uma tarefa ambiciosa que implica a capacidade de gerir múltiplas tarefas simultaneamente (THREADS), que é como quem diz ser capaz de processar dados de sensores exteriores e fundi-los de forma a se mover autonomamente. É basicamente o que o cérebro de um ser humano faz quando, quando recebe a informação sensorial dos olhos e depois a transmite aos membros inferiores encarregues da locomoção.

A visão num robot aumenta significativamente as suas capacidades sensoriais e consequentemente operacionais, bem como a sua versatilidade e segurança de funcionamento.

Pretende-se com a visão robótica poder determinar as posições e propriedades dos objectos, assim como as relações entre os mesmos e o ambiente que os cerca, e assim poder realizar tarefas ou "missões", tais como, navegar entre os objectos, detectá-los num determinado percurso predefinido, tal como se pretende no FNR; é o caso dos semáforos, as linhas da pista, a passadeira, pontos únicos da pista como a intersecção das linhas interiores (fazem um ângulo muito fechado) e outros consoante as "missões".

Figura 2

3. Descrição dos objectivos

Os objectivos do projecto são identificar objectos, determinar as suas propriedades e disponibilizá-las às camadas superiores do sistema global dos robots "Runner".
A prova de condução autónoma do FNR, é uma pista em forma de oito com dimensões conhecidas (capítulo 3), e é nesta pista que vamos identificar os vários "targets".

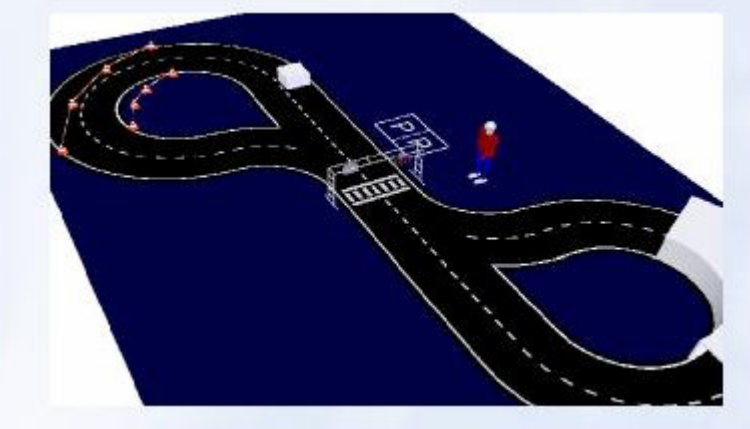


Figura 3

- **Targets do Sistema de Visão:**
  - Linhas da pista e determinar, ao longo do percurso percorrido pelo robot, se estas são rectas ou curvilíneas.
  - Zona de parque
  - Passadeira
  - Semáforos
  - Zona de obras
  - Túnel
  - Obstáculos

4. Estado da arte

Com o desenvolvimento de tecnologias de navegação para veículos autónomos e o aumento da capacidade de processamento dos computadores, apareceram os primeiros robôs móveis industriais, embora o seu âmbito de aplicação seja muito mais alargado. Um robot móvel procura conjugar num só dispositivo de automação a mobilidade de um veículo autónomo e a capacidade de manuseamento e manipulação dos robôs. Neste domínio, o AGV (Automatic Guided Vehicle) e o AS/RS (Automatic Storage/Retrieval System) são os dispositivos com maior aplicação em empresas industriais ou de distribuição. Em Portugal, a empresa EFACEC – Automação e Robótica, é pioneira na área de robótica móvel e é autora de inúmeros projectos com aplicação a nível nacional e internacional.

Um sistema flexível de fabrico consiste num conjunto de máquinas e num sistema de transporte e manuseamento de materiais, ligados e controlados através de uma rede de computadores. Antes do aparecimento da robótica móvel, o transporte e manuseamento de materiais era realizado com uma forte intervenção humana (ex.: veículos guiados manualmente). Com o desenvolvimento de tecnologias de navegação para veículos autónomos e o aumento da capacidade de processamento dos computadores, apareceram os primeiros robots móveis industriais, com o objectivo de reduzir a intervenção humana nos sistemas flexíveis de fabrico.
A utilização de robots móveis permite:

- Aumentar o grau de automação e flexibilidade, facilitando a integração total e o controlo optimizado do sistema, através de uma rede de computadores.
- Optimizar o fluxo de materiais, através de um correcto escalonamento das tarefas, contribuindo para uma melhoria significativa de produtividade global do sistema.
- Eliminar a presença humana em ambientes potencialmente agressivos e perigosos para a saúde (ex.: industria pesada, química, nuclear, etc.).

Um robot móvel procura conjugar num só dispositivo a mobilidade de um veículo autónomo e a capacidade de manuseamento e manipulação dos robots convencionais (ex.: braços robots, etc.).

O aparecimento de robots móveis trouxe consigo reconhecidas vantagens, mas também inúmeros desafios tecnológicos, como sejam:

- A concepção de veículos autónomos, alimentados a partir de baterias de tracção recarregáveis com maior autonomia possível;
- O desenvolvimento de sistemas de navegação automáticos, que permitam a sua mobilidade automática de forma eficiente, flexível, tolerante a falhas e segura;
- O controlo eficiente de frotas constituídas por vários robots moveis, resolvendo problemas como o escalonamento optimizado, o encaminhamento e a gestão de tráfego.

Assim o desenvolvimento de robots móveis é uma tarefa fortemente interdisciplinar, envolvendo áreas tecnológicas tão diversas como: sensores e actuadores, electrónica de potencia, energia, projecto mecânico, cinemática, dinâmica, teoria de controlo, escalonamento em tempo-real, investigação operacional, sistemas de informação, telecomunicações, etc.
A nível internacional, o âmbito de aplicação dos robots móveis não se tem confinado á indústria, abrangendo áreas de logística (distribuição e armazenagem), exploração subaquática e oceanografia, exploração planetária, bem como aplicações militares.

Em Portugal, os projectos industriais de robótica móvel já realizados tiveram aplicação principalmente na indústria (fabricas e células flexíveis de fabrico), na logística de cadeias de distribuição e armazenagem e nos serviços. Neste tipo de aplicações destacam-se dois tipos de dispositivos: o AGV (Automatic Guided Vehicle) e o AS/RS (Automatic Storage/Retrieval System), vulgarmente designado por Armazém Automático.
Os primeiros projectos de AS/RS's realizados em Portugal tiveram lugar na década de 80. Os primeiros projectos de AGV's tiveram lugar no início da década de 90. Em ambos os casos a empresa EFACEC – Automação de Robótica foi pioneira em Portugal, tendo já realizado inúmeros projectos nessa área, com aplicações em Portugal e no estrangeiro.

AGV

O AGV é um robô móvel utilizado para transporte e manuseamento automático de materiais. As suas principais características são:
- Poder movimentar-se de forma autónoma ao longo de um layout de trabalho, sem qualquer intervenção humana;
- Poder responder a uma tarefa de movimentação entre dois pontos, seleccionando uma trajectória ou caminho, percorrendo esse caminho de forma autónoma, parando com precisão na posição final, onde procede de forma automática a uma operação de carga ou descarga de uma carga a transportar.

Sistemas de navegação

Existem diversos métodos de navegações que permitem a um AGV seguir um caminho fixo ou um caminho dinâmico. A determinação de caminhos fixos ou dinâmicos depende dos custos de instalação, dos requisitos de flexibilidade e da necessidade ou não da futura expansão do sistema. Os sistemas com caminhos fixos são menos dispendiosos, todavia inviabilizam a possibilidade de reagir a alterações de layout de trabalho sem interromper o seu funcionamento, acarretando por isso custos adicionais.

O sistema filoguiado é um exemplo de um método de caminhos fixos, que é usado em grande escala, devido a sua robustez e simplicidade. Baseia-se no seguimento do campo magnético criado por condutores implantados no solo e percorridos por uma corrente eléctrica sinusoidal. O campo magnético é detectado por antenas, que seguem a frequência correspondente ao caminho a seguir. Estes sistemas tem sido largamente utilizado devido a sua simplicidade e robustez, mas tem a desvantagem de não permitir a reconfiguração do layout.
Os sistemas de caminhos dinâmicos são mais flexíveis perante modificações no layout e nos requisitos de capacidades, mas são mais complexos e dispendiosos. Navegação por visão artificial, por laser e por GPS (em aplicações exteriores), são exemplos de sistemas de caminhos dinâmicos.

Dentro deles, o sistema de navegação laser é o que tem tido mais implantação no mercado, estando a substituir rapidamente o sistema filoguiado, com as vantagens já apontadas. Neste tipo de sistema, o AGV é equipado com um laser scanner, que realiza constantemente um varrimento rotativo, detectando a posição de painéis reflectores colocados ao longo do layout. A posição do AGV é determinada com base na triangulação dos feixes de luz reflectidos. Este sistema permite reconfigurar facilmente o layout, bastando para isso alterar a colocação dos painéis reflectores e reprogramar a configuração do AGV.

AS/RS

O AS/RS vulgarmente designado por Armazém Automático é um sistema fundamental de suporte ao manuseamento e armazenagem automáticos de materiais em sistemas de fabrico automatizados e em cadeias de distribuição de produtos automatizados.
A sua utilização tem como principais vantagens:
- A possibilidade de automatizar as operações de armazenagem e de integrar o controlo automático dessa operações no controlo global do sistema;
- A redução de custos e o aumento da produtividade em operações de armazenagem;
- A automatização total do controlo de inventários;
- O aumento da capacidade de armazenagem, resultante de uma melhor utilização do espaço disponível;
- Redução dos danos (físicos e materiais) resultantes de acidentes ou situações anómalas nas operações de armazenagem.

5. Projecto Runner – Robot de Condução Autónoma

- Descrição do Robot

Para entender todos os conceitos utilizados na realização deste trabalho e assim poder aprender mais sobre veículos de condução autónoma foi necessário tentar identificar/entender a estrutura tanto de software como hardware dos robots "Runner". A bibliografia usada para perceber toda a arquitectura dos robots foram os relatórios dos alunos dos anos anteriores responsáveis pelo desenvolvimento dos mesmos, o seu testemunho e experiência transmitidos e o software desenvolvido pelos membros do grupo de investigação do Laboratório de Sistemas Autónomos, LSA.

Na parte de hardware identificamos os seguintes componentes:

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

Na parte de software identificamos as seguintes áreas:

- Hodometria
- Sistema de Visão
- Fusão sensorial

5.1 - Descrição do hardware do Robot

A arquitectura do hardware do robot é constituída basicamente pelos seguintes blocos: alimentação, computador, sensores, actuadores e comunicação. Na figura xx é possível visualizar a interligação entre os blocos, de forma a constituir a arquitectura do hardware:

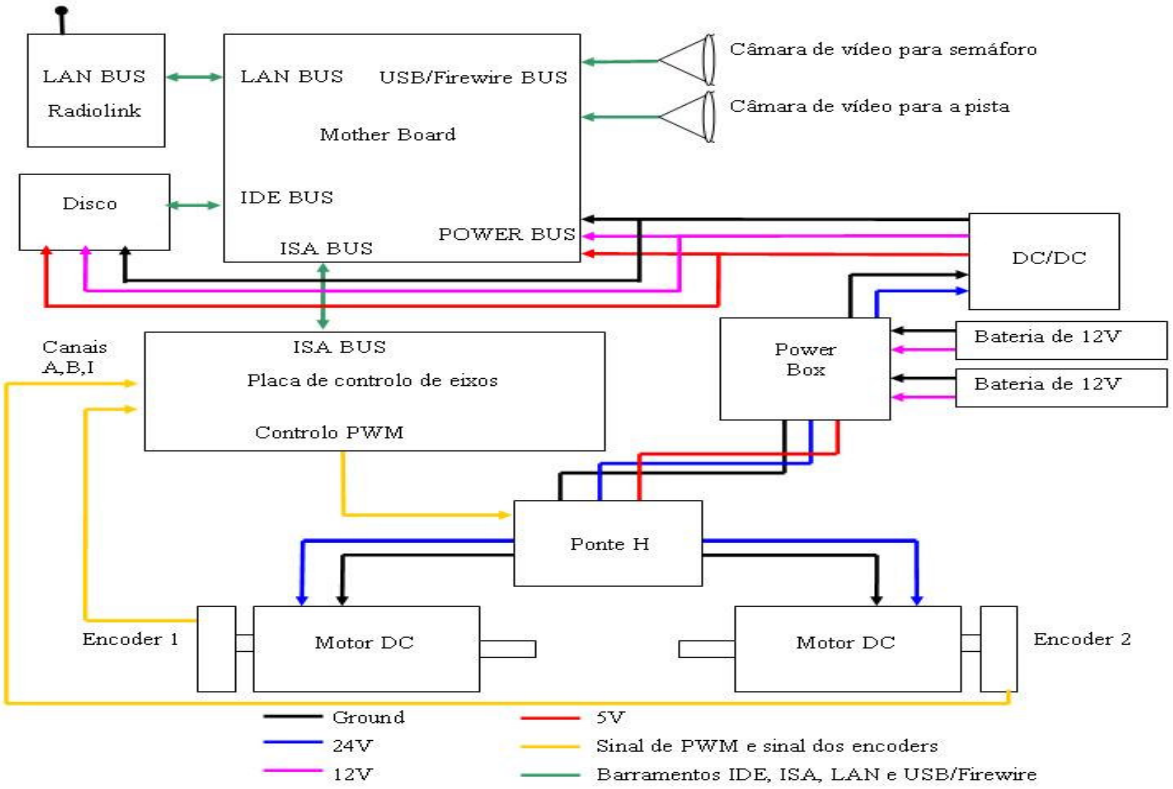


Figura 4: Esquema geral do hardware do robot

5.2 - Arquitectura do software do Robot

O software utilizado na implementação do sistema de navegação do robot foi desenvolvido em linguagem C. A escolha deste tipo de linguagem de programação é justificada pelo facto de ser uma linguagem bastante poderosa e familiar.

O software desenvolvido é constituído por dois blocos relativos aos sensores utilizados: Hodometria, e Visão. A informação proveniente dos sensores é fundida para a determinação do estado do veículo. O bloco da Fusão sensorial é o responsável pela determinação da posição e orientação do robot – navegação do robot.

A estimativa da localização do veículo foi aperfeiçoada pela utilização do Filtro de Kalman Extendido *(Extended Kalman Filter)* que lineariza as equações do estado e funde a informação sensorial.

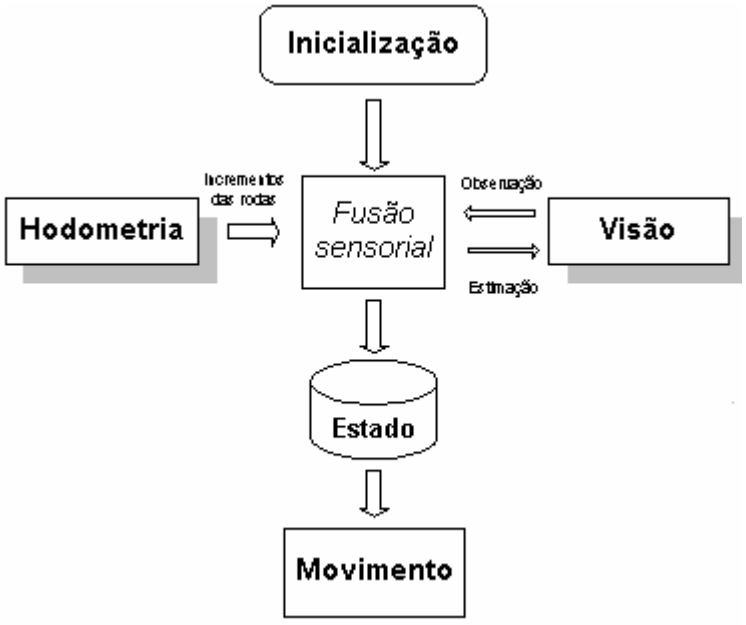


Figura 5: Arquitectura do software do sistema de navegação

Os blocos de software ilustrados na figura anterior são descritos nas secções seguintes:

5.2.1 - Hodometria

A inicialização da hodometria do robot é realizada pela função hodom_init(). Nessa função é criado o thread da hodometria que lança a função hodom(), a qual executa em ciclo fechado a leitura dos incrementos das rodas, a partir das leituras dos encoders ópticos. A leitura desses valores é efectuada pela função axisctr_rd_cnt().

O interface da informação hodometria com o filtro de Kalman é realizado na função update_robot_pos() .Esta função actualiza o estado do robot com os valores da previsão do filtro.

5.2.2 - Sistema de Visão

O sistema de visão é constituído por dois threads, um dedicado à aquisição cíclica de imagens e outro dedicado ao processamento de imagem. Este threads são criados na função vision_init(). A função que realiza a aquisição é a vision_acquire_head(). O processamento é realizado pela função vision_process_head(), o qual é constituído por três blocos: Pesquisa do alvo, Compensação da distorção e Conversão dos referenciais. O interface do resultado final do processamento de imagem com o filtro de Kalman é realizado na função pose_estimation().

O bloco da Calibração não faz parte do sistema de visão porque é realizada em off-line, no qual a intervenção humana é necessária. Este assunto é abordado pormenorizadamente na secção da Calibração do Sistema de Visão.

A figura 6 ilustra a arquitectura do sistema de visão:

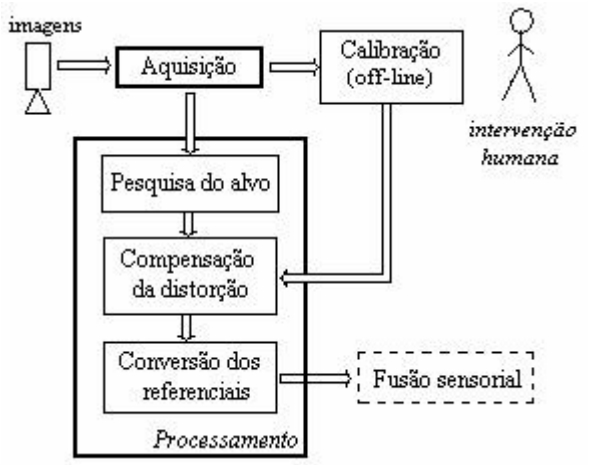


Figura 6: Arquitectura software do sistema de visão

5.2.2.1 - Controlo e classificação de imagem

Um dos requisitos da implementação do sistema de visão é desenvolvimento de funções que permitam o controlo de imagem e a classificação correcta das cores. As funções que solucionam esse requisito são vis_read_cam_opt() e vis_make_color_img(), respectivamente .

O controlo de imagem é um dos problemas da implementação de um sistema de visão, dada a sua dependência com o meio ambiente. A má iluminação e os reflexos da pista de teste exigiu um minucioso ajuste das características da imagem, de forma a que o contraste entre a cor do chão e das linhas seja bem definido. Das muitas tentativas realizadas, é importante ilustrar os três casos seguintes:

Caso 1 – Brightness: 60000; Contrast: 65535; Hue: 65535; Colour: 0

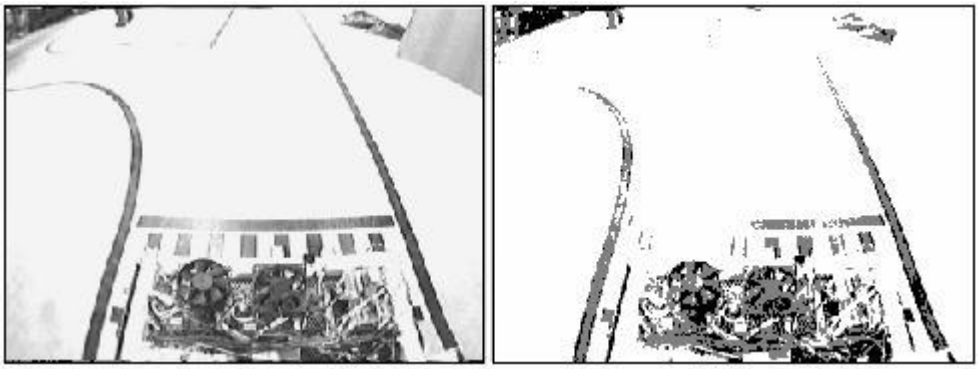


Figura 7: a) imagem original; b) imagem classificada

Observação – No caso 1, o excesso de brilho tem como consequência a má definição das linhas da imagem classificada. As linhas são quase imperceptíveis.

Caso 2 – Brightness: 20000; Contrast: 65535; Hue: 65535; Colour: 0

Figura 8: a) imagem original; b) imagem classificada

Observação – No caso 2, o pouco brilho utilizado torna a imagem muito escura, tornando difícil a distinção das linhas em determinadas zonas.

Caso 3 – Brightness: 35000; Contrast: 65535; Hue: 65535; Colour: 0

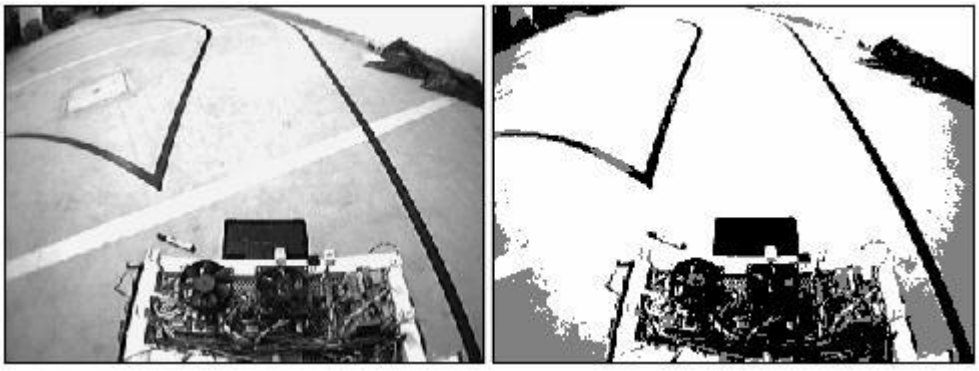


Figura 9: a) imagem original; b) imagem classificada

Observação – No caso 3, o contraste entre as linhas e o chão da pista está bem definido, apesar de se verificar o reflexo da iluminação na zona em linha faz um "v". Os valores utilizados no controlo da imagem podem variar entre 0 a 65535 (mínimo e máximo, respectivamente).

5.2.2.2 - Algoritmo de Pesquisa do Alvo

O algoritmo de pesquisa dos pontos das linhas limites da pista baseia-se numa procura de pixels pretos ao longo de um segmento. A procura dos pixels pretos da linha é realizada numa aérea de pesquisa constituída por duas zonas (esquerda e direita). A restrição da área de pesquisa visa diminuir o tempo necessário para varrer todos os pixels dessa área, optimizando o processamento de imagem. A divisão da área de pesquisa em duas zonas tem como objectivo facilitar a distinção lógica das linhas, ou seja, identificar se é a linha esquerda ou direita relativamente ao referencial do robot.

A validação da linha em cada zona é efectuada pela continuidade do percurso de pesquisa num dado número de linhas da imagem. O valor do número de linhas que validam o segmento é definido pela diferença de linhas entre o ponto inicial e final. Se não existir continuidade entre esses limites o segmento não válido, logo não é considerado como pertencente à linha.

A função que realiza a análise dos pixels ao longo das linhas é a vis_search_vertical() . A análise do pixels ao longo das colunas é feita pela função vis_search_horizontal().

Na figura abaixo é possível verificar que na zona esquerda o algoritmo realiza por acção da função vis_search_line() um varrimento horizontal de várias linhas até encontrar pixels pretos de uma mancha escura do chão. Quando é detectado o primeiro pixel dessa mancha, as funções anteriores traçam um percurso assinalado de vermelho ao longo da mancha. No entanto, essa mancha de pixels pretos não é validada com um segmento das linhas da pista, devido à não continuidade num número mínimo de linhas da imagem.

Na zona direita da área de pesquisa a validação do segmento de recta foi positiva, visto que o percurso efectuado pelo algoritmo ao longo dessa mancha de pixels pretos atingiu o número mínimo de linhas da imagem.

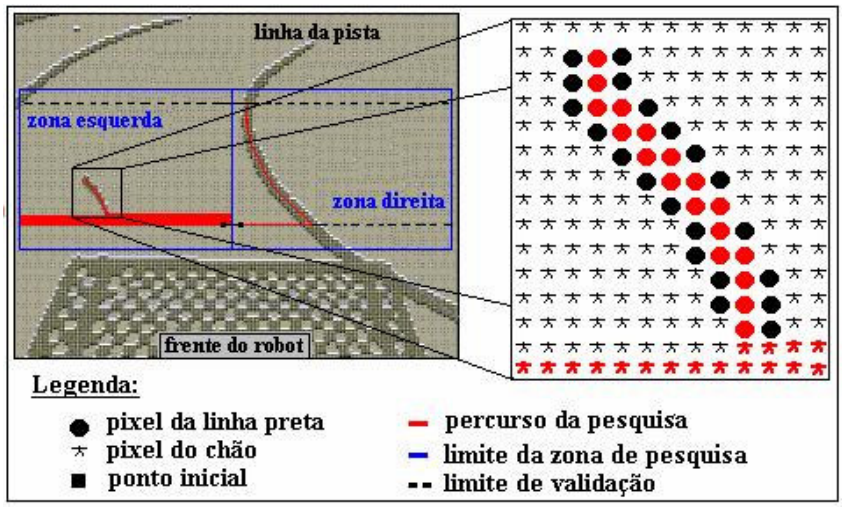


Figura 10: Pesquisa dos pontos da linha na imagem

Quando um segmento de pixels pretos é validado como parte de uma linha da pista, determinados pixels são convertidos em pontos na imagem e depois compensados pela Look Up Table (matriz) de calibração. Esta compensação é realizada pela função v_calib_func(). Após a devida compensação da distorção, provocada pela lente da câmara, o pontos no referencial da imagem são convertidos para o referencial do robot pela função v_world_xyz().

5.2.3 - Fusão sensorial

A fusão sensorial tem como base as duas funções do filtro de Kalman: time_update() e measurement_update(). A primeira realiza a previsão do estado do robot e da confiança do sistema (matriz P) a partir da informação da hodometria (incrementos dos encoders-rodas). A segunda realiza correcções da posição e orientação do robot segundo os erros entre as observações e as estimativas da posição de alvos (landmarks) no mundo, que neste caso são as linhas. As estimativas da posição dos alvos no mundo são realizadas pela função pose_estimation(). Por outras palavras, esta função faz o matching (emparelhamento) do pontos observados no mapa do mundo.

A figura 11 ajuda a compreensão da fusão da informação proveniente dos sensores utilizados:

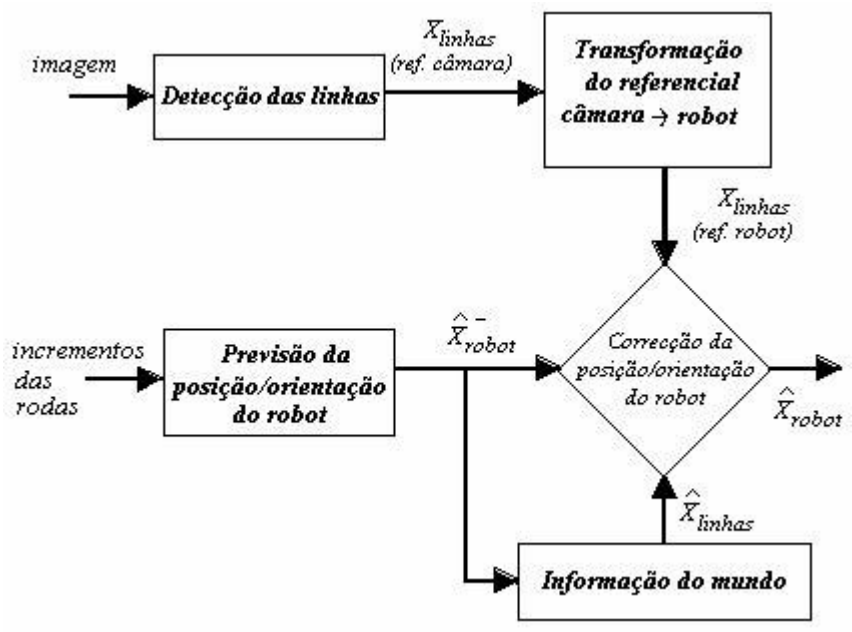


Figura 11: Estimação da posição/orientação do robot

A grande vantagem da fusão sensorial é a utilização simultânea muitos sensores com características que podem ser semelhantes ou completamente diferentes.

CAPÍTULO 2

No capítulo 2 são apresentados os **fundamentos teóricos,** desde conceitos base a sistemas de coordenadas, características do sistema operativo e formalismos matemáticos de alguns métodos para a identificação de rectas, curvas e pontos únicos, tais como Transformada de Hough e Método dos Mínimos Quadrados.
Estes fundamentos, sem os quais não poderíamos progredir, vão servir de base para o desenvolvimento do projecto.

1. Conceitos Base: Edges, Blobs

Os Edges e Blobs são fulcrais para o entendimento e execução do trabalho daí que, tivemos de estudar o que são e quais as suas aplicações.

Edges

Um "Edge" numa imagem é a fronteira entre dois valores significativamente diferentes do pixel (preto - branco ou branco - preto). Assim, deve-se procurar na imagem estas mudanças a fim de identificar "Edges".

A detecção de "Edges" é uma ferramenta eficaz para muitos aplicações da visão da máquina. A detecção de "Edges" fornece a aplicação com informação sobre a posição dos limites dos objectos e da presença das descontinuidades. Normalmente usa-se a detecção de edges nas seguintes áreas de aplicação:

- Calibrando, detecção, e alinhamento.
- Pode-se usar posições múltiplas de Edges para calcular medidas como pontos de intersecção, projecções, e ajustes do círculo ou da elipse

Blobs

" A group of pixels organized into a structure is commonly called a blob"

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

Nas aplicações onde há uma variação significativa na forma ou na orientação de um objecto, a análise de "Blobs" é uma maneira poderosa e flexível de procurar pelo objecto. Pode-se usar uma combinação das medidas obtidas com a análise de "Blobs" para definir uma característica ajustada que defina excepcionalmente a forma do objecto. Se o objectivo for contar todos os "Blobs" que se encontram numa imagem, a primeira etapa é identificar os "Edges" de cada objecto.

2. Sistema de coordenadas

Os sistemas de coordenadas desempenham um papel muito importante na localização do robot em relação ao mundo, localização dos objectos em relação ao robot e localização dos objectos em relação à câmara.
São sistemas que representam pontos no espaço a 2 ou 3 dimensões. Entre 1596 e 1650 René Descartes introduziu sistemas de coordenadas baseadas em coordenas ortogonais (ângulos rectos).

2.1 – Sistemas cartesianos

A utilização de uma câmara no robot requerer a definição de um sistema de coordenadas próprio para a câmara, que geralmente é diferente dos sistemas de coordenadas do robot. Desta forma temos quatro sistemas de coordenadas cartesianas: *sistema de coordenadas do mundo, do robot, da câmara e da imagem.*

2.1.1 – Sistemas de coordenadas do mundo

Face à necessidade de localização do robot no meio envolvente, que neste caso é a pista do "oito", fixou-se a origem e a orientação dos eixos do sistema de coordenadas do mundo como ilustra a figura 3. Neste sistema é descrito o meio envolvente, ou seja o mundo, que é constituído por objectos, robot e câmara. Este sistema é definido pelos eixos de coordenadas X_w, Y_w e Z_w.

2.1.2 – Sistemas de coordenadas do robot

É definido por conveniência e sem perda de generalidade, a partir do eixo de hodometria e não do centro de massa do robot. A área ou o volume ocupado pela estrutura no meio envolvente é localizada em função do eixo de hodometria. Este sistema é definido pelos eixos de coordenadas X_r, Y_r e Z_r.

2.1.3 – Sistemas de coordenadas da câmara

É definido a partir da posição da câmara no robot. É com base neste sistemas de coordenadas que a projecção e o campo de visão é definido. Este sistema é definido pelo eixo de coordenadas X_c, Y_c e Z_c.

A figura seguinte ilustra os sistemas de coordenadas descritos anteriormente:

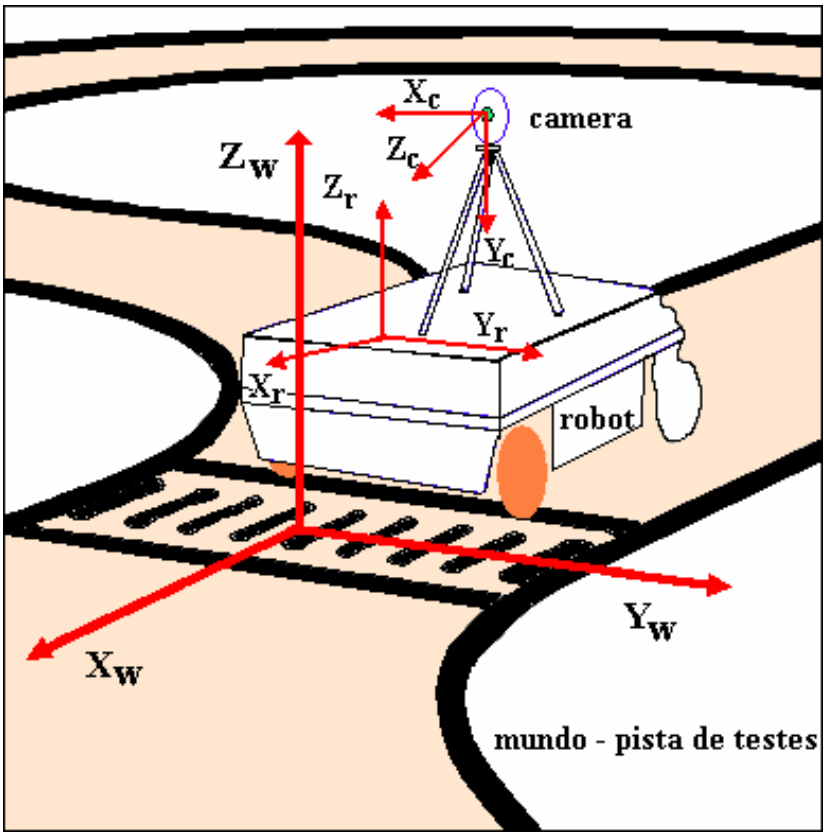


Figura 1: Sistemas de coordenadas

2.1.4 – Sistemas de coordenadas da imagem

Nas câmaras digitais, os sensores CCDs estão ordenados em forma matricial. Assim o sistema de coordenadas da superfície do sensor é solidário aos foto-detectores, possibilitando a realização da orientação interna do sensor sem ser necessário recorrer a marcas de referência (marcas fiduciais). A partir do sistema de coordenadas da superfície do sensor pode-se, por transformação, obter as coordenadas de todos os pontos (pixels) no sistema de coordenadas da imagem u,v (figura 3). È necessário garantir que o sensor permaneça fixo e perpendicular em relação ao eixo óptico do sistema óptico.

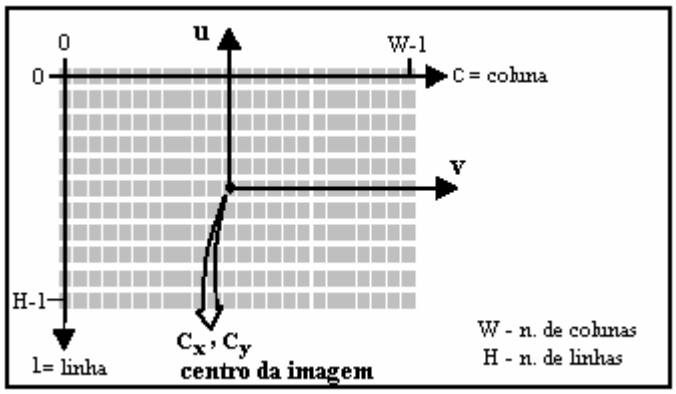


Figura 2: Sistema de coordenadas da imagem

Um ponto P_w no referencial do mundo $\{O_w, X_w, Y_w, Z_w\}$ é representado no referencial da câmara $\{O_c, X_c, Y_c, Z_c\}$ através de duas matrizes. Essas matrizes são a matriz de *Rotação (R)* e a matriz de *Translação (T)*. A matriz *R* tem dimensões 3x3 e descreve a orientação da câmara relativamente ao mundo. A matriz *T* tem dimensões 3x1 e descreve a posição da câmara no mundo. Estas duas transformações constituem os *parâmetros extrínsecos* da câmara.

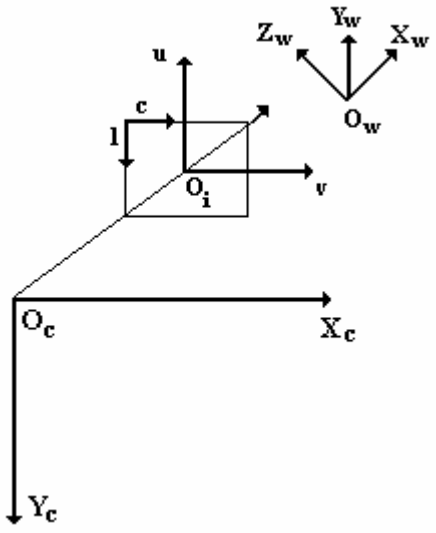


Figura 3: Sistema de coordenadas

A representação do ponto P_w no referencial da câmara (P_c) é dado pela seguinte equação:

$$P_c = R * P_c + T$$

O ponto P_c é representado no referencial da imagem $\{O_i, u, v\}$ através do modelo utilizado para a projecção em perspectiva descrito anteriormente.

$$u = f \frac{X_c}{Z_c}$$
$$v = f \frac{Y_c}{Z_c}$$

É importante salientar que o sistema de coordenadas também está relacionado por uma matriz de rotação (3x3) e uma de translação (3x1). Desta forma existem as seguintes matrizes:

R_{wr} - matriz de rotação entre o mundo e o robot

R_{rc} - " de rotação entre o robot e a câmara

R_{wc} - " de rotação entre o mundo e a câmara

T_{wr} - matriz de translação entre o mundo e o robot

T_{rc} - " de translação entre o robot e a câmara

T_{wc} - " de translação entre o mundo e a câmara

A relação entre os pontos será sempre dada pela equação:

$$P' = R * P + T$$

As figuras 4 e 5 exemplificam a posição actual da câmara no robot. O sistema de visão tem a câmara fixa, ao contrário de outros sistemas que utilizam câmaras rotativas. O facto da câmara encontrar-se fixa tem um motivo que é a diminuição de incertezas nas observações efectuadas pelo sistema de visão.

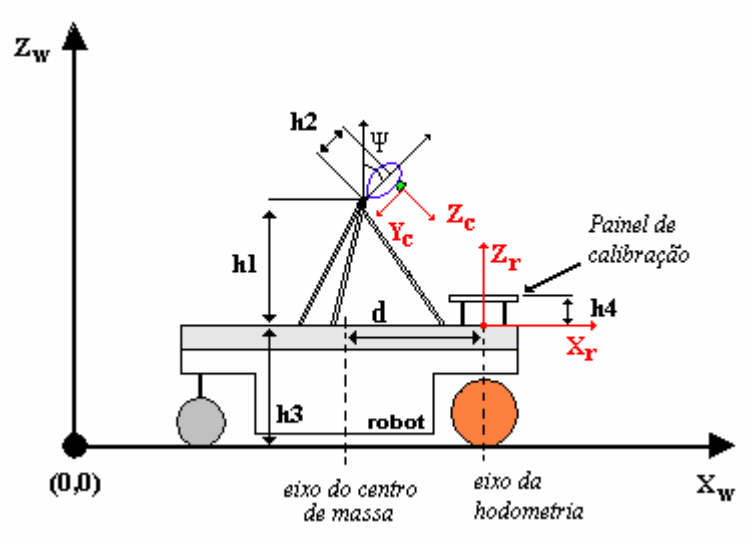


Figura 4: Vista de perfil do robot

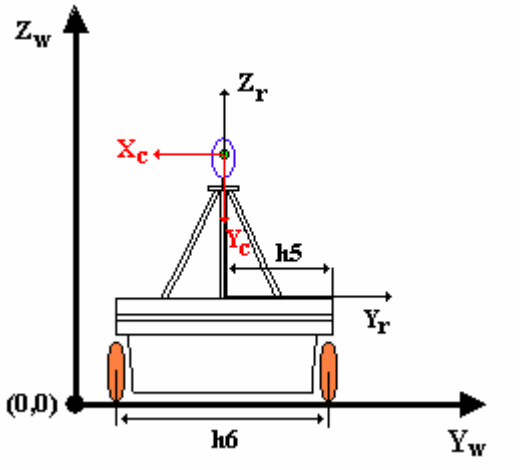


Figura 5: Vista de frente do robot

3. Sistema Operativo

Achamos importante conhecer algumas características importantes do sistema operativo no qual vamos desenvolver os algoritmos que vão fazer parte integrante da arquitectura do Sistema de Visão.

Na nossa opinião a arquitectura deve ser projectada de forma a tiramos o máximo partido do sistema operativo, ou seja, explorar todas as virtudes que o sistema operativo Linux (distribuição Fedora Core) dispõe tais como o facto de ser um sistema operativo com característica de tempo real e politicas de escalonamento entre outras.
Este sistema operativo foi escolhido, porque para além de ser possuidor de vantagens, sobre outros sistemas, vem no seguimento dos objectivos do projecto.

- **Vantagens do Linux em relação aos outros sistemas operativos:**

  - 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.
  - Possibilidade de trabalhar com vídeo.
  - Apoio a nível mundial.
  - Vantagens didácticas.
  - Sistema operativo de Tempo Real (*SOTR*).

Sistemas Tempo Real

Sistemas computacionais de tempo real são definidos como aqueles submetidos a requisitos de natureza temporal. Nestes sistemas, os resultados devem estar correctos não somente do ponto de vista lógico, mas também devem ser gerados no momento correcto. Os aspectos temporais não são limitados a uma questão de maior ou menor desempenho, mas estão directamente associados com a própria funcionalidade do sistema.
Um sistema de tempo real é um sistema em que os limites de tempo são muito importantes, são sistemas que reagem, possuem tarefas que são executadas concorrentemente, e não são determinísticos. Portanto, modelos de sistemas em tempo real necessitam de especificações para requisitos de tempo, manipulação de eventos assíncronos, comunicação, concorrência e

sincronização. Como os sistemas tempo reais são frequentemente distribuídos também precisam mostrar a distribuição do sistema.

- Requisitos Temporais: O sistema executa suas funções dentro de limites de tempo específicos ("tempo de resposta"). Todos os limites (*deadlines*) devem ser obrigatoriamente cumpridos.

- Reactivo: O sistema é reactivo, ou seja, está continuamente respondendo a eventos do ambiente externo.

- Concorrência: Executa tarefas (*threads*) de controlo de forma concorrente, onde diferentes partes do software estão a ser processadas em paralelo. O paralelismo pode ser real (quando existem vários processadores) ou simulado (por sistemas operacionais tempo real nos quais as tarefas são escalonadas para compartilhar o processador).

- Não determinismo: Representa a impossibilidade de provar formalmente que um sistema irá trabalhar em todas as situações sob todas as condições. Isto deve-se à complexidade da concorrência, aos eventos assíncronos externos que podem ocorrer em qualquer ordem e ao hardware que esta envolvido.

Sistemas em tempo real podem ser classificados em críticos (*hard real-time*) e não críticos (*soft real-time*). Nos sistemas críticos, um atraso ou resposta incorrecta é considerado um erro inaceitável que pode resultar em danos graves. Alguns exemplos são os softwares de controlo de tráfego aéreo, bem como sistemas de controlo de aparelhos médicos. Sistemas em tempo real não críticos podem aceitar ocasionalmente uma resposta atrasada, por exemplo um sistema de comunicação digital ou um sistema de vídeo-conferência.
O uso de sistemas em tempo real simplifica o projecto de um sistema. De um modo geral, um sistema pode sempre ser decomposto num conjunto de processos. A função do sistema em tempo real é gerir esses processos atribuindo-lhes "espaço" para que cada um deles execute, sem que isso destrua a integridade temporal do sistema, isto é, prioridade para os processos críticos.

Escalonamento em Linux

O problema básico de escalonamento em sistema operativo é como satisfazer simultaneamente objectivos conflituantes: tempo de resposta rápido, bom *throughput* para processos *background*, evitar postergação indefinida, conciliar processos de alta prioridade com de baixa prioridade, entre outros.
Tradicionalmente, os processos são divididos em três grandes classes: interactivos, *batch* e tempo real. O escalonador do Linux não distingue processos interactivos de processos *batch*, diferenciando-os apenas dos processos em tempo real.
O escalonador do Linux é baseado em *time-sharing*, ou seja, o tempo do processador é dividido em fatias de tempo (*quantum*), as quais são alocadas aos processos. Se, durante a execução de um processo, o *quantum* é esgotado, um novo processo é seleccionado para execução, provocando uma troca de contexto. Esse procedimento é completamente transparente ao processo e baseia-se em interrupções de tempo.

O algoritmo de escalonamento do Linux divide o tempo de processamento em épocas (*epochs*). Cada processo, no momento da sua criação, recebe um *quantum* calculado no início de uma época. Diferentes processos podem possuir diferentes valores de *quantum*. O valor do *quantum* corresponde à duração da época, e essa, por sua vez, é um múltiplo de 10ms inferior a 100ms. Outra característica do escalonador Linux é a existência de prioridades dinâmicas. O escalonador do Linux monitora o comportamento de um processo e ajusta dinamicamente sua prioridade, racionalizando o uso do processador pelos processos. Processos que recentemente ocuparam o processador durante um período de tempo considerado "longo" têm uma prioridade baixa. De forma análoga, aqueles que estão há muito tempo sem executar recebem um aumento na sua prioridade, sendo então beneficiados em novas operações de escalonamento.

Na realidade, o sistema Linux trabalha com dois tipos de prioridades: estática e dinâmica. As prioridades estáticas são utilizadas exclusivamente por processos de tempo real correspondendo a valores na faixa de 1-99. Nesse caso, a prioridade do processo tempo real é definida pelo utilizador e não é modificada pelo escalonador. Somente utilizadores com privilégios especiais têm a capacidade de criar e definir processos tempo real.

O esquema de prioridades dinâmicas é aplicado aos processos interactivos e *batch*. Aqui, a prioridade é calculada, considerando-se a prioridade base do processo e a quantidade de tempo restante em seu *quantum*.

O escalonador do Linux executa os processos de prioridade dinâmica apenas quando não há processos de tempo real. Em outros termos, os processos de prioridade estática recebem uma prioridade maior que os processos de prioridade dinâmica. É importante referir o Linux usa a mesma representação interna para processos e *threads.* Um *thread* é simplesmente um novo processo que partilha parte do contexto do processo pai.
Para seleccionar um processo para execução, o escalonador do Linux prevê três políticas diferentes:

- **SCHED_FIFO**: Essa política é valida apenas para os processos de tempo real. Na criação, o descritor do processo é inserido no final da fila correspondente à sua prioridade. Nessa política, quando um processo é a locado ao processador, ele executa até que uma de três situações ocorra: um processo de tempo real de prioridade superior torna-se apto a executar; o processo libera espontaneamente o processador para processos de prioridade igual à sua; o processo termina, ou bloqueia-se, em uma operação de entrada e saída ou de sincronização.

- **SCHED_RR**: Na criação, o descritor do processo é inserido no final da fila correspondente à sua prioridade. Quando um processo é alocado ao processador, ele executa até que uma de quatro situações ocorra: seu período de execução (*quantum*) tenha se esgotado nesse caso o processo é inserido no final de sua fila de prioridade; um processo de prioridade superior torna-se apto a executar; o processo libera espontaneamente o processador para processos de prioridade igual a sua; o processo termina, ou bloqueia-se, em uma operação de entrada e saída ou de sincronização. Essa política também só é válida para processos de tempo real.

- **SCHED_OTHER**: Corresponde a um esquema de filas multinível de prioridades dinâmicas com *timesharing*. Os processos interactivos e *batch* recaem nessa categoria.

Observação: A **SCHED_FIFO** é política de escalonamento dos processos utilizada neste trabalho.

O escalonador controla a execução:

- Tarefas correspondentes a processos a nível utilizador;
- Tarefas de sistema: relacionadas com a parte da rede, gestão de memória;
- Tarefas que executam internamente por conta de *device drivers*.

4. Métodos matemáticos para detecção de rectas, curvas e pontos únicos

É nosso objectivo estudar métodos matemáticos que nos permitam detectar formas ou objectos, e assim poder criar algoritmos (Capítulo 3) para os blocos da arquitectura do Sistema de Visão. Esses algoritmos, na sua maioria, vão ser constituídos por um conjuntos de testes e dados úteis e têm como objectivo a identificação de "targets" conhecidos.

4.1 – Estudo da 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. Geralmente, a transformada é aplicada após a imagem sofrer um pré processamento, normalmente a detecção edges. Também se pode aplicar uma transformada de Hough generalizada a formas, ou objectos, que não sejam representados facilmente em termos analíticos.
As principais vantagens da transformada de Hough são o facto de ser um método robusto, tolerante a falhas em alguns pontos da imagem e por ser muito pouco afectada pelo ruído da imagem.

Na generalidade, pode-se pensar na transformada de Hough da seguinte forma:

Dada uma função implícita que parametrize a forma pretendida, $\boldsymbol{f(v, \Phi)} = \mathbf{0}$, em que **v** é um ponto da curva e **Φ** é um vector de parâmetros, e dada uma função $\boldsymbol{g(v, \Phi)}$ que vale **1**

quando o ponto pertence à forma e **0** caso contrário, a transformada de Hough consiste em determinar, para cada forma, o valor de:

$$H(\Phi) = \sum_{v} g(v, \Phi)$$

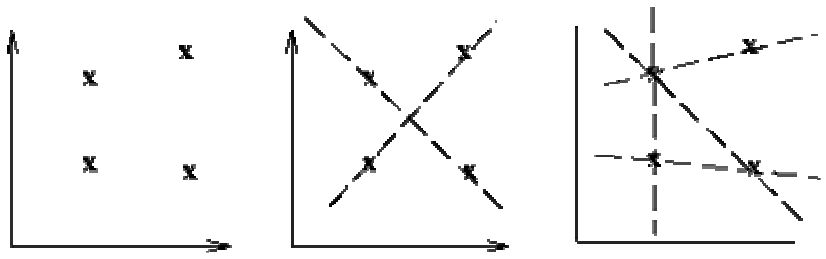
Ao conjunto de todos os H(Φ) chama-se espaço de acumuladores.
O valor da função no "espaço Hough" dá a densidade ao longo da linha no espaço

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.

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

Podemos analiticamente descrever um segmento de linha de várias formas, contudo a equação mais usada é a que usa a forma paramétrica.

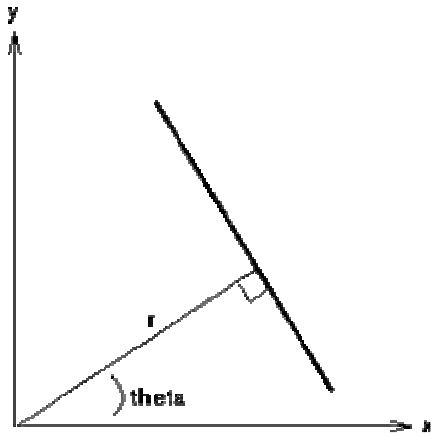


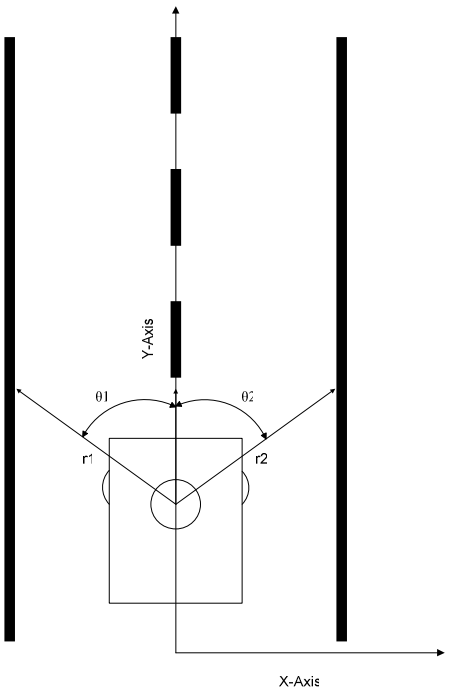
Figura 6: Representação paramétrica de um segmento de linha.

A parametrização é, assim, dada por:

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

Na análise de uma imagem, as coordenadas dos pontos, guardados nos edges (x_i , y_i) 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)**.

Na figura 7 pode-se ver de que forma se pode usar a informação da transformada de Hough, para saber quais as linhas detectadas.



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

Figura 7

Acumulador:

A transformada é implementada, quantificando o "espaço Hough" em intervalos finitos ou em acumuladores.

O algoritmo usado deverá percorrer todos os pontos (x_i,y_i) 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.

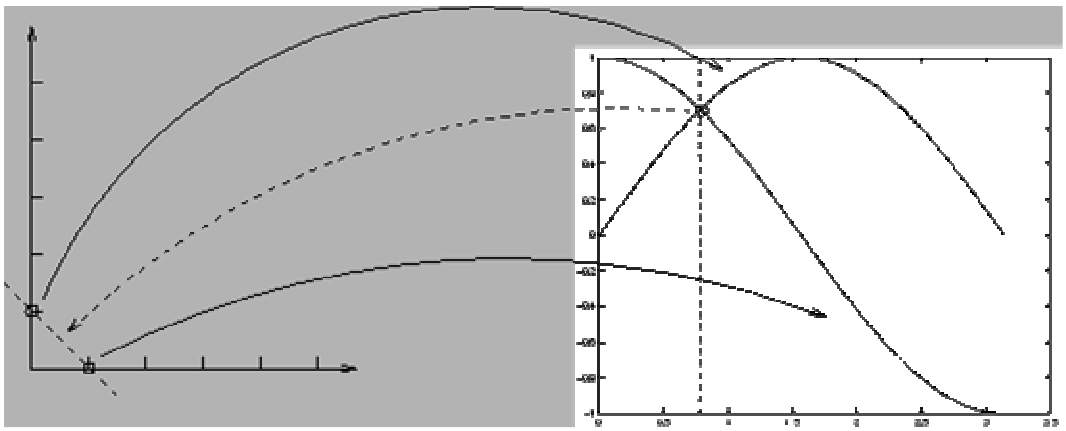


Figura 8: Cada ponto na imagem da esquerda é projectado numa curva sinusoidal no acumulador, à direita, usando a equação (A). A intersecção das curvas representa a linha que conecta os pontos.

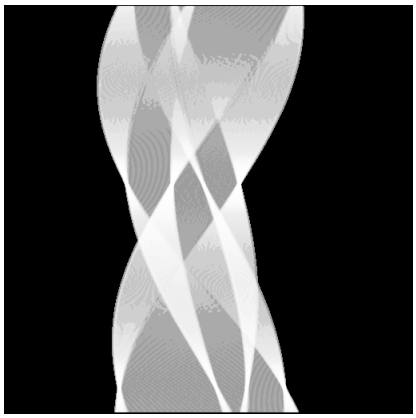


Figura 9: Exemplo do "espaço Hough"

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.
O input de uma operação de thresholding é tipicamente um grayscale ou uma imagem da cor.
Na sua implementação mais simples, o output é uma imagem binária que representa a segmentação. Os pixels pretos correspondem ao fundo e os pixels brancos correspondem à imagem ou vice-versa. A segmentação é determinada por um único parâmetro conhecido como

"intensity threshold". 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.

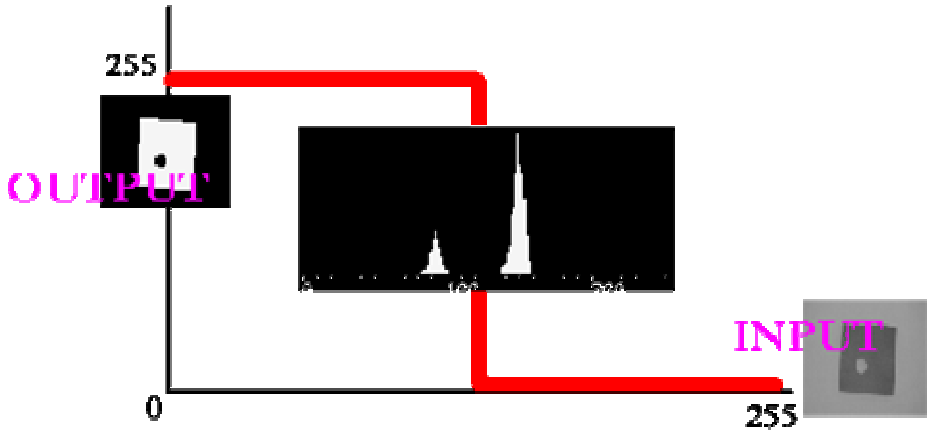


Figura 10: Threshold

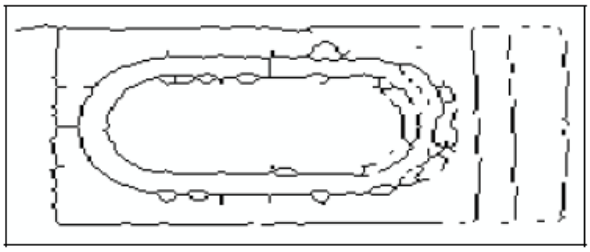
Podemos usar o mesmo procedimento para detectar circunferências, visto que a única coisa que muda é a equação paramétrica e o facto de não existir orientação.

Detecção de circunferências

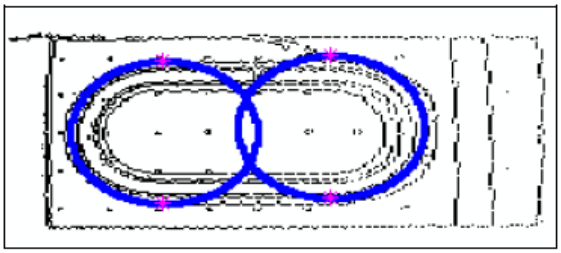
Uma circunferência define-se facilmente com 3 parâmetros: as coordenadas *x* e *y* do centro e o raio. Para determinar o espaço de acumuladores, surge uma forma bastante rápida de calcular, se tivermos em conta que, para cada raio ***r***, um ponto **v** pertence a todas as circunferências que têm centros nos pontos que estão a uma distância ***r*** de **v**.
Isto define uma circunferência à volta de **v** com raio *r*. Assim, para calcular o espaço de acumuladores, limitamo-nos a, para cada raio, desenhar uma circunferência em torno de cada ponto **v** do espaço.

A parametrização é: $(x - p)^2 + (y - q)^2 = r^2$

Exemplo de detecção de círculos

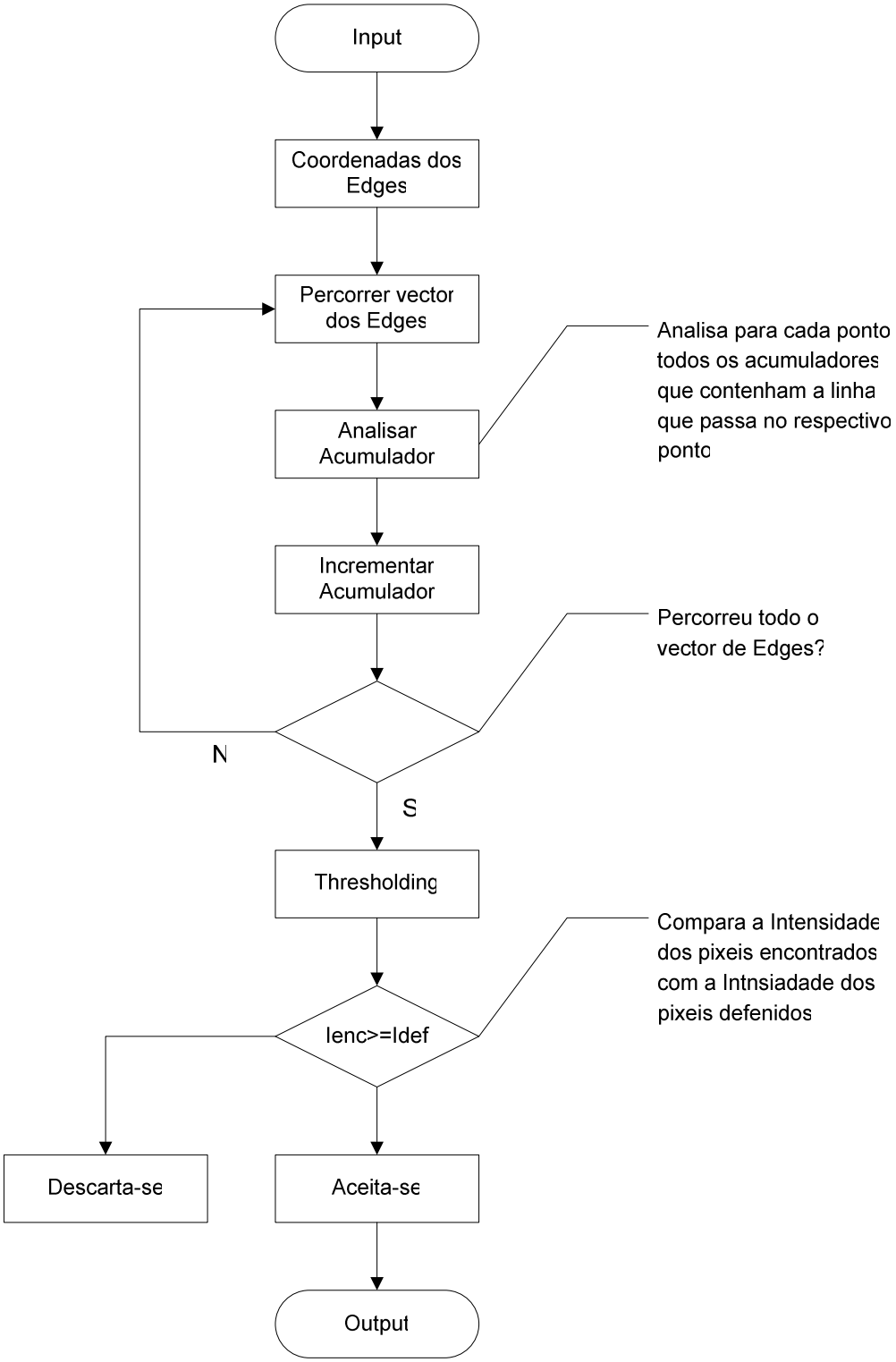


a) Imagem original



b) Circunferências detectada

Algoritmo da transformada de Hough



4.2 – Método dos Mínimos Quadrados

Supondo que o conjunto de dados consiste dos pontos (x_i, y_i) com $i = 1, 2, \ldots, n$ deseja-se encontrar uma função **f** tal que: $f(x_i) \approx y_i$.

Para se obter tal função, nós supomos que a função **f** é de uma forma particular contendo alguns parâmetros que necessitam ser determinados. Por exemplo, supor que ela é quadrática, significa que $f(x) = ax^2 + bx + c$, onde a, b e c não são conhecidos. O que procuramos são os valores de a, b e c que minimizam a soma dos quadrados dos resíduos:

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

Vamos estudar a aproximação de funções numa perspectiva diferente da interpolação. Por exemplo, se tivermos apenas os valores da função em certos pontos, não vamos exigir que a função aproximada interpole a função dada nos pontos. Exigimos apenas que essa função aproximada tome valores (nesses pontos) de forma a minimizar a distância aos valores dados... falamos em minimizar, no sentido dos mínimos quadrados!

Isto é importante em termos de aplicações, já que podemos ter valores obtidos, experimentalmente, com uma certa incerteza. Ao tentar modelar essa experiência, com uma certa classe de funções, seria inadequado exigir que a função *a*proximada interpolasse esses pontos. Um caso simples, em que se aplica esta teoria é o caso da *regressão linear*, em que tentamos adaptar a um conjunto de pontos e valores dados, à "melhor recta", que (neste caso) será a recta que minimiza a soma quadrática das diferenças entre os valores dados ao valores da recta, nesses pontos.

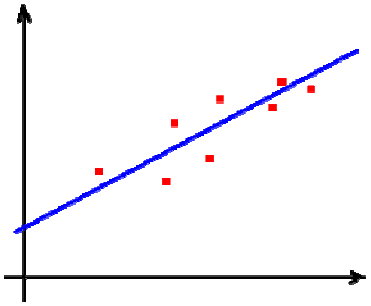


Figura 11:Regressão Linear: Neste caso pretendemos encontrar a função do tipo a + b x (... ou seja, a recta) que "melhor se adapta" aos valores dados.

4.3 – Perspectivas de utilização dos métodos matemáticos

Foram apresentados alguns métodos matemáticos que poderão ser utilizados para detectar rectas e curvas. Nesta fase ainda não é possível escolher o método mais adequado, pois só após implementação pratica de cada uma delas é que poderemos avaliar com exactidão qual deles o melhor para os nossos objectivos.
De um ponto de vista imediato qualquer um deles poderia ser utilizado, tanto a Transformada de Hough, como os Mínimos Quadrados, são métodos bastante utilizados na detecção de objectos, mas existem certas questões que nos parecem relevantes e que precisam ser analisadas e respondidas tais como:

- Qual o método que permitirá uma implementação mais fácil e robusta?
- Qual o método que será mais flexível, para que possa ser fiável em qualquer situação?
- Qual será o método 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?

Estas são algumas das questões mais importantes que só depois da implementação é que poderemos responder com mais exactidão.

CAPÍTULO 3

Arquitectura de software é o estudo da organização global dos sistemas de software bem como do relacionamento entre seus subsistemas e componentes. É importante ter uma noção do objectivo (função) de cada subsistema integrado no sistema de software, o porquê de existirem vários subsistemas ou camadas, a forma como informação é transmitida e que tipo de informação é transmitida para cada camada ou nível.

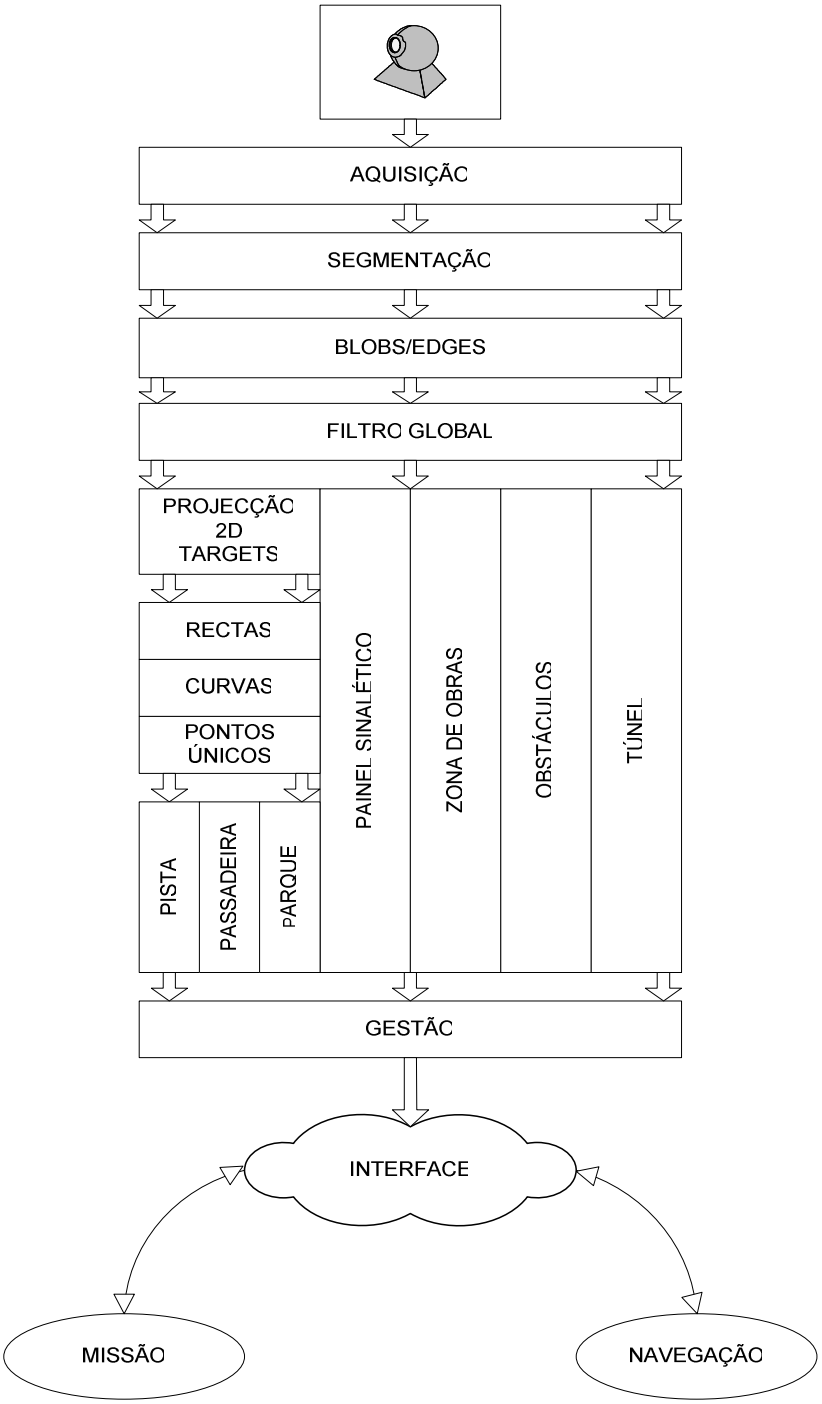
1. Arquitectura do Sistema de Visão

O sistema de Visão tem uma arquitectura própria e está incluído na arquitectura global do software dos robots "Runner".
O principal objectivo do módulo de Visão é identificar "targets" (objectos) encontrados na imagem e disponibilizar às camadas superiores informações úteis sobre estes, tais como: identificar a posição destes em relação ao referencial do robot, identificar a cor e a forma dos "targets" (linhas, conjunto de linhas (passadeira), semáforos, obstáculos, túnel, zona de obras e outros aspectos que possam interessar.
Uma parte significativa do nosso trabalho foi desenvolver o conjunto de camadas do módulo de Visão responsáveis pelo seu funcionamento e definir a função de cada camada.

Seguidamente indicamos os blocos que fazem parte do diagrama do Sistema de Visão apresentado na figura 7.

- Aquisição/Processamento
- Segmentação
- Detecção de blobs e edges
- Filtro Global
- Projecção
- Detecção dos diferentes "targets"
- Módulo de gestão

Diagrama de Blocos do Sistema de Visão



Seguidamente procederemos à descrição dos aspectos importantes de cada um dos respectivos blocos do Sistema de Visão, tentaremos caracterizar a tarefa de cada um no Sistema de Visão e tentaremos criar um algoritmo. Um algoritmo não representa, necessariamente, um programa de computador, e sim os passos necessários para realizar uma tarefa.

1.1 – Aquisição de Imagens/Processamento

Convém referir, antes de mais, que a calibração das câmaras é realizada em off-line, no qual a intervenção humana é necessária, o que implica que o bloco da Calibração não faça parte do Sistema de Visão.

O sistema de visão é constituído por dois threads, um dedicado à aquisição cíclica de imagens e outro dedicado ao processamento de imagem. Existem conjuntos de threads síncronos e conjuntos de threads assíncronos na arquitectura global de software, os threads ligados à Visão são assíncronos. Cada thread tem uma dada prioridade estática que depende do papel que desempenha. A atribuição dos valores da prioridades é o resultado um processo de experiência e análise da performance do sistema.
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. É um sistema muito 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.

1.2 – Segmentação

A imagem obtida pela câmara deve ser segmentada para se proceder à identificação dos "targets" pretendidos. Estes objectos podem ser simples ou compostos, isto é, formados por outros objectos.
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 o já recebemos a imagem segmentada e com informação acerca de blobs e edges como falaremos mais à frente. Em baixo temos um exemplo de uma imagem segmentada.

Brightness: 35000; Contrast: 65535; Hue: 65535; Colour: 0

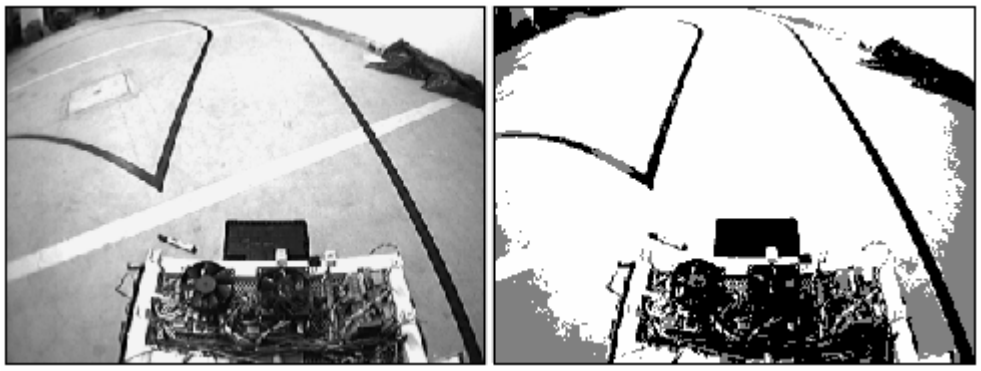


Figura 1: a) imagem original; b) imagem classificada

1.3 – Detecção de blobs e edges

Após a segmentação da imagem, é necessário proceder a detecção de blobs e edges. A detecção de edges irá fornecer informação sobre a posição dos limites dos objectos e da presença de descontinuidades e a detecção de blobs irá fornecer informação em relação ao objecto. Estas funções já se encontram criadas, e nós iremos utilizar a informação disponível por estas funções para proceder à detecção e classificação dos diferentes objectos.

- 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.

- Se encontrarmos um edge que achamos que pode ser uma recta, procura nos pares qual é que faz "match". Definimos uma recta e escolhemos um ponto do edge par, no qual iremos calcular a distância desse ponto a recta. Isto irá permitir calcular distâncias.

1.4 – Filtro Global

Depois de detectados e identificados os edges e os blobs, é necessário aplicar um filtro global, cujo objectivo é reduzir ou eliminar a distorção e ruído da imagem. 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".

Características do filtro global

- Tem de ser um filtro flexível, pois os pixeis analisados vão depender da perspectiva da câmara, isto é, **n** pixeis com uma coordenada em y "alta", tem uma dimensão mais pequena do que **n** pixeis com coordenada em y "baixa".

- Definir um número de pixeis, e todos os blobs com um número inferior a esses pixeis são eliminados.

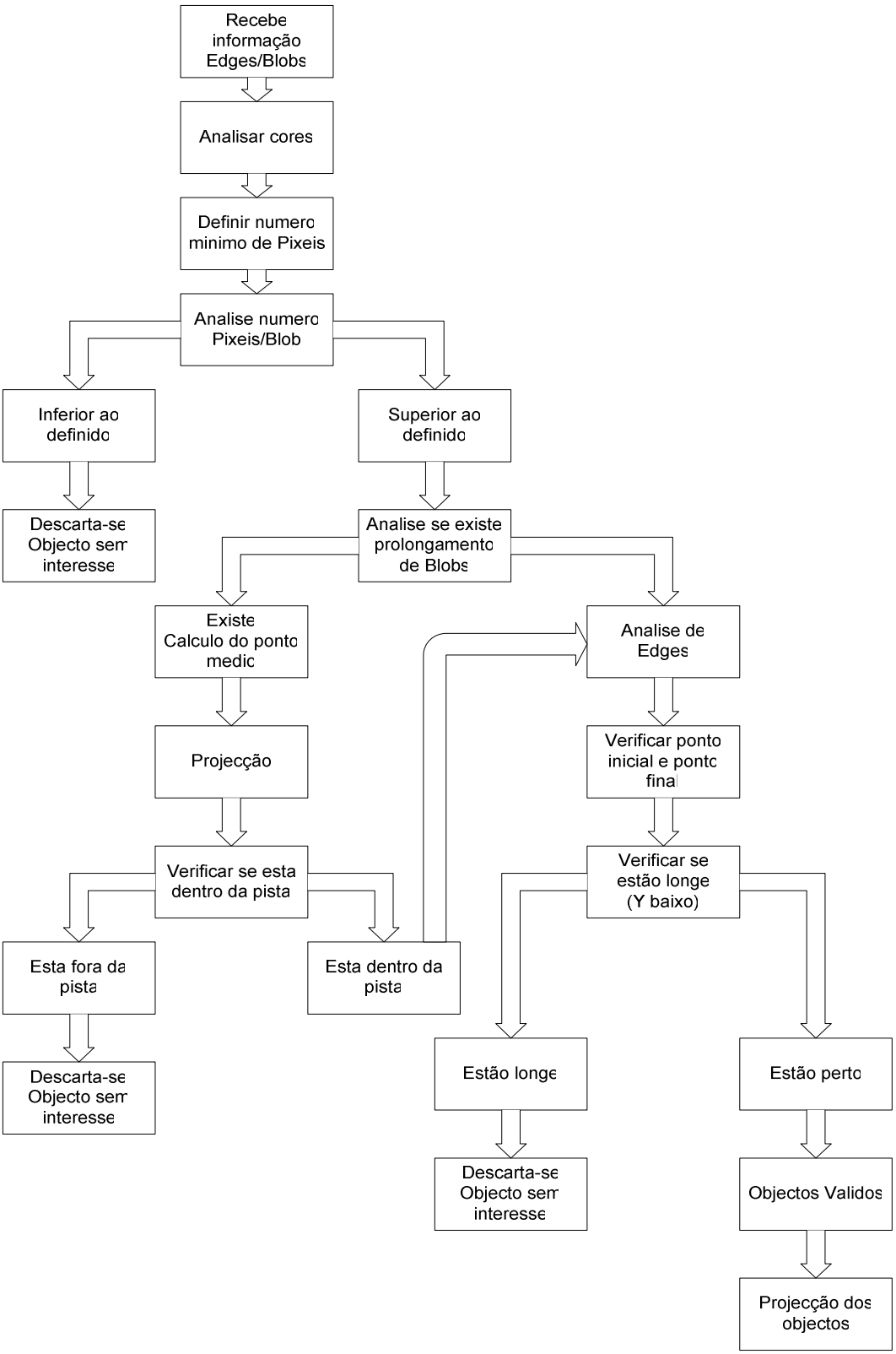
- Descartamos edges com pontos iniciais e finais que estejam longe (coordenadas em y "baixas").

- Se um ponto inicial tiver perto (coordenada em y "alta"), então poderá ser um objecto de interesse.

- Ter em atenção que dependendo da perspectiva com que o robot vê o obstáculo, o blob que identifica esse obstáculo pode ser muito maior do que se poderia pensar, pois pode existir um prolongamento desse blob com outros da mesma cor, dai que a solução poderá passar pelo calculo do ponto médio do blob e verificar que se está dentro da pista (após projecção).

- Deve possibilitar a "filtragem" por cores isto é, deve fazer uma distinção entre as diferentes cores dos blobs e edges para que a identificação dos semáforos e da zona de obras (cones), seja mais rápida, melhor dizendo deve detectar uma transição anormal e guardar os dados para serem analisados pelas respectivas funções. As transições normais seriam cores de preto e branco.

Diagrama de blocos



1.5 – Projecção

Hipótese 1:

- Aplicar Filtro Global.
- Projectar pontos de interesse em relação ao referencial do robot.
- Validar rectas, curvas e pontos únicos.
- Detectar "targets".

Hipótese 2:

- Aplicar Filtro Global.
- Validar rectas, curvas e pontos únicos.
- Projectar pontos de interesse em relação ao referencial do robot.
- Detectar "targets".

Após uma análise cuidada das duas hipóteses, optamos pela hipótese 1.
A nossa escolha parece-nos a mais acertada pois, na nossa opinião é mais útil trabalhar com coordenadas reais (em relação a um referencial) do que trabalhar com pixeis, para determinar distâncias entre edges e blobs e a partir daí poder identificar rectas e curvas… O referencial escolhido foi o do robot e não o da câmara.
A hipótese 2 tinha o inconveniente de se ter que aplicar métodos matemáticos (pesados) antes da projecção, o que não corresponderia a um referencial real, isto poderia originar erros na projecção.
O código fonte para fazer a projecção em relação ao referencial do robot já se encontra feito.

A projecção dos "targets" em relação ao referencial do robot é uma parte muito significativa do Sistema de Visão, porque além de ser importante para identificar os próprios "targets" é necessária para fazer o "match" do robot com o mundo e saber se a posição projectada corresponde à real. Não faz parte das tarefas da visão fazer esse "matching", mas sim enviar informação acerca da posição do robot para a Navegação. A Navegação é a responsável pela localização do robot no mundo.

1.6 – Pista (linhas, segmentos de rectas, curvas)

Identificação e Caracterização

A pista desenrola-se dentro de uma área de (11 x 16.4) m, possui o formato de estrada e é delimitada por duas linhas laterais e paralelas. Uma imagem representativa do respectivo traçado é apresentada na Figura 2. Os raios de curvatura interior e exterior dos traçados curvos são respectivamente de 1.5m e 3.1m. O troço rectilíneo tem 10m de comprimento. A distância entre linhas, no bordo interior, é de 150cm, sendo estas de 5cm de largura. No centro da pista existe uma linha tracejada que a divide em duas faixas de rodagem iguais.
Os troços brancos desta linha têm 5cm de largura por 20cm de comprimento e encontram-se espaçados de 15cm.

Cores

O chão da pista é de cor preto, absorvente de infravermelhos. As linhas laterais são brancas e reflectoras de infravermelhos. A zona exterior à pista, mas adjacente à linha externa, numa banda com pelo menos 20cm de largura, é da mesma cor do chão da pista.

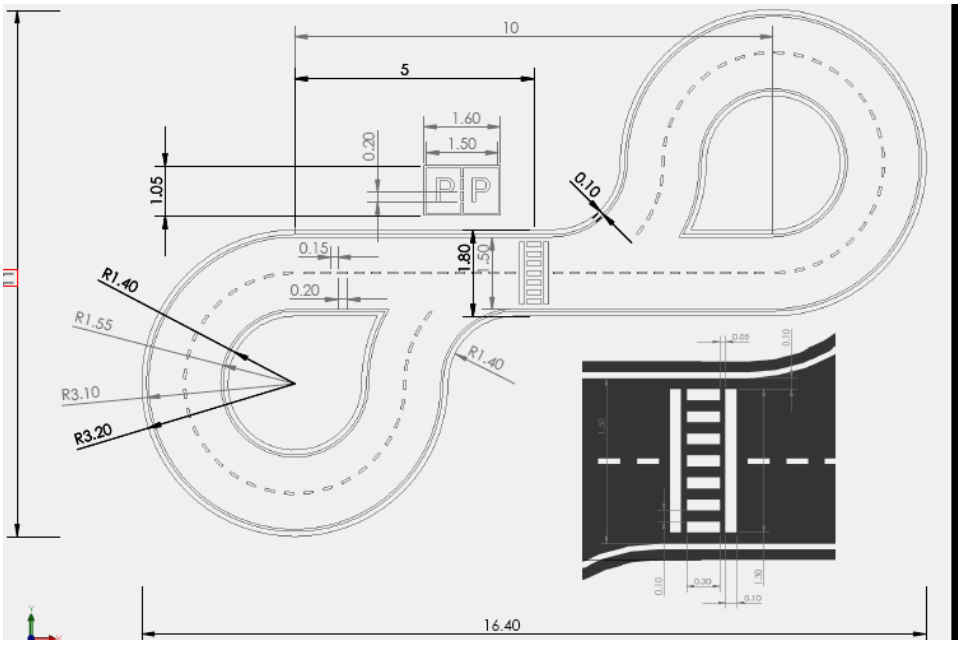


Figura 2: Pista do FNR

1.6.1 – Detecção de rectas, curvas e pontos únicos na pista

- A detecção de rectas curvas e pontos únicos é realizada depois da projecção 2D em relação ao referencial do robot.

- Vão ser usados métodos matemáticos (interpolação ou mínimos quadrados e/ou transformada de Hough) para determinar as equações das rectas, das curvas e estimação do valor das coordenadas para o caso dos pontos únicos.

- Os métodos matemáticos não podem ser "pesados" para não prejudicar o funcionamento do Sistema de Visão, pois não nos interessa a exactidão mas sim uma boa estimativa, uma vez que estamos sempre a adquirir novas imagens em movimento.

- Usaremos informação das coordenadas dos edges nos métodos matemáticos o que vai tornar todo o processo mais rápido e dependendo da boa estimativa e robustez dos métodos podemos identificar as rectas, curvas e pontos únicos da pista.

- As linhas da pista serão talvez as mais fáceis de identificar e também uma das mais importantes, uma vez que poderemos usar a informação dos edges relativamente à transição de cor preto-branco e branco-preto, edge PB e BP, bem como calcular as distancias entre edges para saber se a distancia corresponde à real. Claro que os valores não serão completamente iguais, daí que usaremos uma margem para validar se são as linhas desejadas. Já existem funções para determinar a distância entre os edges, pelo que quando for necessário calcular esse valor é só preciso chamar a respectiva função.

- Estas linhas poderão ser úteis, no futuro, se utilizarmos o mapeamento do percurso na pista. Na primeira volta fazíamos o mapeamento, ficávamos a saber as condenadas de todas as linhas descontínuas, e assim nas outras voltas o robot já não se perdia.

- Também poderão ser úteis, para dar uma informação complementar, que ajudara na inércia e hodometria visual. Entre dois frames, poderemos saber as linhas descontínuas que tínhamos num frame e as que tínhamos no outro. Como sabemos o que andamos de uma frame para a outra, se o que andamos for muito diferente do que a hodometria andou, quer dizer q a hodometria patinou, o que permite corrigi-la.

- Um ponto único importante será o vértice da pista de mudança de sentido. Este ponto é importante porque dará uma coordenada absoluta, que permite saber com precisão em que sitio do mundo o robot está e corrigir eventuais erros acumulados durante o percurso.

1.7 – Passadeira, Zona de partida/chegada e Parque de estacionamento

Identificação e Caracterização

A meio da recta central, na junção dos dois troços circulares, está colocada uma passadeira como a representada na Figura 3.A zona "zebrada" é constituída por sete traços brancos com 10x30cm, espaçados de 10cm entre si. Esta zona encontra-se afastada das linhas delimitadoras da pista de 10cm. A passadeira é limitada de ambos os lados por dois traços brancos com 10x130cm (centrados na pista). Todos os traços brancos da passadeira são reflectores de infravermelhos.
A zona de partida/chegada corresponde a uma área imediatamente anterior à passadeira, considerando o sentido de movimento do robô, e que intercepta o 1º traço transversal desta, de acordo com as regras estabelecidas no capítulo "competição".
O parque de estacionamento está colocado depois da passadeira. Consiste em dois rectângulos contíguos onde se encontra inscrita a letra "P". As respectivas medidas e distâncias à pista podem ser consultadas na Figura 2.

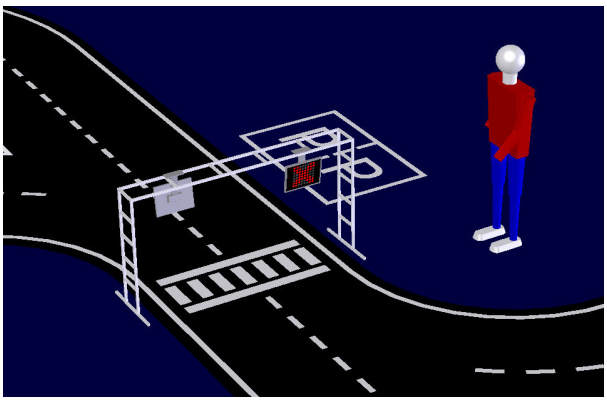


Figura 3: Passadeira e zona de partida/chegada

1.7.1 – Características do algoritmo da passadeira

- Para a detecção da passadeira vamos usar a informação fornecida pelas funções de detecção de rectas.

- Existirá uma função que receberá como argumento a informação das rectas identificadas e através de condições detectara as que correspondem à passadeira.

- Analisar as características da passadeira (numero de rectas verticais paralelas, rectas horizontais, tamanho das rectas, distancia entre elas).

- Verificar quantas transições de cor existem (edges), entre as diferentes rectas.

- Definir um número mínimo de transições de cor, para que possamos garantir que estamos perante a passadeira. Porque dependendo da posição do robot, poderá ver mais ou menos rectas verticais paralelas.

- Medir a distância entre rectas, e a largura das próprias rectas.

- Através do edge da linha da pista (BP), e o par desse edge (BP) da linha horizontal da passadeira, podemos calcular a distância entre eles. Como sabemos a distância real da linha da pista ao início da linha horizontal da passadeira, a diferença entre as duas distâncias dá-nos o comprimento da linha horizontal da passadeira.

1.7.2 – Características do algoritmo do parque

- Durante o percurso da pista podemos identificar o parque, porque nos poderá disponibilizar informações úteis.

- Na saída da curva, em que o parque se encontra de frente, o vértice mais próximo da passadeira poderá ser um ponto de referência. Será um edge com qualidade suficiente para caracterizar esse ponto, e a direcção da recta que une o vértice.

1.8 – Painel Sinalético

Descrição

Sobre a passadeira serão montados dois painéis TFT com 15” de dimensão diagonal, em posição invertida e verticalmente alinhados com cada uma das faixas de rodagem.

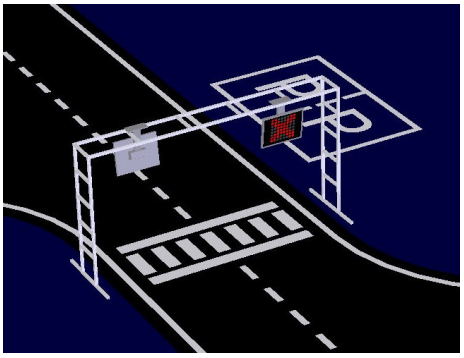


Figura 4: Painel Sinalético

Características dos sinais apresentados

Os sinais apresentados nos painéis são gerados a partir de um formato de imagem com 1024x768 pixels. Os símbolos propriamente ditos ocupam uma área quadrada com 600x600 pixels. As cores usadas, apresentadas sobre fundo negro, são respectivamente o vermelho, o verde e o amarelo. Os sinais do painel sinalético podem ser observados na Figura 5.

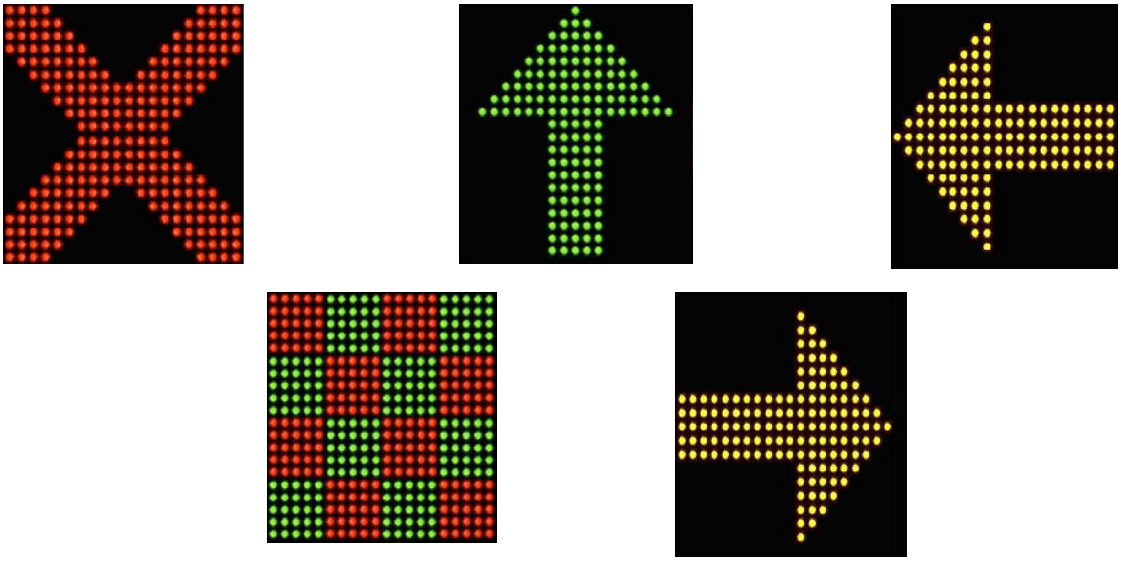


Figura 5: Sinais do painel sinalético

Função e sinalética

A função dos painéis sinaléticos é a de orientar a prova dos robots dando-lhes ordem para:

1 – Parar;
2 – Seguir em frente;
3 – Seguir sobre a esquerda
4 – Fim de passagem,
5 – Seguir para o Parque

A correspondência entre cada uma destas funções e a informação luminosa apresentada no painel sinalético é a seguinte:

Função	*Correspondência*	*Sinalética Semáforos*
1	Parar	um "X" (em luz vermelha)
2	Seguir em frente	uma seta vertical (em luz verde)
3	Seguir sobre a esquerda	uma seta horizontal para a esquerda (em luz amarela)
4	Fim de passagem	bandeira xadrez,(em luzes verde e vermelha piscando alternadamente estando cada configuração acesa durante 0,25 s)
5	Seguir para o parque	uma seta horizontal para a direita(luz amarela)

Figura 6

1.8.1 – Características do algoritmo Painel Sinalético

- Os semáforos serão detectados através da cor e através da forma.

- Para a cor, poderá definir-se um número mínimo de pixeis e máximo, em que caso seja encontrado um valor diferente poderá descartar-se esse objecto, pois não será o nosso semáforo.

- Isto previne uma situação em que se encontre um objecto da mesma cor fora da pista.

- Para se validar um objecto por cor, neste caso o nosso semáforo, recorre-se a informação fornecida pela função de blobs, que nos indicara o número de pixeis da mesma cor.

- Outro critério de validação será pela forma, em que para isso iremos recorrer a informação fornecida pelas funções que identificam rectas, e que analisaremos se estamos perante uma recta que corresponda ao sinal do semáforo.

- Usamos a informação dos edges para calcular distâncias entre rectas.

- Para o sinal de "Fim de passagem", termos blobs alternados (vermelhos-Verdes), que juntamente com a informação dos edges, poderemos verificar se existe transição de cores, e se estão alternadas confome nos mostra a figura 10.

1.9 – Zona de Obras

Descrição

A zona de obras corresponde a uma alteração ao trajecto original da pista. Esta zona tem um comprimento e uma forma desconhecida das equipas, sendo a sua colocação revelada pela organização após o encerramento do parque fechado. A zona de obras poderá coincidir com uma zona em curva ou com uma zona em recta do percurso original.

Materiais e cores

A zona de obras é limitada à entrada e à saída por dois pares de cones cor de laranja (adaptados dos utilizados nas obras das estradas), colocados sobre a linha delimitadora pista. O percurso em obras será delimitado à esquerda e à direita por uma sequência cones semelhantes aos já referidos e espaçados entre si de uma distância igual a cerca 1m. Cada par de cones consecutivos encontra-se unido por uma fita com 5cm de largura, colocada na vertical e de cor alternadamente vermelha e branca (as cores são aproximadas – a fita é semelhante à versão comercial usada para delimitar obras em vias públicas). base desta fita ficará a cerca de 10cm do solo. Este percurso inicia-se nas linhas delimitadoras da pista, afasta-se delas para o exterior (interior) da mesma, regressando posteriormente ao traçado original. Na Figura 7 é possível observar uma imagem exemplificativa de uma zona de obras.

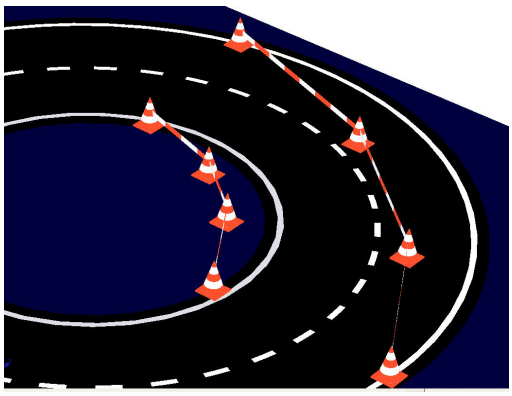


Figura 7: Zona de obras

1.9.1 – Características do algoritmo de Zona de Obras

- Como o primeiro cone da zona de obras coincide com a linha da pista, iremos encontrar um blob branco maior que a linha, que poderá nos indicar a presença de algo coincidente com a pista, o que convêm validar, pois poderá ser o nosso cone.

- A fita que une os cones funciona como uma linha, com cores alternadas, em que os critérios de validação de rectas, poderão ser aplicados neste caso.

- O robot continuara a ver e a identificar as linhas da pista, mos só enviará informação para o controlo referente ao cones e fita que os une.

- Como os cones se tratam de objectos com volume, de igual forma como o obstáculo, poderá ser interessante projectar edges verticais. Verifica-se um conjunto de pontos e vê-se a viabilidade de se ter uma coluna vertical.

- Para se concretizar esta técnica será necessário construir umas funções que permitam esta projecção.

1.10 - Obstáculos

Descrição

Será colocado na pista um obstáculo que impedirá a circulação dos robôs por uma das faixas de rodagem (ver capítulo sobre a competição). O obstáculo é um paralelepípedo rectângulo de base quadrada com 60cm de lado e uma altura mínima de 20cm. Um exemplo de colocação deste obstáculo pode ser observado na Figura 8.

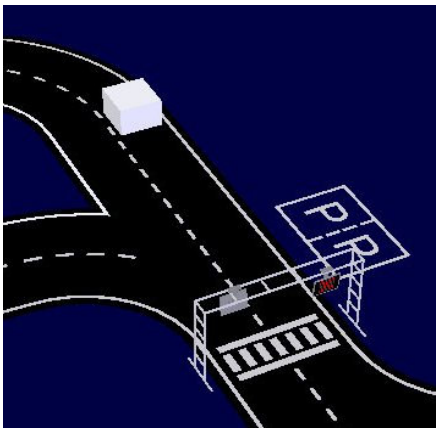


Figura 8: Obstáculo Cúbico

1.10.1 – Características do algoritmo do Obstáculo

- Dependendo do ângulo de visão em que vamos encontrar o obstáculo, poderá haver um prolongamento do blob do obstáculo com o blob das linhas ou de algo dentro do campo de visão do robot. Para tentarmos solucionar este problema podemos calcular o ponto médio desse blob, e verificar se está dentro da pista.

- O obstáculo terá de ser detectado um pouco antes da sua aproximação, para se poder efectuar a manobra de desvio do mesmo.

- Como o obstáculo é um objecto com volume, será vantajoso projectar edges verticais. Verifica-se um conjunto de pontos e vê-se a viabilidade de se ter uma ou várias colunas verticais.

1.11 – Túnel

Dimensões

O túnel será colocado na pista num dos seus sectores circulares e cobrirá um raio de curvatura de cerca de 90º. As suas dimensões interiores são de 150cm de largura e aproximadamente 100cm de altura. O seu comprimento médio é de cerca de 3,6m. Os bordos verticais de entrada e/ou saída do túnel terão uma largura mínima de 5cm para facilitar o seu reconhecimento.

Material e cor

Os bordos verticais de entrada/saída e as paredes interiores serão de cor brancos, de forma a reflectirem luz infravermelha. O tecto poderá ser em qualquer material, não necessariamente branco, nem necessariamente plano.

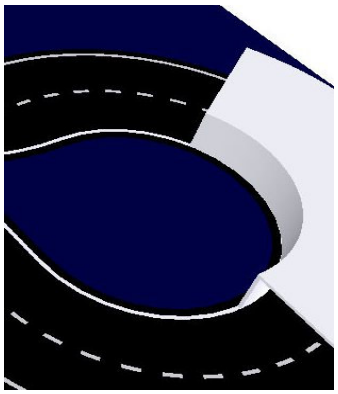


Figura 9: Túnel

1.11.1 – Características do algoritmo do Túnel

- Para detectar o túnel, será vantajoso projectar edges verticais. Verifica-se um conjunto de pontos e vê-se a viabilidade de se ter uma ou várias colunas verticais.

2. 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.
Tentamos imaginar a arquitectura global do software dos robots "Runner" e pensamos em lalgumas allternativas:

- Filas de mensagens
- Modelo Publisher/Subscriber Ocera
- Zona de memória partilhada + semáforos

Filas de mensagens

Os sistemas baseados em filas de mensagens permitem a interacção entre componentes em tempo diferido e são uma alternativa viável ao modelo de chamadas a procedimentos remotos na construção de aplicações em que os clientes e os servidores não necessitam de interagir de modo síncrono, entenda-se que o os clientes são as camadas superiores e o servidor é o sistema de visão.
O modelo é particularmente adequado para as aplicações que estão sujeitas a uma conexão intermitente ou baixo débito nas ligações.

Zona de memória partilhada + Semáforo de exclusão mútua

Criação de uma zona de memória partilhada para disponibilizar uma lista de "targets" entre outras informações úteis para que as camadas superiores do sistema global de software pudessem consultar (ler) dados sempre que necessitarem. Teríamos de garantir que as camadas superiores não pudessem ler essa zona de memória partilhada, quando o Sistema de Visão tivesse informações novas e quisesse actualizar, o que significa que iríamos ter um semáforo de exclusão mútua que bloquearia essa zona enquanto estivesse a escrever os dados novos e voltaria a torna-la acessível quando acabasse a operação de escrita Também teria de

se criar um sistema para avisar as outras camadas que novos dados estavam disponíveis para tornar o sistema mais rápido (signal).

Modelo Publisher/Subscriber Ocera

O modelo de tempo real Publisher/Subscriber (RT P/S) da Ocera é um meio de comunicação entre processos, do mesmo computador ou de diferentes computadores (ligados em rede), em tempo real. Com este modelo pretende-se criar um interface entre as camadas superiores e o sistema de visão para trocar informações úteis. Este modelo tem a vantagem de estar criado e de ter funções de registo importantes para atender exclusivamente a camada que pretende informação ou transmitir informação como por exemplo ligar/desligar funções.
Neste modelo:

" A Publisher can also be a subscriber and vice-versa."

3. Estruturas das funções

Os algoritmos desenvolvidos para os respectivos targets, filtro global, etc, são úteis para criar as funções que vão fazer parte do sistema de Visão. Indicaremos os parâmetros de entrada e de saída das funções do sistema.
A finalidade de cada função é identificar o respectivo target:

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

Vis_Fglobal

Objectivo:

Limpar imagem, eliminar o que não interessa

Input:

Imagem, blobs, edges.

Output:

Nova imagem + limpa, menos distorção e ruído

Vis_Hough

Objectivo:

Identificar rectas/curvas

Input:

Vector de edges preto-branco e branco-preto

Output:

Rectas/curvas e distância em relação ao referencial do robot

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;
```

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;

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;

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;
```

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;
```

Vis_Gestao

Objectivo:

Enviar/receber informação

Input:

Estrutura global Vis_Info

Output:

Enviar informação obtida do sistema de visão para as camadas superiores (Read/write)

4. Conclusão

Ao longo do trabalho desenvolvido aprendemos que a Visão Computacional é uma área de pesquisa multidisciplinar, e que por vezes para resolver um determinado problema tivemos de abordar diversas áreas para o compreender e soluciona -lo.

Ficamos com alguma experiência em conceitos fundamentais para a visão robótica e parte do nosso trabalho foi aplicar esses conceitos e desenvolver um Sistema de Visão.

Não podemos dizer que o trabalho está concluído, longe disso, mas temos a certeza que o caminho e a abordagem feita ao trabalho foram bons, apesar de ainda termos um longo caminho a percorrer para finalizar o projecto.

Bibliografia

- Relatórios 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:

- http://www.inf.ufsc.br/~visao/2000/Hough/index.html
- http://www.inf.ufsc.br/~visao/ia.html
- http://www.sbf1.sbfisica.prg.br/eventos/enfpc/xx/pracs/res143/
- http://www.inf.ufsc.br/~visao/1999/abu/#Hough
- http://homepages.inf.ed.ac.uk/rbf/HIPR2/hough.htm
- http://www.cogs.susx.ac.uk/users/davidy/teachvision/vision4.htm
- http://iris.usc.edu/Vision-Notes/bibliography/edge260.html
- http://omni.isr.ist.utl.pt/~alex/tfcs/COLORTRACK0506/
- http://cs-alb-pc3.massey.ac.nz/notes/59318/l11.html
- http://www.di.ufpe.br/~jf143/projectos/segmenta
- http://www.ocera.org
- http://www.sop.inria.fr/robotvis/personnel/zzhang/Publis/Tutorialtim/
- http://cs-alb-pc3.massey.ac.nz/notes/59318/l11.html
- http://homepages.inf.ed.ac.uk/rbf/HIPR2/hough.htm
- http://www2.lut.fi/~kyrki/ipcv02/hough/exercise/exercise.html
- http://visl.technion.ac.il/labs/anat/12-Hough/vhd1.txt

- PDF's:
  - RegrasExpecificaçõesCA2006-v11Out2005