AGVSim - Simulador de AGV com controlo difuso

Luís António (34113, lma@coral.inesc.pt), Rui Carmo (34179, rcarmo@coral.inesc.pt)

IST, Lisboa, 1994 - 1º Trabalho da Cadeira de Sistemas Flexíveis de Fabricação

Abstract

A lógica difusa pode revelar-se uma abordagem simples e eficaz ao controlo de sistemas, reduzindo a complexidade e permitindo simultaneamente manipular directamente conceitos difíceis de exprimir em termos precisos.

No entanto, o desenho de um sistema de controlo difuso é ainda mais arte do que ciência, devido ao grande número de factores que influenciam a performance *do sistema.*

Neste documento descreve-se não só a aplicação de simulação desenvolvida, como também o estudo levado a cabo sobre o problema específico do controlo de um AGV e as opções tomadas em ambos.

1. Utilização do Programa

1.1. Requisitos Mínimos

Para funcionar correctamente, o AGVSim necessita no mínimo de um PC 386 com co-processador matemático e memória RAM suficiente para correr o Windows 3.1 em «Enhanced Mode».

1.2. Instalação

Para instalar o AGVSim, basta inserir a *diskette* no *drive* A: ou B: e executar o programa SETUP.EXE. A aplicação e os documentos de exemplo serão copiados para o disco rígido e será criado um novo grupo no Program Manager.

1.3. A janela do AGVSim

Sendo uma aplicação Windows, o AGVSim apresenta um *interface* com o utilizador em tudo semelhante ao habitual neste ambiente.

Conforme se pode ver na Fig. 1.1, a janela divide-se em diversas zonas, cada qual com a sua funcionalidade específica. Assim, e para além dos elementos de *interface* comuns a qualquer aplicação Windows, temos ainda uma área destinada à edição e visualização do AGV e da sua trajectória, uma outra para a observação das variáveis ao longo da simulação e outras duas áreas que representam a imagem colhida pela câmera do AGV, antes e depois de analisada.

O programa lê e escreve documentos de extensão **`.sim`** (através dos comandos habituais), que contêm num formato próprio toda a informação pertinente a uma dada simulação (trajectória, posição e orientação do AGV, constantes, etc) *excepto* as regras do motor de inferência.

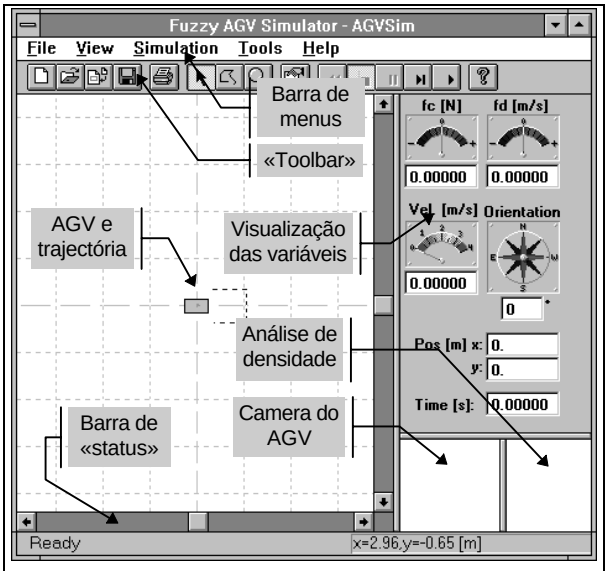


Fig. 1.1 - A janela do AGVSim

Estas são «importadas» (através da opção **`File|Import Rules...`**) de documentos de extensão **`.fdl`**, que contêm a especificação das variáveis e regras *fuzzy* na linguagem FDL (*Fuzzy Description Language*).

Estes ficheiros de regras podem ser editados com qualquer editor de texto (por ex., o *Notepad*), sendo o seu formato descrito detalhadamente na secção 2.5.2.

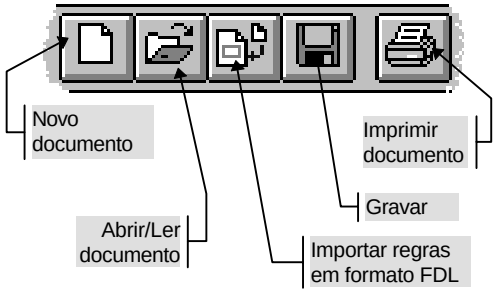
Ao iniciar o programa, este cria automaticamente um documento novo, contendo apenas o AGV (colocado no centro do referencial). Para efectuar uma simulação, o utilizador necessita efectuar dois passos: Desenhar a trajectória que o AGV deve seguir e importar um conjunto de variáveis e regras definidas em FDL.

1.3.1. A *toolbar*

Os comandos mais utilizados possuem um botão equivalente na *toolbar*, pelo que se apresenta aqui uma breve descrição da sua funcionalidade:

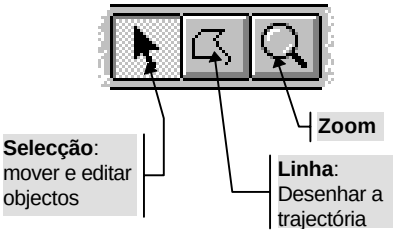
a) Documentos

Eis aqui os botões da «toolbar» relacionados com a manipulação de documentos:



Estes comandos são equivalentes aos existentes debaixo do menu File, e permitem criar, abrir, gravar e imprimir documentos **`.sim`**, para além de importar a base de regras.

b) Ferramentas



O utilizador tem ao seu dispor três ferramentas: A ferramenta de selecção, que serve para «agarrar», editar e posicionar objectos (o AGV e a trajectória pretendida), a ferramenta de desenho, que é utilizada para desenhar a trajectória, e a ferramenta de *zoom*, que permite aumentar ou diminuir a escala de visualização. Estas ferramentas comportam-se de uma forma em tudo idêntica às ferramentas semelhantes encontradas em editores gráficos de todos os tipos, i.e., é possível, por exemplo, seleccionar o AGV, arrastá-lo com o rato até uma nova posição e, fazendo *double-click*, alterar as suas propriedades, ou então seleccionar a trajectória e mover os seus pontos de controlo.

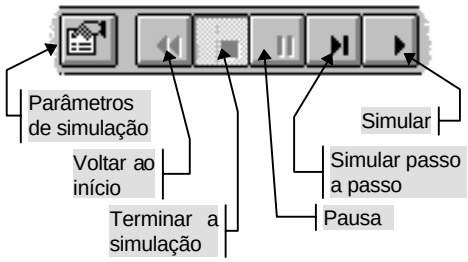
A ferramenta de desenho permite desenhar segmento a segmento (carregando sucessivamente nos locais por onde se pretende fazer passar a trajectória) ou continuamente (movendo o rato e mantendo o botão premido)[1], terminando-se o desenho com um *double-click* no ponto final.

Quanto à ferramenta de *zoom*, basta dizer que permite fazer *zoom in* com um *click* e *zoom out* com *Ctrl+click*.

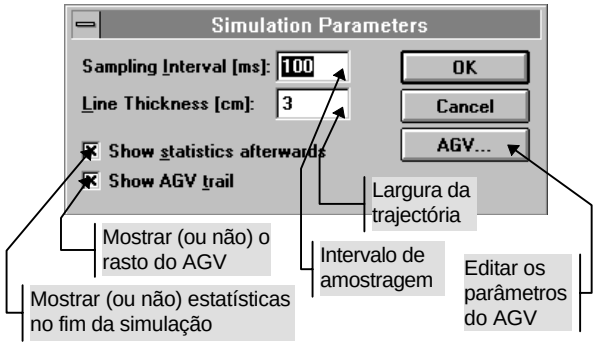
1 De notar que, em máquinas lentas, é pouco recomendável fazer movimentos demasiado rápidos, por forma a permitir ao computador actualizar o ecran.

c) Controlo da simulação

Este conjunto de botões permite controlar o progresso da simulação de forma intuitiva:

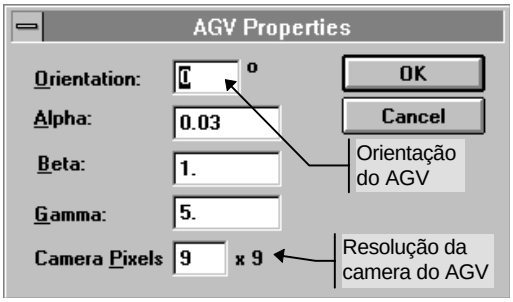


Carregando no botão dos parâmetros de simulação, surgirá a seguinte *dialog box*, que permite alterar directamente o intervalo de amostragem e a largura da trajectória «pintada» no chão:



Pode-se ainda especificar se se pretende ver o «rasto» que o AGV deixa no seu trajecto e se se pretende obter um conjunto de estatísticas no fim da simulação.

Carregando no botão **`AGV...`**, podemos ainda alterar as propriedades do AGV:



(Esta *dialog box* também surge quando fazemos *double-click* no AGV com a ferramenta de selecção).

1.3.2. A simulação

a) Os ficheiros `.fdl`

Como já foi dito anteriormente, para efectuar uma simulação é necessário desenhar uma trajectória para o AGV seguir e importar um documento **`.fdl`** contendo regras e variáveis difusas. O ficheiro pode conter qualquer número de variáveis e regras, mas a aplicação apenas lerá e atribuirá valores às seguintes variáveis:

- Variáveis de entrada (correspondentes aos sectores da camera):

c00	**c01**	**c02**
c10	**c11**	**c12**
c20	**c21**	**c22**

- Variáveis de saída: **fc** e **fd**.
- Variável de *feedback* (declarada como uma variável de entrada): **Velocity**

Não é necessário definir todas estas variáveis - é perfeitamente possível definir um AGV que «veja» apenas os três sectores do topo da camera, por exemplo, ou até um AGV que não vire... A aplicação afixará um aviso por cada variável não definida, mas funcionará correctamente.

A variável **Velocity** merece aqui uma atenção especial. Esta variável exprime a velocidade actual do AGV em m/s, e é calculada internamente a partir das equações da dinâmica do AGV. Não é uma variável de entrada «pura», visto que depende directamente das variáveis de saída, e é tipicamente utilizada para exercer um controlo mais «fino» sobre a aceleração do AGV.

É portanto da responsabilidade do utilizador definir convenientemente o documento **.fdl** conforme os resultados pretendidos. No caso de existirem erros no ficheiro (por exemplo, o uso numa regra de uma variável não declarada), será emitido um aviso apropriado.

b) O decorrer da simulação

Após desenhar a trajectória pretendida e importar um conjunto de variáveis e regras, pode-se então iniciar a simulação com o comando **Play**.

Durante a simulação, é possível fazer *zoom* sobre o centro do AGV utilizando os comandos **View|100%**, **View|200%** e **View|400%** (ou **Ctrl+1**, **2** e **4**), ou manter o AGV no centro da janela com o comando **Simulation|Track AGV** (não muito recomendável em máquinas lentas).

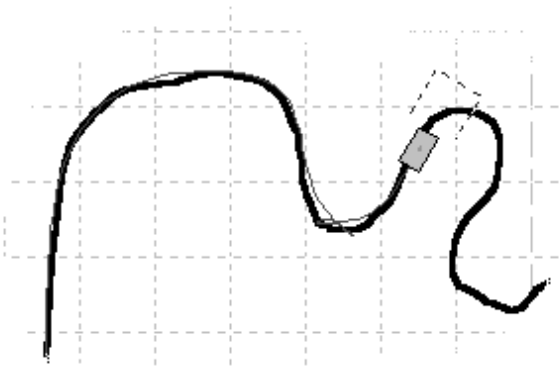


Fig. 1.3 - O AGV a seguir uma trajectória, com o seu «rasto» visível.

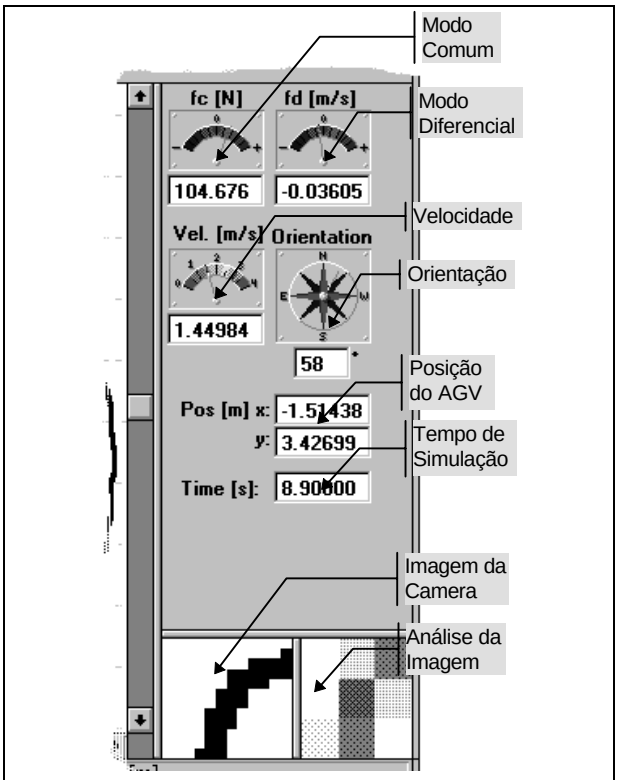


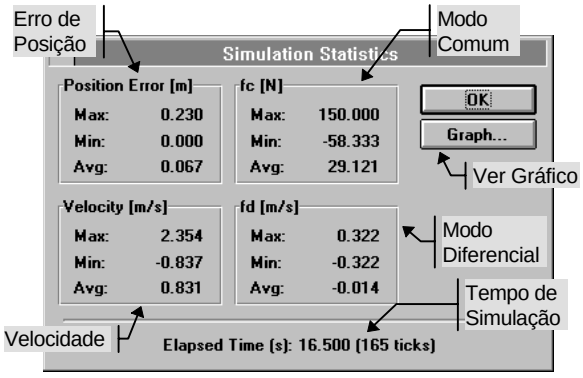
Fig. 1.2 - O painel de visualização das variáveis e as zonas da camera.

Os valores actuais de posição, velocidade e orientação, bem como das variáveis de saída, estão patentes no painel lateral e são constantemente actualizados. Sob este painel podem-se ainda observar a imagem recolhida pela camera do AGV e a análise efectuada (os valores de densidade obtidos são posteriormente atribuídos às variáveis de entrada do controlo).

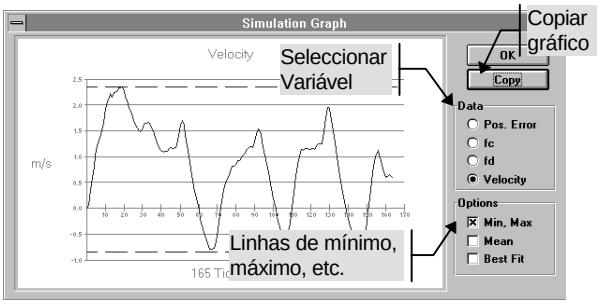
Através do comando **Pause**, pode-se parar temporariamente a simulação e fazer quaisquer pequenas alterações aos parâmetros do AGV ou à trajectória.

c) Estatísticas

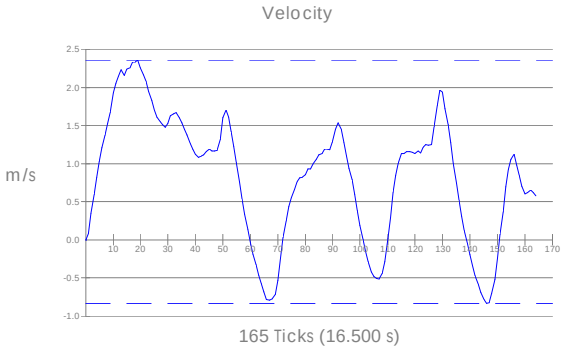
Ao terminar ou recomeçar a simulação (através dos comandos **Stop** e **Rewind**, respectivamente), surge a seguinte *dialog box* com os resultados da recolha de dados durante a simulação:



Esta *dialog* apresenta apenas os valores mínimo, máximo e médio do erro de posição, da velocidade, do modo comum e do modo diferencial. Carregando no botão **`Graph...`**, é possível ver o gráfico de qualquer uma destas grandezas em relação ao tempo:



Carregando no botão **`Copy`**, gráfico será copiado para o «Clipboard», podendo então ser inserido em qualquer outra aplicação - por exemplo, eis aqui o mesmo gráfico da *dialog box*, inserido neste documento de Word:



2. Relatório técnico

2.1. Dinâmica do AGV

Dadas as equações da dinâmica do AGV presentes no enunciado, torna-se necessário atribuir valores às constantes α, β e γ por forma a que a simulação se comporte de acordo com o mundo real. A escolha destes valores foi feita tendo em conta as propriedades físicas do veículo e estabelecendo uma relação entre as equações do enunciado e as equações da cinemática:

2.1.1. Grandezas físicas do AGV

Neste caso, as grandezas físicas escolhidas para o AGV foram as seguintes:

- Comprimento: 50 cm
- Largura: 30 cm
- Distância entre rodas motrizes: 20 cm (com o eixo 15 cm à frente do centro de massa)
- Massa: 30 Kg

2.1.2. Determinação das constantes

A equação

$$\frac{dv}{dt} = \alpha f_c - \beta v = a(t)$$

do enunciado dá-nos a aceleração do AGV, que pode ser separada em duas componentes: αf_c, que é a aceleração dada pelos dois motores, e $-\beta v$, que é a componente de atrito.

Dado que $F = ma$ e que neste caso a força é dada por f_c, vem que

$$\alpha f_c = \frac{f_c}{m} \Leftrightarrow \alpha = \frac{1}{m}$$

ou seja, α é o inverso da massa do AGV. No nosso caso, $\alpha = 1/30 \approx 0.03$.

Para a componente de atrito, foi escolhido um valor de $\beta = 1$, dado que não se dispunha de dados para concluír sobre o coeficiente de atrito aplicável ao AGV (excepto que consiste num valor linearmente dependente da velocidade).

Tratando-se de um sistema de primeira ordem, e após aplicar a transformada de Laplace, verifica-se que tem um pólo em β. Desta forma, foi escolhida uma frequência de amostragem dez vezes superior ao valor de β ($f = 10Hz$), que equivale a um período de simulação de 100 ms.

Quanto a γ, basta aplicar a equação da velocidade angular:

$$\frac{d\theta}{dt} = \gamma f_d \Leftrightarrow \omega = \frac{v}{r}$$

Considerando $f_d = v$ (ou seja, um dos motores parado), verifica-se que γ é inversamente proporcional à distância entre as rodas (no nosso caso para r = 0,2 m, $\gamma = 5$).

2.2. Representação interna dos objectos

Os objectos directamente manipuláveis pelo utilizador (o AGV e a trajectória a seguir) são subclasses de **`CDrawObject`**, uma classe que implementa toda a funcionalidade de interacção com o utilizador (mover, «abrir», editar, etc.).

2.2.1. A trajectória

Tanto a trajectória a ser seguida pelo AGV como o «rasto» que este deixa são representadas internamente por objectos da classe **`CPolyline`**. Estes objectos, que implementam uma linha poligonal genérica, contêm um *array* de objectos **`CPoint`** (os vértices da linha poligonal), permitindo representar a trajectória do AGV

em números inteiros com a precisão (à escala escolhida) de 1 cm. Esta representação foi escolhida devido principalmente a questões de implementação:

- As primitivas gráficas de Windows só trabalham com **`CPoints`**, pelo que a rapidez de actualização do ecran seria comprometida pelo uso de representações intermédias.
- No caso de ser necessária mais precisão (por exemplo, para o cálculo do erro de posição), um **`CPoint`** pode ser implicitamente convertido num **`CVector`** - uma classe originalmente desenvolvida para facilitar a rotação do AGV, que representa um vector em vírgula flutuante.

A classe **`CPolyline`** é ainda responsável pelo cálculo do erro de posição do AGV, cujo algoritmo é descrito na Secção 2.4.

2.2.2. O AGV

A classe **`CAGV`** é responsável pelo cálculo da dinâmica do veículo, pela sua representação gráfica e pelo processamento da imagem da camera, actualizando-os a cada *tick* do relógio de simulação.

Um objecto desta classe mantém informação sobre a posição do centróide do AGV, a sua orientação, a sua velocidade actual e o resultado da análise da camera.

2.3. A camera

Ao comparar os diversos métodos de controlo para AGVs, o uso de uma camera é, sem dúvida, o mais versátil. Comparado com, por exemplo, o controlo por indução (que dá informação unidimensional - o desvio - e a uma distância fixa do centro de massa do AGV), a camera permite recolher informação ao longo da «profundidade» do seu campo visual, permitindo um melhor controlo nas curvas.

O uso da camera implica, no entanto, que o sistema de controlo não tenha explicitamente como entradas os erros de posição e de orientação. Em contrapartida, o que sucede é que estes são calculados implicitamente pelo controlo difuso a partir da imagem.

2.3.1. Rasterização

A rasterização da imagem da camera envolve um conjunto de passos:

- Em primeiro lugar, é calculado o rectângulo envolvente da camera (que abrange toda a àrea da camera e um pouco mais em redor) no espaço de coordendadas do documento de simulação.
- Em seguida, utilizando o algoritmo de Cohen-Sutherland para *clipping* de uma recta por um rectângulo [1], são identificados os segmentos da trajectória que intersectam esse rectângulo envolvente.
- Esses segmentos são então transformados para o espaço de coordenadas da camera e desenhados com a primitiva **`LineTo`** num *device context* criado em memória.
- Esse *device context* possui um espaço de coordenadas lógicas e uma resolução física idênticos à da camera e um *bit* de profundidade, contendo assim o *bitmap* resultante da rasterização.

Desta forma, a rasterização propriamente dita (a atribuição de valores aos *pixels*) é feita pelo GDI do Windows, com todas as vantagens que daí advêm (rapidez, eficiência e precisão).

2.3.2. Método de análise

No entanto, a informação recolhida pela camera é inútil sem uma pré-análise adequada, uma vez que o motor de inferência pouco ou nada sabe de imagens, *pixels* e outras coisas que tais... - a imagem deve, portanto, ser transformada num conjunto de variáveis de entrada. Resta portanto escolher um método de análise adequado, i.e., que possa ser implementado com o mínimo de recursos computacionais e que não leve a um número exagerado de variáveis de entrada.

Foi decidido dividir a camera em 9 zonas de iguais dimensões e medir a densidade de pontos negros por zona - atribuíndo o valor de densidade de cada zona a uma variável de entrada.

E porquê 9 zonas? Tomemos um exemplo concreto: Se a camera fosse dividida em apenas 4 zonas, a única diferença entre este tipo de controlo e o controlo por indução seria uma maior «profundidade» - algo como um segundo conjunto de sensores colocado mais à frente - enquanto que um número superior a 9 tornaria a escrita das regras muito mais difícil - dado que, ao aumentarmos linearmente o número de variáves, aumentamos exponencialmente a complexidade das regras (para cobrir todos os casos).

As 9 zonas fornecem ao AGV informação suficiente para saber se está ou não a seguir correctamente a trajectória, se pode ou não acelerar com confiança e que grau de viragem deve empregar para efectuar as curvas (consoante a maior ou menor proximidade do ponto de desvio). Permitem-lhe ainda ultrapassar «cruzamentos» com relativa facilidade (o que constitui um problema grave para um AGV com apenas 4 zonas).

2.3.3. Resolução e largura da trajectória vs. densidade

A camera abrange uma área de 70×70 cm ligeiramente à frente do AGV, e pode ter qualquer resolução entre 9×9 e

27×27 *pixels* - ou seja, cada *pixel* da camera pode abranger uma área que varia entre 7,7×7,7 e 2,6×2,6 cm.

A relação entre a resolução da camera, a largura da trajectória e os valores de densidade por zona é relativamente complexa:

Por exemplo, desprezando os erros introduzidos pela rasterização e fixando a largura da trajectória em 3 cm, com uma resolução de 9×9 *pixels* o AGV «verá» uma linha com 1 *pixel* de largura. Ora, uma linha de 1 *pixel*, numa camera com esta resolução, induz um valor máximo de densidade (num sector de 3×3 *pixels*) de 0,3(3) - isto porque, em cada momento, a linha cobrirá no máximo 1/3 dos *pixels* de qualquer sector.

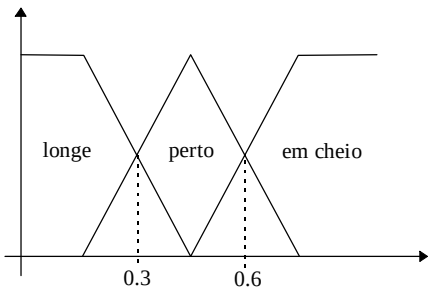
Aumentando a resolução da camera para o máximo de 27×27 *pixels* (mantendo fixa a largura da trajectória), a densidade máxima por sector baixa para 0,1(1) - pois agora apenas 1/9 dos *pixels* está coberto pela linha.

Portanto, para uma mesma largura de linha, os valores de densidade por zona variam inversamente com a resolução da camera. Este facto reflete-se no controlo, que necessita assim de ser alterado para diferentes características da camera e de largura da trajectória.

2.3.4. Densidade vs. funções de pertença

A grande variação de valores de entrada em função das características da camera e largura da trajectória obriga a definir o número e a forma das funções de pertença das variáveis de entrada por forma a que o valor de activação propagado ao(s) consequente(s) da(s) regra(s) seja adequado.

Por exemplo, para o caso de uma linha com 12 cm de largura e uma camera de 9×9 pixels, as funções de pertença «longe», «perto» e «em cheio» de uma dada zona da camera poderiam ser as seguintes:



Se a linha estivesse «longe» desta zona, haveria menos do que 3 *pixels* a negro, enquanto que se passasse «em cheio» pela zona haveria certamente mais do que 6 *pixels* a negro.

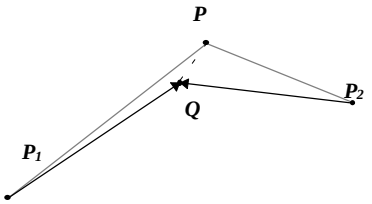
No entanto, se a linha tivesse metade da grossura (e portanto ocupasse no máximo 3 *pixels* da zona), o valor de densidade nunca ultrapassaria 0,3 - pelo que as formas das funções de pertença teriam de ser ajustadas convenientemente, sob pena de qualquer regra que dependesse de «em cheio» propagar valores quase nulos ao consequente.

2.4. Determinação do erro de posição

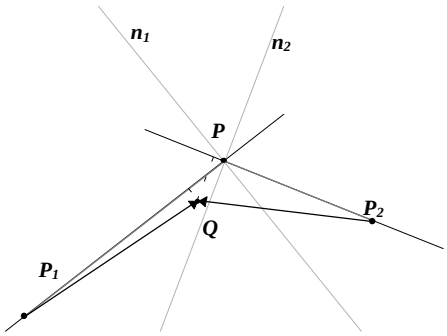
O algoritmo de determinação do erro de posição baseia-se em calcular a distância de um ponto (o centróide do AGV) à linha poligonal que representa a trajectória a seguir.

Mas tal não é um problema trivial, pois calcular a distância ao vértice mais próximo ou à recta definida pelos dois vértices mais próximos dá resultados incorrectos.

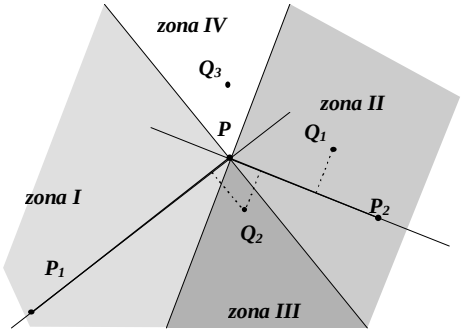
Por exemplo, consideremos $\boldsymbol{Q}$ (o centróide do AGV) e três vértices da trajectória, tais que $\boldsymbol{P}$ é o vértice mais próximo de $\boldsymbol{Q}$ e $\boldsymbol{P_1}$ e $\boldsymbol{P_2}$ os vértices antes e depois de $\boldsymbol{P}$, com $\boldsymbol{P_1Q} > \boldsymbol{P_2Q}$:



Neste caso, é óbvio que a distância $\boldsymbol{PQ}$ não é a distância mais curta entre $\boldsymbol{Q}$ e a linha poligonal. Nem, como se pode ver na figura seguinte, o é a normal ao segmento entre os dois vértices mais próximos:



Como se pode ver, neste caso é a normal a $\boldsymbol{P_1P}$ que representa a distância mais curta entre $\boldsymbol{Q}$ e a linha poligonal. Mas que dizer das normais a $\boldsymbol{P_1P}$ e $\boldsymbol{P_2P}$ que passam por $\boldsymbol{P}$? Estas normais delimitam o espaço em quatro zonas distintas:



Esta divisão reduz o problema a quatro casos distintos:

- Se ***Q*** está na zona I, então a distância pretendida é a normal a $\boldsymbol{P_1Q}$.
- Se ***Q*** está na zona II, então a distância pretendida é a normal a $\boldsymbol{P_2Q}$.
- Se ***Q*** está na zona III, então a distância pretendida é a menor das duas normais.
- E, finalmente, se ***Q*** está na zona IV, então a distância pretendida é ***PQ***.

A decisão é feita utilizando operações entre vectores: calculam-se as projecções dos vectores $\boldsymbol{P_iQ}$ sobre os vectores $\boldsymbol{P_iP}$, e compara-se o comprimento das projecções ao dos vectores - se a projecção é maior do que o vector $\boldsymbol{P_iQ}$, então ultrapassou-se a normal $\boldsymbol{n_i}$ - isto evita calcular explicitamente as normais.

2.5. Controlo por lógica difusa

2.5.1. Descrição do sistema

Nesta secção descrevem-se as opções tomadas a nível de implementação do sistema de controlo por lógica difusa como sendo a representação de funções de pertença e os métodos utilizados para a desfuzificação.

a) Funções de pertença e fuzificação

As funções de pertença são definidas por uma linha poligonal que varia no eixo dos *xx* entre o mínimo e o máximo dos valores nítidos da variável *fuzzy* respectiva, e no eixo dos *yy* entre 0 e 1. As razões que levaram a esta escolha foram principalmente a facilidade de representação interna e a facilidade de manipulação. Além disso, é comum aproximar por segmentos de recta uma função de pertença que idealmente seria curvilínea, dada a pequena diferença em relação à função original e a maior simplicidade de definição.

Foi ainda ponderada a possibilidade de representar apenas funções de pertença com forma trapezoidal e triangular (à semelhança de outros sistemas estudados), mas esta opção é demasiado limitativa - dado não permitir, por exemplo, a representação de uma função que seja a «negação *fuzzy*» de uma outra com forma triangular.

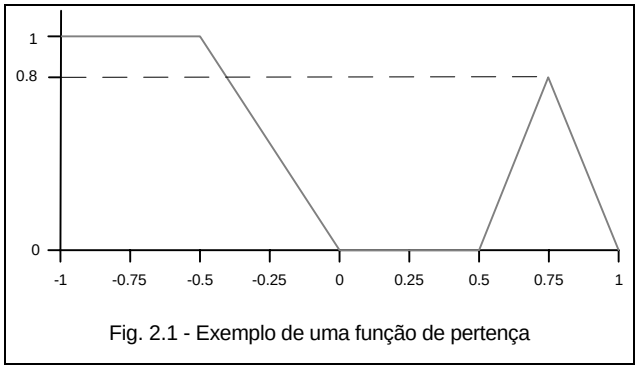


Fig. 2.1 - Exemplo de uma função de pertença

A definição de uma função de pertença é feita por um conjunto de pontos (pares (*x*,*y*)), em que cada par de pontos define um segmento de recta da função. Como as funções de pertença têm de possuir valores de *y* para todos os valores possíveis de *x*, admitem-se inicialmente definidos os pontos (<*min*>, *0*) e (<*max*>, *0*), em que <*min*> e <*max*> indicam respectivamente os valores mínimo e máximo da variável. Estes pontos podem ser redefinidos indicando um novo ponto com o mesmo valor *x* e um outro valor *y*. Desta forma, uma função de pertença para a qual não sejam indicados quaisquer pontos é simultaneamente constante e nula.

A título de exemplo, os pontos (*-1, 1*), (*-0.5, 1*), (*0,0*), (*0.5, 0*), (*0.75, 0.8*) e (1, 0) definem a função apresentada na Fig. 2.1 .

O nível de pertença de um determinado valor nítido é calculado a partir dos dois pontos com valor de *x* mais próximo tais que o valor nítido esteja entre eles - o processo consiste em determinar o segmento de recta definido por estes dois pontos e achar o valor de *y* do ponto de intersecção deste segmento com a recta vertical que passa pelo valor nítido dado.

b) Variáveis *Fuzzy*

A definição de uma variável *fuzzy* consiste em três valores nítidos (mínimo, máximo e inicial) e num conjunto de funções de pertença associadas a essa variável - no caso de se tratar de uma variável de saída, é necessário ainda indicar o método de desfuzificação.

c) Desfuzificação

A transformação de um valor difuso para um valor nítido (apenas efectuada para as variáveis de saída), consiste numa sequência de passos (correlação, recombinação e cálculo do valor nítido), existindo diversas alternativas para efectuar cada passo.

i. Correlação

Para efectuar a correlação de uma função de pertença de saída por um nível de activação foi usado o método do produto, que consiste em multiplicar toda a função pelo nível de activação. A razão desta escolha prende-se meramente com a maior facilidade de implementação, dada a estrutura de representação interna das funções de pertença.

Estas são representadas por conjuntos de pontos, pelo que implementar o método escolhido consiste apenas em multiplicar o valor de *y* de cada ponto pelo nível de activação. Para implementar o método de correlação mínima, seria necessário introduzir novos pontos na função (onde a recta que passa pelo valor de pertença

intersecta a função) e retirar outros (os situados acima dessa recta).

ii. Recombinação

Para a recombinação das várias funções de pertença que resultaram não nulas do passo anterior, foi usado o método da união. Novamente, a simplicidade de implementação foi um factor de peso na escolha. Para a implementação de qualquer outro método haveria a necessidade de construir uma nova função de pertença à custa de todas as que definem a variável, verificar as intersecções entre elas e criar novos pontos.

iii. Cálculo do valor nítido

Neste passo é calculado o valor nítido da variável consoante o resultado dos passos anteriores. Dado que é o mais importante do processo de desfuzificação, e para que possa ser estudada a sua influência, foram implementados quatro métodos diferentes:

- ***Modal***

Este método consiste em escolher o máximo da função que resulta da recombinação. Podem portanto surgir situações de conflito, quando mais do que uma função de pertença possui o mesmo nível de activação ou quando existem vários valores nítidos para o máximo nível de activação. Em ambos os casos é escolhido o valor nítido mais à esquerda no eixo dos *xx*.

Este método tem a desvantagem de ignorar acções potencialmente importantes, pois mesmo que a função tenha outros máximos locais próximos do máximo, estes não são considerados no resultado final.

- ***Média ponderada***

Este método é um pouco mais elaborado que o anterior e pondera os máximos de cada uma das funções de pertença da variável com o respectivo nível de activação desde que este último seja não nulo. Deste modo, as acções que anteriormente eram completamente ignoradas têm agora um peso no resultado final correspondente ao seu nível de activação.

- ***Singleton***

Conforme descrito em [3], consideram-se as funções de pertença como sendo definidas apenas por um segmento de recta vertical, sendo o valor nítido calculado através da fórmula

$$X = \frac{\sum_{i=0}^{n} x_i . n_i}{\sum_{i=0}^{n} n_i}$$

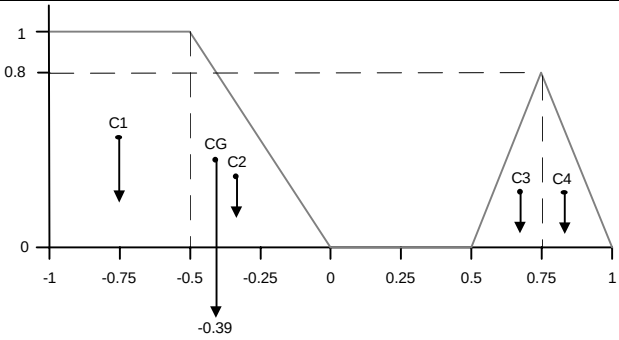


Fig. 2.2 - Exemplo do cálculo do valor nítido pelo método do centro de gravidade

- em que x_i representa os valores no eixo dos *xx* dos máximos das várias funções de pertença e n_i os respectivos níveis de activação.

De notar que, para não obrigar a que exista um modo particular de definir as funções de pertença só para este método, utiliza-se o segmento de recta que corresponde ao máximo valor difuso. *É da responsabilidade de quem define as funções de pertença fazê-lo de modo a que possuam um único máximo quando se pretender usar este método de desfuzificação.*

- ***Centro de gravidade***

Este é o método mais elaborado de todos os implementados, consistindo em calcular o centro de gravidade da função recombinada, sendo o valor nítido escolhido a coordenada *x* desse mesmo centro.

Para efectuar este cálculo, o problema de calcular o centro de gravidade de uma função definida por segmentos foi subdividido no problema de calcular centros de gravidade de figuras geométricas dadas por dois pontos consecutivos, que vão sendo combinados para dar o centro de gravidade total - desta forma o cálculo reduz-se à determinação dos centros de rectângulos e triângulos rectângulos.

2.5.2. A linguagem FDL

Para permitir o teste de diferentes tipos de controlo, existe a necessidade de alterar com frequência todos os seus parâmetros, desde a forma e número das funções de pertença até às regras. Posto este problema, foi definida uma pequena linguagem, baptizada FDL - *Fuzzy Description Language* - que permite especificar completamente um sistema de controlo por lógica difusa. Nesta secção explicam-se sucintamente a sintaxe e a semântica desta linguagem:

a) Definição de variáveis

Os pontos de entrada e saída do sistema de controlo são as variáveis difusas. Neste nível existe a necessidade de distinguir entre variáveis de entrada e de saída, visto que

as últimas têm associado um método de desfuzificação. Este é especificado independentemente para cada variável (em vez de ser único para todo o sistema), permitindo assim a combinação de vários métodos num mesmo sistema. A principal vantagem desta abordagem consiste em permitir a escolha de métodos mais simples (e portanto mais rápidos e menos precisos) para as variáveis menos críticas do controlo e métodos mais elaborados para as variáveis onde é necessário um valor de actuação mais preciso.

A sintaxe da definição de uma variável de saída é a seguinte:

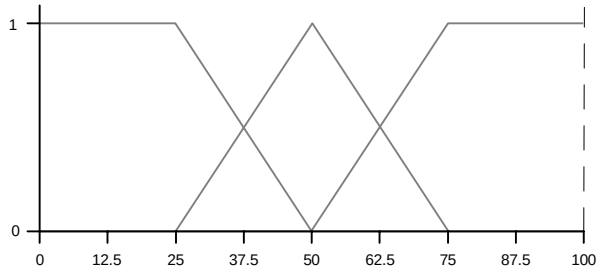
```
OUTVAR SINGLETON fc
  MIN 0
  MAX 100
  DEFAULT 0

  MEMBER Small
    POINTS 0,1 25,1 50,0
  END

  MEMBER Medium
    POINTS 25,0 50,1 75,0
  END

  MEMBER Large
    POINTS 50,0 75,1 100,1
  END
END
```

Esta listagem define a seguinte variável de saída:



Passando a analisar a sintaxe:

- A palavra reservada **OUTVAR** (seguida do nome da variável, que serve de identificador na definição das regras de controlo), indica que se trata de uma variável de saída, que é obrigatoriamente seguida da indicação do método de desfuzificação - neste caso **SINGLETON** (os métodos são indicados pelas palavras reservadas da Tab. 2.1).

MODAL	Modal
AVERAGE	Média ponderada
SINGLETON	Singleton
COG	Centro de gravidade

Tab. 2.1 - Palavras reservadas dos métodos de desfuzificação

Para a definição de uma variável de entrada, utilizar-se-ia a palavra reservada **INVAR** em vez de **OUTVAR** e não se indicaria qualquer método de desfuzificação.

- Em seguida, as palavras reservadas **MIN**, **MAX** e **DEFAULT** indicam respectivamente os valores nítidos mínimo, máximo e inicial.

Define-se então a lista de funções de pertença da variável.

- A definição de uma função de pertença começa pela palavra reservada **MEMBER**, seguida do seu nome, e termina com **END**.
- A palavra reservada **POINTS** inicia a indicação dos pontos que definem a função.

De forma análoga ao nome da variável, o nome da função de pertença é utilizado na definição das regras para identificar uma determinada função de pertença dentro de uma variável.

Cada variável pode ter um número arbitrário de funções de pertença a ela associadas.

b) Regras

A definição das regras de controlo de um sistema difuso começa com a palavra reservada **FUZZY** seguida de um identificador e termina com **END**. Cada regra tem a forma geral

```
RULE <identificador>
  IF <expressão> THEN <expressão>
END
```

onde **<expressão>** é, ou uma referência a uma função de pertença de uma variável expressa como:

```
<id var> IS <id membro>
```

ou então uma combinação de expressões utilizando os operadores AND, OR e NOT:

```
<expressão> AND <expressão>
<expressão> OR  <expressão>
<expressão> NOT <expressão>
```

com a particularidade de que *no consequente das regras apenas é admitido o operador **AND***.

3. Estudo dos parâmetros de controlo do AGV

3.1. Considerações

Como parâmetros que influenciam o controlo do AGV, temos:

- O número de variáveis de entrada do sistema

- O número e a forma das funções de pertença das variáveis
- O método de desfuzificação usado
- A estrutura das regras, que podem ser arbitrariamente numerosas e complexas.

Dado que existe um número muito elevado de combinações destes factores, não os podemos estudar a todos. É portanto necessário definir uma estratégia que permita tirar o máximo de conclusões possíveis sobre os aspectos relevantes do controlo.

A estratégia escolhida consistiu em definir várias trajectórias com um nível de dificuldade gradualmente crescente, e ir refinando o controlo por forma a que o AGV consiga seguir todas essas trajectórias.

3.2. Os testes

3.2.1. teste01

Foram inicialmente consideradas apenas as variáveis de entrada c_{00}, c_{01} e c_{02}, correspondentes ao «topo» da camera - este tipo de controlo assemelha-se ao controlo por indução, em que o AGV dispõe apenas de informação sobre o desvio.

Para essas variáveis, foi escolhido apenas um adjectivo **«Crowded»**, que indica se o sector da camera tem ou não *pixels* a negro. Para *fc*, também um único adjectivo **«Something»**, que permite indicar simplesmente se existe ou não *fc* - desta forma o valor de *fc* vai depender unicamente do nível de activação das regras em que esta figura no consequente.

fd, por sua vez, tem dois adjectivos **«Left»** e **«Right»**, para permitir a mudança de direcção:

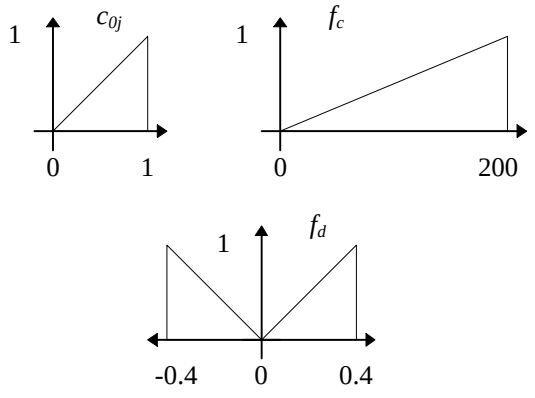


Fig. 3.1 - funções de pertença do ficheiro **`teste01.fdl`**

Foram definidas apenas três regras, uma para virar à esquerda, outra para virar à direita e outra para acelerar:

```
RULE TurnLeft
  IF (c00 IS Crowded)
  THEN(fd IS Left)
END

RULE TurnRight
  IF (c02 IS Crowded)
  THEN (fd IS Right)
END

RULE Go
  IF (c01 IS Crowded)
  THEN (fc IS Something)
END
```

Este tipo de controlo seguiu correctamente a trajectória do primeiro nível, dada a simplicidade desta. No entanto, no segundo nível «perdeu» a curva mais apertada. Um pequeno ajuste nas funções de pertença de c_{0j} para a forma da Fig. 3.2 (**`teste01b.fdl`**) permitiu ao AGV simultaneamente acelerar mais e virar mais nas curvas, pelo que conseguiu então seguir correctamente o segundo nível, com os seguintes resultados:

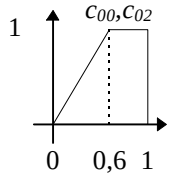


Fig. 3.2 - ajuste efectuado a c_{00} e c_{02}

Erro máximo:	0.324 m
Erro médio:	0.117 m
Velocidade média:	2.431 m/s
Tempo (1 volta):	15,5 s

Tab. 3.1 - resultados do **`teste01b.fdl`**

Dado que este valor de erro médio era relativamente elevado (uma vez que o AGV estava a «cortar» as curvas), foi decidido utilizar apenas as variáveis do «fundo» da camera, na tentativa de contornar este problema (**`teste01c.fdl`**).

Uma vez que a distancia entre os «sensores» e o centro de rotação do AGV foi reduzida, este passou a virar mais «em cima» da trajectória. Para este teste, o AGV fez uma volta completa com um valor de erro médio de 0.041m e uma velocidade média de 2.196 m/s - reduziu-se bastante o erro mantendo aproximadamente a mesma velocidade. No entanto, o AGV despistou-se logo no ínicio da segunda volta...

Tentou-se ainda utilizar apenas os sectores médios (**`teste01d.fdl`**), mas com resultados ainda piores, uma vez que o AGV acelerava mais logo de início e perdia a trajectória logo na primeira curva - e isto porque é no sector central que a densidade de pontos é maior em média.

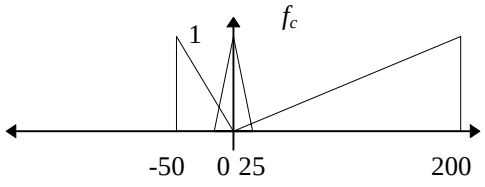
- De notar que nstas simulações foram utilizados todos os métodos de desfuzificação suportados, sem grande influência no resultado.

Conclui-se portanto que é necessário controlar a velocidade do AGV por forma a reduzi-la nas curvas.

3.2.2. teste02

Pelas regras dos testes anteriores, quando não existiam *pixels* no sector central *fc* era nulo. De modo a reduzir a velocidade nas curvas, é portanto necessário dar a *fc* valores negativos, por forma a que o AGV trave realmente.

Para tal, modificou-se a variável de saída *fc*, definindo as funções de pertença «**Negative**», «**Zero**» e «**Positive**», em que o adjectivo **Positive** é idêntico ao «**Something**» dos testes anteriores e adicionou-se a regra de travagem (ver Fig. 3.3). Todas as outras variáveis e regras permaneceram inalteradas.




```
RULE Break ; Travar se...
  IF NOT (c21 IS Crowded)
  THEN (fc IS Negative)
END
```

Fig. 3.3 - Alterações realizadas em **teste02.fdl**

Como resultado destas alterações, o AGV deixou de perder a trajectória, reduzindo ligeiramente o erro para 0.036 - mas a velocidade média diminuiu também para 1.715m/s, dado que o AGV passou a «travar» nas curvas.

Para tentar repor a velocidade média anterior, modificou-se novamente *fc* aumentando o seu máximo para 300N (**teste02b.fdl**), com resultados desastrosos. O AGV perdeu a trajectória por acelerar demais, e devido à regra de travagem que força o *fc* a valores negativos, começou a recuar sem voltar a encontrar a trajectória.

Optou-se então por travar ainda mais nas curvas, passando o valor mínimo de *fc* para -100N (**teste02c.fdl**). O resultado foi que o AGV seguiu sempre a trajectória, mas sem aumentar a velocidade média.

Em simultâneo com as alterações em *fc*, foi testada a influência dos métodos de desfuzificação. Neste conjunto de testes, a forma de desfuzificação da variável *fc* tem grande influência (mantendo a forma de desfuzificação de *fd*), dado que estão duas regras a influênciá-la - o que torna a ponderação destas regras um factor determinante no resultado final do valor da variável.

Por exemplo, dado que o método **MODAL** não faz qualquer tipo de ponderação, é incapaz de fazer com que o AGV ande sequer, pois acelera e trava alternadamente (quando a densidade de pontos no centro é ou não superior a 0.5), sempre com os valores máximo e mínimo.

Por sua vez, o método **AVERAGE** faz com que a velocidade média seja da ordem dos 0.7m/s. Isto porque, como nenhuma das regras que influenciam *fc* está desactivada, existe sempre uma ponderação de -100 com 300 pelo nível de activação destas - o que faz com que *fc* só varie entre -21N e 56N.

O método **SINGLETON** produz os melhores resultados, dado que o seu tipo de ponderação faz com que *fc* varie entre -100N e 200N.

O método de centro de gravidade, devido à forma do adjectivo «**Positive**», tende neste caso a dar a *fc* valores maiores em média - o que faz com que o AGV acelere demais e perca a trajectória.

Obviamente que a desfuzificação de *fd* também tem influência, dado que faz com que o AGV vire mais ou menos (e consequentemente acompanhe a trajectória). Neste caso, dado que as regras que actuam em *fd* raramente estão activas simultaneamente, o método **SINGLETON** não faz nenhuma ponderação e dá sempre valores extremos de *fd* - comportando-se de forma idêntica ao **MODAL**. Isto torna as viragens do AGV mais bruscas, mas permite-lhe seguir curvas mais apertadas. Os outros métodos comportam-se de forma diferente, dando valores intermédios a *fd*, pelo que o facto do AGV perder ou não a trajectória depende principalmente da desfuzificação de *fc*.

É necessário neste ponto salientar que os testes acima foram efectuados com regras que possuíam apenas um antecedente e um consequente. Foi tentada outra abordagem, que consistiu em combinar várias variáveis de entrada no antecedente e várias de saída no consequente das regras - na tentativa de, ao jogar com a combinação dos níveis de activação nas entradas, obter um controlo melhor.

As regras de viragem pretendiam fazer com que o grau de viragem do AGV implicasse um grau de travagem proporcional, visto que é propagado o mesmo nível de activação para ambas as variáveis do consequente. A regra de «**KeepStraight**» pretendia diminuir o nível de activação de «**Positive**» nas curvas, para que este não aumentasse a velocidade (contrariando as outras regras). Deixou assim de existir uma regra de «travagem» (**teste02d.fdl**):

```
RULE TurnLeft
  IF (c20 IS Crowded)
  THEN ((fd IS Left) AND
        (fc IS Negative))
END
```

```
RULE KeepStraight
  IF ((c21 IS Crowded) AND
      NOT(c20 IS Crowded) AND
      NOT(c22 IS Crowded))
  THEN (fc IS Positive)
END


RULE TurnRight
  IF (c22 IS Crowded)
  THEN ((fd IS Right) AND
        (fc IS Negative))
END
```

Verificou-se que este tipo de controlo continua ainda dependente do método de desfuzificação das variáveis de saída, embora no caso de se usar **SINGLETON** em ambas se tenham obtido os valores de 2.787 m/s para a velocidade média e 0.045 m para o erro de posição, em 2 voltas completas ao nível 2 (**level02.sim**) sem perder a trajectória - mas com o AGV a fazer viragens muito bruscas.

Deste conjunto de testes pode-se concluir que é difícil aumentar a velocidade média sem aumentar o erro ou mesmo perder a trajectória. Consequentemente, para manter o erro constante é necessário manter uma velocidade relativamente constante.

Ao testar estes tipos de controlo no nível 3 (**level03.sim**), observou-se que nenhum deles conseguia fazer com que o AGV seguisse correctamente a trajectória na zona de curvas apertadas e atingir simultaneamente uma velocidade média razoável no resto da trajectória. Isto deve-se ao facto de o AGV «olhar» apenas para a linha inferior de sectores da camera, não conseguindo distinguir entre curvas mais ou menos apertadas - e consequentemente «travar» mais numas que noutras.

3.2.3. teste03

Para tentar resolver este problema, passou a utilizar-se todos os sectores da camera, aumentando a informação disponível a nível do controlo e incorporando os melhores aspectos de alguns dos testes anteriores.

Decidiu-se utilizar a desfuzificação por centro de gravidade, dado que fornece resultados mais «correctos» intuitivamente - tornando-se portanto mais fácil escrever regras que dêem o comportamento desejado.

As funções de pertença foram refinadas para a forma da Fig. 3.4, aumentando o seu número e sobrepondo-as parcialmente, por forma a se obterem variações mais suaves e se tentar exercer um controlo mais preciso sobre *fc* e *fd*.

As regras do AGV foram então reescritas (aumentando consideravelmente em número) para utilizar estas funções de pertença, de acordo com dois princípios básicos: Manter a linha no centro do campo de visão e virar proporcionalmente ao desvio desta.

Desta forma, o AGV aumenta a velocidade quando os sectores centrais da camera (c_{01}, c_{11} e c_{21}) estão preenchidos, abrandando sempre que pelo menos o sector do topo (c_{01}) «perde» a linha, e vira mais ou menos consoante a linha está mais perto ou mais longe do seu centro de massa (ver o ficheiro **teste03.fdl** para mais detalhes).

Com os novos adjectivos, e utilizando a combinação de variáveis de saída nos consequentes, o controlo de viragem foi francamente satisfatório em termos de suavidade e precisão (o erro de posição médio baixou para 0.030 m na trajectória **level02.sim**, com uma velocidade média de 2.053 m/s).

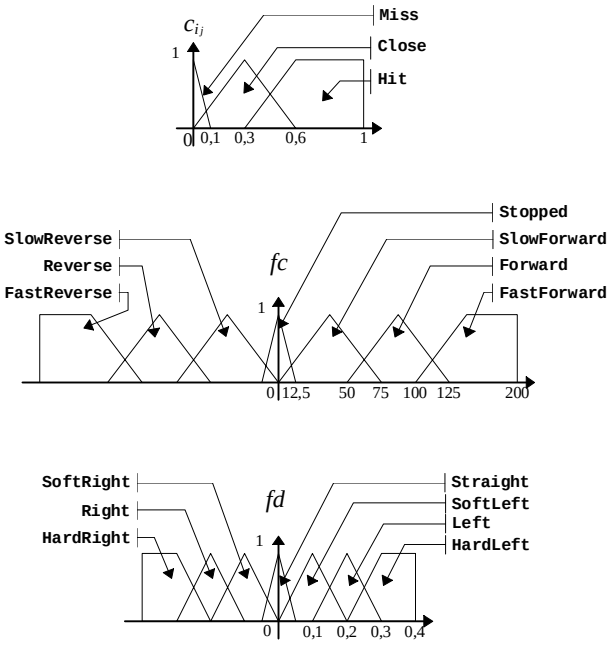


Fig. 3.4 - funções de pertença de **teste03.fdl**

Passando à trajectória **level03.sim**, o controlo de velocidade continuava a levantar problemas. O AGV, apesar de realmente travar nas curvas e as descrever melhor, tendia a exceder muito rapidamente os 2.0 m/s em qualquer troço mais «recto», o que constitui uma velocidade crítica para a abordagem das curvas mais apertadas deste trajecto.

As regras foram então alteradas para o AGV travar ao mínimo desvio da trajectória - o que baixou a velocidade média para o valor de 1.601 m/s, com um valor de erro médio de 0.060 m.

Neste ponto, o AGV era claramente capaz de negociar trajectos difíceis, mas continuava a demonstrar uma tendência para acelerar muito rapidamente em qualquer trajecto mais rectilíneo.

3.2.4. teste04

Dado que o AGV não tinha «noção» da sua velocidade (e nem mesmo da sua aceleração, pois, como é lógico, não se podem escrever regras em função das variáveis de saída), decidiu-se neste ponto incorporar uma variável de *feedback* no prorgrama (**`Velocity`**), para tentar efectuar um controlo mais realista.

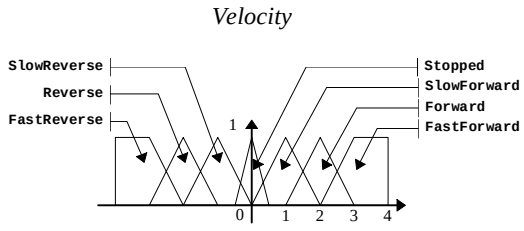


Fig. 3.5 - Funções de pertença da variável de *feedback*

Esta variável, que surge em FDL como uma variável de entrada, é na realidade implementada na aplicação como uma função directa de *fc*, representando a velocidade (em m/s) do AGV no instante imediatamente anterior à avaliação das regras.

Combinando a variável de *feedback* com as entradas da camera, conseguiu-se que o AGV variasse o seu grau de viragem e a sua aceleração também de acordo com a sua velocidade actual, com os melhores resultados desta série de testes para uma volta no nível 3: uma velocidade média de 1.980 m/s e um erro de posição de apenas 0.040m.

As alterações consistiram apenas em duplicar o número de regras de viragem, criando uma conjunto de regras para virar a velocidades elevadas e outro para virar a baixa velocidade.

3.2.5. Limitações

Apesar de tudo, foi impossível escrever um conjunto de regras que permitisse ao AGV descrever uma volta completa ao nível 4 (**`level04.sim`**), dado que, ao perder a trajectória num ângulo agudo, o AGV não conseguia recuar e retomar o trajecto de forma correcta. Isto deve-se ao facto de ser muito difícil exprimir em regras o processo correcto de superar esta situação (pode-se forçar uma viragem quano o AGV não «vê» a trajectória, mas isso levanta demasiados problemas).

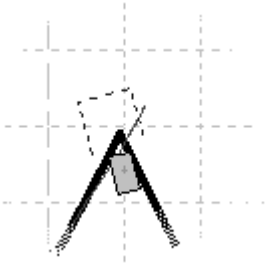


Fig. 3.6 - O AGV preso num ângulo agudo...

Existem duas soluções possíveis: Aumentar a àrea de visão do AGV, expandindo-a para os lados, ou manter informação de estado relativamente aos desvios detectados na trajectória (por exemplo, na figura acima, o AGV «lembrar-se-ia» de «ter visto» um desvio para a direita antes de perder a trajectória, e em vez de recuar viraria para a direita).

No entanto, não se pode esperar que um AGV típico, num ambiente industrial, encontre alguma vez um problema semelhante...

3.3. Conclusões

Deste estudo podemos concluir que o uso de uma camera como *input* do controlo de um AGV permite criar um sistema flexível, preciso e robusto, sendo a lógica difusa um meio de controlo mais do que adequado para a tarefa do controlo propriamente dito.

Isto é tanto mais verdade quando se considera a relativa pobreza da informação processada - sem utilizar dados absolutos de posição, orientação ou velocidade (ou mesmo todos os sectores da camera), e com conjuntos de regras e variáveis muito simples, o AGV foi imediatamente capaz de descrever satisfatoriamente a maior parte das trajectórias.

Pode-se também concluir que o uso de informação de profundidade não é estritamente necessário para um AGV industrial, pois a *performance* da série de testes **`teste02x.fdl`** foi mais do que adequada nas trajectórias **`level01.sim`** e **`level02.sim`** (as trajectórias típicas de ambientes industriais não têm concerteza curvas tão apertadas como **`level03.sim`**...).

No entanto, o uso de todos os sectores da camera tem vantagens inegáveis: O AGV definido no teste 3 foi capaz de negociar com relativa facilidade a trajectória do nível 3 e grande parte da do nível 4. O erro de posição foi consideravelmente reduzido (devido a travar consoante a inclinação da curva) e a velocidade média foi aumentada (por acelerar com «confiança» nos trajectos rectilíneos).

Verificou-se ainda que, no caso específico do controlo de *fc*, a introdução de uma variável de *feedback* de velocidade permitiu exercer um grau de controlo mais preciso, pois a informação que o AGV possuia era ainda insuficiente para lidar convenientemente com curvas muito apertadas.

Do ponto de vista da lógica difusa propriamente dita, verificou-se que não existe uma forma absolutamente correcta para especificar tanto o número como a forma das funções de pertença, embora seja claro que o aumento do número de funções de pertença e a sua sobreposição parcial aumentam a suavidade e a precisão do controlo.

Por outro lado, a combinação de variáveis de saída no consequente das regras demonstrou ser uma forma muito útil (e intuitiva) de controlar o AGV, desde que se

utilizasse a desfuzificação pelo método do centro de gravidade - que forneceu os resultados mais «suaves» e coerentes.

Tirando partido de ambos estes aspectos, foi ainda possível verificar que o número de regras não necessita acompanhar o aumento do número de funções de pertença, dado que se torna desnecessário especificar exaustivamente todas as combinações.

Em suma, o desenho de um sistema de controlo por lógica difusa pode ser abordado de diversas formas, podendo resultados semelhantes ser obtidos alterando as funções de pertença, o método de desfuzificação ou a estrutura e número das regras - é, acima de tudo, uma questão de equilíbrio e bom senso.

Luís António e Rui Carmo ✍

Bibliografia e Referências

[1] Foley, van Dam, Feiner, Hughes; Computer Graphics - Principles and Practice; A.-Wesley, 1992

[2] David J. Kruglinski; Inside Visual C++; Microsoft Press, 1993

[3] David Brubaker; Fuzzy-logic basics: intuitive rules replace complex math; EDN, Jun. 1992

[4] D. Brubaker, C. Sheerer; Fuzzy-logic system solves control problems; EDN, Jun. 1992

[5] Greg Viot; Fuzzy Logic in C; Dr. Dobb's Journal, Feb. 1993

[6] J.A.R. Tucker, P.E. Fraley, L.P. Swanson; Fuzzy Logic in C: An Update; Dr. Dobb's Journal, Apr. 1994