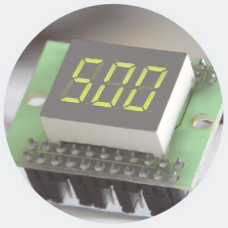
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES






OPTIMIZAÇÃO DO FLUXO DE UM SISTEMA DE TRÁFEGO DE VEÍCULOS

TIAGO ALEXANDRE PINHEIRO DE ALMEIDA
novembro de 2018

POLITÉCNICO
DO PORTO

OPTIMIZAÇÃO DO FLUXO DE UM SISTEMA DE TRÁFEGO DE VEÍCULOS

Tiago Alexandre Pinheiro De Almeida

Mestrado em Engenharia Electrotécnica e de Computadores
Área de Especialização de Automação e Sistemas
Departamento de Engenharia Electrotécnica
Instituto Superior de Engenharia do Porto
2018

Este relatório satisfaz, parcialmente, os requisitos que constam da Ficha da Unidade Curricular de Tese / Dissertação, do 2º ano, do Mestrado em Engenharia Electrotécnica e de Computadores

Candidato: Tiago Alexandre Pinheiro De Almeida, Nº1080542, 1080542@isep.ipp.pt

Orientação Científica: Isabel Maria de Sousa de Jesus, ISJ@isep.ipp.pt

Mestrado em Engenharia Electrotécnica e de Computadores

Área de Especialização de Automação e Sistemas

Departamento de Engenharia Electrotécnica

Instituto Superior de Engenharia do Porto

25 de novembro de 2018

“Being young lets us take risks that we wouldn’t do otherwise,

let’s us actually go and try to change the system”

Arjun Singh

“Whether you believe you can do a thing or not, you are right.”

Henry Ford

Agradecimentos

Trabalhar com quem admiramos transforma inevitavelmente a nossa obra, tornando o seu processo de concepção mais estimulante e enriquecedor.

Assim, nunca será demais agradecer a colaboração inestimável da Professora Isabel de Sousa de Jesus pela sua disponibilidade, rigor e amabilidade. Estou grato pelo esforço desenvolvido na leitura do relatório e suas eventuais correcções, permitindo enriquecer dessa forma o teor deste documento.

À minha família, pelo apoio e motivação na prossecução dos meus estudos ao longo deste percurso académico e na minha via profissional.

Por fim, a todos aqueles amigos que me ajudaram e motivaram durante este percurso e que sempre mostraram todo o gosto em contribuir para o enriquecer.

A todos eles, imensos agradecimentos. Sem eles, certamente o resultado final não seria digno de orgulho.

Resumo

O congestionamento de tráfego automóvel tem-se tornado uma das grandes preocupações das grandes cidades a nível internacional. Desde teorias de locomoção de veículos em estruturas a nível tridimensional, onde o asfalto teria variados níveis verticais, até tuneis subterrâneos em construção na cidade de Los Angeles, a criação de um sofisticado sistema de monitorização e controlo de tráfego resultaria numa solução efectiva para esta problemática. Um vasto número de assuntos em engenharia de transportes são comumente caracterizados como vagos, ambíguos e subjectivos. No controlo de sinais de tráfego luminosos, os variados fluxos de veículos forçam a comutação dos sinais das respectivas faixas, numa luta de prioridades. Num fluxo convencional de veículos, os sinais luminosos comutam baseados numa constante de tempo, contrariamente à necessidade actual. De forma a colmatar esse facto, um pequeno número de controladores implementam uma comutação baseada num horário diário, utilizando dados e padrões de tráfego pré-analisados. Outros sistemas avançados permitem, de igual forma, o controlo baseado em critérios simultâneos e em tempo real, como o tempo médio de fila de espera ou a percentagem de veículos em paragem. Com vista à melhor optimização, é previsível que um sistema com recurso ao controlo difuso possa oferecer competitividade em situações reais onde o uso de controladores com métodos tradicionais seja problemática.

Os controladores modernos programáveis que permitem um número de parâmetros ajustáveis podem servir adequadamente o propósito deste processo. De forma a obter bons resultados, é necessário um controlador desenvolvido à medida da situação e um ajuste directo no terreno. Com a lógica difusa, encontra-se provada a eficácia e o sucesso na resolução de problemas onde o processo possa ser controlado por mão humana, contudo, o modelo do sistema real é difícil ou impossível. É perante estas falhas que um controlador difuso se excede. Na verdade, um dos exemplos mais reconhecidos neste tipo de controlo é numa intersecção de duas faixas de um sentido cada. Mesmo neste caso mais simples, o controlo difuso obteve o mesmo desempenho que um controlador adaptativo tradicional.

Desta forma, este documento consiste no estudo da implementação do controlo da lógica difusa nos sistemas mais variados. No seguimento desta ideia, são utilizados esses recursos

estudados, com o objectivo de implementar um controlador, tendo por base um *software* da área de Engenharia, o MATLAB. Com este propósito, é efectuado um estudo teórico de introdução ao tema, assim como um estudo da *toolbox* a partir da qual será implementado o sistema de controlo. O sistema base é criado como um sistema discreto de veículos, entre várias intersecções, sendo controlado com recurso a um controlador difuso.

A segunda parte do projecto será a de integrar o modelo e controlo anteriormente referido num ambiente gráfico 3D, recorrendo á toolbox *Simulink* 3D. Aqui, poderá ser analisado a animação respectiva e a verificação dinâmica do comportamento do sistema, recorrendo a ambiente virtual.

Palavras-Chave

MATLAB, *SimEvents*, *SimMechanics*, *Simulink 3D Animation*, *Stateflow*, Lógica difusa, controlo

Abstract

Traffic congestion has become one of the biggest cities concern on an international level. From theories of vehicle locomotion at three dimensional levels, where the asphalt would have varied vertical levels, to underground tunnels in construction in the city of Los Angeles, the development of a sophisticated traffic monitoring and control system would result in an effective solution to this problem. A large number of subjects in transport engineering are common characterized as vague, ambiguous and subjective.

In the control of traffic light signals, the various vehicle flows force the switching of the respective band signals in a priority conflict. In a conventional flow of vehicles, the traffic light signals are switched based on a constant time frame, contrary to the current needs. In order to address this concern, a small number of real-life controllers implement a switch based on a hourly schedule, using pre analysed data and traffic patterns. Other advanced systems also allow a control based on real-time criteria based control such as the average queue time or the percentage of vehicles at standstill. In order to better optimize, it is foreseeable that a system using diffuse control can offer competitiveness in real time situations where the use of controllers with traditional methods is problematic.

Modern programmable controllers that offer a number of adjustable parameters can adequately deliver the purpose of this process. In order to obtain a good outcome, a controller developed to a specific situation and a direct adjustment in the field are required.

With *Fuzzy Logic*, efficiency and success in solving problems where the process can be controlled by human hand is proven. However, developing a real system model is difficult or nearly impossible. Facing these results, a fuzzy controller can surpass. In fact, one of the most recognized examples in this type of control is at an intersection of two lanes of one direction each. Even in this modest example, *Fuzzy Logic* achieved the same performance as a traditional adaptive controller.

In this way, this document consists in studying the implementation of a diffuse logic in the most varied systems. Following this idea, these resources are used with the aim of implementing a controller, based on a software from the Engineering area, MATLAB. With

this purpose, was carried out a theorical study of introduction to the subject, as well as a study of the toolbox from which the control system will be implemented. The base system is created as a discrete vehicle system, having several intersections, and controlled using *Fuzzy Logic*.

The second part of this project will be to integrate the previously mentioned model and control in a 3D graphic environment, using the *Simulink 3D Animation* toolbox. With regards to this, the user can analyse the respective animation and the dynamic verification of the system behaviour, using a virtual environment.

Keywords

MATLAB, *SimEvents*, *SimMechanics*, *Simulink 3D Animation*, *Stateflow*, *Fuzzy Logic* control

Índice

Índice de Figuras

Índice de Tabelas

Acrónimos

ALLONS-D	–	*Adaptive Limited Look-ahead Optimization of Network Signals – Decentralized*
ANFIS	–	*Adaptive Neuro-Fuzzy Inference System*
C1	–	Cruzamento 1
C2	–	Cruzamento 2
FIS	–	*Fuzzy Inference System*
FIFO	–	*First In First Out*
IoT	–	*Internet of Things*
LD	–	Lógica Difusa
LIFO	–	*Last In First Out*
SAC	–	Sistemas adaptativos complexos
SCATS	–	*Sydney Coordinated Adaptive Traffic System*
SCOOT	–	*Split Cycle Offset Optimisation Technique*
SIT	–	Sistemas Inteligentes de Transporte
VRML	–	*Virtual Reality Modeling Language*

1. INTRODUÇÃO

Este capítulo tem como objectivo fazer uma apresentação ao trabalho realizado, assim como o enquadramento na disciplina, identificando quais os objectivos a desenvolver e uma breve abordagem a cada um dos capítulos seguintes.

Um controlador é um dispositivo, ou um conjunto de dispositivos, programados para efectuar uma tarefa, baseada em informação recebida. Existem controladores programados para verificar e alterar a variável segundo padrões de lógica ou controlo sequencial, enquanto outros modificam o valor a implementar baseado em como o sistema se está a comportar, também intitulados por controladores de realimentação. Os controladores que recorrem a lógica difusa, sendo uma combinação dos anteriormente referidos, começaram por ser propostos e analisados na década de 70, estando presente num objecto de um dia a dia comum, como nos termóstatos, aquecedores, suspensões de veículos, etc.

Foi criado para colmatar uma falha existente no controlo lógico, onde muitas vezes este era ineficiente. Este introduz novas possibilidades na criação de variáveis que um controlo binário não consegue. De igual forma, é inexistente a necessidade de modelar um sistema de forma matemática com o objectivo de o controlar.

Estes podem ser desenvolvidos recorrendo a conceitos abstractos e sem fronteira definida, resultando, frequentemente, numa vantagem computacional e resultados mais eficientes.

Os programas de simulação de sistemas difusos são utilizados em diversos tipos de industrias desde o seu desenvolvimento inicial para o metro de Sendai, no Japão, onde viria a melhorar a economia e o trajecto deste.

Deste modo, a possibilidade de implementação de um controlador, a criação do sistema adequado e a visualização 2D ou 3D representativa do sistema foram pontos a ter em conta quando se optou escolheu o *software* mais adequado para o desenvolvimento de um controlo de semáforos, visando uma melhoria no tráfego de veículos.

1.1. CONTEXTUALIZAÇÃO

No âmbito da unidade curricular de Tese / Dissertação, este trabalho surgiu pela necessidade de estudar e entender como se processaria um controlo difuso a partir de uma das ferramentas mais utilizadas na área, o MATLAB. O trabalho está inserido numa área de relevo da engenharia, que permite que um sistema autónomo seja dominado a partir de um padrão imposto inicialmente. Isto é realizado utilizando controladores que modificam o comportamento do sistema à medida que este vai evoluindo. Com isto, existe uma adaptação dos padrões impostos no circuito em função do tempo. A título de exemplo, um ar-condicionado totalmente regulado consegue modificar a temperatura de uma divisão ao longo do tempo. Assim, foi proposto um trabalho que envolvesse o estudo de um sistema controlável, utilizando ferramentas e *software* específicos – MATLAB – da área analisada, o qual permitisse controlar o tráfego rodoviário, através do controlo de sinais luminosos.

1.2. OBJECTIVOS

É pretensão do corrente trabalho a implementação de um sistema difuso usando a *toolbox Fuzzy Logic* do MATLAB, o qual permita controlar os sinais luminosos, visando uma melhoria no tráfego rodoviário. Tendo em conta o trabalho adjacente à implementação de um sistema difuso, optou-se por estabelecer os seguintes objectivos:

- Pesquisa teórica sobre lógica difusa;
- Estudo da *toolbox Fuzzy Logic* do MATLAB, de forma a desenvolver o controlador.
- Estudo da *toolbox SimEvents*, com o objectivo de criar um sistema discreto.
- Implementação de um sistema difuso.
- Visualização da simulação desenvolvida recorrendo à toolbox *Simulink 3D Animation*

1.3. ORGANIZAÇÃO DO RELATÓRIO

Este relatório é composto por 6 capítulos: Introdução, Lógica Difusa, MATLAB, Desenvolvimento do projecto, Resultados e Conclusões.

Neste Capitulo que se encerra, foi apresentada uma breve contextualização sobre a lógica difusa, os objectivos do presente projecto e a organização do relatório.

No Capítulo 2, “Lógica Difusa”, é abordado, como o próprio nome indica, a lógica difusa. Nele pretende-se fazer uma pequena introdução à logica difusa salientando os objectivos para a qual é utilizada mas também apresentando objecções a este método. É, também, feita

referência ao método de inferência, no qual são enunciados os métodos realizados de forma a obter uma saída difusa. São também apresentadas algumas aplicações onde este tipo de controlo é utilizado, assim como a evolução no desenvolvimento deste tipo de controladores, nomeadamente no controlo em sistemas inteligentes de transporte.

O Capítulo seguinte, "*MATLAB*", pretende guiar o leitor através das várias potencialidades da *toolbox* que se encontra presente no MATLAB para o desenvolvimento de sistemas difusos. Nele são abordadas as interfaces gráficas disponíveis, as funções para a linha de comandos e os blocos existentes para utilizar no *Simulink*®. São apresentadas as diversas funções das toolboxs:

- *SimEvents*, com o propósito de criar o sistema discreto
- *Stateflow*, parte onde são transaccionados os estados discretos
- *Fuzzy Logic*, onde se encontra desenvolvido o controlador difuso
- *Simulink 3D Animation*, permitindo visualizar todo o sistema previamente criado

No Capítulo 4 faz-se referência ao desenrolar do trabalho e são descritos todos os aspectos a ter em conta aquando do desenvolvimento do sistema e sua visualização.

No Capítulo "Resultados" pretende-se comparar os dois controladores implementados, de forma a ser possível comprovar a eficiência de um controlador difuso quando comparado com um controlador pré-programado.

Por último, são retiradas as conclusões relativas ao desenvolvimento do projecto e são perspectivados futuros desenvolvimentos.

2. LÓGICA DIFUSA

Neste capítulo é feita uma introdução à teoria da lógica difusa. Isto terá como objectivo preparar o leitor para um dimensionamento acertado do controlador nos capítulos seguintes. É explicada a sua origem, assim como feito um paralelismo com outros tipos de lógica, incluindo a álgebra booleana, com os elementos do conjunto {0,1}. As operações de relacionamento com diferentes funções da lógica difusa também são referidas. Igualmente, são apresentados alguns objectivos que este método se propôs cumprir, assim como algumas objecções por críticos da área lógica. Por fim, são enunciadas as maiores aplicações onde esta lógica tem efeito no quotidiano.

2.1. INTRODUÇÃO À LÓGICA DIFUSA

O conceito de lógica difusa foi desenvolvido por Lofti Zadeh e apresentado como uma forma de processar informação. Neste contexto, a lógica difusa é uma metodologia de resolução de problemas de um sistema de controlo implementado em sistemas físicos, que podem variar desde simples microcontroladores a enormes estações de trabalho baseadas em aquisições de dados. O conceito pode ser implementado em *hardware*, *software*, ou uma combinação de ambos, fornecendo uma maneira simples para chegar a uma conclusão, baseada em dados vagos, ambíguos ou até à falta destes na entrada do sistema. Desta forma, o conceito é posto

em prática aproximando o controlo pretendido à forma de como uma pessoa o abordaria, porém de uma forma mais rápida.

As raízes da lógica difusa (LD) levam-nos de volta ao tempo em que existia evolução nas formas de vida inteligentes e onde todas estas podiam ser classificadas como sistemas difusos. Quando Aristóteles nos presenteou com um sistema de duas variáveis lógicas, foi Plato que estudou o conceito que viria a servir de base para o que é conhecido como lógica difusa, propondo um termo intermédio, ou terceiro valor, entre a *verdadeiro* e o *falso*, podendo este, ter termos verdadeiros e outros não tão verdadeiros. O assunto nunca foi verdadeiramente estudado, até por volta do início século XX, quando surgiu a ideia lógica de Lukasiewicz de estudar a *verdade*, o *possível* e o *falso* no conjunto {0,1,2}. Décadas depois, Lofti Zadeh, em 1965, introduziu o termo lógica difusa. Este observou que a lógica tradicional de dois valores lógicos [0 e 1], não conseguia manipular termos ou ideias como o *algo quente* ou *extremamente frio*. Não era possível caracterizar, nesta nomenclatura, cores como o cinzento, pois só poderia existir o preto e o branco. Foi assim introduzida a lógica difusa, com valores entre 0 e 1, como a conhecemos actualmente. A maneira tradicional de funcionamento de um processador é baseada na lógica binária de 0 ou 1, não existindo um valor intermédio. Existem consideráveis vantagens na representação digital, contudo, no quotidiano, as pessoas estão habituadas a tratarem de conceitos de uma forma analógica, a partir da linguagem verbal.

Enquanto os cientistas e investigadores dos Estados Unidos da América não acreditavam no sucesso da implementação do método, outros estudavam-no e faziam experiências. Em Londres, E. Mandani construiu o primeiro controlador, baseado na lógica difusa, para aplicar na manipulação da velocidade de um comboio a vapor, tendo resultado num grande sucesso. Na mesma altura, o Japão implementava um controlador similar, com regras tão regulares e suaves, que os passageiros não conseguiriam sentir se o comboio estaria parado ou em movimento, a não ser que observassem o exterior do comboio.

Assim, a lógica difusa pode ser vista como um maior conjunto de regras Booleanas, tendo vários valores lógicos. São acrescentados graus entre a verdade e a falsidade absolutas, de forma a abranger a verdade parcial. De outra forma, a lógica difusa envolve classificar termos e funções em conjuntos, que possam ser traduzidos em linguagem verbal. É uma forma de raciocínio onde não há lugar somente para o exacto e o impreciso. Por exemplo, o *muito quente* e o *devagar* são expressões que não dão lugar à ideia de um absoluto, pois não

as conseguimos quantificar. Enquanto 25ºC podem ser normais para uma pessoa europeia, para outra do árctico pode ser classificado como demasiado calor, ou calor insuportável. Assim, não é possível classificá-los em conjuntos rígidos, mas sim com limites definidos, que suporta uma ideia de imprecisão.

2.1.1. OPERAÇÕES DA LÓGICA DIFUSA

As operações da lógica difusa são as transformações a que um conjunto de sinais se sujeita, de forma a obter um único sinal de saída. Neste caso, serão explicadas as operações de união, intersecção e complemento, pois embora existam mais operações, estas são as principais e as que mais se reflectem no trabalho proposto nos próximos capítulos.

A união de duas funções lógicas A e B, com funções de pertença A e B, respectivamente, é definida como o máximo das duas funções de pertença individuais. Isto é intitulado de critério máximo na lógica difusa, e é o equivalente a uma operação OR na álgebra Booleana, como pode ser visto na Figura 1.

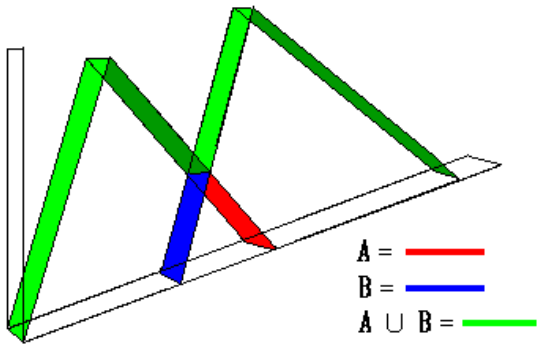


Figura 1 Lógica de união

A intersecção de duas funções lógicas A e B, com funções de pertença A e B, respectivamente, é definida como o mínimo das duas funções individuais. É o equivalente a uma operação AND na álgebra Booleana, como pode ser visto na Figura 2.

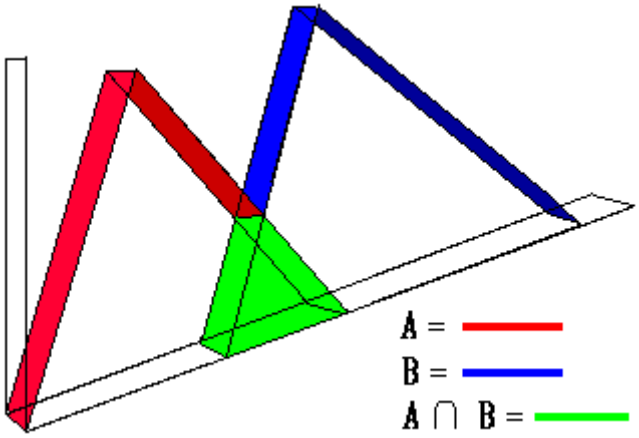


Figura 2 Lógica de intersecção

A função da lógica de complemento de um conjunto lógico A, com função de pertença A é definida como a negação dessa mesma função. Esta função é o equivalente a uma operação NOT na álgebra Booleana, como se pode ver na Figura 3.

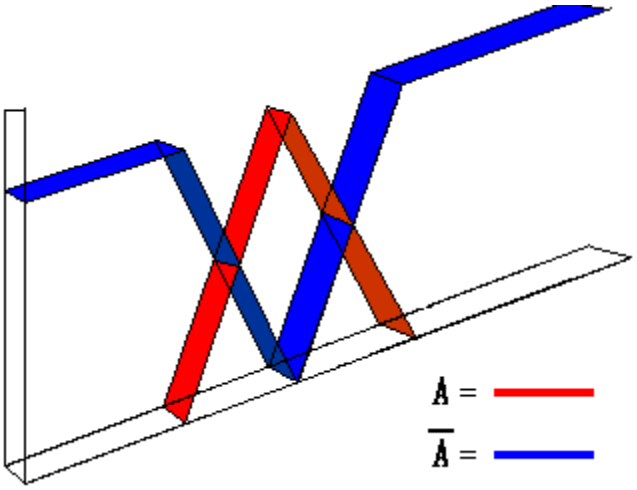


Figura 3 Lógica de complemento

2.1.2. OBJECTIVOS REAIS

O controlo do ambiente para grandes sistemas computacionais é, usualmente, mais complicado que o controlo de divisões frequentado por pessoas. Não só os sistemas fazem aquecer o ar da divisão, como são recomendados pelo fabricante para serem mantidos dentro de uma temperatura com um limite de 1ºC. O controlo de humidade é, também, um desafio, pois os elevados níveis causam corrosão e até bloqueiam sistemas mecânicos ou existe a maior possibilidade de descarga estática com níveis baixos. As especificações neste tipo de controlo, mantêm-se muitas vezes, por volta dos 50% de humidade, tendo 3% de margem de manobra.

Além disso, a concepção de um sistema de controlo ambiental de precisão, também enfrenta não linearidades, causada por um comportamento do sistema, tais como atrasos no fluxo do ar e os padrões de distribuição irregulares. Incertezas nos parâmetros do sistema também acontecem frequentemente, como o tamanho e forma da divisão, a localização do sistema de ventilação ou massa térmica do equipamento e das paredes.

A lógica difusa incorpora um método, baseado nas regras de dependências funcionais *se x e y, então z,* para resolver um problema de controlo, em vez de tentar modelar o sistema matematicamente. O modelo difuso é empírico, baseado na experiência do utilizador, em vez da sua compreensão técnica do sistema. Por exemplo, em vez de ligar com controlo de temperatura em termos de T = 25ºC, T < 30ºC, ou 20 < T < 25, existem termos como *se objectivo está quente* e *o objectivo está a ficar frio,* então *aumentar a temperatura*.

Estes termos são imprecisos mas ainda muito descritivos do que devia realmente acontecer. Deste modo, o objectivo desta lógica é o de simular o comportamento humano em diversas situações: ao tomar banho, se a água está fria, a torneira é regulada para a temperatura desejada. A lógica difusa é capaz de reproduzir este tipo de comportamento, porém a uma taxa de velocidade muito mais elevada.

2.1.3. INFERÊNCIA DIFUSA

Com as variáveis do sistema definidas, é necessário agora saber o que fazer de forma a termos acesso às variáveis de saída na forma difusa. Num sistema deste tipo, a inferência é decomposta em 5 partes: a *fuzzificação* das entradas, aplicações dos operadores, aplicação do método, agregação das saídas e a desfuzificação.

- *Fuzzificação* das entradas

Neste subprocesso, as funções de pertença definidas das variáveis de entradas, são correspondidas aos seus valores reais, de forma a determinar o grau de verdade de cada regra. Antes das regras serem avaliadas, é necessário saber a que valor corresponde cada valor de entrada. Num veículo, por exemplo, ao dizer que este é *lento, mediano* ou *rápido*, quando esta variável tem um universo de discurso de [0;20], é necessário distinguir se *rápido* se encontra perto do valor de 10 ou de 20. Consoante onde este se encontre, irá ter pesos diferentes, determinados pelo espaço da área referente à função de pertença associado, possivelmente até afectado pelos universos de discurso de outras funções de pertença. Todos

os passos seguintes são influenciados pelo resultado desta etapa, pelo que este é um erro comum a problemas em vários sistemas.

- Aplicação dos operadores

Uma vez que todas as entradas tenham sido *fuzzificadas*, é sabido o peso que os valores de entrada exercem no sistema. O objectivo dos operadores é unir os pesos das diferentes funções de pertença associadas a uma entrada. Neste caso, ao examinar a Figura 4, podemos inferir que utilizando o operador OR, este irá ter como saída o valor do peso mais alto de todas as variáveis. Como a comida é deliciosa, e tem valor de entrada de 8, numa gama de valores de [0;10], o peso que esta toma é de 0.7. Como o serviço é pobre, tem um valor de entrada de 3 numa gama de valores de [0;10], o peso correspondente é 0. Assim, com um operador lógico OR, o resultado do operador difuso irá ser 0.7. Também outro operador lógico disponível seria o AND. Neste caso, porque o serviço foi pobre, e como o peso da regra tem o valor 0, faria com que comprometesse o resultado, pois este seria multiplicado pelo peso de 0.7, ficando como nulo o resultado da multiplicação.

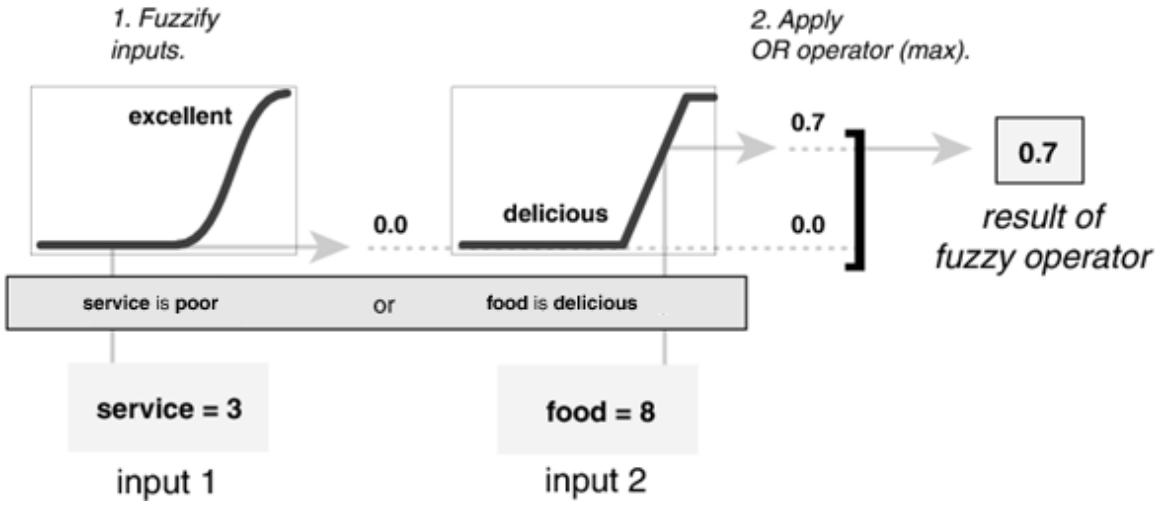


Figura 4 Exemplo de operação OR

- Aplicação do peso

Depois de cada peso ser calculado, este é associado a uma função difusa na saída do controlador. A entrada neste passo de inferência, é representada por um peso de valor variante entre [0;1] e a saída é representada por um conjunto difuso. Esta função irá ser afectada pelo peso anteriormente calculado, fazendo decrescer, em casos de pesos menores

que 1, a função de saída, como exemplifica a Figura 5, onde traduz o peso resultante da operação na função de saída.

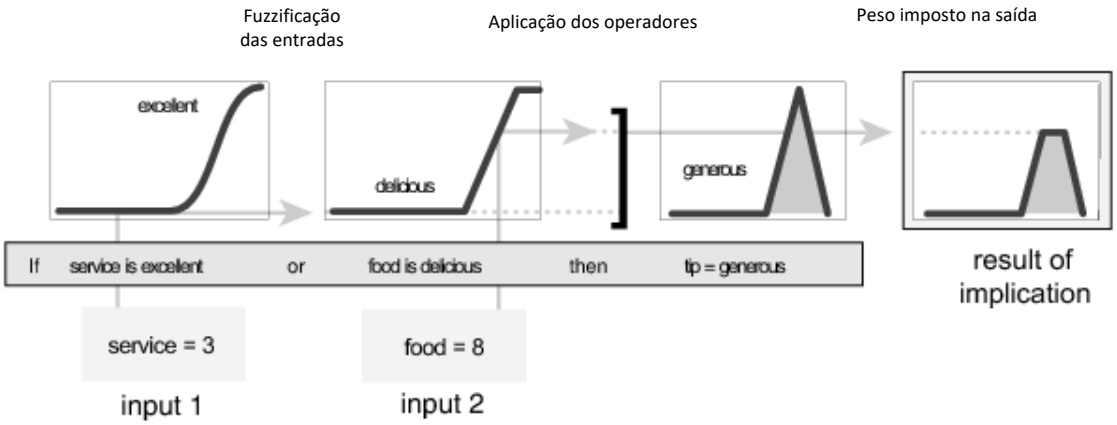


Figura 5 Aplicação do método

- Agregação das saídas

Nos sistemas baseados no conhecimento, frequentemente mais do que uma regra conduz à mesma conclusão. Embora isto não represente um problema para os sistemas clássicos, este caso tem de ser tratado separadamente nos sistemas difusos.

Se a conclusão de uma regra tem um peso de 0.7, mas também um peso de 0.3 com outra regra, então torna-se necessário que os diferentes pesos sejam reunidos num só. Isto é realizado pelo processo de agregação das saídas, o qual corresponde à unificação dos resultados individuais com o operador lógico OR.

- Desfuzificação

O resultado do processo de inferência, usualmente, tem de ser traduzido desde a lógica difusa para outras palavras, tais como uma indicação da acção a executar. Uma série difusa de saída da variável linguística é, por conseguinte, a união de todos os seus termos. Esta série difusa de saída, tem uma função de pertença que é calculada a partir das funções de pertença e do grau de pertença dos diferentes termos. A tarefa da desfuzificação, é transformar as funções de pertença da série difusa num resultado único. O método de desfuzificação é, portanto, a função das possíveis funções de pertença da série difusa de saída.

- Diagrama de inferência

Neste exemplo da Figura 6, está retratado, de forma simplista, todo o processo atrás referido. De notar a existência de duas variáveis de entrada em dois tempos distintos. Pode ver-se também, a aplicação do peso na altura em que a função de saída sofre uma compressão na sua amplitude. Por fim, estes dois sinais são sobrepostos, criando uma única função que é por fim *desfuzificada* numa única saída.

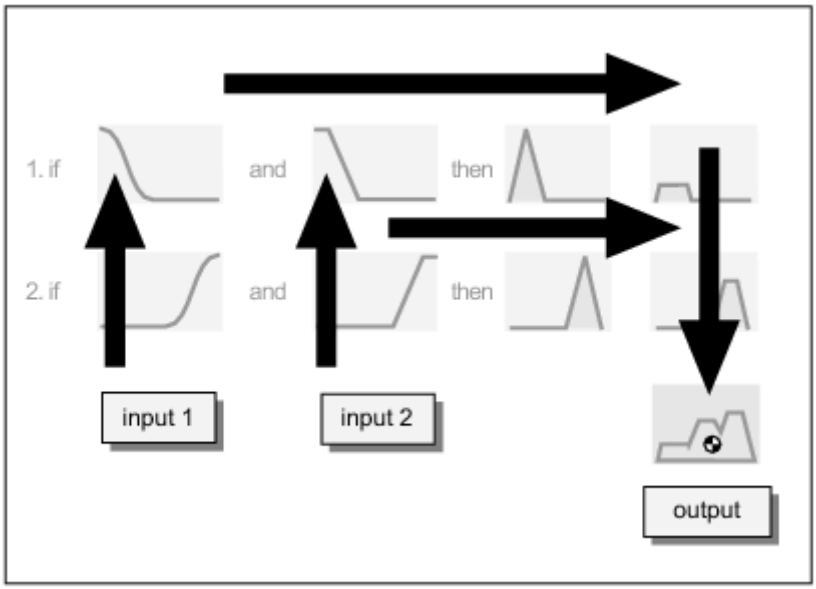


Figura 6 Diagrama de inferência

2.1.4. OBJECÇÕES

À medida que esta lógica foi descoberta, surgiram várias objecções e razões pelas quais esta não deveria ser usada, em mais do que uma ou duas situações esporádicas. Embora tenha havido quem se opusesse devido à complexidade do processo, derivado da atribuição de valores a termos linguísticos, também há quem defenda que podem ser usadas outro tipo de lógicas a aplicar nos sistemas e, portanto, a lógica difusa pode não ser necessária.

Também têm existido objecções relativamente aos valores tidos pela *verdade* e pela *falsidade* da lógica difusa. Haack [1] acredita que estes termos são discretos. No exemplo *o céu é azul*, qualquer aplicação da lógica difusa só derivaria da imprecisão na definição do termo *azul*, e não na natureza dos factos, onde é mais, ou menos azul. Também se opõe à utilidade desta lógica quando se justifica que, na manipulação de dados, a introdução deste método de cálculo não simplifica os resultados obtidos, tornando-os ainda, mais complexos. Por este motivo, argumentam alguns, esta lógica torna-se desnecessária.

Contudo, os engenheiros definem a lógica difusa como uma ferramenta útil, com diversas aplicações, especialmente na resolução de problemas que os computadores não são capazes de resolver. Um exemplo disto é a capacidade de reconhecimento de imagem, uma vez que enquanto o cérebro humano é capaz de distinguir um objecto de uma imagem, mesmo que desfocado, enquanto um computador, o que analisa, é apenas um conjunto de pixéis.

Esta lógica também garante os meios necessários para analisar a ambiguidade existente, associada com a vagueza, imprecisão ou falta de informação. Ao considerar uma *pessoa baixa*, o adjectivo pode ser caracterizado de forma diferente por várias entidades. Este adjectivo, também chamado de linguística, transmite a mesma mensagem para pessoas diferentes, contudo não garante uma definição única. Para solucionar isto, um computador compararia o valor termo *baixo* com o valor pré-definido da altura, imposto pelo utilizador. Esta variável linguística representa a indefinição existente no sistema.

Este método foi concebido como uma melhoria relativamente aos métodos habituais em casos de triagem ou análise de dados, porém tem provado ser uma grande valia para muitos sistemas de controlo, pois, novamente, imita a lógica de controlo humano.

2.2. APLICAÇÕES

Dadas as suas grandes potencialidades, a lógica difusa é utilizada para controlar máquinas e produtos de consumo. Nas aplicações adequadas, os sistemas difusos são de simples desenvolvimento e podem ser entendidos e implementados por pessoas que não são especialistas na área do controlo. Na maioria dos casos, uma pessoa com conhecimentos técnicos médios é capaz de projectar um sistema difuso. O sistema de controlo não será óptimo mas será aceitável. A lógica difusa também é utilizada em aplicações onde o poder computacional é limitado e apenas é necessário um controlo simples. Este tipo de controlo não é uma resposta para todos os problemas mas apenas para aqueles onde a simplicidade e a rapidez de implementação são importantes. A seguir são apresentadas algumas aplicações que usam a lógica difusa:

- Controlo do ambiente:
  - Equipamentos de ar condicionado;
  - Humidificadores.
- Electrodomésticos:
  - Máquinas de lavar e secar roupa;

  - Aspiradores;
  - Torradeiras;
  - Microondas;
  - Frigoríficos:
- Equipamento electrónico:
  - Televisões;
  - Fotocopiadoras;
  - Máquinas fotográficas e de vídeo com focagem automática, exposição automática e anti-vibração;
- Industria automóvel:
  - Controlo do clima de um automóvel;
  - Caixas de velocidades automáticas;
  - Direcção nas quatro rodas;
  - Sistemas de controlo dos assentos e espelhos.

Para melhor entender como funciona a lógica difusa, é explicado sucintamente o que acontece nas máquinas de lavar, aspiradores, câmaras de vídeo, equipamentos de ar condicionado e nas câmaras fotográficas:

- Numa máquina de lavar, é feita a distinção entre a sujidade da roupa e o tipo de tecido para determinar, automaticamente, as necessidades de água, detergente e potência;
- No caso do aspirador, estes são capazes de variar a pressão da sucção tendo em conta a existência de sujidade no chão e a quantidade de pó no mesmo;
- As câmaras de vídeo permitem gerar imagens "limpas" eliminando as distorções causadas pela não habilidade do utilizador ou pela movimentação dos objectos;
- Os equipamentos de ar condicionado variam a intensidade da operação dependendo do número de pessoas presentes no espaço, mantendo a temperatura homogénea constantemente;
- Por sua vez, nas câmaras fotográficas a autofocagem, *zoom* e a auto-exposição é feita através da lógica difusa.

Como se pode ver, trata-se de um método de controlo usado em diversas aplicações devido à sua simplicidade e ao seu baixo custo.

2.3. LÓGICA DIFUSA APLICADA A SISTEMAS REAIS

Uma das tarefas mais importantes e promissoras dos sistemas de transporte em cidades modernas é o poder de satisfazer as necessidades de ambos o país e os cidadãos num acordo mutuo de serviços de transporte eficientes. Outros estudos demonstram métodos de melhoria operacional eficiente de gestão de veículos em diferentes países [2]. Os resultados demonstram que a solução mais promissora decorre no desenvolvimento e implementação de sistemas inteligentes de transporte (SIT). Os SIT servem como uma integração de tecnologias de informação e comunicação, gerindo tráfego, infrastruturas, veículos e utilizadores, com o objectivo de aumentar a segurança e eficácia nas vias de serviço.

Hoje em dia, o âmbito da implementação de SIT varia desde a resolução de desafios na gestão de sinais luminosos e na segurança na estrada de forma a aumentar a eficácia existente do sistema de transporte. As capacidades dos SIT não são limitadas a aumentar a segurança e eficiência do processo de transportes mas também a reunir informação dos variados sistemas, focado na identificação de veículos, análise de situações de tráfego e detecção de violações do código da estrada. Alem disto, os SIT podem permitir a localização de veículos ou criminosos suspeitos. Quaisquer operações deste nível são conseguidas recorrendo a monitorização de vídeo remoto e ao processamento de largas quantidades de informação, assim como relatórios analíticos gerados de forma automática.

Durante esta pesquisa, a revisão dos projectos existentes de SIT provou que os últimos avanços nas áreas de tecnologia de informação, *data mining* ou teoria de controlo de sistemas, não se encontram implementados na gestão de tráfego rodoviário. A razão deve-se à falta de experiência e de e também à falta de métodos eficientes de múltiplos níveis de controlo baseado em parâmetros rodoviários. Os estudos existentes concentram-se na automação e desenvolvimento de sistemas de controlo e seus componentes, encontrando-se o foco no desenvolvimento e na melhoria do sistema de gestão do tráfego nas auto-estradas e na gestão adaptativa que solucionam problemas de fluxo de trânsito [3] ou a optimização na gestão de sinais luminosos [4]. Outros estudos analisam o desenvolvimento de métodos técnicos de forma a medir com precisão os parâmetros do processamento de tráfego rodoviário [5]. Contudo, estes estudos visam resolver problemas pontuais de controlo automático de tráfego, não tendo em consideração que o melhoramento numa área em particular pode levar à deterioração do tráfego noutro lugar próximo.

Contudo, com o avanço da tecnologia e das cidades emergentes recorrendo a *Internet of Things* (IoT), problemas como o tráfego rodoviário, vigilância de segurança podem agora ser estudados de uma forma global, auxiliando nos estudos prévios e na falta de informação. Com este objectivo, a maior parte das cidades urbanos fazem grandes investimentos nos tempos mais recentes, como mencionado anteriormente, em SIT. Também com o desenvolvimento de sistemas adaptativos, os actuais avanços de sensores de IoT começaram a tomar forma. Hoje em dia, com as vantagens da comunicação social, a distribuição e desenvolvimento na atribuição de aparelhos de IoT tem também aumentado [6].

Por sua vez, o congestionamento de tráfego em ruas e avenidas constituiu um problema critico e agravado pelo número crescente de veículos em urbanizações recentes e modernas. O ritmo lento de desenvolvimento de novas infrastruturas próprias para veículos e a oposição pública no aumento de avenidas pré existentes leva à reconsideração dos recursos actuais com o propósito, de forma efectiva, de gerir o fluxo de veículos. De igual forma, o tempo decorrente de um não melhoramento destas vias pode afectar directamente a produção, produtividade, desempenho e a maior utilização de combustível. O controlo de sinais luminosos é um dos objectos de estudo de sistemas inteligentes e avançados a serem investigados, pois tem impacto directo na eficácia dos sistemas de transporte urbanos.

No geral, os sinais luminosos recorrentes operam em um de três estados pré definidos: pré-programados, por accionamento ou em modo adaptativo.

Pré-programados consiste numa série de intervalos que são fixos na sua duração, repetindo-se continuamente. Em contraste aos anteriores, os sinais que operam por accionamento têm a capacidade de actuar perante a presença ou ausência de veículos ou pedestres na intersecção. Estes sinais luminosos capazes de actuar por accionamento requerem a presença de um veículo em uma ou mais vias, de forma a que certos padrões pré configurados possam ser cumpridos. Estão também equipados com detectores e a lógica de controlo necessária para serem capazes de cumprir o propósito, comutando conforme a necessidade do tráfego automóvel. O controlo por accionamento usa a informação actual e precisa, obtendo-a a partir de sensores presentes dentro da intersecção, alterando o comportamento de um ciclo de comutação dos sinais luminosos. Esta alteração é, por conseguinte, controlada por exigência do tráfego.

O modo adaptativo, também intitulado sistemas adaptativos complexos (SAC), compreende os últimos desenvolvimentos no controlo de tráfego de veículos. O SAC calcula e analisa, continuamente, os sinais recebidos baseados no volume de tráfego de uma respectiva área e, simultaneamente, implementa as resoluções. Estes sistemas são primariamente usados para, de forma eficaz, o modelo ser capaz de se adaptar às rápidas alterações das condições de tráfego dinâmico, estando associado ao sistema inteligente de controlo dinâmico em tempo real.

Estes sistemas utilizam a informação recolhida a partir de sensores de veículos e optimizam o tráfego em tempo real, daí intitulando-se modelos adaptativos em sistemas de tempo real.

Estes sistemas de tráfico adaptativo têm sido enunciados e utilizados na literatura desde o inicio dos anos 1970. Devido à sua aproximação com a realidade e a capacidade de prever fluxos de congestionamento, estes são normalmente complexos e incluem módulos de previsão e estimativas. No ínicio da década de 80, Nathan Gartner propõe uma politica de estratégia e optimização para sistemas adaptativos de controlo, desenvolvida e testada nos Estados Unidos da América. Esta estratégia inclui um controlo capaz de se aproximar de um desempenho teórico ideal, requerendo dados constantes de sensores das intersecções vizinhas [7].

De igual forma, o *Split Cycle Offset Optimisation Technique* (SCOOT) foi desenvolvido na década de 80 no Reino Unido pelo *Departamento e Desenvolvimento de Transportes*, continuando a ser aperfeiçoado nos dias correntes. O SCOOT, como um sistema adaptativo, permite o ajuste dos sinais luminosos às condições de tráfego humanas, utilizando sensores que medem o número de veículos. Estes sensores colocados em cruzamentos [8], formam regiões de tráfego, optimizando o fluxo de veículos num espaço inter-regiões. Testando a eficiência deste sistema, a Universidade de Leeds comprovou um aumento de eficiência, traduzindo-se num fluxo menos congestionado do número de veículos, em aproximadamente 15% [9].

Um novo sistema desenvolvido na Austrialia, o *Sydney Coordinated Adaptive Traffic System* (SCATS), sendo mais recente, não é assente em modelos de tráfego de veículos mas contém uma biblioteca de planos, das quais selecciona tendo, de igual modo, como dados de análise, o número de veículos nas intersecções.

Além destes apresentados, outros exemplos foram sendo desenvolvidos como o *Adaptive Limited Look-ahead Optimization of Network Signals – Decentralized* (ALLONS-D) ou um modelo estocástico, criado por Yu e Recker [10], baseado num sistema de decisão de Markov. modelando a tomada de decisão em situações onde os resultados são aleatórios e com a ajuda de um processo cognitivo de tomada de decisão. Contudo, o controlo de fluxo nas intersecções em conjunto com o número de estados possíveis cria um grande fluxo computacional.

Desta forma, este tipo de métodos ainda encontra uma grande barreira. Em certas circunstâncias, o excessivo poder computacional necessário torna que certos métodos baseados no sistema de decisão de Markov ou em programação dinâmica, requeiram informação actualizada e precisa durante os minutos seguintes à análise, de forma a determinarem os melhores planos de controlo. Com adição ao anteriormente referido, os dados de entrada nos sistemas são valores reais. De forma a aumentar a eficiência dos controladores de tráfego, o método da variável difusa, ao invés da real, diminui o impacto computacional.

Esta variável difusa permite a implementação de regras muito similares com o processo de pensamento humano. Desta forma, os controladores difusos são capazes de decidir perante dados de entrada mesmo que com informação incompleta. Durante este século, encontram-se a ser desenvolvidos ainda mais controladores para controlo de tráfego, permitindo a evolução dos algoritmos e o reduzido tempo de espera entre comutação dos sinais luminosos, e resultando numa maior dinâmica de condução. A motivação num controlo auxiliado por lógica difusa é a existência de imprevisibilidade no controlo dos sinais, sendo as decisões do controlador baseadas em informação imprecisa.

3. MATLAB

No âmbito deste projecto, o desenvolvimento do sistema foi abordado recorrendo às diferentes funcionalidades do MATLAB, um software interactivo, capaz de simular e simplificar diferentes tarefas diárias da área de engenharia.

Esta aplicação dispõe de diversas extensões de simulação, adaptando as necessidades do utilizador às variadas situações. Durante este projecto, com o objectivo de monitorizar um sistema de tráfego de veículos, foram utilizadas diversas extensões ou *toolbox*, correlacionando sistemas discretos com sistemas contínuos, sendo estas:

- *SimEvents*
- *Stateflow*
- *Fuzzy Logic*
- *Simulink 3D Animation*

De seguida, e como forma introdutória, são formuladas descrições e exemplos, seguindo de forma similar a aprendizagem do aluno à habituação do trabalho. As extensões serão apresentadas na mesma ordem previamente descritas, contendo uma breve descrição da mesma e, em seguida, o detalhe necessário para o desenvolvimento do projecto.

Estas extensões são organizadas por um sistema de bibliotecas específicas e partilhadas, as quais podem oferecer acesso a blocos para a modelação de sistemas discretos, controlo de estados, criação de lógicas, animação de modelos, entre outros.

Para aceder às bibliotecas das extensões mencionadas, existem três diferentes soluções:

- Abrir a biblioteca de blocos do *Simulink* e procurar *SimEvents*/*Simulink 3D Animation*/ *Fuzzy Logic Toolbox* ou *Stateflow*.
- Pelo menu inicial do MATLAB: *Start -> Simulink -> SimEvents*/*Simulink 3D Animation*/ *Fuzzy Logic Toolbox* ou *Stateflow*.
- Digitando, na linha de comandos do MATLAB: `SimEvents`, `vrlib`, `fuzzy` ou `sflib`, `respectivamente`.

3.1. *SIMEVENTS*

Esta extensão permite a criação e simulação de eventos discretos no tempo. É uma biblioteca especializada para a optimização de mudanças de estado no tempo. Desta forma, é criada a possibilidade de manipulação de entidades em determinados estados de tempo durante a simulação. Um exemplo pode ser demonstrado de seguida:

Se o objectivo de uma simulação consistir em, por exemplo, averiguar o tempo despendido por um cliente enquanto aguarda numa agência bancária, um evento discreto permite simular uma entidade que aguarda na fila, levanta o dinheiro ao balcão e se dirige à saída.

Contudo, se o interesse consistir em calcular a trajectória do cliente na agência bancária, ou qualquer desafio que envolva equações diferenciais, será mais adequado utilizar parâmetros contínuos.

De seguida são descritas as particularidades da expansão *SimEvents*, detalhando as diferenças entre entidades e eventos e exemplificando a modelação de um sistema.

3.1.1. ENTIDADES

Em grande parte dos sistemas discretos, a criação destes eventos, compreende a simulação com entidades, onde uma classe é usada desde o início da simulação até ao seu final, passando por vários pontos (ou etapas) intermédios. A essa entidade é permitida a atribuição

de informação, que pode ou não ser utilizada durante o resto da simulação. No quadro seguinte, para uma melhor compreensão do leitor, são criados vários exemplos de entidades, o seu contexto e os seus atributos.

Tabela 1 Exemplo de entidades e seus atributos

Entidade	Contexto	Atributo(s)
Utente	Utentes dentro de um elevador	Peso, quantidade de malas
Peça de montagem	Linha de montagem	Altura, peso, linha de destino
Avião	Aviões na pista de descolagem	Destino, número de passageiros, prioridade

Estes atributos são característicos da entidade que, após ter ultrapassado um ponto intermédio durante o seu percurso no sistema, podem ser alterados.

3.1.2. EVENTOS

Durante a simulação de um sistema discreto, um evento é uma ocorrência instantânea, habilitado para alterar um estado de um atributo, de uma entrada/saída do sistema ou da entidade. Podem ser parametrizados eventos quando uma entidade muda de estado, por exemplo, e com referência à Tabela 1, quando o utente sai do elevador ou a peça muda de linha de montagem. Neste exemplo, existe uma alteração na posição da entidade do sistema e uma alteração no atributo da entidade, respectivamente.

Estes eventos servem para criar uma sequência de alteração de estados das entidades durante a simulação. Sendo assim, um evento ocorre sempre que se verifique uma alteração durante toda a simulação.

Esta parametrização de eventos pode ser realizada recorrendo aos blocos da extensão do *SimEvents*. Assim, de seguida são especificados os blocos que podem levar a eventos discretos no âmbito desta ferramenta, com referência à Figura 7.

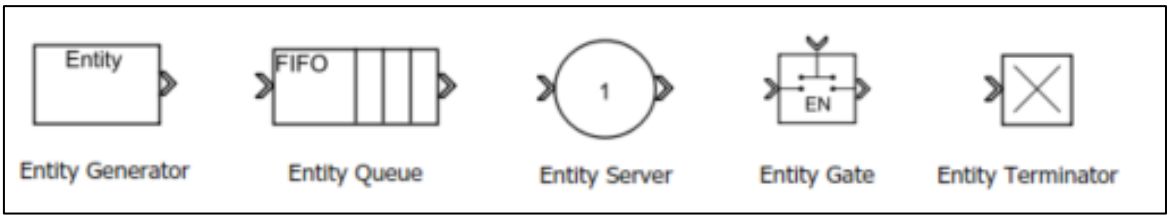


Figura 7 Blocos do *SimEvents* nos quais é permitida a criação de eventos

Em cada um destes blocos, um evento é passível de ser criado durante a criação ou saída de uma entidade. Recorrendo aos exemplos anteriormente mencionados (Tabela 1), pode ser possível alterar a prioridade (atributo) do avião (entidade) durante uma alteração de blocos.

3.1.3. MODELAÇÃO

Após aceder à biblioteca do *SimEvents*, como indicado no Capítulo 3, é apresentado o conjunto de blocos constituintes desta extensão. Neste projecto irão ser necessárias os blocos mencionados na Figura 7, pelo que se optou por fazer referência a estes nas descrições seguintes.

- Criação de entidades ou *Entity Generator*

Com o propósito de gerar entidades no decorrer de uma simulação, este bloco permite a sua criação em resposta a um evento ou com base periódica, como é representado pela Figura 8.

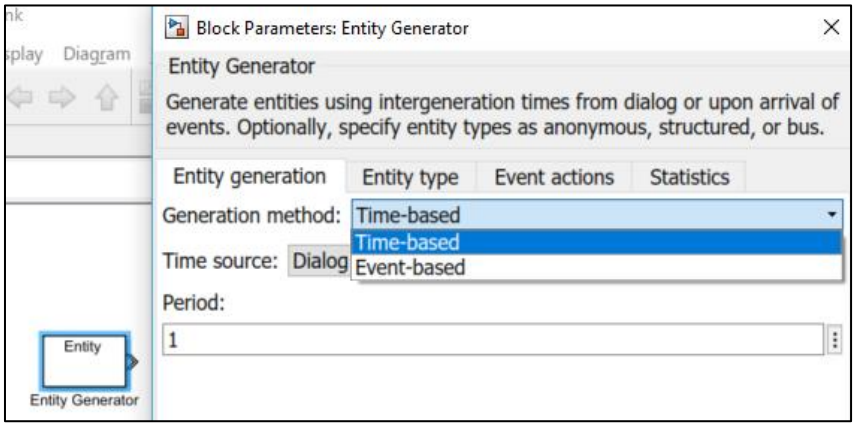


Figura 8 Parâmetros do bloco *Entity Generator*

Um método periódico temporal permite a configuração da frequência de eventos. Esta pode ser de tempo fixo (e.g. 5 em 5 segundos) ou configurável recorrendo, por exemplo, a uma função de MATLAB.

De igual forma, o tipo de entidade também é passível de ser configurado, como indicado na Figura 9. O tipo *Structured* permite a inicialização de atributos que podem ser modificados, como ilustrado no campo *Attribute Name*, onde é permitido adicionar vários atributos com um valor inicial configurado.

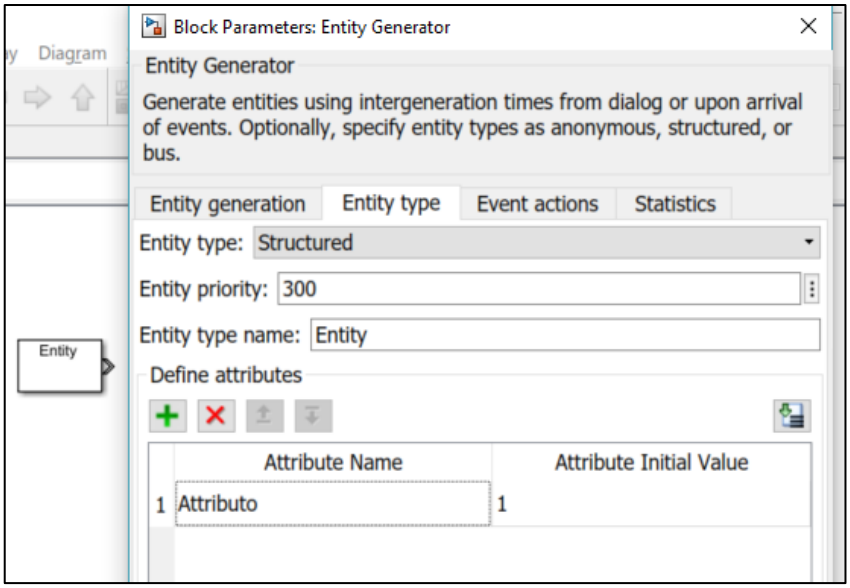


Figura 9 Parâmetros do bloco *Entity – Entity Type*

Estes parâmetros atribuídos inicialmente na criação da entidade, seguem acoplados a esta durante toda a simulação. De tal forma, é possível serem alterados em caso de necessidade.

Também este bloco dispõe de um separador comum a todos os outros referidos na Figura 7, o *Event actions*. Esta opção permite desenvolver uma acção quando a entidade é gerada ou quando a entidade sai do bloco. Isto resulta, por exemplo, na possibilidade de invocar uma acção do MATLAB quando a entidade é criada ou atribuir um valor diferente a um atributo quando a entidade parte do bloco.

- Listagem de entidades ou *Entity Queue*

Este bloco é utilizado sempre que existir a necessidade de guardar ou agrupar as entidades, por ordem de chegada ou atribuindo uma prioridade. Após as entidades serem geradas, é, por norma, sugestivo, o agrupamento destas de forma a serem distribuídas para outros blocos. Este bloco pode actuar como, por exemplo, na espera de utentes a serem atendidos ou um conjunto de aviões a aguardar a descolagem.

Como pode ser analisado na Figura 10, este bloco tem a capacidade de configuração máxima de entidades em espera ou de diferenciar o tipo de fila (*First In First Out* (FIFO) ou *Last in First Out* (LIFO)).

De notar que no caso de exceder a capacidade, o bloco prévio retém a entidade até que exista uma vaga na *Entity Queue*.

De igual forma que existe a necessidade de espaço para a entidade no bloco a receber, o bloco seguinte também terá de dispor de capacidade para uma entidade ser removida deste.

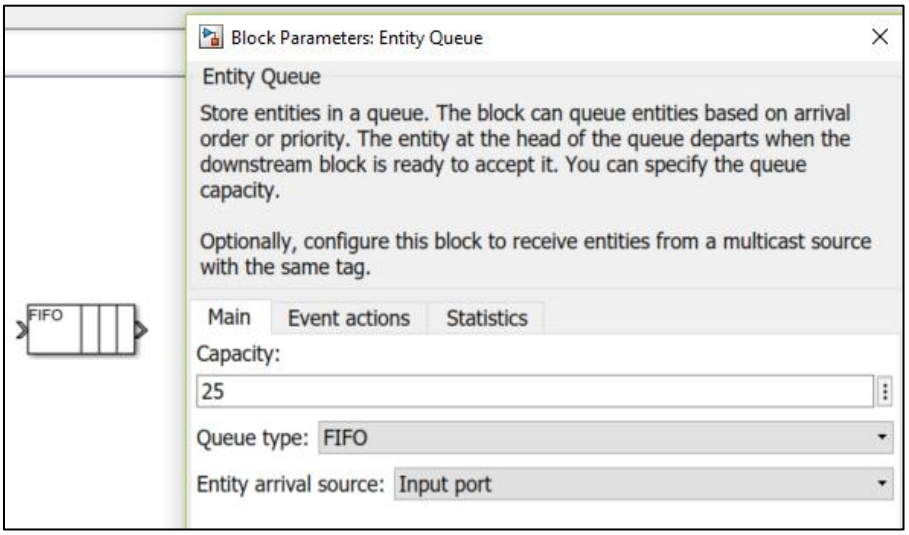


Figura 10 Parâmetros do bloco *Entity Queue*

- *Entity Server*

O propósito deste bloco é o de realizar trabalho durante um período de tempo. Após isso, retira a entidade, passando-a para o bloco seguinte. Pode-se considerar um exemplo como um utente, numa agência bancária, a ser atendido.

De forma similar ao *Entity Queue*, também é possível configurar a capacidade, sendo o tempo de serviço tratado individualmente (ao surgirem duas entidades no mesmo espaço de tempo, o tempo de serviço corre paralelamente para cada uma destas).

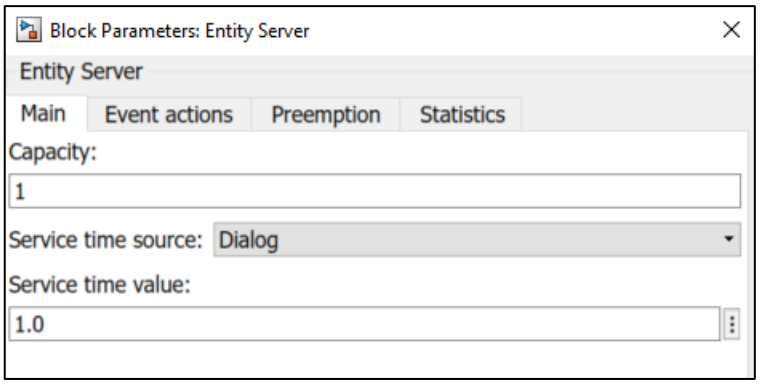


Figura 11 Parâmetros do bloco *Entity Server*

De notar que a origem do tempo de serviço pode ser escolhida de entre uma função MATLAB ou um atributo pré-definido de uma entidade. Isto permite a personalização do tempo, relativamente à entidade criada. Um exemplo seria associar um atributo à entidade que indicasse a tarefa desta a efectuar no banco. Seria assim possível associar tempos de serviço inferiores para, por exemplo, o levantamento de dinheiro e tempos superiores para, por exemplo, abertura de contas.

- Cancela de Entidades ou *Entity Gate*

Este bloco é utilizado como um interruptor, permitindo a passagem ou a retenção de entidades no bloco anterior. De notar que dispõe de uma porta adicional, que deve ser utilizada como porta de controlo. De acordo com o parâmetro configurado no bloco e na informação recebida na porta de controlo, o bloco executa diferentes tarefas. Pode ser utilizado um de três modos de operação:

  - *Enable gate* – permite a passagem de entidades quando este bloco receber, a partir da porta de controlo, uma mensagem com um valor positivo. (Ex: é permitido executar a tarefa quando o utilizador carregar no botão)

  - *Release gate* – permite a passagem de uma entidade por mensagem recebida, a partir da porta de controlo. (Ex: por cada vez que o botão é pressionado, pode ser executada uma tarefa)

  - *Selection gate* – permite a passagem de uma entidade quando o seu atributo corresponder ao valor recebido a partir da porta de controlo (Ex: quando a impressão digital do utilizador coincidir com o esperado pelo sistema, é autorizada a sua saída).

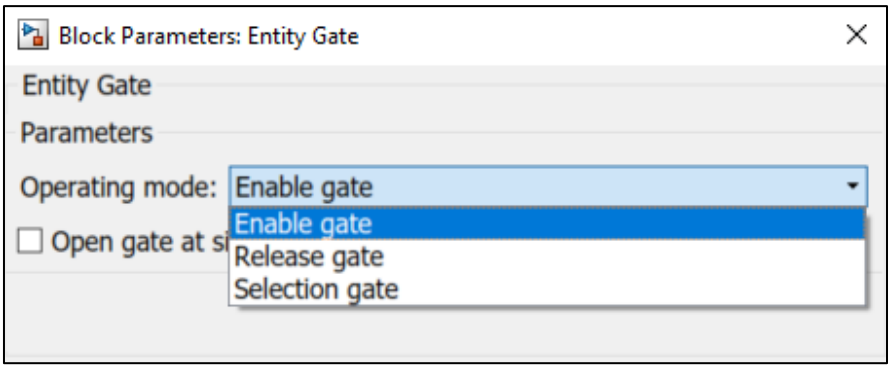


Figura 12 Parâmetros do bloco *Entity Gate*

Um exemplo da utilização deste bloco num contexto real, pode ser tido como a permissão para o cliente sair da agência bancária, somente após ter carregado no botão para abertura da porta.

- Término de Entidades ou *Entity Terminator*

No final do fluxo, quando não existe trabalho adicional a realizar pela entidade, esta deve ser descartada do sistema. Todas as entidades que são colocadas neste bloco são eliminadas do sistema e não podem ser recuperadas.

De notar que em todos os blocos anteriormente mencionados, no momento em que uma entidade seja transferida para um bloco, é possível tomar acções de desenvolvimento, como exemplifica a Figura 13. Neste exemplo, é tomada uma acção quando a entidade entra no bloco. Esta acção faz com que seja atribuída ao atributo *Tempo*, previamente configurado como um atributo da entidade, o resultado da função *levantardinheiro()*;

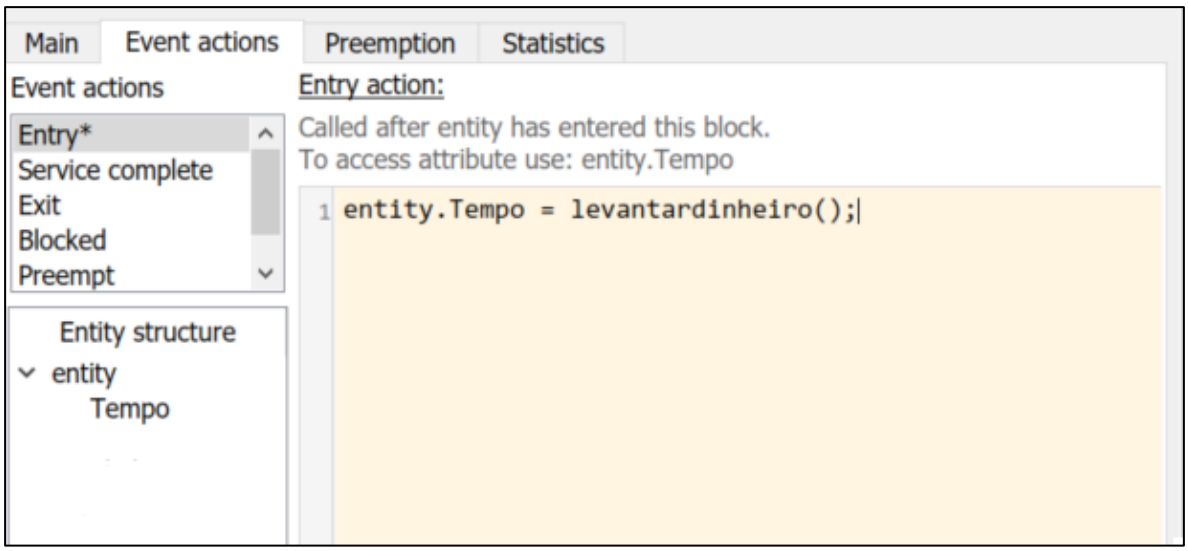


Figura 13 Exemplo de uma acção durante um evento

As funções invocadas a partir do menu de acções de cada bloco, como por exemplo a função *levantardinheiro()*, necessitam de ser previamente desenvolvidas e inicializadas no espaço do *Simulink* de forma a poderem ser utilizadas.

3.2. SIMULINK 3D ANIMATION

Com o objectivo de visualizar o sistema, suas posições e possíveis erros de implementação, optou-se por recorrer à extensão do MATLAB capaz de transformar sinais em modelos 3D. Este sistema 3D é assim constituído por diversos objectos independentes, interligados por um nó único central e que, quando em conjunto, formam uma única estrutura de visualização. O modelo é desenvolvido seguindo a linguagem *Virtual Reality Modeling Language* (VRML), onde cada nó representa uma funcionalidade no ambiente gráfico.

Nesta linguagem, consideram-se seis diferentes categorias ao classificar as suas funcionalidades:

- Objecto – Representa qualquer estrutura física, desde cones, quadrados e cilindros.
- *Transform* – Define posições, translações, escalas e cria sub-nós.
- Material – Permite alterar o tipo de material do objecto em estudo, desde a sua cor à sua textura.
- *Fog* – Altera as propriedades ópticas do sistema.
- Sensor de proximidade – Define uma interacção com o utilizador, a partir dos periféricos do sistema ou o sistema recebe um sinal quando envolve determinados parâmetros.
- Luz direccional – Representa um foco de luz no sistema. É utilizado para iluminar uma certa área de interesse.

Como estes nós servem para definir hierarquias, qualquer valor alterado num deles, irá influenciar os nós dependentes. Isto permite criar posições, objectos ou luzes relativas ao nó pai, em projectos de alta complexidade.

3.2.1. EXEMPLO DE MODELO

Neste trabalho, a interacção com o mundo virtual 3D é realizada a partir do *Simulink*. Os restantes blocos das diferentes extensões utilizadas neste projecto (*SimEvents*, *Stateflow* e *Fuzzy Logic*) constituem um agregado de dados guardados numa base de dados local. Baseado nesta informação, os dados são carregados para o mundo virtual com a adição de um bloco específico do *Simulink 3D Animation*.

Como iremos comprovar no decorrer deste relatório, esta extensão é assente num editor de texto e num compilador, que conseguem transformar as acções do ambiente gráfico ou de texto na linguagem de MATLAB para modo texto VRML. Sendo assim, é fundamental a adição do editor para que sejam introduzidas as bibliotecas necessárias ao funcionamento do programa. Com este propósito, é necessário digitar na linha de comandos do MATLAB `vrinstall('-install','editor')`.

Sendo agora possível iniciar a extensão, é possível escolher a forma de desenvolvimento do ambiente gráfico pretendida, de entre duas opções possíveis:

- Em código VRML, baseando o desenvolvimento em escrita em modo de texto, sendo a extensão do ficheiro associada por *.wrl*.

- Em *3D World Editor*, que representa uma ferramenta de criação de imagens VRML e X3D nativa, fornecendo uma interface para a sintaxe dessas linguagens, sendo este o método utilizado durante o desenvolvimento deste projecto.

Para inicializar a extensão é necessário seleccionar, no menu inicial do MATLAB -> APPS , expandir a selecção e, dentro de *Simulation Graphics and Reporting*, seleccionar *3D World Library*. O editor é aberto e o utilizador selecciona *New File*.

As simulações nesta extensão são feitas recorrendo ao conceito de hierarquia em árvore ou em nós. Ao criar um ficheiro, dispõe-se de um nó intitulado *Root*, como ilustra a figura abaixo.

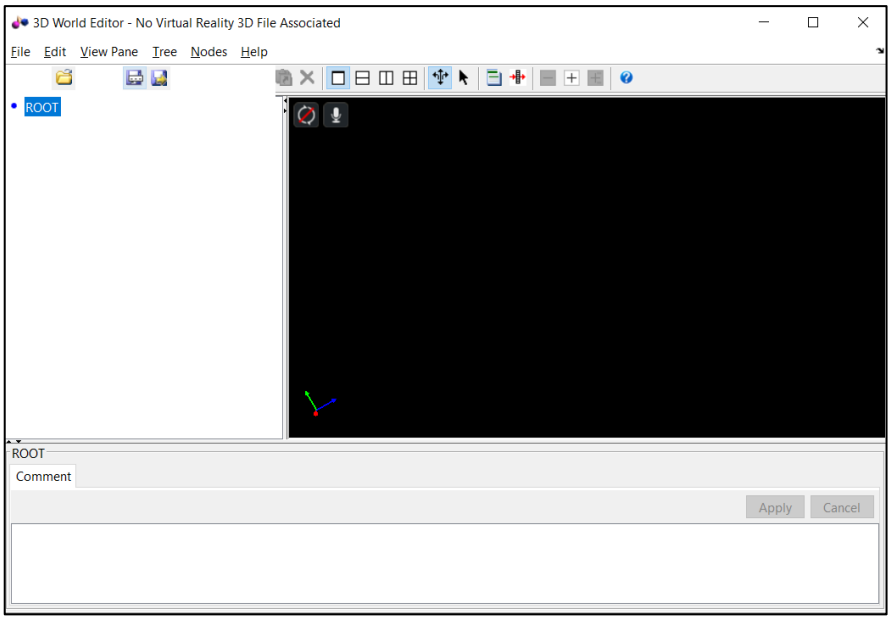


Figura 14 3D World Editor – ecrã inicial

Para iniciar a construção do modelo, é necessário criar diversos nós, representando os objectos destino. Para isto, o utilizador selecciona a opção *Nodes -> Add -> Group -> Transform*. Este objecto dispõe agora de coordenadas de centro, rotação, translação, entre outros.

Um dos parâmetros mais utilizado durante a execução deste projecto é o parâmetro translação. Este é composto por valores nos 3 eixos [x,y,z] e disponíveis para o ambiente *Simulink*.

Após isto, depende do utilizador a escolha do elemento 3D quer representar no espaço. Cada objecto tem de conter uma forma ou *shape* associada. Para isso, com o nó *children* seleccionado, deverá adicionar o elemento que possibilite a adição de uma forma, de modo similar ao passo anterior. Contudo, desta vez deve seleccionar os seguintes exemplos, como ilustrado na Figura 15.

- *Nodes -> Add -> Geometry*, se o utilizar for criar um objecto (e alterar as dimensões deste).

- *Nodes -> Add -> Appearance*, se o utilizador pretender, por exemplo, alterar o material ou a textura do nó.

- *Nodes -> Add -> Light*, se o objectivo for a criação de uma luz direccional, entre outros.

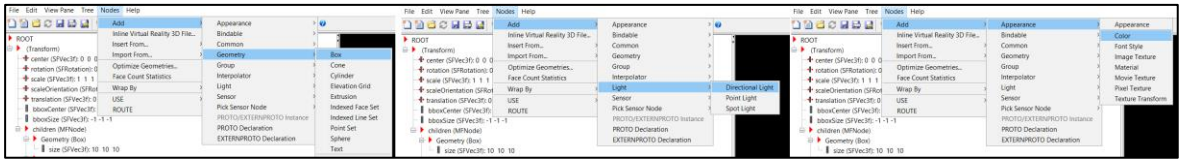


Figura 15 *Simulink 3D Animation* – Modelação de um objecto

Estando agora seleccionadas todas as opções desejadas, é possível alterar os valores destas. A geometria, por exemplo, como visível na Figura 16, tem três valores, correspondente às grandezas físicas constituintes do volume do objecto (CxLxA).

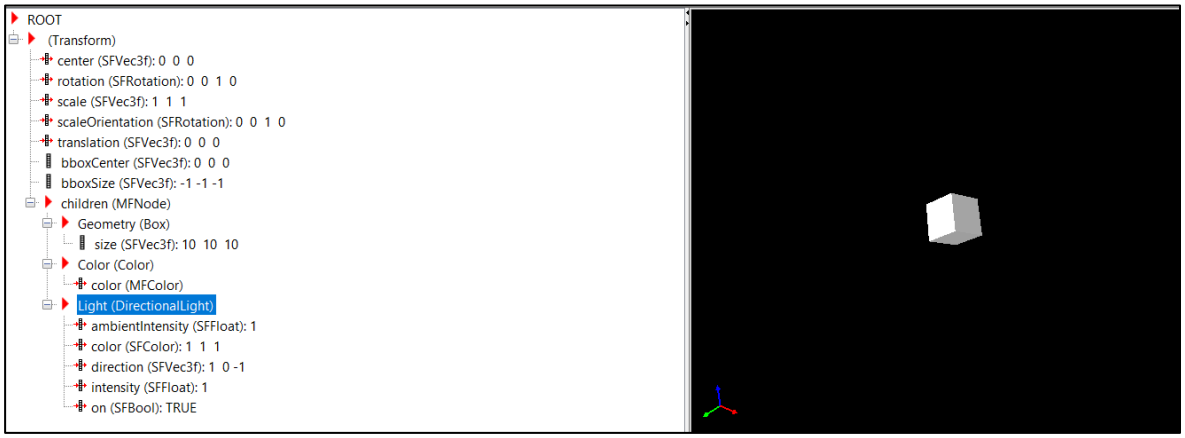


Figura 16 *Simulink 3D Animation* - Criação de um objecto

O objecto encontra-se assim criado e qualquer alteração necessária ao modelo pode ser realizada a partir do menu lateral esquerdo. Estes exemplos espelham o que é possível construir num modelo 3D recorrendo a esta ferramenta.

Estando os objectos criados e alterados os parâmetros constituintes destes, a alteração será reflectida no espaço virtual e os sinais reportados para o ambiente *Simulink*. Estes podem ser ligados a qualquer bloco do *Simulink* que contenha a mesma dimensão (seja unitária ou vectorial).

Desta forma, todos os sinais passíveis de serem alteráveis durante a simulação são convertidos recorrendo a um bloco - *VR Sink* - originário da biblioteca do *Simulink 3D Animation* como demonstrado a partir da figura abaixo.

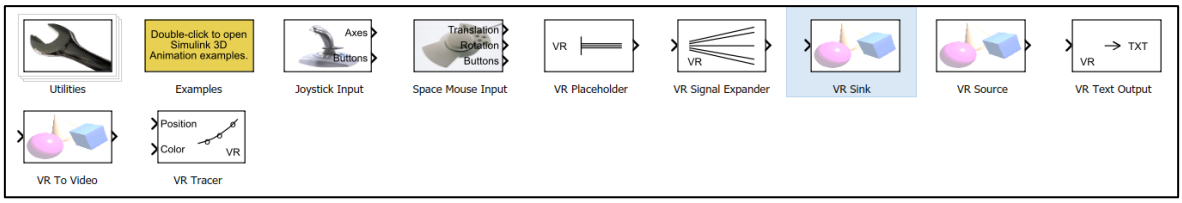


Figura 17 *Simulink 3D Animation* – Biblioteca

Uma vez carregado o ficheiro .wrl no bloco *VR Sink*, como se pode comprovar na Figura 18 (Cubo.wrl), é necessário expor os sinais que se pretende que sejam alterados durante a simulação (translação de um objecto, rotação de uma bola, *outputs* de um sensor). Neste exemplo, comprova-se os objectos hierarquicamente criados (*Geometry*, *Appearance* e *Light*) e sua estrutura. Ao lado de cada parâmetro encontra-se uma caixa de selecção. Esta, quando seleccionada, é actualizado o *Simulink* com as novas entradas no bloco, restando ao utilizador ligar o sinal à entrada do bloco *VR Sink.*

Neste exemplo, optou-se por extrair os sinais de translação e tamanho do cubo. De notar que o bloco foi actualizado, permitindo ao utilizador ligar os sinais de entrada.

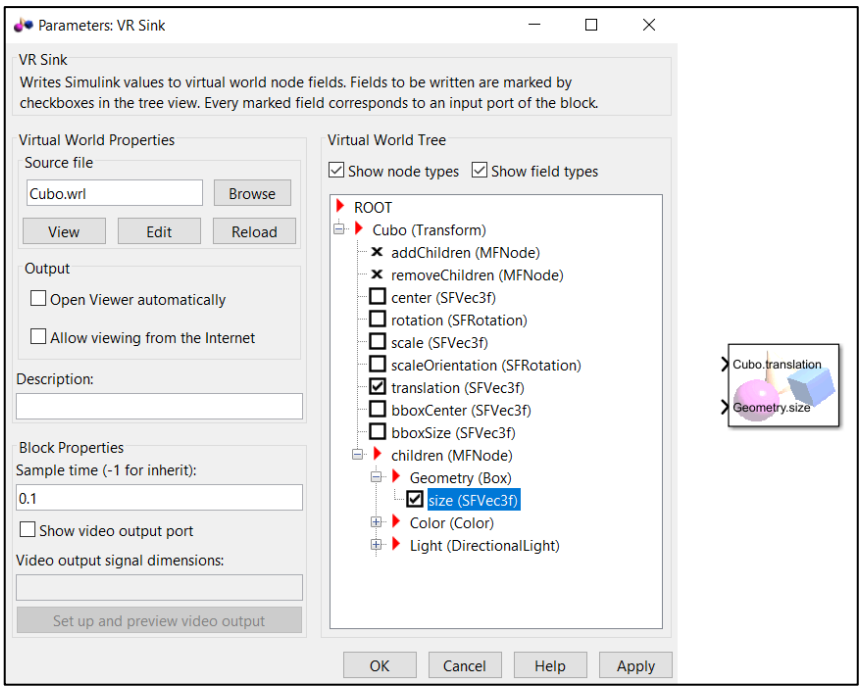


Figura 18 *Simulink 3D Animation* – Parametros VR Sink

Durante a simulação, existe agora um paralelismo entre o valor do sinal de entrada e a translação, num eixo individual ou em conjunto, do Cubo, no sistema 3D.

Um outro elemento essencial de referir no desenvolvimento deste projecto é um bloco *VR Signal Expander*.

Este bloco permite expandir um vector com múltiplos elementos para um único sinal de VRML. Prosseguindo o exemplo anterior da Figura 18, o utilizador só dispõe de uma entrada para a translação do objecto, que compreende um vector de 3 elementos. Este *Signal Expander* consiste em mapear valores individuais a serem utilizados numa única porta de saída. Caso não fosse utilizado, o utilizador não conseguiria mapear valores individuais variáveis, com origens em diferentes fontes, para um único canal.

Na figura seguinte, apresenta-se um exemplo da configuração deste bloco e suas descrições.

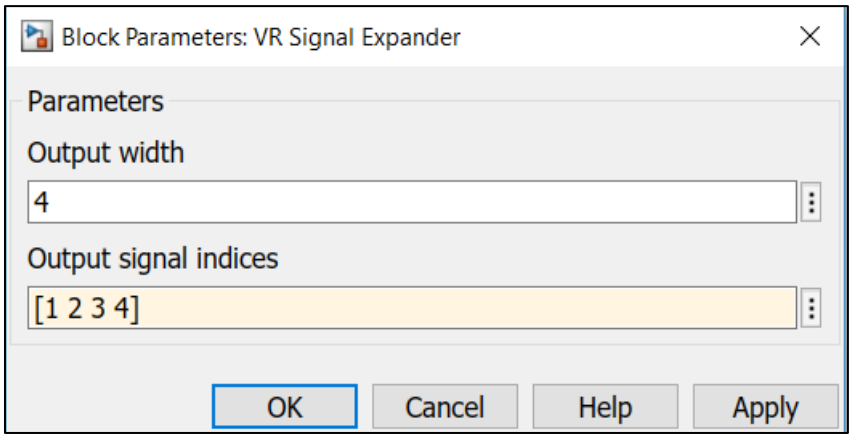


Figura 19 ***Simulink 3D Animation – VR Signal Expander***

Este bloco dispõe de um sinal de entrada e vários possíveis de saída. A concatenação é realizada dependendo do parâmetro escolhido no *tamanho do vector de saída* e *índices de saída*.

- O *output width* deve ser configurado tendo por base o bloco de destino. No exemplo mencionado anteriormente, visto que a entrada do sinal para a translação do cubo requer um vector com 3 elementos, deve ser configurado com este valor.

- O *output signal índices* permite alterar o mapeamento entre os sinais de entrada e de saída, no caso da ordem do vector de entrada ser incorrecta para o bloco destino.

3.3. *FUZZY LOGIC* TOOLBOX

No seguimento do enunciado no Capítulo 3, a expansão *Fuzzy Logic* providencia um interface gráfico, funções do MATLAB e blocos específicos para criar um sistema difuso. Este é útil para quando o utilizador quer desenvolver modelos de sistemas não lineares, onde os limites numéricos não são precisos ou são demasiado rígidos. Durante a criação de um sistema difuso, recorrendo a esta expansão, o utilizador deve seguir regras delineadas de forma a garantir a consistência do processo e a não existência de erros por falta de elementos anteriores ao desenvolvimento. Desse modo, a criação do exemplo durante este capítulo rege-se pela seguinte ordem:

- Definição de entradas e saídas do sistema
- Criação de funções de associação
- Definição e criação de regras
- Simulação do resultado no sistema difuso

Todos estes pontos podem ser desenvolvidos recorrendo a uma linha de comandos. Contudo, é facilitada a utilização recorrendo a um interface gráfico. Ao iniciar a expansão de sistemas difusos, como descrito no inicio do Capítulo 3, é apresentada ao utilizador uma janela, tal como é representada na Figura 20.

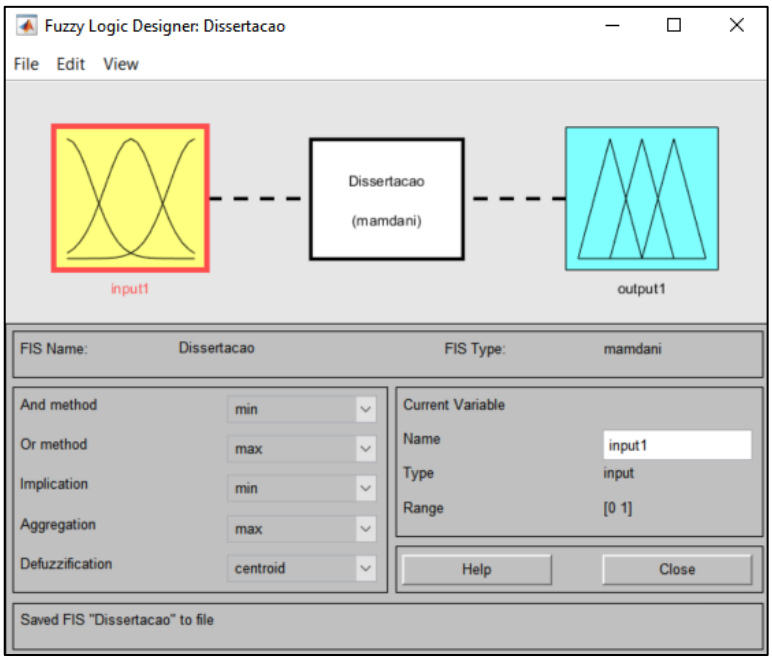


Figura 20 *Fuzzy Logic* Toolbox – Inicialização

O editor *Fuzzy Inference System* (FIS) suporta e ajuda na resolução dos principais problemas do sistema, tais como o número de sinais de entrada, configuração de regras ou vista gráfica. De notar que esta expansão não limita o número de entradas e/ou de saídas escolhidas pelo utilizador.

Seguindo o descrito anteriormente, a primeira etapa será a de definir as entradas e as saídas do sistema. Para isso, o utilizador terá de navegar até à barra de ferramentas e escolher *Edit – Add Variable – Input*, como representado na Figura 21.

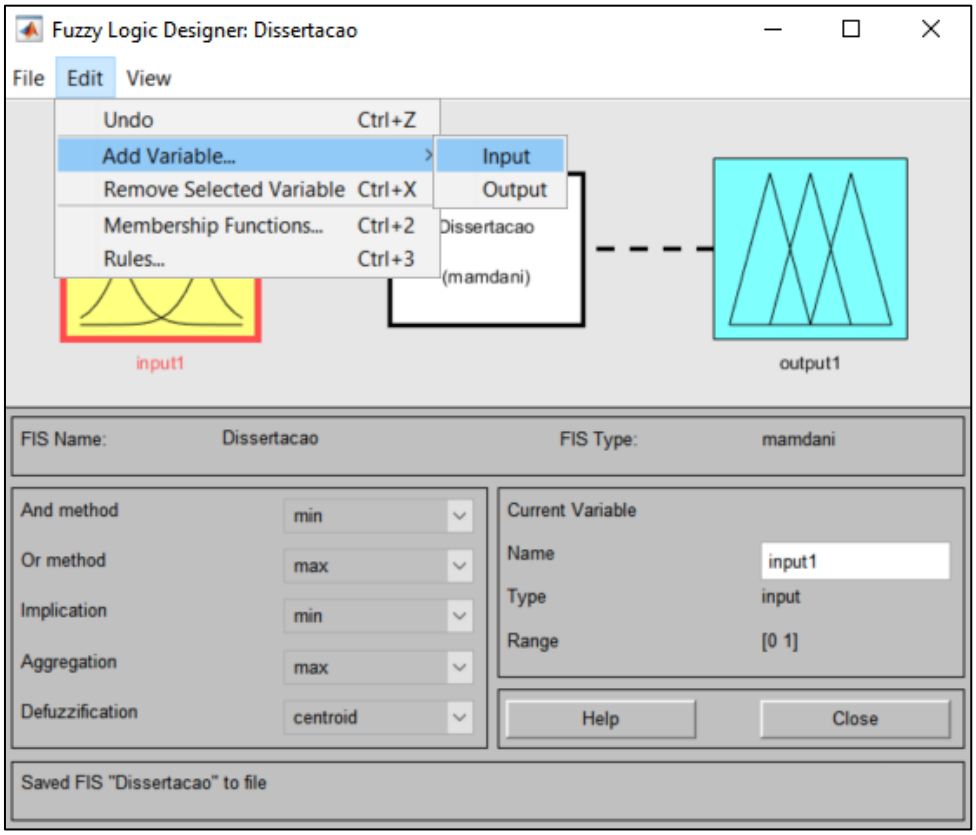


Figura 21 *Fuzzy Logic* Toolbox – Definição de entradas / saídas

O diagrama do sistema expõe o nome do sistema e o tipo de inferência usado. Neste exemplo, o nome do sistema está representado como *Dissertacao* e utilizada a inferência tipo *mandani*.

Na metade inferior da janela encontram-se expostos diversos parâmetros configuráveis pelo utilizador, nomeadamente nome da entrada, desfuzificação, agregação, entre outros. Estes podem ser configuráveis e até associados a uma regra pré definida, como se demonstra na Figura 22.

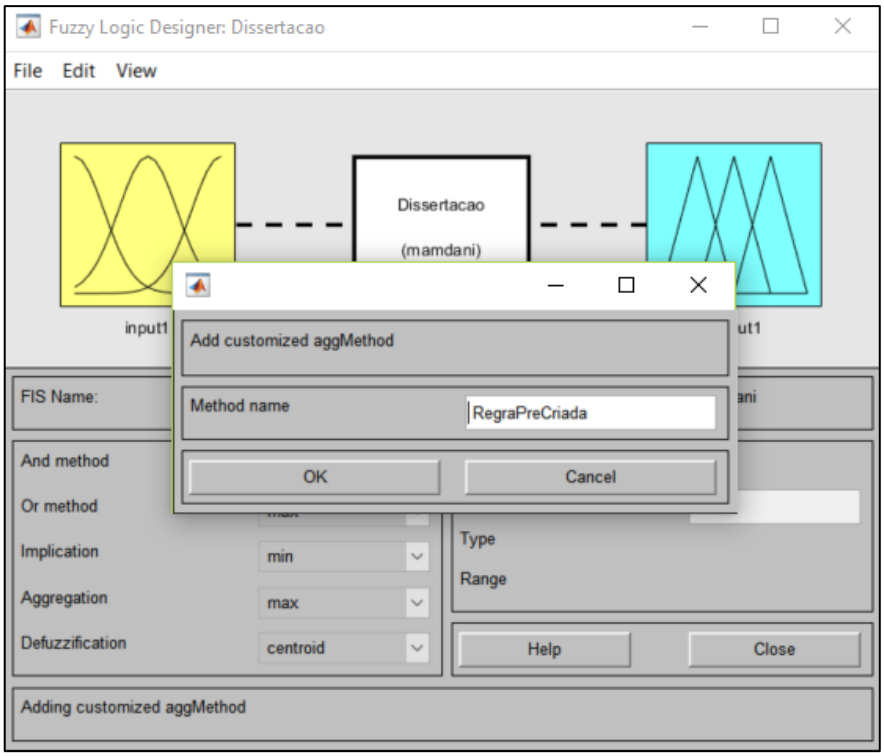


Figura 22 *Fuzzy Logic* Toolbox – Parâmetros configuráveis de *input*

Após terem sido criadas as entradas e as saídas necessárias, é, como referido no início da subsecção 3.3, essencial continuar a configuração das funções de associação.

Com este propósito, o utilizador selecciona o sinal respectivo, onde é apresentada a função, juntamente com os parâmetros de configuração, como demonstrado na Figura 23. Este editor permite a alteração de todas as funções associadas às variáveis do sistema difuso. Isto permite ao utilizador a manipulação das funções já existentes de forma a acomodar o propósito do problema.

Desta forma, para cada sinal de entrada, é possível escolher um tipo de função de associação, sendo esta pré definida ou customizada pelo utilizador. Este painel permite, de igual forma, escolher os limites de entrada, que devem corresponder aos valores mínimo e máximo possíveis que esse sinal pode receber.

No exemplo seguinte, escolheu-se um tipo de função com distribuição *gausseana*, com nome *Baixa*, e parâmetros de desvio padrão 5 e valor mínimo 0.

Se o utilizador pretender adicionar ou remover funções de associação para cada uma das variáveis, a opção está disponível na barra de separadores – *Edit* – *Add MFs* ou *Remove Selected MF*, respectivamente.

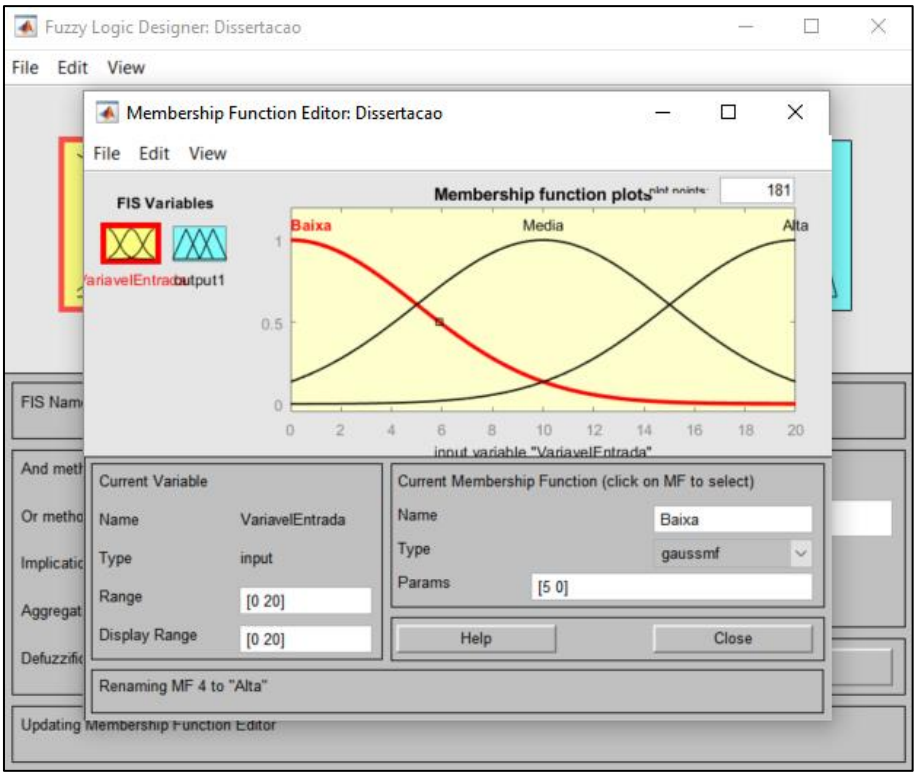


Figura 23 *Fuzzy Logic* Toolbox – Funções de associação

As mesmas operações devem ser realizadas para todos os sinais de entrada e de saída, bem como as respectivas funções de associação respectivas.

O próximo passo no desenvolvimento do sistema difuso é a definição de regras. Para isto, o utilizador selecciona, no *FIS Editor*, a janela do sistema difuso, sendo prontamente apresentada a janela de configuração, intitulada *Rule Editor*, como representa a Figura 24.

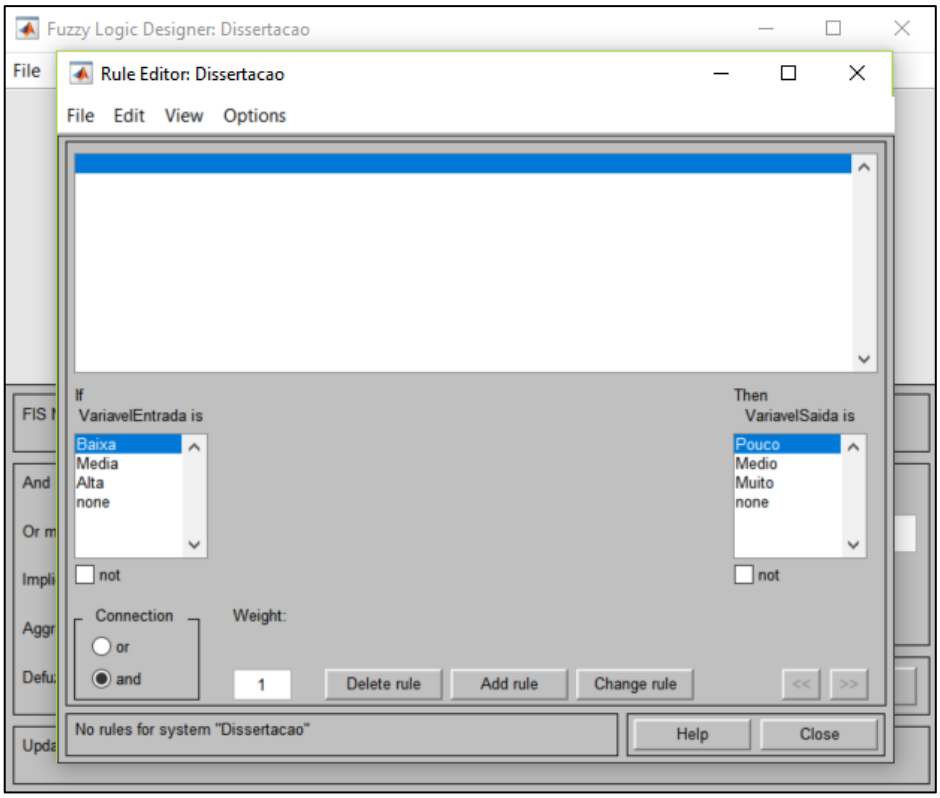


Figura 24 *Fuzzy Logic* Toolbox – Editor de regras

O editor permite a configuração de conceitos automaticamente, seleccionando uma função de associação de cada sinal de entrada e de saída e um parâmetro de ligação.

Para isto, o utilizador deve ter, pré definidos, os conceitos que quer utilizar para o seu sistema, visto que, conhecer a relação entre as entradas e as saídas é das principais razões pela qual se opta por o desenvolvimento de um sistema difuso.

Após seleccionar uma função da variável de entrada e uma função da variável de saída, a regra será exibida no diagrama de regras. Para isso, deverá adicionar a regra na caixa de texto *Add rule*. O utilizador deve desenvolver, no mínimo, uma regra que envolva cada uma das funções presentes no sistema, de forma a ser assegurado um resultado para cada cenário.

No caso de existirem múltiplas variáveis de entrada ou de saída, é possível isolar uma variável, excluindo a não pretendida, bastando seleccionar a função *none* para o sinal correspondente.

Por fim, após estarem desenvolvidas e configuradas todas as funções e regras associadas ao sistema, o resultado final pode ser analisado em função das regras ou da superfície ocupada pelos mapeamentos entrada-saída.

Deste modo, o utilizador selecciona no painel inicial a opção *View – Rules*, que irá ser representada por um interface gráfico *Rule Viewer*, como demonstrado na Figura 25.

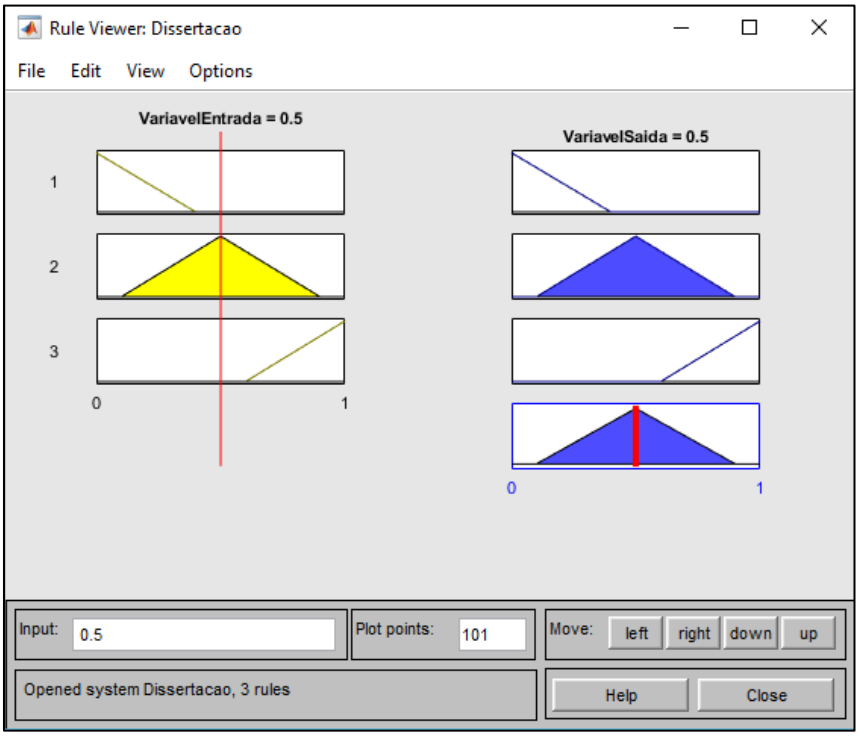


Figura 25 *Fuzzy Logic* Toolbox – *Rule Viewer*

Cada coluna demonstra o conjunto de funções de associação para cada variável de entrada ou de saída. Neste exemplo, o sistema contem três funções de associação para o sinal de entrada e, respectivamente, três para o sinal de saída.

Cada função deste editor está associada com uma regra particular e mapeia variáveis de entrada para a variável de entrada de regras. Isto pode ser visto, na Figura 25, como o número de linhas (1, 2 e 3) estando associada a cada regra criada previamente. Por sua vez, a coluna de saída (VariavelSaída) demonstra como as regras são mapeadas para as variáveis de saída. Por fim, o último gráfico da coluna da direita demonstra o resultado final agregado, tendo por base as funções de associação e regras respectivas, como a desfuzificação identificada por a linha vermelha.

Para este exemplo, para uma variável de entrada de 0.5, o resultado da VariavelSaída é 0.5.

Uma outra alternativa para rever o resultado gráfico é abrir o interface gráfico de superfície ou *Surface Viewer*, como resulta na Figura 26.

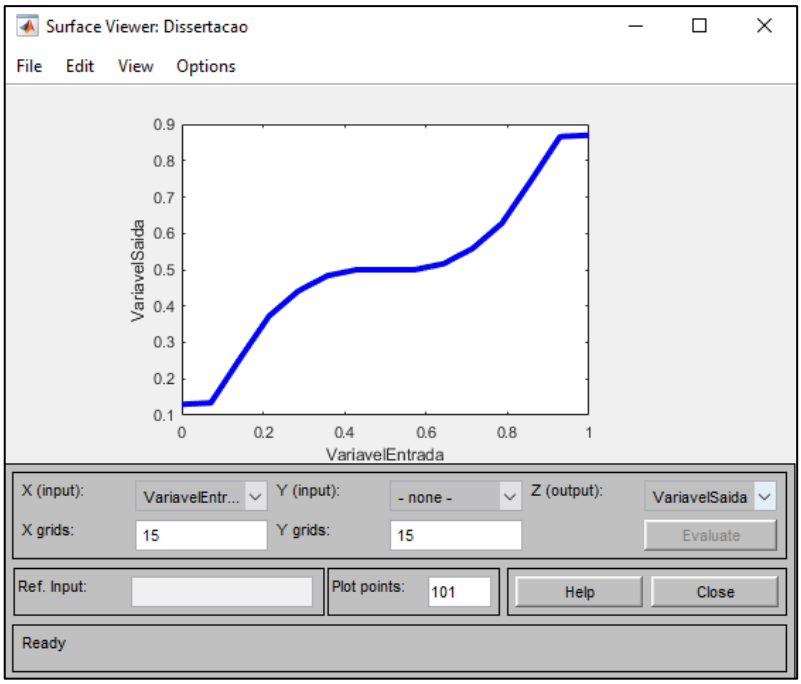


Figura 26 *Fuzzy Logic – Surface Viewer*

Este permite o desenho em 3D (no caso de existirem mais que duas variáveis) ou 2D e uma visualização da variável de saída em relação à variável de entrada.

Consequentemente, a biblioteca da expansão *Fuzzy Logic* dispõe de um bloco para a interface com o *Simulink*, intitulado *Fuzzy Logic Controller*, como representado na Figura 27. Este, quando incluído no projecto e carregado com o ficheiro .fis, é actualizado com as variáveis de entrada e de saída definidas pelo utilizador no ecrã inicial, como representado previamente na Figura 20.

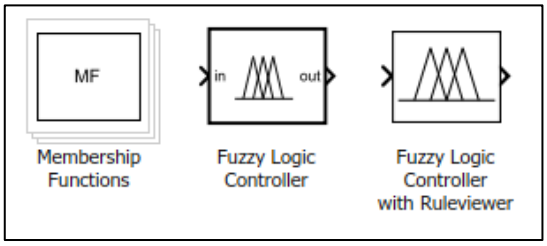


Figura 27 *Fuzzy Logic* – Biblioteca de blocos

3.4. *STATEFLOW*

O *Stateflow* é uma expansão do *Simulink* que permite modelar ambientes para o desenvolvimento de sistemas de tomada de decisão. Isto permite criar uma lógica baseada em sinais discretos ou contínuos, assim como baseada no tempo de simulação, e adaptar, consequentemente, o sistema.

A biblioteca do *Stateflow* contêm blocos para modelar o estado de variáveis, tabelas de mudança de estado ou tabelas de verdade, como demonstrado na Figura 28.

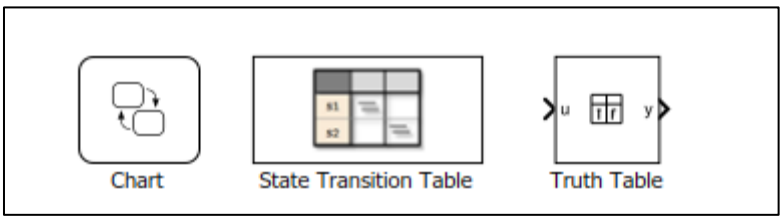


Figura 28 *Stateflow* – Diagrama de blocos

Ao iniciar o bloco de estado, irá ser apresentado ao utilizador um editor, sem lógica adicionada e com uma barra lateral de opções. Neste editor, são desenvolvidos sistemas de estado ao arrastar objectos gráficos (como *states* ou *junctions*), da barra lateral para o editor.

De seguida, e com referência à Figura 29, irão ser descritos os vários objectos que permitem ao utilizador configurar um sistema de estados.

1. Objecto de estado – Estes objectos permitem a introdução de lógica e a mudança de estado sempre que estiverem activos, sendo as acções executadas dentro destes objectos reflectidas imediatamente nas variáveis de saída. Dentro destes objectos, podem existir subestados, que são activos quando o estado hierarquicamente superior estiver activo.

2. Junções – Permitem a comunicação entre diferentes objectos de estado. Também é possível configurar condições ou acções durante a mudança. Como exemplo pode-se considerar a mudança do EstadoA para o EstadoB somente quando a condição imposta na junção se concretizar (x>1).

3. Estado por defeito – Permite a indicação ao bloco de estados de qual o objecto de estado que deve ser considerado por defeito. Sendo só possível adicionar um estado

por defeito, no caso de se adicionarem suplementares, terão de ser indicadas condições especificas a serem consideradas.

4. Caixa de agrupamento – Permite aglomerar diferentes objectos de estado de forma a separar funções e objectos dentro e fora da caixa. É utilizado como forma de conceder visibilidade a diferentes partes do sistema.

5. Função do MATLAB – Quando se transacciona de um estado para este objecto, é executada a função correspondente. Após isto, o sistema tem continuidade somente se esta função estiver ligada a um novo objecto de estado.

6. Tabela de estados – Estas funções implementam lógica associada a uma tabela gráfica. Estas tomadas de decisão podem, por exemplo, ser utilizadas em alteração de estados com lógica oportuna.

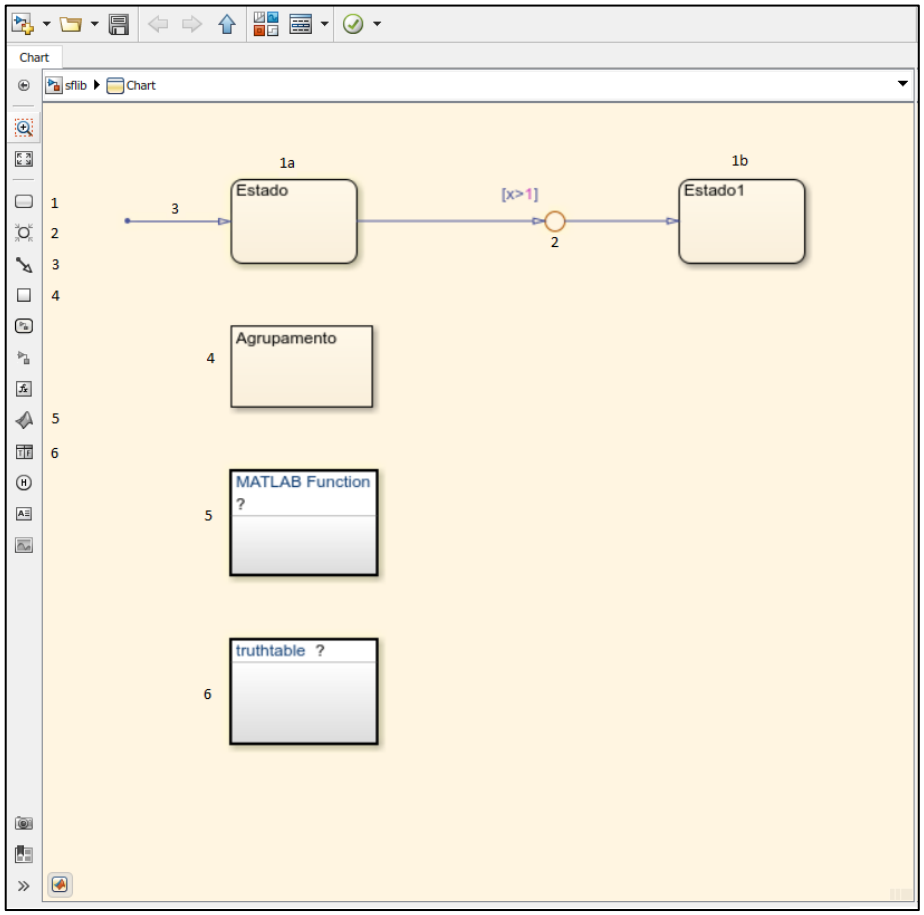


Figura 29 *Stateflow* – Objectos gráficos

Como forma de reforçar o exemplo anteriormente referido, foi criado um sistema que altera a variável de saída como consequência da variação do sinal de entrada.

Para isto, foi aplicado um sinal sinusoidal na entrada do sistema, foram configurados os objectos de estado e analisado o sinal de saída, como se comprova na Figura 30.

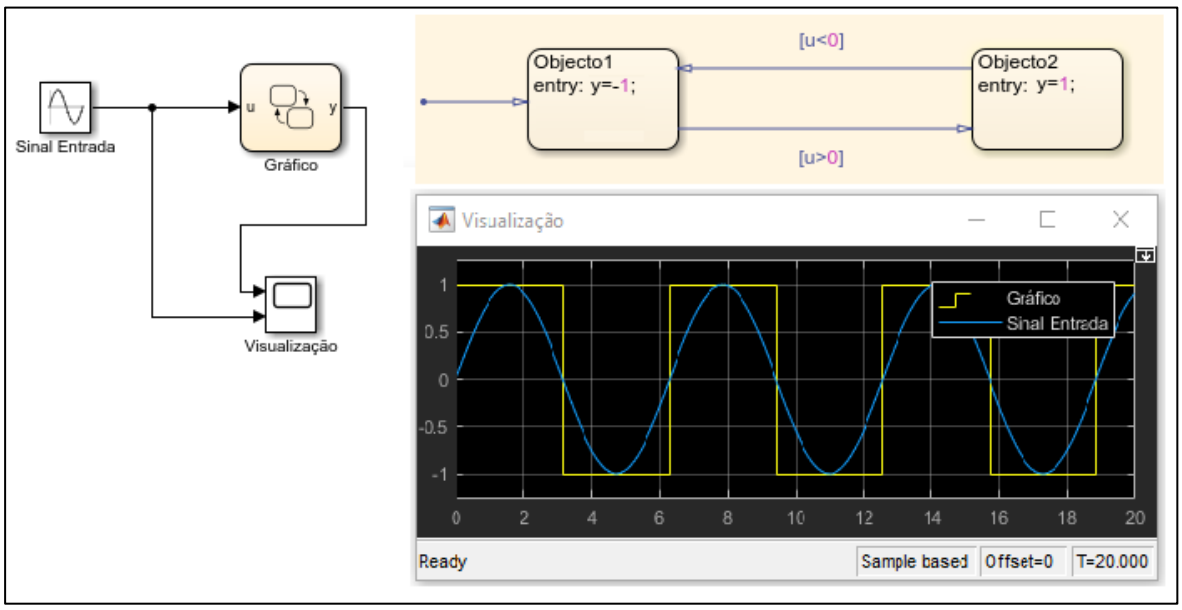


Figura 30 *Stateflow* – Simulação de um sistema sinusoidal

Como pode ser analisado, quando o sinal de entrada é positivo, o objecto de estado *Objecto2* é activado e a saída, adequa-se, actualizando o valor para 1.

O inverso também se verifica, correspondente ao sinal de entrada negativo, o objecto de estado é alterado para *Objecto1*, tendo o sinal de saída o valor de -1.

De forma a configurar o compilador com os sinais adequados, é também necessário indicar a funcionalidade destes. Neste caso, o utilizador terá que seleccionar, na barra de ferramentas, *Tools -> Model Explorer*, no qual será apresentada uma janela, como indicado na Figura 31.

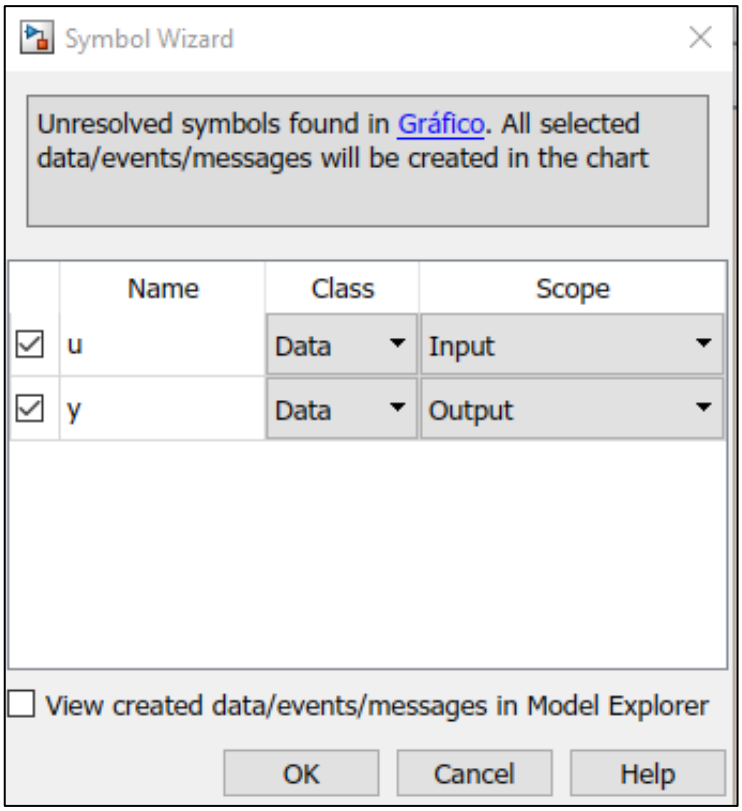


Figura 31 *Stateflow* – Configuração dos sinais de entrada/saída

3.5. SIMULINK

Sendo o *Simulink* uma ferramenta que agrega várias expansões do MATLAB, e tendo ligação directa também com estas, são utilizados no sistema os blocos que não se podem agrupar em nenhum dos subcapítulos anteriores. Por esta razão, e porque estes pertencem à biblioteca do *Simulink*, optou-se por efectuar uma descrição de cada um deles numa subsecção separada das restantes.

Apresentam-se, assim, de seguida, os restantes blocos significativos, utilizados ao longo do projecto no desenvolvimento do sistema.

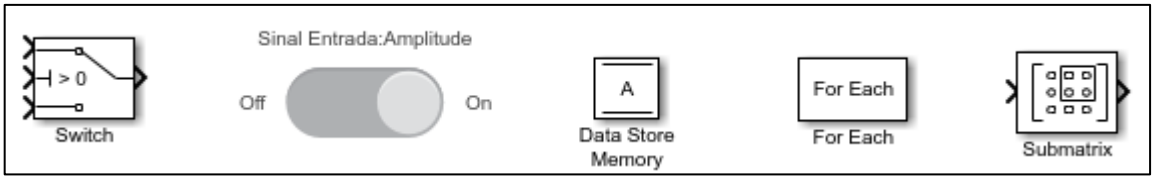


Figura 32 *Simulink* – Restantes blocos essenciais

- *Switch*

Permite a comutação entre duas variáveis de entrada, tendo por base a satisfação do critério configurado. Desta forma, é possível configurar a passagem do sinal *Input 1* quando o valor for superior, inferior ou diferente do *Input 2*. No caso de não satisfazer a condição prévia, a saída corresponde ao *Input 3*.

- *Slider Switch*

Com o objectivo de alterar uma variável, com valores pré definidos, entre simulações, o *slider* permite ao utilizador uma comutação, com o objectivo, por exemplo, para analisar diferentes resultados dos diferentes variáveis de entrada.

- *Data Store Memory – Data Store Read – Data Store Write*

O bloco *Data Store Memory* define um espaço de memória para ser utilizado por ambos os blocos *Data Store Read* e *Data Store Write*. Estas variáveis globais podem ser utilizadas como um elemento individual, conjunto ou em matriz. Quando uma variável é inicializada com recurso ao *DataStore Memory*, pode ser acedida em qualquer localização do sistema com recurso aos outros dois blocos.

- *For Each*

Habilita os blocos dentro do subsistema de processamento dos elementos de entrada independentemente. Isto permite a separação de uma matriz, intitulada *Teste* no exemplo abaixo descrito, de [2x1] em dois sistemas similares, onde a entrada do elemento [1x1] é tratada individualmente, como é representado na Figura 33.

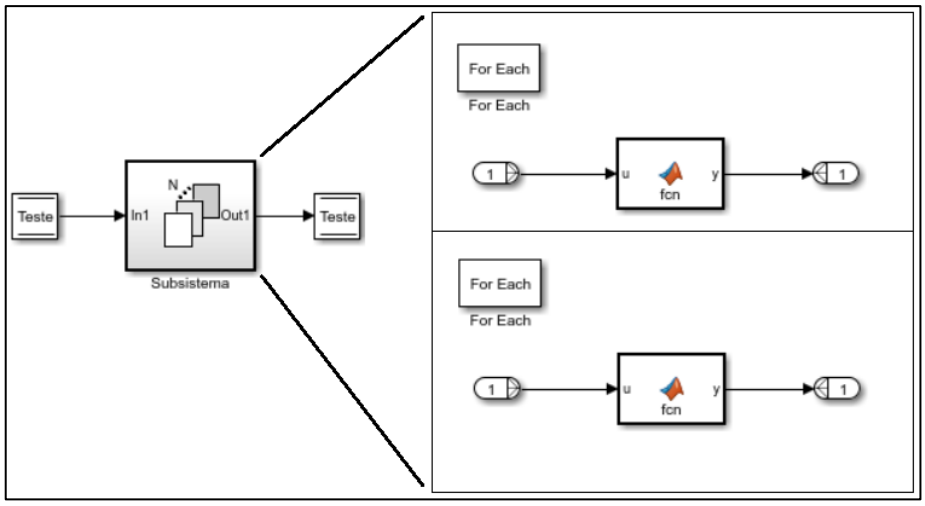


Figura 33 *Simulink* – For Each block

Na saída, ambos os sinais são concatenados e a matriz *Teste* restaura o seu tamanho inicial.

- *SubMatrix*

Divide a matriz de entrada em diversos elementos, configuráveis. Isto permite seleccionar a primeira coluna de uma matriz [NxN] para obter o sinal de saída [Nx1].

4. DESENVOLVIMENTO DO SISTEMA

Neste capítulo são descritos os passos necessários à criação, configuração e simulação do sistema, assim como à interligação das diferentes extensões do MATLAB, desde eventos discretos, com o *SimEvents*, a eventos contínuos, com o *Simulink 3D Animation*.

Após o processo de desenvolvimento se encontrar descrito, são comparadas as soluções entre ambos os controladores difusos ou temporais, apresentando o tempo médio de espera ou o número de carros que se encontram dentro do sistema.

4.1. MODELO GERAL

O desenvolvimento do sistema foi dividido em diferentes partes, cada uma correspondente a diferentes comportamentos do sistema. Como pode ser analisado na Figura 34, foram geradas variáveis globais que são utilizadas em todo o sistema. Estas são:

- S1 – Variável que retém o valor dos semáforos associados ao Cruzamento 1 (C1).
- S2 – Variável que retém o valor dos semáforos associados ao Cruzamento 2 (C2).
- Lane – Variável utilizada para guardar o destino dos veículos gerados na Faixa 1.

- C – Matriz que possui a informação dos veículos gerados na Faixa 1. Contém a posição no eixo xx, posição no eixo yy, faixa de destino e o número de identificação.

- D - Matriz que possui a informação dos veículos gerados na Faixa 2. Contém a posição no eixo xx, posição no eixo yy e o número de identificação.

- E - Matriz que retêm a informação dos veículos gerados na Faixa 3. Contém a posição no eixo xx, posição no eixo yy e o número de identificação.

- F - Matriz que guarda a informação dos veículos gerados na Faixa 4. Contém a posição no eixo xx, posição no eixo yy e o número de identificação.

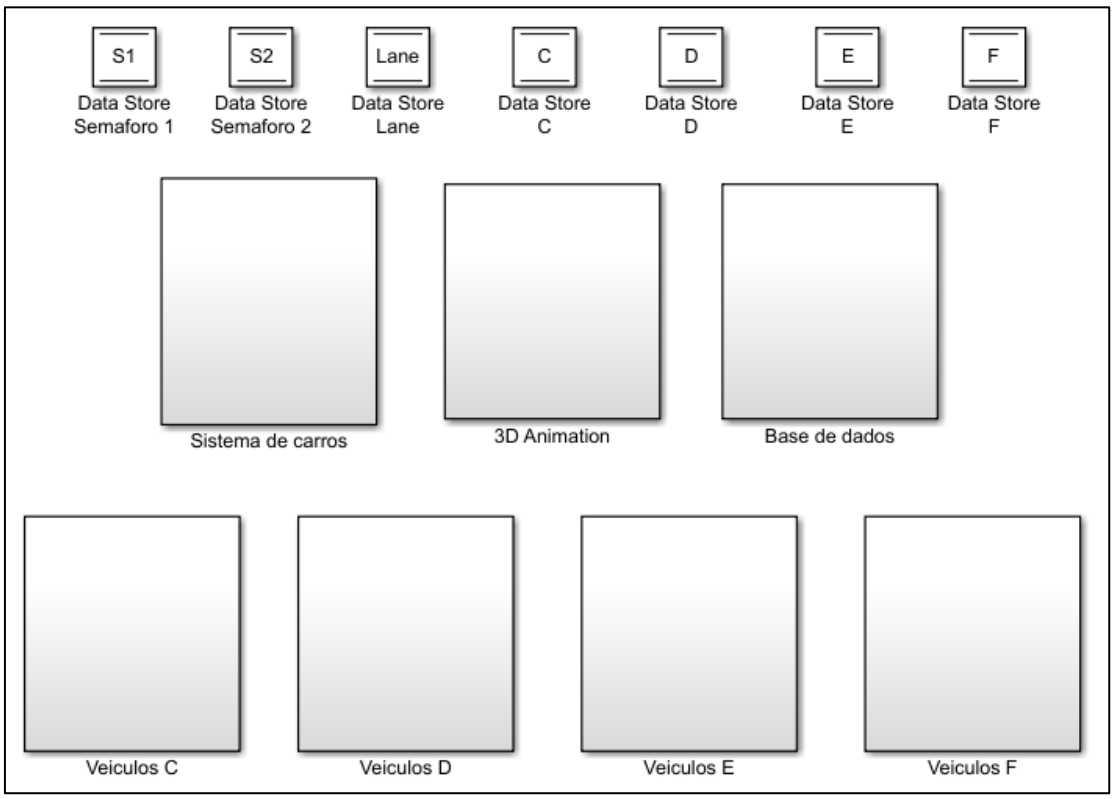


Figura 34 Visão geral do sistema discreto e contínuo

Foi criado um subsistema, intitulado *Sistema de carros* que engloba todos os eventos discretos gerados e manipulados durante a simulação. Neste bloco, são criados e removidos os veículos do sistema, é aplicada e alterada a lógica dos controladores dos semáforos, e são visualizados todos os dados estatísticos referentes à simulação (tempo médio de espera e número de veículos em diversos pontos do sistema).

O bloco *3D Animation* contém o *interface* entre as bases de dados e a simulação 3D. Em tempo real, e enquanto os valores anteriormente descritos vão sendo actualizados na base de dados recorrendo às dinâmicas dos veículos criados, estes também são actualizados no ambiente gráfico recorrendo ao interior deste bloco.

O bloco *Base de dados* permite a visualização de todos os dados das variáveis globais, em tempo real, à medida que estes são actualizados dentro dos blocos *Sistema de Carros* e *Dinâmica de Veículos*, os quais são descritos de seguida.

Os restantes quatro subsistemas (Veículos C/D/E/F) compreendem a modelação das dinâmicas dos veículos criados nas vias 1, 2, 3 e 4, respectivamente. Servem também para aplicar o sistema de colisões criado para evitar contactos entre diferentes objectos do sistema.

Desta forma, para enumerar o escrito previamente com o modelo 3D criado, na Figura 35 ilustram-se os conceitos mais importantes, relacionando-os com a visão geral do sistema da Figura 34.

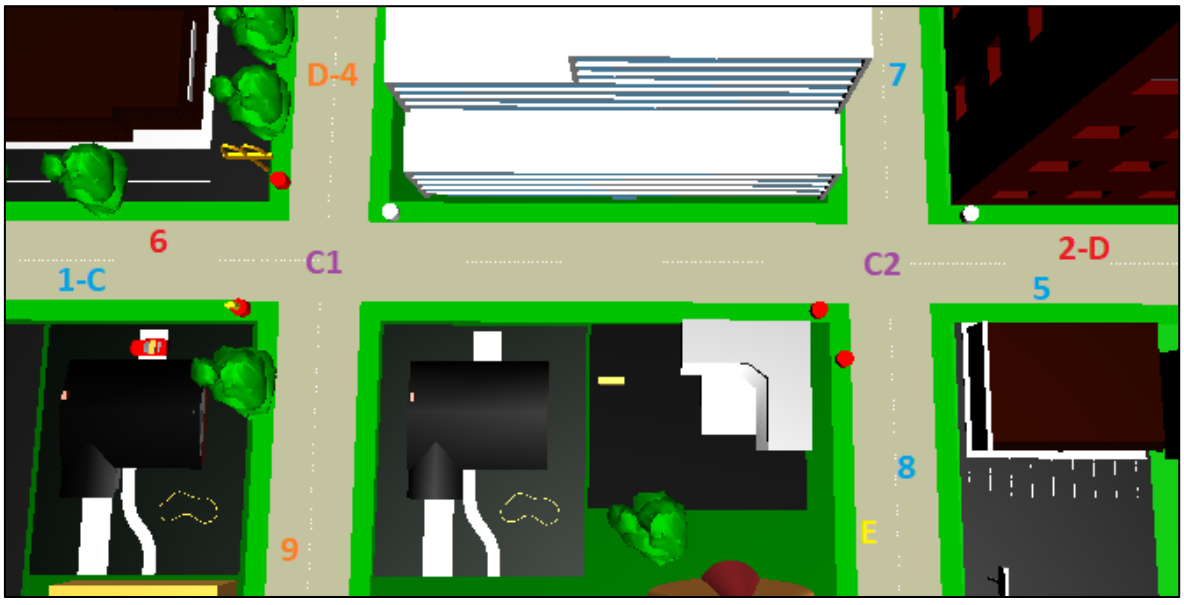


Figura 35 Visão geral do sistema 3D

Neste modelo, o processo de condução e escolha dos veículos é autónomo e aleatório. Os veículos são colocados nas suas vias aleatoriamente e em intervalos de tempo distintos e não relacionados. De igual forma, dentro dos destinos possíveis desenvolvidos, as escolhas são também aleatórias, não existindo manipulação por parte do utilizador.

Com referência à Figura 35, o modelo foi desenvolvido seguindo um mapeamento o mais aproximado à realidade possível. Assim, os condutores devem respeitar os sinais luminosos, não sendo permitido fazer inversão de marcha ou conduzir em faixas de sentido proibido e evitam-se as colisões entre os veículos, parando estes quando se encontram demasiado próximos de um outro veículo.

Deste modo, de seguida enumeram-se as regras base dos veículos para o desenvolvimento deste modelo:

- Os veículos C, gerados na faixa 1-C, podem ter como destino as faixas 5, 7 e 8.
- Os veículos D, gerados na faixa 2-D, têm como destino a faixa 6.
- Os veículos E, simulados como veículos de emergência e devendo ter prioridade sobre qualquer outro da faixa 1 e 2, são gerados e mantêm-se na direcção Norte, movendo-se para a faixa 7.
- Os veículos D, gerados na faixa D-4, têm como destino a faixa 9.

Deste modo, a Figura 36 elucida os trajectos possíveis para os veículos das faixas anteriormente mencionadas.

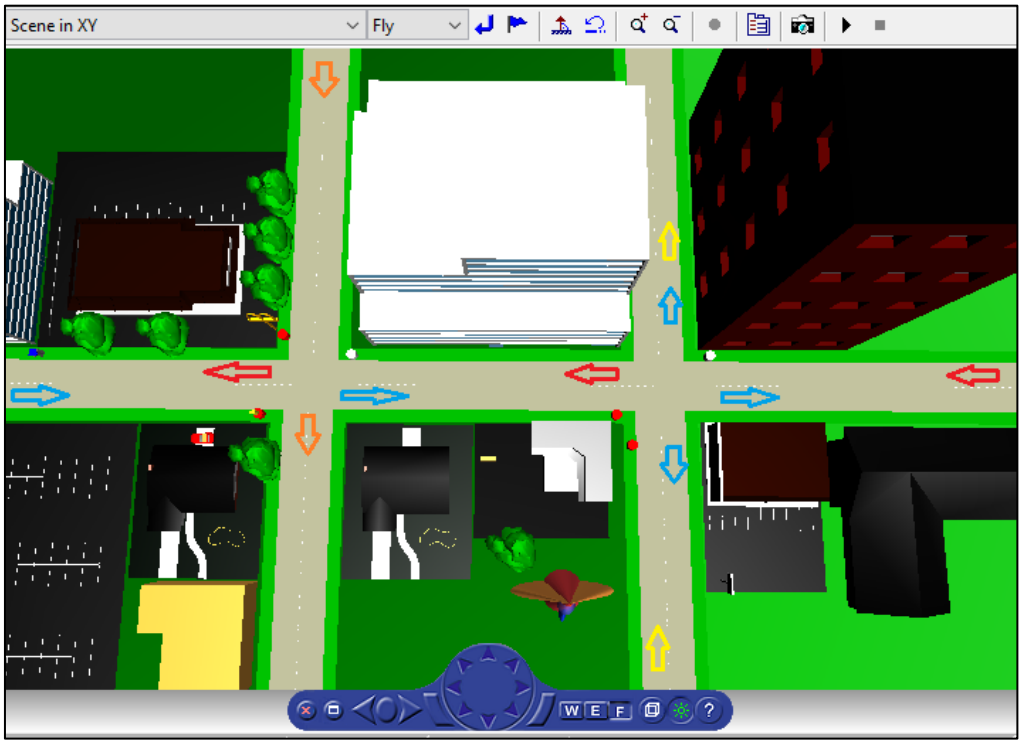


Figura 36 Faixa de destino dos veículos gerados

Assegura-se, assim, uma completa aproximação a um cenário real, onde os condutores podem efectuar mudanças de direcção à esquerda, direita ou prosseguir na mesma via.

Consequentemente, com as mudanças de direcção e cruzamentos, torna-se imperativo o controlo recorrendo a sinais luminosos. Apesar de o sistema possuir uma lógica que previne a colisão entre veículos, a não utilização de sinais luminosos não é adaptada à realidade, onde estes existem em grande parte dos cruzamentos das grandes cidades [14]. Em situações em que estes são inexistentes, os cruzamentos são substituídos por rotundas, não se adaptando ao âmbito deste projecto.

Desse modo, foram criadas regras de mudança de estado dos sinais luminosos adaptados ao sistema desenvolvido. Estas permitem a circulação de veículos e mudança de faixas sem colisões, permitindo também adaptar os controladores desenvolvidos durante a realização deste projecto a ambos os cruzamentos.

Com referência à Figura 37, consideram-se as seguintes regras:

- O Cruzamento 1, identificado por C1, dispõe de três sinais luminosos: Norte, Este e Oeste. O comportamento dos sinais Este e Oeste são similares e coincidentes e permitem a passagem dos veículos da faixa 1 para oeste e da faixa 2 para este. Inversamente, encontra-se o semáforo Norte, permitindo a passagem dos veículos da faixa 4 para a faixa 9. Conclui-se, assim, que quando o sinal Este comuta para verde, o de Oeste tem o mesmo comportamento. Paralelamente, o Norte comuta para vermelho, de forma aos veículos não colidirem.

- O Cruzamento 2, identificado por C2, dispõe, igualmente, de três sinais luminosos (Este, Oeste e Sul). Nestes, ao contrário de C1, não existem sinais conjuntos e todos dispõe da sua lógica individual. Isto significa que, no momento em que o sinal de Oeste se encontrar verde, os restantes deverão encontrar-se vermelhos. A mesma lógica foi desenvolvida para os sinais de Este e Sul.

No seguimento da explicação prévia, demonstra-se um exemplo na Figura 37.

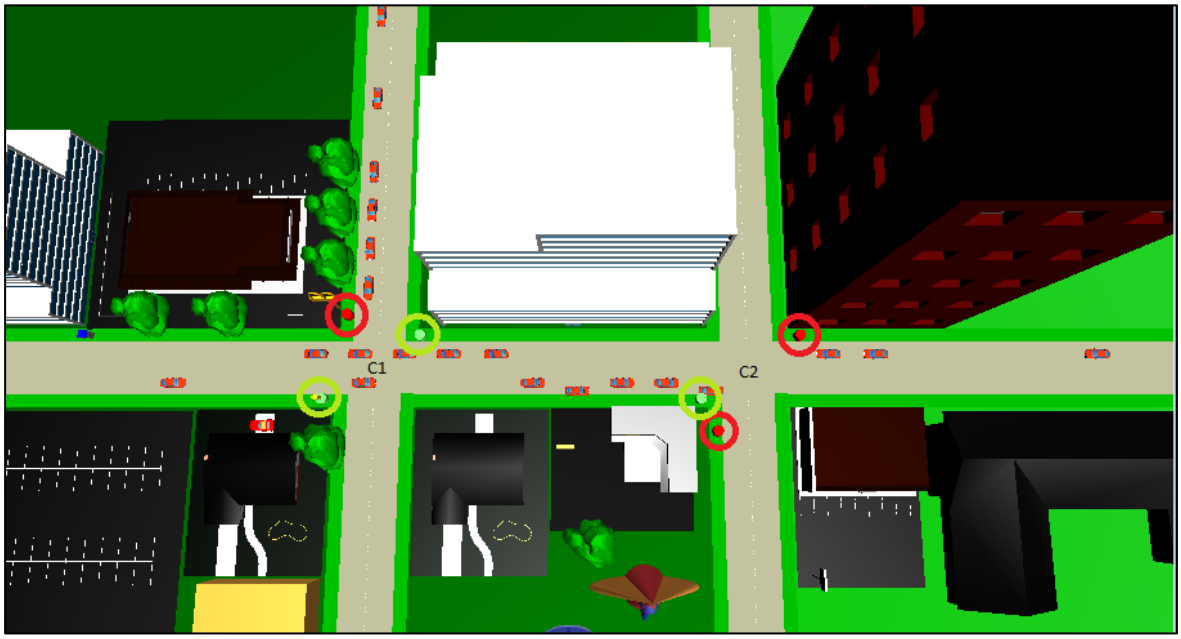


Figura 37 Comutação dos sinais luminosos

Qualquer veículo que encontre um sinal luminoso constituinte deste sistema segue as regras de trânsito comuns:

- Ao encontrarem um sinal luminoso vermelho não ultrapassam a linha de cruzamento.
- Durante um sinal amarelo, os veículos prosseguem até este ser alterado para vermelho. Em nenhuma condição, inclusive durante a comutação, o veículo ultrapassa a linha de cruzamento estando o sinal vermelho gerado. Independentemente da lógica utilizada para controlar os sinais luminosos, esta cor é sempre mantida durante 3 segundos após a comutação. Isto assegura o tempo suficiente para que os veículos que se encontram a atravessar o cruzamento cheguem à via seguinte.
- Durante um sinal verde, não existem restrições.

Após uma visão geral do sistema, irão ser, de seguida, detalhados os diversos blocos constituintes do mesmo, já mencionados anteriormente.

4.2. SISTEMA DE CARROS

Este sistema compreende a modelação discreta das entidades, desde a sua criação ao seu término, contendo os controladores desenvolvidos e a lógica de actuação destes nos sinais luminosos.

A esta modelação do sistema discreto, encontra-se associada a sua representação no ambiente virtual 3D. Com referência à Figura 38, os diversos blocos encontram-se associados a diversos pontos do sistema:

- O bloco *Entity Generator* é utilizado na criação de veículos. Foi utilizado um bloco por cada faixa onde sejam geradas entidades, resultando nos *Entity Generator* C, D, E e F.

- De seguida, a cada bloco anteriormente descrito, encontra-se ligado um *Entity Queue*. Este tem a função de reter as entidades antes destas ultrapassarem o sinal luminoso.

- O *Entity Server* é utilizado para transpor as entidades que se encontram antes do sinal luminoso para uma nova faixa.

- O *Entity Gate*, não tendo relação directa modelada no sistema 3D, encontra-se na qualidade de uma "cancela", a qual permite aos veículos, após a comutação do sinal luminoso para verde, continuar o seu trajecto.

- Nos subsistemas C e D, onde os veículos podem transpor dois conjuntos de sinais luminosos, o ciclo anterior, com excepção do bloco de criação de entidades, repete-se, existindo novos blocos de *Entity Queue*, *Entity Server* e *Entity Gate*.

- Após o término do ciclo, quando os veículos chegarem ao fim da sua respectiva via, é utilizado, em todos os subsistemas, um *Entity Terminator*, removendo as entidades discretas do sistema do *SimEvents* e, como descrito futuramente na Figura 47, removendo a identificação da base de dados e os veículos do sistema 3D.

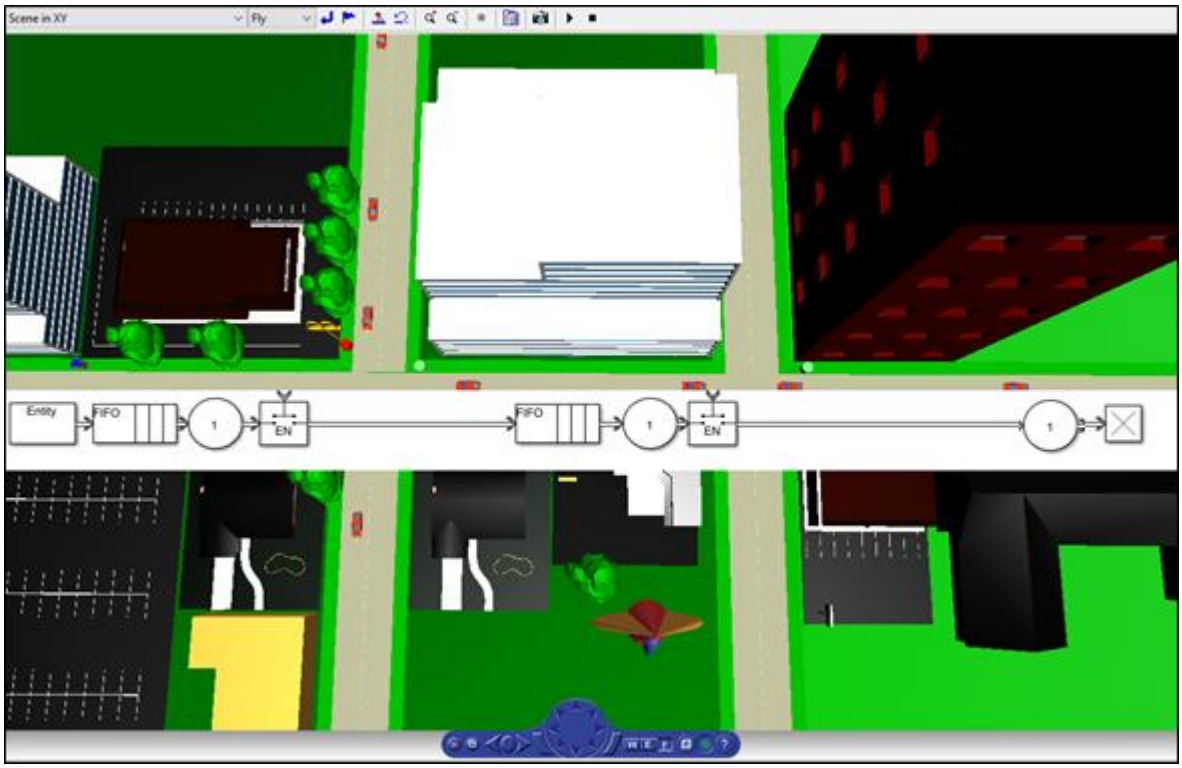


Figura 38 Modelo 3D com correspondência ao *SimEvents*

Como visto anteriormente no Capítulo 3, as entidades servem para representar objectos contáveis aos quais se possam associar atributos manipuláveis. Foi esta a forma desenvolvida para produzir os veículos e alterar os parâmetros destes durante as simulações, sendo gerados como entidades, a partir da extensão *SimEvents*,

A cada sistema de veículos (C, D, E e F) está associado um *Event Generator*, responsável por criar a entidade e a sua parametrização, como pode ser comprovado na Figura 39.

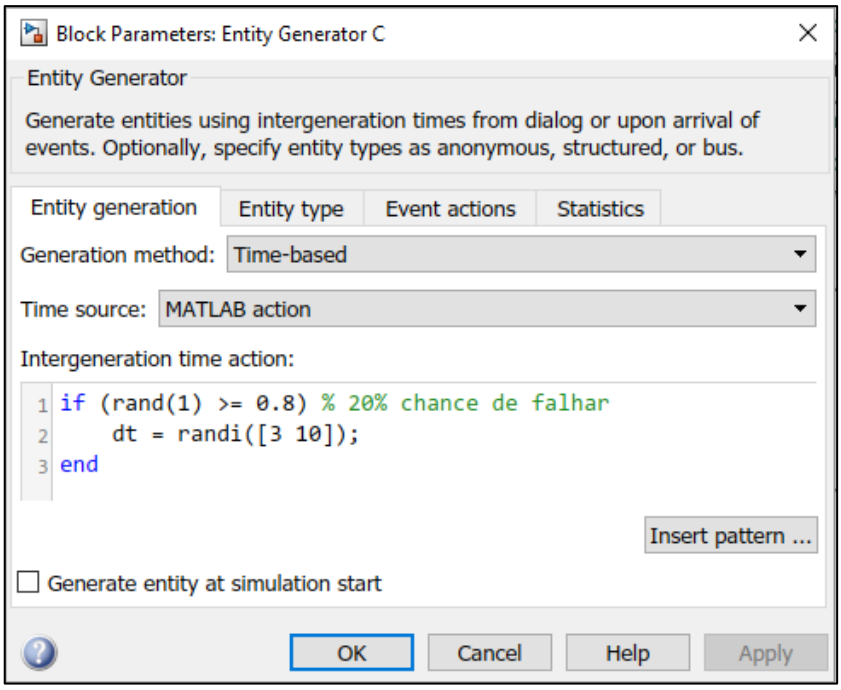


Figura 39 Criação de entidades da via C, D e F

O intervalo de tempo considerado entre a criação dos veículos é aleatório e tenta assemelhar-se a uma simulação real. Neste projecto, considerou-se um intervalo de tempo entre 3 e 10 segundos, com uma probabilidade de 20% de não existir nenhum carro gerado. Esta lógica foi introduzida para os blocos de criação de entidades dos sistemas C, D e F, sendo estes os veículos normais. Contudo, para o veículo de emergência da via E não existe possibilidade de falha, sendo este gerado obrigatoriamente, uma vez de 30 a 60 segundos, de forma a introduzir instabilidade no sistema sempre que possível.

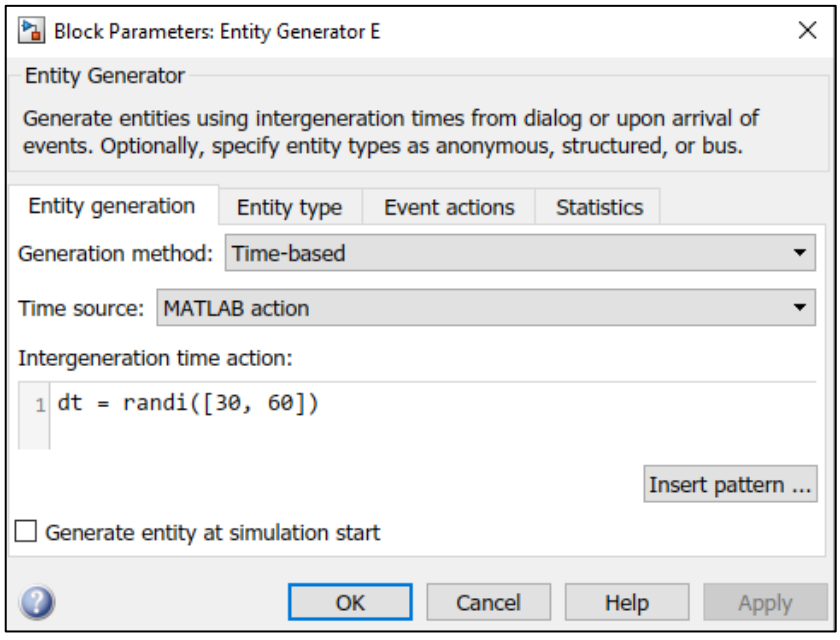


Figura 40 Criação de entidades da via E

No momento da criação da entidade são também associados parâmetros que servem para manipulação e lógica durante o resto da simulação:

- **entity.Lane** – Somente inicializado nos veículos C. Define aleatoriamente a faixa de destino da entidade, podendo conter os valores 1000, 1001 ou 1002. O valor 1000 tem o significado do veículo se dirigir para a faixa 5, enquanto o valor 1001 representa a faixa 7 e o valor 1002 representa a faixa 8.

- **entity.Timestamp** – Número de identificação da entidade durante toda a simulação. Esta é usada para enumeração durante todo o processo discreto, para o preenchimento da informação na base de dados.

- **entity.Data** – Transforma a via onde o veículo é criado em valores numéricos a serem utilizados posteriormente no modelo. A cada via é associado um valor predefinido de 500, 600, 700 e 800 para as vias C, D, E e F, respectivamente.

As bases de dados que correspondem a cada via são preenchidas com recurso a uma função do MATLAB (Anexo A) invocada na criação da entidade, como pode ser analisado na Figura 41.

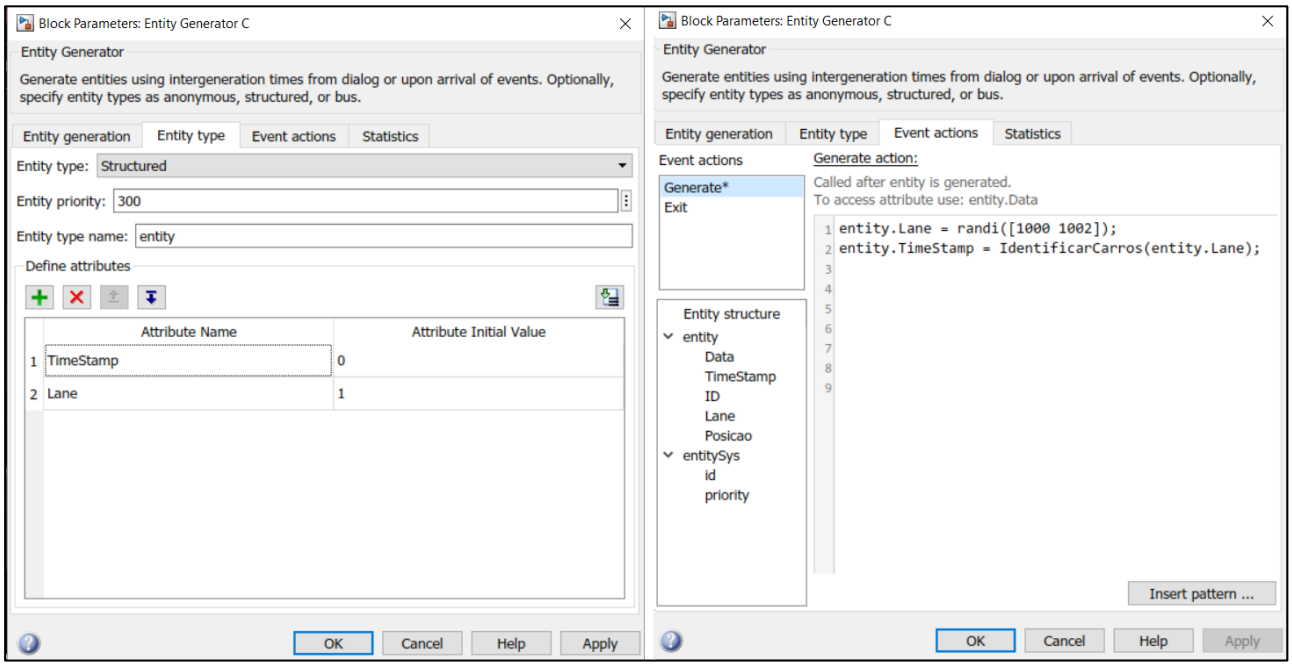


Figura 41 Entity Generator - Criação dos parâmetros da entidade

Esta função tem como entrada o parâmetro *Lane*, previamente e aleatoriamente gerado entre os valores 1000 e 1002. Um dos entraves encontrados na associação entre sistemas discretos e funções do MATLAB, é a impossibilidade de manipulação dos parâmetros das entidades a partir da geração de código. Desta forma, para assegurar que a identificação dos veículos composta a partir da variável *Timestamp* se mantinha crescente durante a simulação, foi criada uma variável global *NumeroCarrosC*. Esta variável é responsável por guardar localmente a contagem do número de carros criada e reverter para o sistema discreto a contagem actual, utilizada como *Timestamp*.

De notar que durante esta operação, a base de dados é preenchida por ordem decrescente e é sempre actualizada na criação de uma entidade.

Para melhor compreensão, a Figura 42 assinala a base de dados dos veículos C, contendo a informação, a cada linha individual, de posição x, posição y, faixa de destino e identificação do veículo, respectivamente. Enquanto o valor da posição em ambos os eixos é actualizado ao instante, as duas últimas colunas são preenchidas durante a criação da entidade, propriamente na função *IdentificarCarros()*;

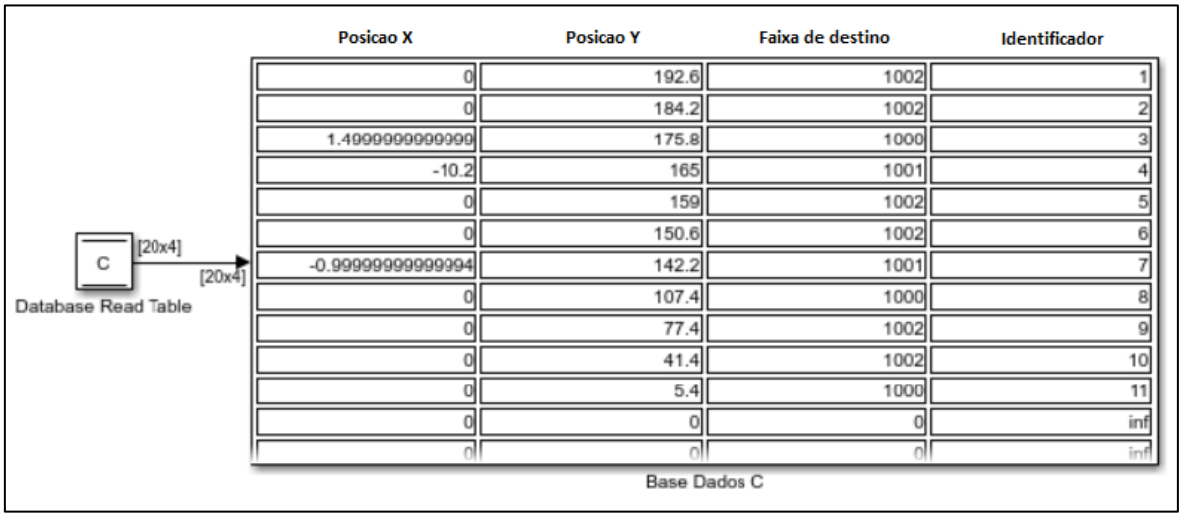


Figura 42 Base de Dados dos veículos C, com informação durante a simulação

O código utilizado encontra-se disponível no Anexo A.

Com acesso ao separador *Statistics*, do bloco *Entity Generator*, é possível aceder ao número de entidades geradas a partir do bloco (*Number of entities departed*), que mais tarde irá ser utilizada de forma a verificar o correcto funcionamento do sistema.

Após a criação e a parametrização das entidades e estas serem acrescentadas à base de dados, são distribuídas para os respectivos *Entity Queues*, associados a cada subsistema (C, D, E e F), como descrito anteriormente.

Estes blocos são configuráveis com o número máximo de veículos permitidos e a sua respectiva ordem dentro do bloco. Para efeitos de simulação, não existe imposição de limite de veículos neste bloco, visto que, comparativamente à realidade, não existe um número limite de veículos num espaço físico, sendo somente limitante a sua grandeza física relativa ao espaço ocupado.

Desta forma, limitou-se a 10, o número de veículos que se encontram no *Entity Queue*, visto ser este o número máximo suportado do sistema 3D, em termos de comprimento, desde o sinal luminoso até ao local onde são inicializados os veículos.

O tipo de fila de espera configura-se como FIFO, pois é a necessidade do sistema que o primeiro veículo a chegar ao sinal luminoso, também seja o primeiro veículo a ultrapassá-lo.

De igual forma que no bloco anterior, no separador *Statistics*, seleccionou-se o tempo médio de espera dos veículos desde que se encontram no bloco. Isto significa que o resultado desta informação trará à simulação a quantidade de tempo desde que estes entram na faixa até que ultrapassam o sinal luminoso. Permite assim ao utilizador analisar o tempo despendido por cada veículo em espera em cada faixa, podendo testar a eficiência do controlador utilizado nos sinais luminosos.

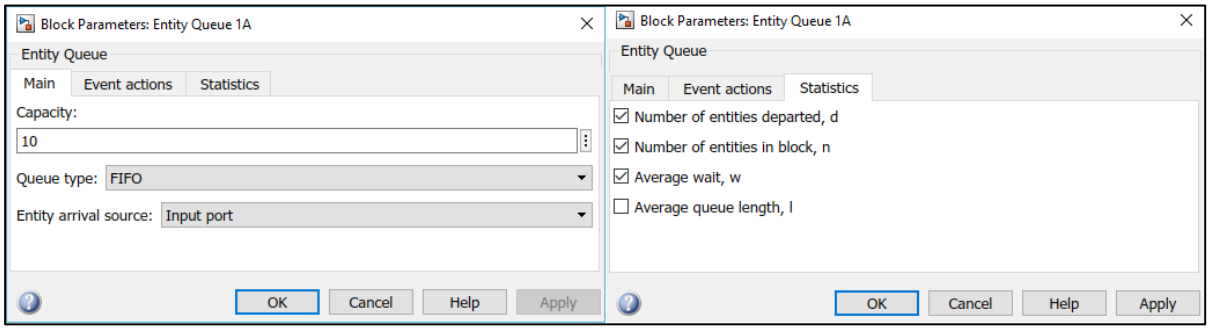


Figura 43 Entity Queue C – Parametrização do bloco

De seguida, o *Entity Server* e o *Entity Gate*, funcionando em conjunto, permitem a saída da entidade da fila de espera e colocam-na nos blocos seguintes.

A lógica que permite a passagem dos veículos nos sinais luminosos encontra-se interligada entre diferentes blocos e diferentes subsistemas. De forma a simplificar a compreensão, foi criado o fluxograma da Figura 44:

- O controlador, analisado posteriormente na Figura 51, sempre que altera o estado de um sinal luminoso, actualiza a sua variável global (S1 ou S2) para um valor predefinido para cada uma das cores.

- A função MATLAB, a correr em plano de fundo, pesquisa, com recurso à base de dados dos respectivos veículos (C, D, E ou F), se algum veículo se encontra nas imediações dos sinais luminosos e se o sinal luminoso se encontra com o valor pré-definido, associada à cor verde.

- A saída da função previamente descrita activa a porta de controlo do *Entity Gate*, presenta na Figura 46. Quando isto acontece, o tempo de serviço, previamente configurado no *Entity Server*, permite a ultrapassagem do veículo enquanto a porta de controlo retém o sinal positivo. Desta forma, é assegurada toda a passagem a veículos que tenham transposto o sinal luminoso.

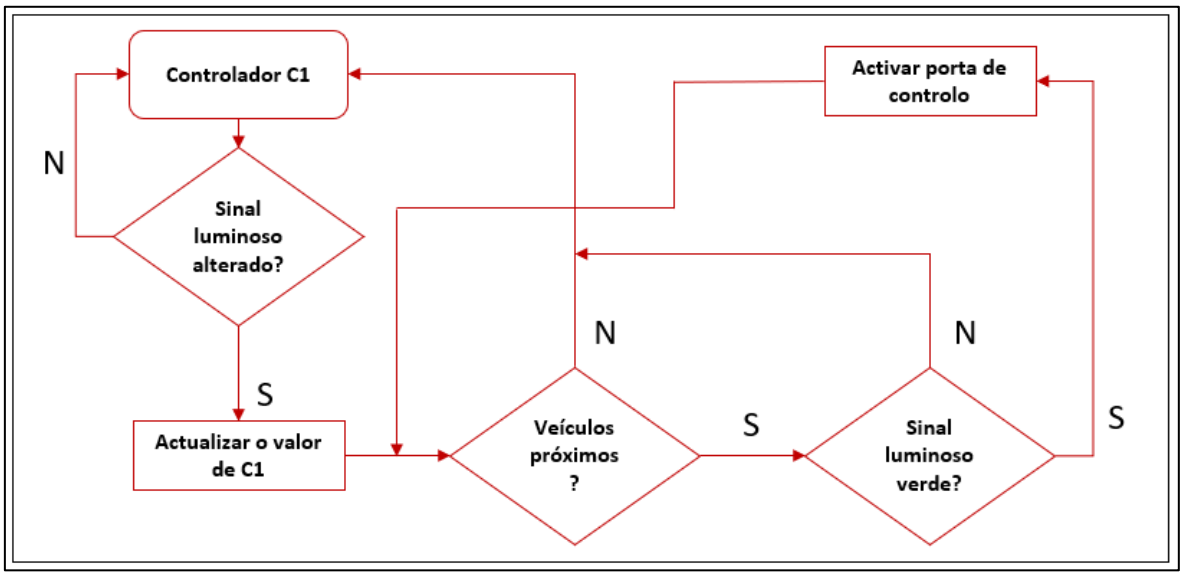


Figura 44 Fluxograma de passagem dos veículos

No caso de o *Entity Server* não se encontrar configurado ou não se incluir no sistema, no momento de abertura da porta de entidades, iriam ser gerados múltiplos eventos, passando as entidades pelo bloco *Entity Gate* sem gestão e de forma incoerente com o modelo 3D.

O *Entity Server* foi assim configurado com o limite de uma entidade e o tempo de serviço de 1 segundo de simulação. Desta forma, assegura-se um fluxo controlado e criterioso, como comprovado na Figura 45. Igualmente ao bloco anterior e de forma a ser adicionado o tempo a este, também se adicionou, no separador *Statistics*, o tempo de espera.

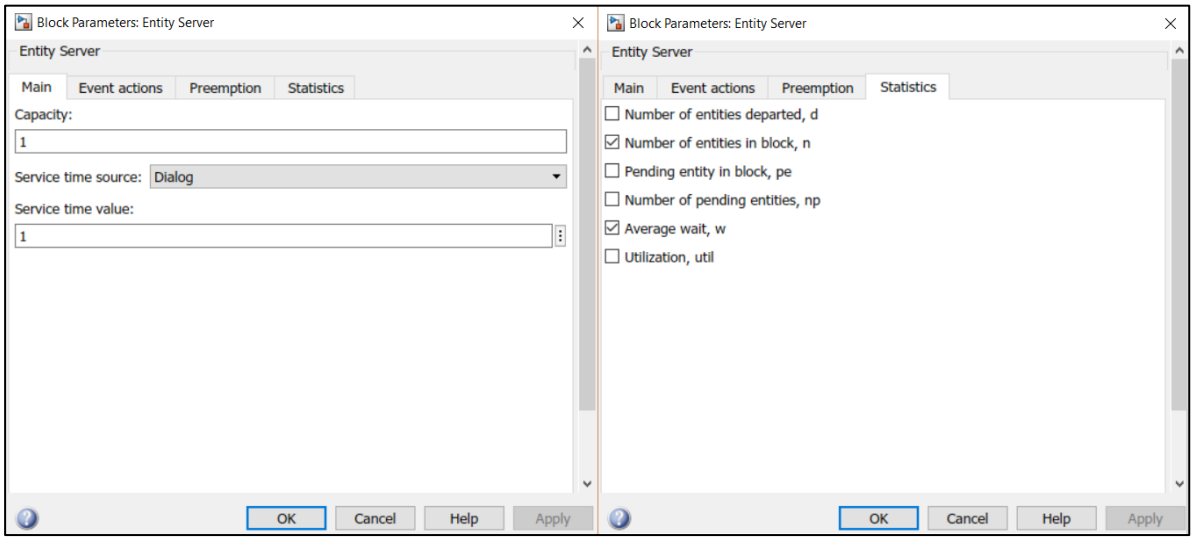


Figura 45 Entity Server – Parametrização do bloco

Assim, o *Entity Server* é activado quando o veículo ultrapassa a linha do sinal luminoso. Para isto acontecer, o *Entity Gate* tem de permitir a passagem, activando a porta quando recebe uma mensagem positiva, a partir da saída da função do MATLAB (Anexo B), como representado na Figura 46.

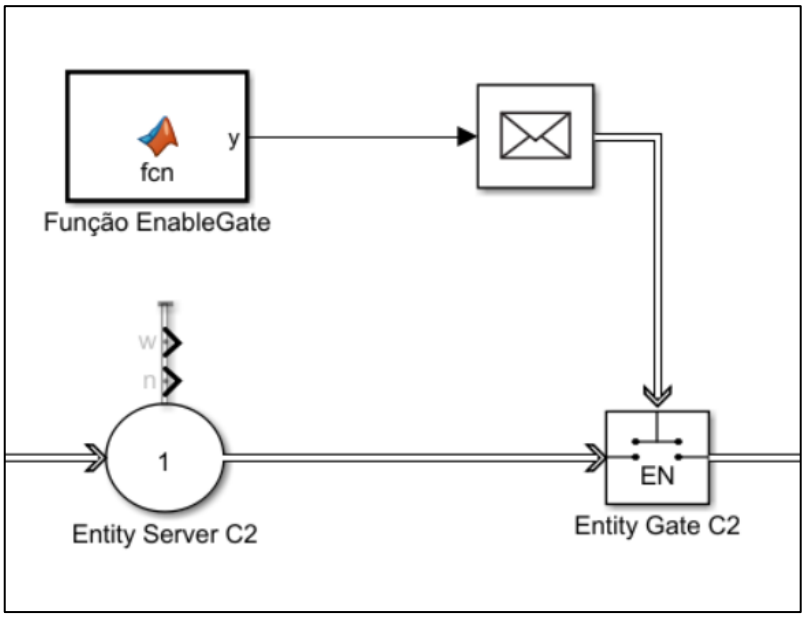


Figura 46 Função de activação da porta de controlo

Do mesmo modo, após a porta de controlo, e somente para as faixas C e D, são introduzidos novos blocos de *Entity Queue*, *Entity Server* e *Entity Gate*, similares aos anteriores e configurados com parâmetros idênticos.

Por fim, após os veículos se encontrarem na sua faixa final de destino, necessitam de ser removidos do sistema discreto e, consequentemente da simulação. Isto é desenvolvido recorrendo ao bloco *Entity Terminator*. Contudo, ao ser removida a entidade quando ultrapassa o último cruzamento, esta desapareceria nesse instante do modelo 3D. Desta forma, há que retardar e remoção e manter a dinâmica do veículo no activo, até este sair do ponto de vista do modelo.

Isto foi atingido recorrendo a um bloco *Entity Server*, onde as entidades realizam serviço durante um intervalo de tempo configurável. Após decorrido o tempo de serviço, a entidade é transferida para o bloco *Entity Terminator*, onde é removida.

O tempo de serviço configurado, como indicado na Figura 47, foi de 20 segundos. Este é o tempo necessário para o veículo ultrapassar o ponto de visão no modelo 3D e poder ser removido sem causar incidentes.

De igual forma, pode-se comprovar a invocação à função *RemoverEntidade* (Anexo C) quando existe um evento, neste exemplo a entrada de uma entidade no bloco. Esta função tem como entradas os parâmetros configurados previamente no bloco *Entity Generator* e que serviram como informação para o preenchimento da base de dados, o *Timestamp* e o *Data*.

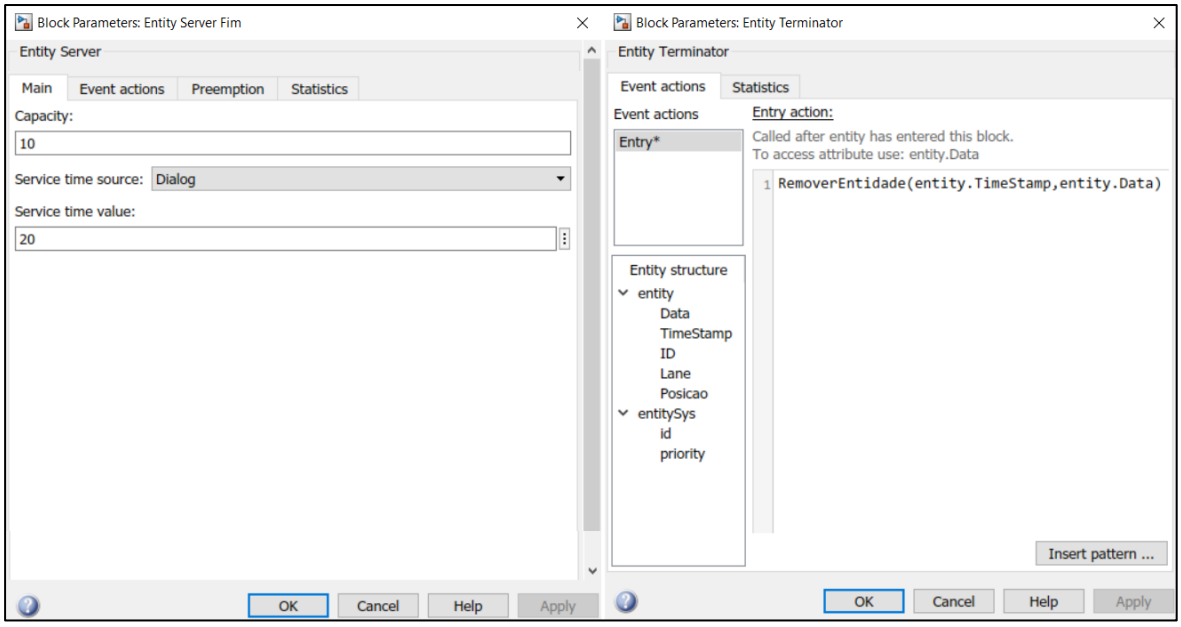
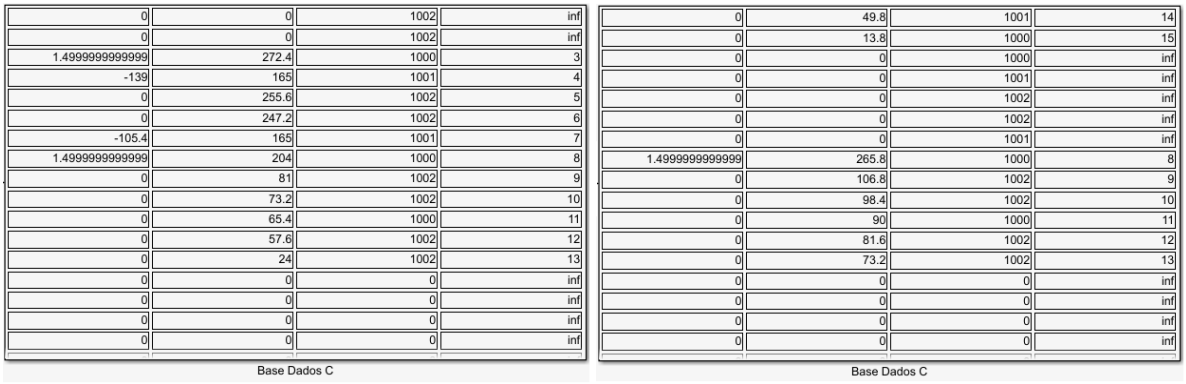


Figura 47 Entity Server Final – Parametrização de valores

O propósito desta função é o de remover a entidade, identificação e consequentemente posições e faixa associada, da base de dados. Como pode ser comprovado na Figura 48, na tabela da esquerda, a identificação do veículo 1 e 2 (linha 1 e linha 2) foi removida com recurso à função. Desta forma, assim quando surgir o veículo nº14 (tabela do lado direito), este irá ocupar o topo da tabela e sucessivamente, substituindo as entidades que vão sendo removidas do sistema discreto e modelo 3D. Também no momento em que a nova entidade é adicionada, são actualizados todos os parâmetros da respectiva linha.

Isto permite a criação de um sistema auto sustentado e renovável, capaz de uma simulação sem impedimentos de tamanho de base de dados.



0	0	1002	inf
0	0	1002	inf
1.4999999999999	272.4	1000	3
-139	165	1001	4
0	255.6	1002	5
0	247.2	1002	6
-105.4	165	1001	7
1.4999999999999	204	1000	8
0	81	1002	9
0	73.2	1002	10
0	65.4	1000	11
0	57.6	1002	12
0	24	1002	13
0	0	0	inf
0	0	0	inf
0	0	0	inf
0	0	0	inf

Base Dados C

0	49.8	1001	14
0	13.8	1000	15
0	0	1000	inf
0	0	1001	inf
0	0	1002	inf
0	0	1002	inf
0	0	1001	inf
1.4999999999999	265.8	1000	8
0	106.8	1002	9
0	98.4	1002	10
0	90	1000	11
0	81.6	1002	12
0	73.2	1002	13
0	0	0	inf
0	0	0	inf
0	0	0	inf
0	0	0	inf

Base Dados C

Figura 48 Base de dados C – remoção de entidade

Consequentemente, conclui-se o desenvolvimento do subsistema discreto de criação, gestão de eventos e remoção das entidades sendo de seguida, apresentado o tipo de controladores dos sinais luminosos de ambos os cruzamentos, recorrendo à *toolbox Fuzzy Logic*, assim como a respectiva lógica criada a partir da *toolbox Stateflow*.

Como previamente descrito, o subsistema de controladores dispõe de sinais de entrada extraídos dos blocos anteriormente mencionados *Entity Queue* e *Entity Server*. Estes, recorrendo ao parâmetro *Number of entities in block*, permitem um número correcto de veículos nas respectivas faixas a qualquer momento da simulação, visto que quando uma entidade é adicionada a este bloco, é contabilizada como entrada para os blocos do controlador. Um exemplo deste modelo pode ser analisado na Figura 49.

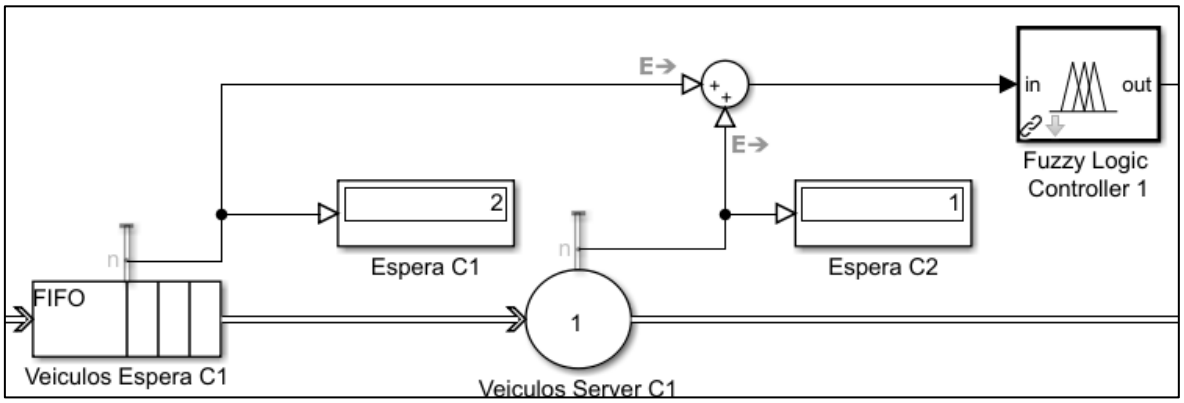


Figura 49 Dados de entrada nos controladores

Associado a cada conjunto de sinais luminosos ou a cada cruzamento, existe um conjunto de controladores a serem utilizados de forma a gerar sinais de saída. Estes sinais, como revisto no capítulo de sistemas difusos no MATLAB, dispõem de um valor ou peso, associado a cada controlador. Pode, portanto, assumir-se, que em cada cruzamento existe um conjunto de controladores, onde, de cada um se extrai o peso relativamente ao número de veículos na entrada.

Para S1, optou-se por criar um peso de veículos por cada faixa associada ao cruzamento. Como pode ser analisado na Figura 50, o Controlador 1 encontra-se associado à faixa 1, o controlador 2 encontra-se associado à faixa oposta e o Controlador 3 à faixa 4.

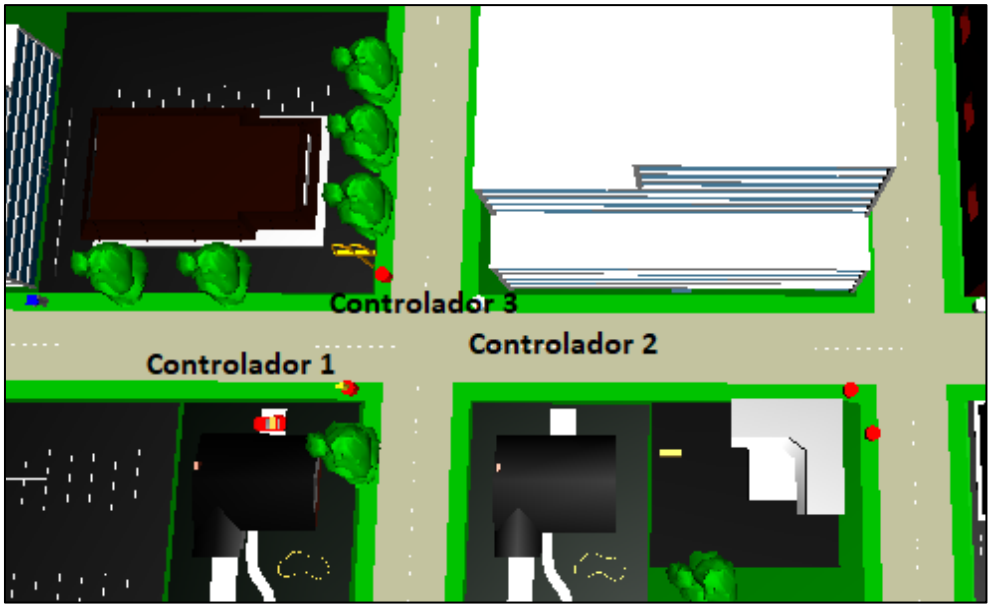


Figura 50 Associação de controladores a cada faixa de rodagem

Desta forma, o número de veículos em cada faixa é traduzido num peso aplicado a partir do controlador difuso.

Utilizando a *toolbox Fuzzy Logic* foi possível configurar, seguindo os passos descritos no capítulo 3.3, as entradas e saídas do controlador.

Começou por se configurar as variáveis de entrada e de saída, conforme se pode comprovar a partir da Figura 51.

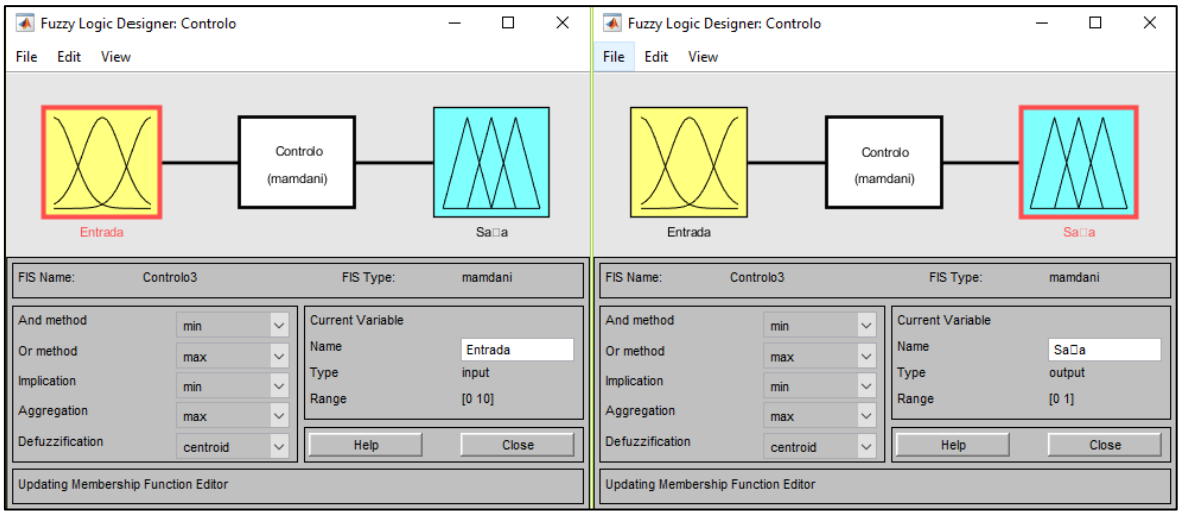


Figura 51 Configuração das variáveis entrada e saída do controlador

A variável *Entrada* foi limitada entre os valores de 0 e 10. Isto porque, segundo a configuração do bloco *Entity Queue*, o número máximo de veículos esperados é de 10.

O valor *Saída*, utilizado depois na lógica de mudança de estados a partir do *Stateflow*, tem um valor mínimo de 0 e um valor máximo de 1.

Partindo desta configuração, pode esperar-se que, em cada faixa, quando não existem carros pendentes em fila de espera, o valor de saída será próximo de 0. Contudo, num momento de congestionamento, em que cada unidade atinja o valor máximo de 10 veículos, é expectável que o valor da saída se altere para perto do máximo, consequentemente, do valor 1.

De seguida, configura-se as funções de associação para cada uma das variáveis de entrada e saída. Com o propósito de aumentar a eficácia do sistema, foram configuradas 5 funções de associação, como pode ser revisto na Figura 52:

- A função *Nenhum* indica a inexistência de veículos e a não necessidade em alterar o sinal luminoso do cruzamento.
- A função *Poucos* significa que o número de veículos não chega a metade do número permitido.
- *Alguns* indica que existe uma acumulação e o sinal somente não é alterado se existir mais congestionamento nas faixas opostas.

- *Muitos* e *Maximo* exprime a necessidade urgente da alteração do sinal luminoso pois verifica-se um aumento significativo na criação de entidades nos blocos de entrada, o que reflete um aumento do fluxo de veículos.

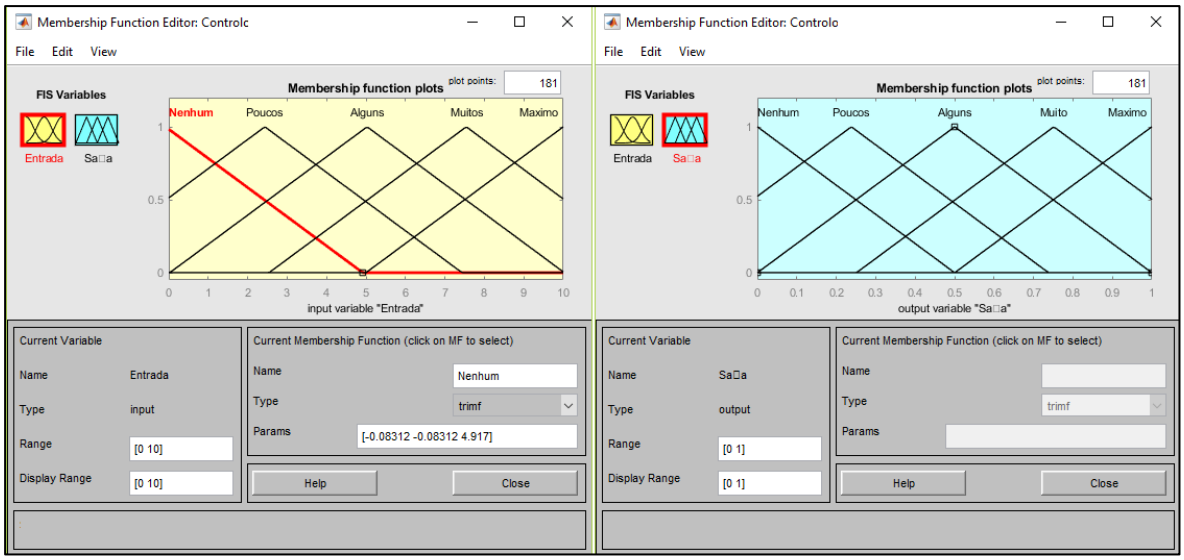


Figura 52 Parametrização das funções de associação

Existindo a necessidade de criar as regras, as entradas e as saídas são relacionadas da seguinte forma:

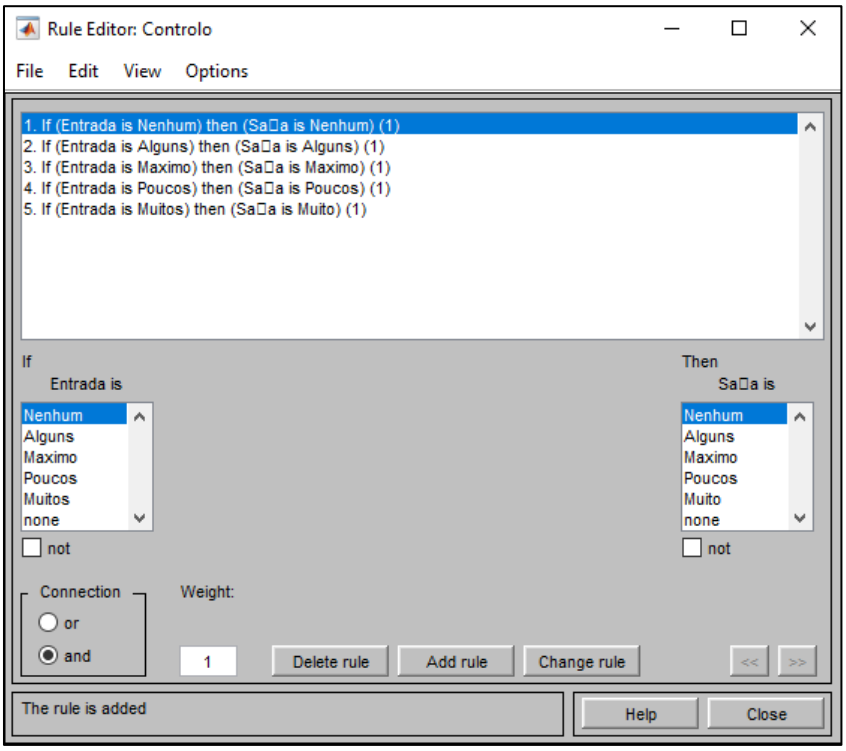


Figura 53 Parametrização das regras

Por fim, como se pode comprovar a partir do gráfico de regras, para um número de veículos igual a 2, a saída respectiva é 0.381. Contrariamente, quando existe um fluxo superior, a saída correspondente é 0.616.

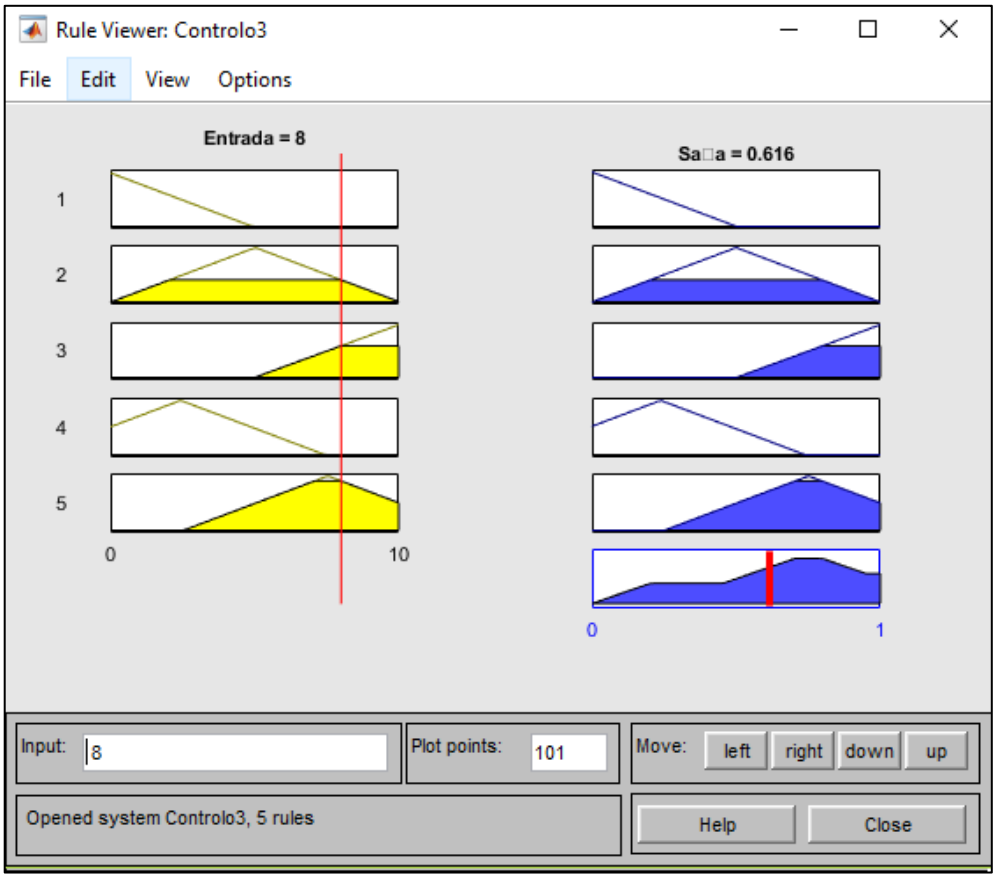


Figura 54 Gráfico de regras

O valor de saída do bloco controlador permite assim aceder a um sinal ponderado pronto a ser utilizado na gestão de lógica a comutar os sinais luminosos. Após isto, uma função do MATLAB é utilizada para gerar um número binário na entrada do bloco do *Stateflow*.

Desta forma, na entrada deste bloco é esperado:

- $x = 0$, se o peso do número de veículos na faixa 4 for superior a ambas as restantes.
- $x = 1$, se o peso do número de veículos na faixa 1 ou na faixa oposta for superior à faixa 4.

Contudo, apesar de o controlador indicar qual o peso associado a cada faixa, o sistema não dispõe da restante informação onde o comportamento possa ser representado num sistema real, como por exemplo a comutação de um sinal vermelho para um sinal amarelo ou um

tempo de espera entre a passagem dos veículos durante os cruzamentos. Por conseguinte, de seguida é descrita a lógica utilizada a partir do sistema, de forma a comutar as variáveis globais S1 e S2, destinadas a alterar a cor dos respectivos sinais no modelo 3D e no controlo da dinâmica de veículos na subsecção 4.3.

Com referência aos objectos de estado da Figura 55 e considerando que a faixa 1 tem um maior número de veículos e, consequentemente, um maior peso no valor de saída do controlador difuso, a lógica traduz-se em:

- Faixa1_4_1 – A saída tem o valor de 1. A faixa 1 e a faixa opostas têm o sinal luminoso verde e é permitida a passagem a estes veículos. Os veículos da faixa 4 aguardam com o sinal vermelho. A cada 3 segundos de simulação é verificada a alteração do sinal de entrada.

- Faixa1_4_2 – O sinal de entrada foi alterado para 0, significando um maior tráfego de veículos na faixa 4. Após 1 segundo de simulação, o objecto de estado é alterado.

- Faixa1_4_3 – A saída do bloco do *Stateflow* tem agora o valor 2, significando que os sinais luminosos anteriormente a verde, agora se encontram a amarelo. A faixa 4 mantém o sinal luminoso a vermelho. Após 3 segundos de simulação, o objecto de estado é alterado.

- Faixa1_4_4 – A saída é alterada para o valor 4, traduzindo-se num sinal luminoso vermelho para todas as faixas.

- Faixa4_1 – Após 3 segundos de simulação, permitindo um intervalo de tempo aos veículos para ultrapassarem o cruzamento, o sinal da faixa 4 é comutado para verde, mantendo-se os restantes a vermelho.

- Faixa4_2 - O sinal de entrada foi alterado para 1, significando um maior tráfego de veículos na faixa 1 ou oposta. Após 1 segundo de simulação, o objecto de estado é alterado.

- Faixa4_3 - A saída do bloco do *Stateflow* tem agora o valor 3, significando que o sinal luminoso anteriormente a verde se encontra agora a amarelo. A faixa 1 mantém o sinal luminoso a vermelho. Após 3 segundos de simulação, o objecto de estado é alterado.

- Faixa4_1 - A saída é alterada para o valor 4, traduzindo-se num sinal luminoso vermelho para todas as faixas.

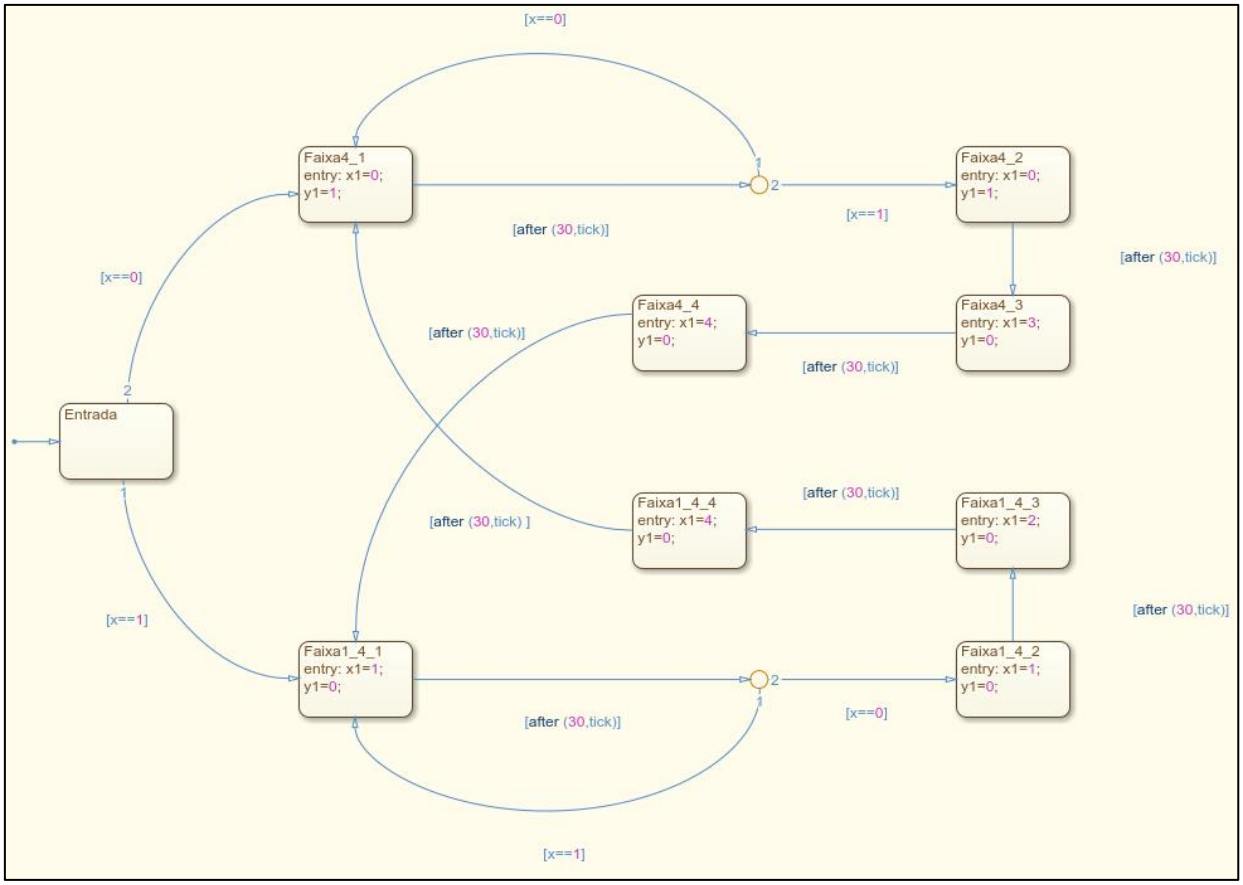


Figura 55 Fluxograma da lógica de controlo difuso – S1

Conforme o fluxograma anterior é actualizado, modificando o valor de saída, este é gravado para a variável global S1, sendo assim utilizada em diferentes pontos do modelo.

Para efeitos de teste do modelo, foi criado um outro controlador, funcionando em simulações paralelas ao controlador difuso, onde os sinais luminosos são comutados periodicamente. Deste modo, é possível comprovar a eficácia do sistema difuso, comparativamente a outro baseado em intervalos de tempo regulares.

Este tipo de comutação é feita recorrendo ao bloco *Slider Switch*, associando ambas as extremidades à constante acoplada ao bloco de *Switch*, como comprova a Figura 56. Desta forma, a selecção na extremidade esquerda impõe o valor de 1 ao bloco constante, permitindo a passagem do valor de saída do bloco da lógica difusa. Contrariamente, quando o utilizador seleccionar a outra extremidade, é activado o controlador temporal.

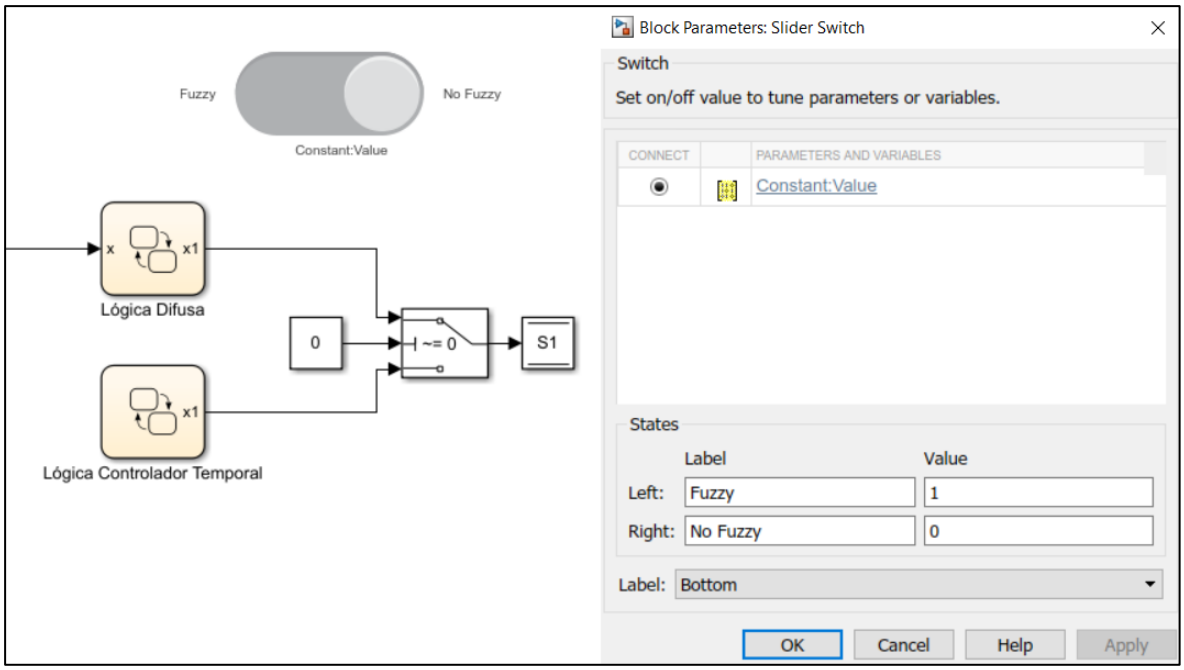


Figura 56 Bloco de comutação e diagramas de lógica

Esta lógica de sinais luminosos controlada com recurso a um tempo fixo é bastante utilizada em locais onde não seja justificada a instalação de outros recursos, como dispositivos que auxiliem um controlador difuso. Por esta razão, foi utilizada uma lógica fixa de 15 segundos de duração do sinal luminoso para cada uma das vias. Desta forma, quando seleccionado o controlador temporal, os sinais luminosos comutam perante um fluxograma diferente, num sistema alternável de cor verde e vermelha, a cada 15 segundos.

Este espaço temporal pode ser alterado à medida das necessidades, como se exemplifica de seguida.

Tendo por consequência a faixa 1 com o sinal luminoso verde e com referência à Figura 57, a lógica resulta em:

- Faixa1_1 – Atribuído o valor de 1 na saída do bloco, comutando para verde o sinal da faixa 1. Após 15 segundos de simulação o objecto de estado é alterado.

- Faixa1_2 – Valor de 2 na saída do bloco, permitindo a comutação para o sinal amarelo durante 3 segundos de simulação.

- Faixa1_3 – Valor de 4 na saída do bloco, alterando para vermelho todos os sinais luminosos, mantendo-se assim durante 3 segundos.

- Faixa1_4 – Valor de 0 na saída do bloco, comutando para verde o sinal da faixa 4 durante 15 segundos.

- Faixa1_5 – Alteração para amarelo durante 3 segundos, seguindo-se o objecto de estado em que coloca todos os sinais luminosos a vermelho, repetindo-se o ciclo.

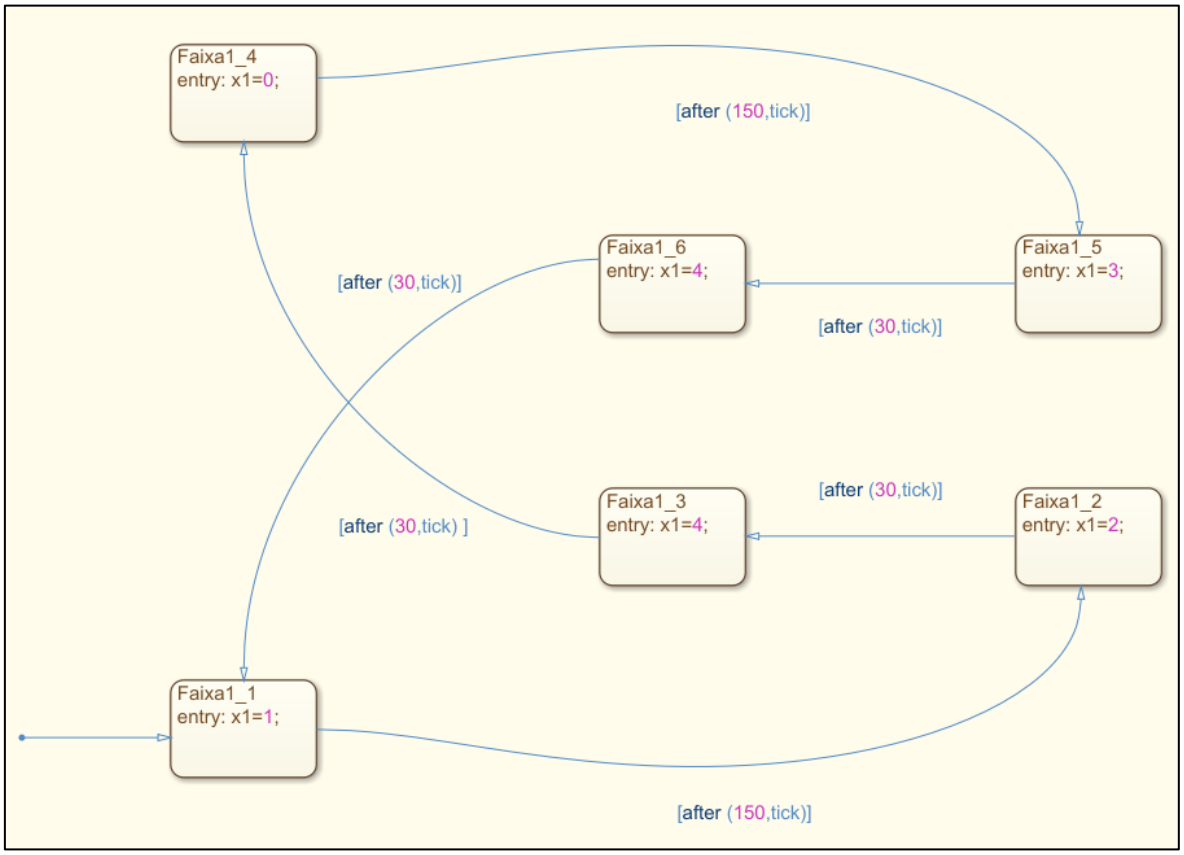


Figura 57 Fluxograma da lógica de controlo temporal

De forma análoga, o cruzamento S2, é controlado por lógica difusa ou temporal, permitindo ao utilizador a comutação, existindo o tempo de espera do sinal amarelo e o tempo permitido para os veículos ultrapassarem o cruzamento sem antes os sinais luminosos transitarem de cor.

Existem, contudo, duas diferenças na modelação do cruzamento, resultando num comportamento e lógicas distintas:

- Os carros da faixa 1, ao ser permitido a sua mudança de direcção, obriga à utilização de três comportamentos distintos dos sinais. De entre os sinais luminosos das faixas 2, 3 e 7, só um poderá estar activo e nunca dois em simultâneo.

- A faixa 3, tendo prioridade sobre qualquer outra, obriga à comutação antecipada de qualquer um dos outros sinais, de forma a existir prioridade para esta via.

Consequentemente, a lógica utilizada em C1 foi modificada de forma a ter em conta o número de veículos urgentes na faixa 3. Isto foi possível devido ao separador de *Statistics* nos blocos *Entity Queue* e *Entity Server* existentes na faixa 3 e que servem de entrada para o bloco de lógica de uma função do MATLAB. Como pode ser analisado na Figura 58, a lógica do *Stateflow* tem como entrada um sinal manipulado. Nesta função do MATLAB, sempre que existir um número positivo em espera na faixa 3, a saída é alterada para um número predeterminado. Desta forma, é possível desenvolver lógica com o *Stateflow* de modo a existir uma transição de estado sempre que a variável de entrada for alterada.

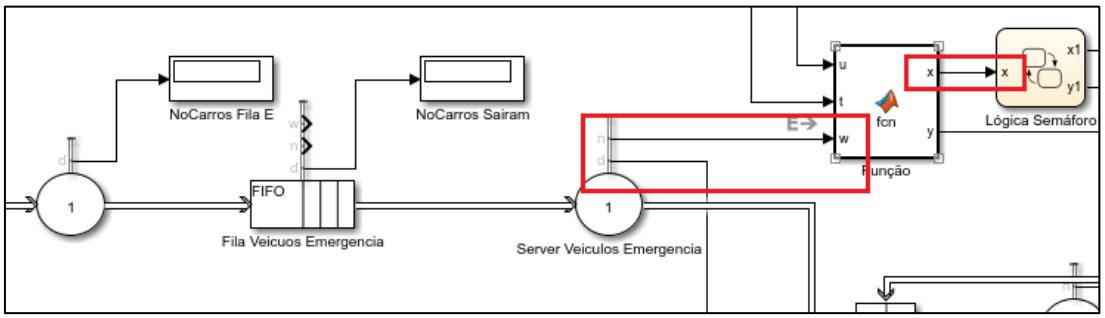


Figura 58 Lógica utilizada em veículos prioritários

Como pode ser comprovado na Figura 59, a cada alteração de estado, é verificado se a entrada corresponde ao valor 2 (Faixa1_1), comutando de seguida para o sinal amarelo (Faixa3_1) e vermelho (Faixa3_2) na via onde os carros se estavam a movimentar, permitindo a passagem com o sinal verde (Faixa3_3) ao veículo de emergência na faixa 3.

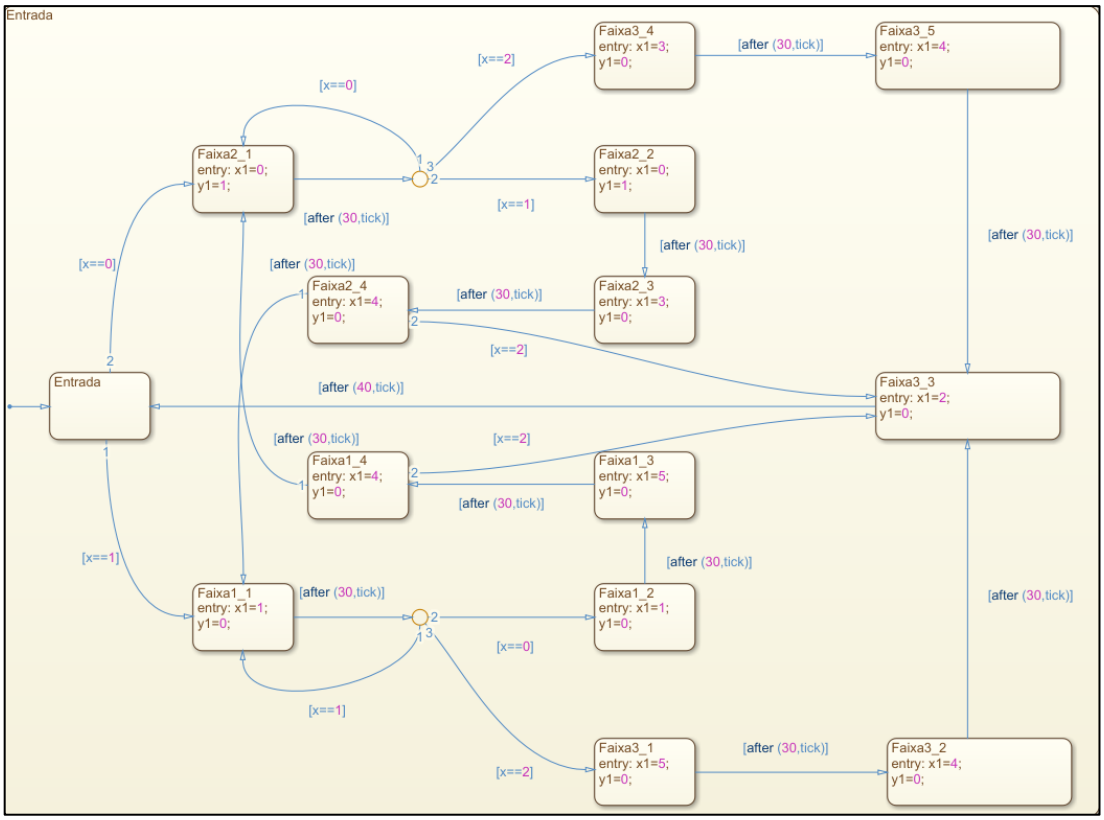


Figura 59 Fluxograma da lógica de controlo difuso – S2

Após a passagem do veículo, o sinal luminoso fica pendente de alteração até não existirem mais veículos nessa faixa de emergência. Nesse momento, é alterado o objecto de estado para *Entrada* e o ciclo repete-se novamente.

Deste modo, o sistema discreto de simulação de veículos fica completo, permitindo a criação, parametrização e remoção das entidades, actuação dos controladores perante o número de veículos em cada via e lógica associada à comutação dos sinais luminosos.

4.3. DINÂMICA DE VEÍCULOS

Com o objectivo de movimentar os veículos segundo as posições destes e as vias aleatoriamente escolhidas, este conjunto de subsistemas permite modelar a velocidade e a paragem dos veículos.

Como descrito anteriormente na Figura 34, existem 4 subsistemas de dinâmicas, cada um destes associado aos veículos criados em cada faixa e tendo em conta as regras previamente descritas, tais como veículos C e D, entre outros, são gerados na via 1 e 2, respectivamente.

As diferenças entre estes subsistemas reside na lógica utilizada para os veículos ultrapassarem os cruzamentos. De tal forma, irá, de seguida, ser descrito o subsistema C, correspondente aos veículos gerados na via 1, visto ser este o mais completo e abrangente.

Estes sistemas contêm uma ligação permanente à base de dados dos veículos correspondentes, com o auxilio dos blocos *Data Store Read* e *Data Store Write*. Como pode ser analisado na Figura 60, o subsistema C contem como entradas a base de dados C, o valor das variáveis dos semáforos S1 e S2 e o correspondente destino dos veículos na variável *Lane*. De igual forma, após serem validadas as posições dos veículos e autorizada a sua velocidade, estas informações são escritas para a base de dados, a partir do *Data Store Write*, de forma a ser utilizada a partir do modelo 3D.

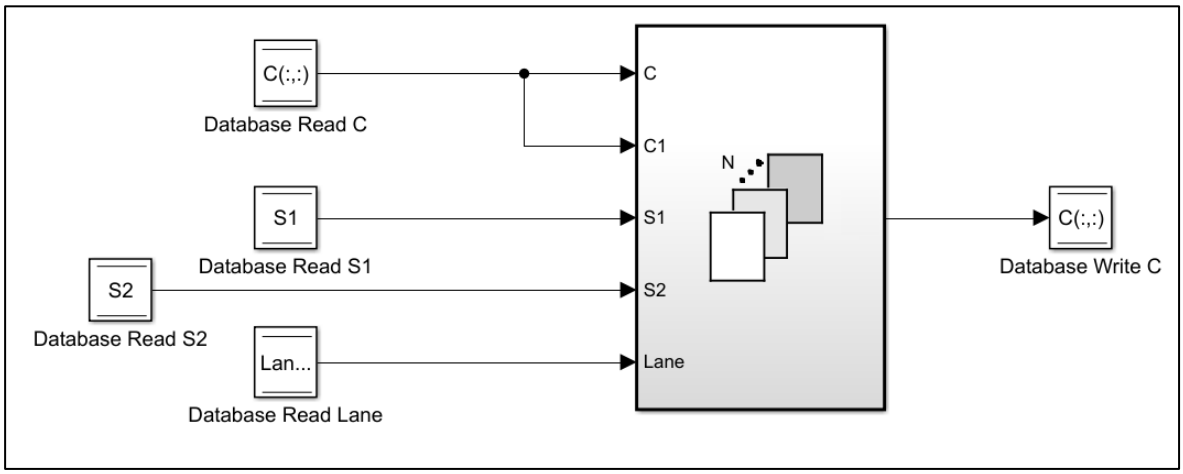


Figura 60 Entradas e saídas do subsistema

No interior deste sistema é utilizado o bloco *For Each* para individualizar a matriz C em veículos individuais. Como se pode analisar na Figura 61, a partir da matriz C são criados subsistemas paralelos com o mesmo comportamento, visto que todos os carros possuem as mesmas características, apenas diferindo nos valores das posições de cada veículo. Desta forma, é possível dividir a matriz C, composta por 20 linhas e 4 colunas (20x4) em 1x4, criando 20 subsistemas similares.

Também permite a separação, com recurso a desmultiplicador, das posições nos diversos eixos dos veículos, assim como a identificação.

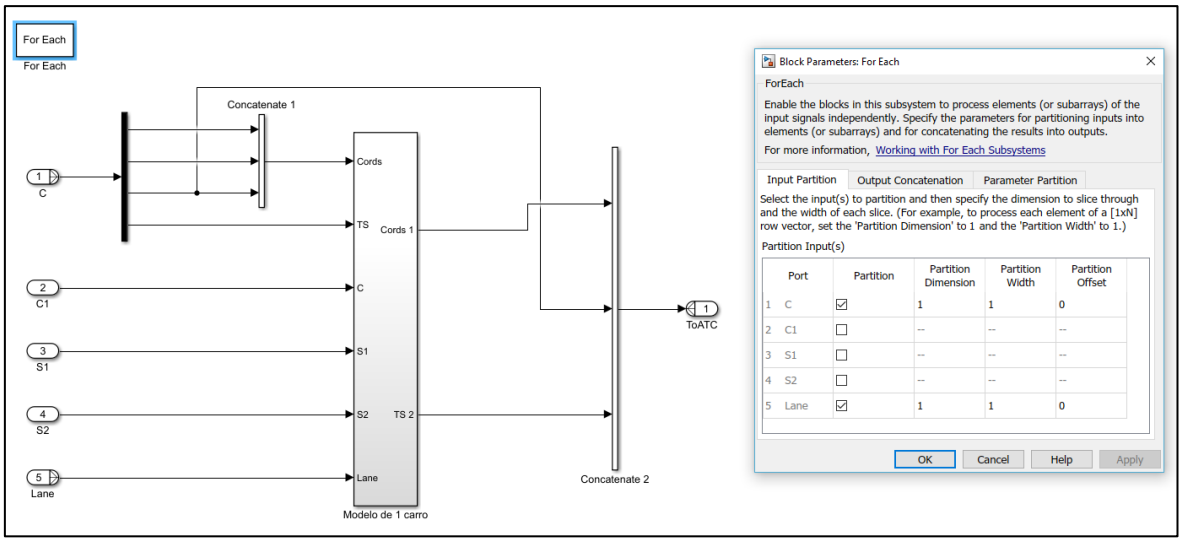


Figura 61 Individualização da matriz C com recurso ao For Each

As posições individuais, assim como a identificação associada, permite gerar um sistema de colisões entre veículos do mesmo sistema. Isto é atingido recorrendo a uma lógica gerida a partir do *Stateflow*, onde as posições de todos os veículos da matriz são comparadas com a de um veículo em particular. Desta forma é permitida a comparação entre, por exemplo, o veículo 2 e o veículo 10, afastando a possibilidade de uma colisão entre qualquer veículo da base de dados. Este cenário assemelha-se ao de um condutor que para se movimentar analisa a posição dos outros veículos.

Com referência à Figura 62, pode analisar-se a lógica de colisões presente para os veículos da via 1:

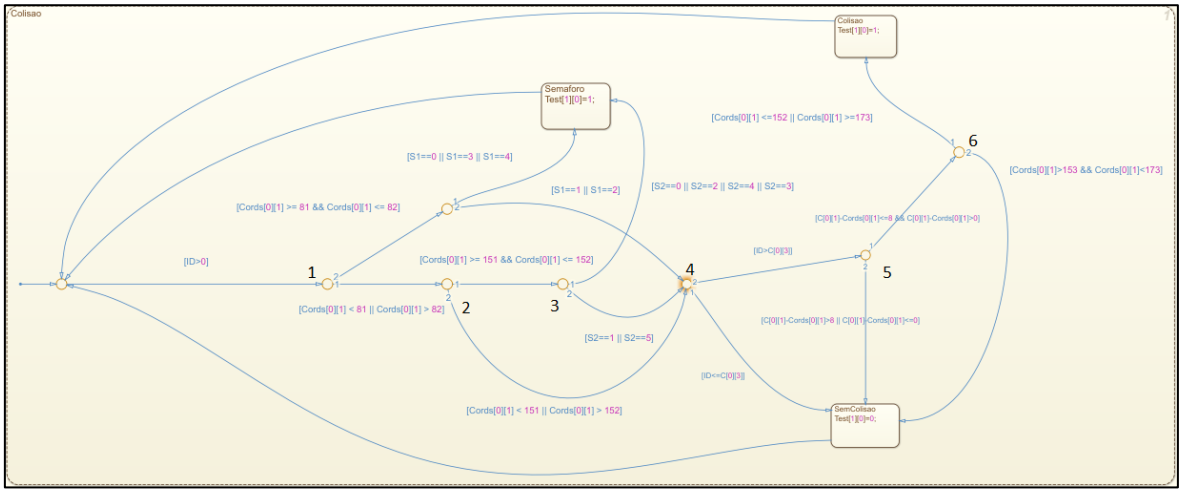


Figura 62 Diagrama de colisões do sistema

- A condição inicial para a entrada neste fluxograma é a necessidade da identificação do veículo ser superior a 0, representado como *ID>0*. Desta forma, a lógica só é considerada para os veículos já inseridos na base de dados.

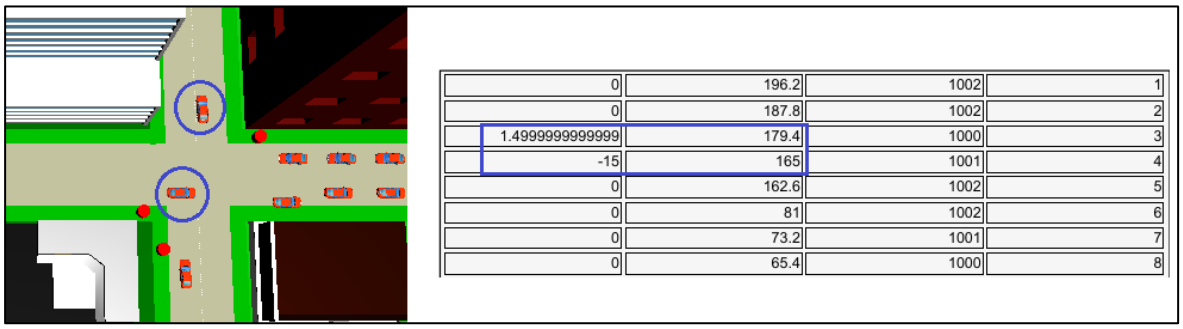
- De seguida, e considerando que o sinal luminoso se encontra na posição x== 83, é efectuada uma verificação da posição do veículo. Ao encontrar-se próximo do sinal, estando este em condições que obriguem à paragem do veículo (vermelho equivale a S1==0, S1==3 ou S1==4), é detectada uma colisão e enviado o sinal de saída equivalente (Colisão == 1), no objecto de estado intitulado *Semaforo*.

- Se o veículo não se encontrar na proximidade de S1 ou de S2 significa que o único obstáculo que possa impedir a sua velocidade é a proximidade de outro veículo.

  De tal forma, na junção 4, é comparada a identificação do veículo com todos os outros do mesmo sistema. No caso de a identificação ser inferior a qualquer outra, significa que este é o primeiro veículo da via, não existindo impedimentos para este prosseguir. Caso contrário, é verificada a posição deste comparativamente com as restantes.

- Na junção 5, encontrando-se o veículo a mais de 8 unidades de distância de qualquer outro, não existe colisão no objecto, passando o estado a ser *SemColisao*.

- Na junção 6, estando o veículo a menos de 8 unidades de distância de outro, é verificada a posição vertical, pois pode um veículo estar em movimento na via 7 e outro em direcção à via 5. Apesar da posição horizontal ser próxima, a posição vertical e, consequentemente, o afastamento dos veículos não deve ser motivo de paragem. Um exemplo pode ser analisado na Figura 63.



0	196.2	1002	1
0	187.8	1002	2
1.4999999999999	179.4	1000	3
-15	165	1001	4
0	162.6	1002	5
0	81	1002	6
0	73.2	1001	7
0	65.4	1000	8

Figura 63 Exemplo das diferenças entre posições

Assim, na saída do diagrama de colisões, o sistema terá um valor pré-definido de 1 se existir uma colisão para o veículo ou 0 se não existir colisão. Consequentemente, este valor servirá para permitir ou negar a activação da velocidade para o veículo de cada subsistema.

O sinal de saída é adicionado a um bloco de *Switch,* que também contem outros blocos, nomeadamente o da velocidade. Como comprovado na Figura 64, quando o aviso de colisão tiver o valor positivo, a saída do bloco *Switch* é utilizada a partir da primeira entrada. De forma contrária, quando existir uma colisão, o valor 0 é servido como input de velocidade, que, posteriormente integrado, não alterará a posição.

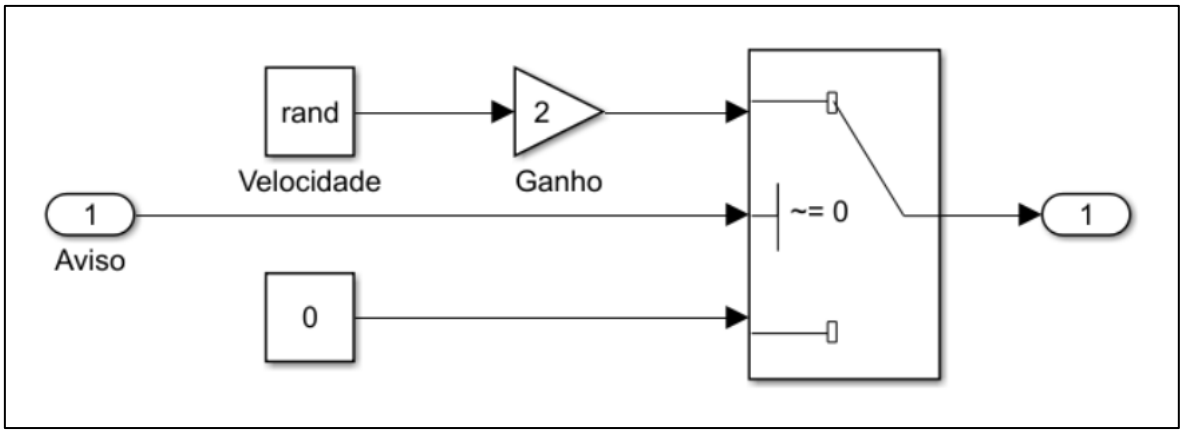


Figura 64 Saída do diagrama e comparação com velocidade

Estando, desta forma, analisado o sistema de colisões e da lógica associada a permitir a velocidade aos veículos de cada sistema, resta explorar o cálculo da nova posição, no caso dos veículos da via 1 alterarem a sua trajectória no cruzamento 2 e a consequente actualização da base de dados.

Isto é atingido com recurso ao uso de um novo diagrama de lógica do *Stateflow*, uma função MATLAB e de um bloco de *Data Base Write*. A Figura 65 ilustra o mencionado.

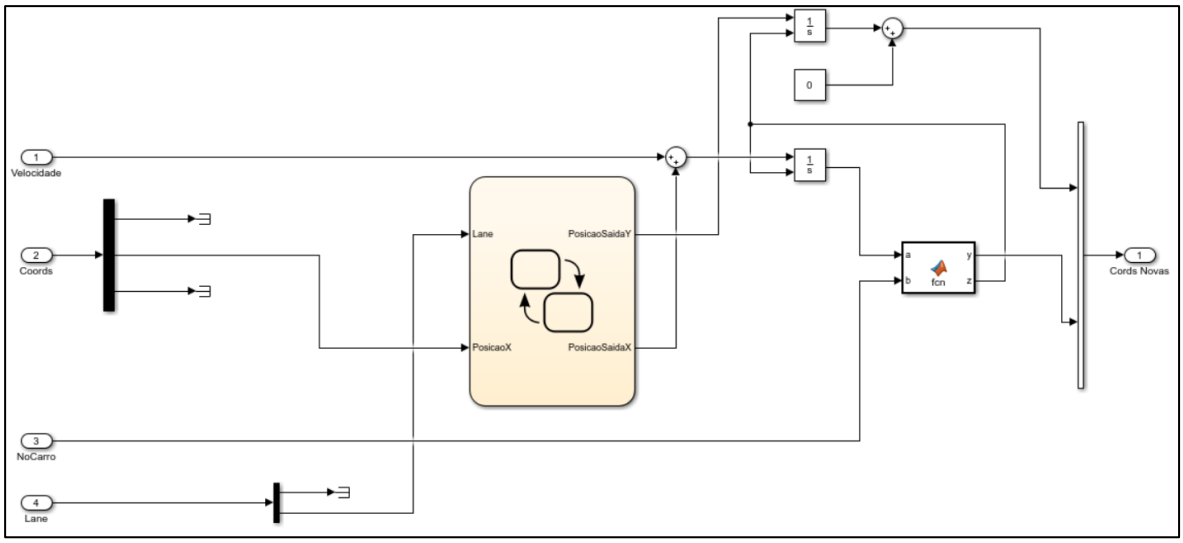


Figura 65 Atualização da velocidade/posição para a base de dados

Durante o cruzamento 2, conforme visto na introdução do Capítulo 0, os veículos C têm a opção de prosseguir na via onde se encontram, virar à esquerda ou à direita. Deste modo, é necessário existir uma actualização da posição em xx e em yy, visto que neste modo existe variações, durante o cruzamento, em ambos os eixos.

A entrada 2 da Figura 65 contém a posição actual do veículo e serve como ponto de entrada ao diagrama do *Stateflow*. Nesta situação, é utilizado o desmultiplexador e considerado o valor da posição no eixo horizontal (eixo de xx). De igual forma é adicionada a via do veículo em causa, a partir da entrada 4.

Este diagrama encontra-se dividido em 4 caixas de agrupamentos, onde cada veículo pode somente aceder a uma durante o tempo que se encontra no sistema. Estas partes são:

- Se a via possuir o valor *Missing*, implica que o veículo não se encontra inicializado no sistema, logo não deverão existir alterações na velocidade e posterior actualização na base de dados. Desta forma, o valor de saída em ambos os eixos deve ser 0, como representado na Figura 66. De notar a importância da ligação de volta à junção inicial. Isto assegura que o subsistema do veículo não fica retido no objecto de estado e pode voltar a ser reutilizado para qualquer das outras caixas de agrupamentos, onde a via do veículo já se encontra escolhida.

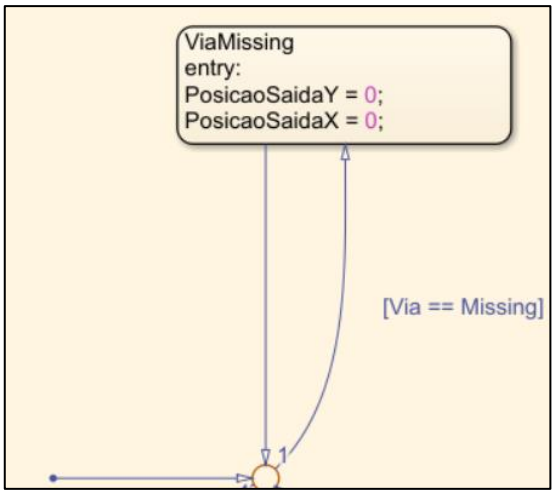


Figura 66 *Stateflow* – Sem alterações na saída

- No caso do veículo prosseguir para a via 5, significa que este continua o seu caminho sem alterar a trajectória no eixo vertical. Contudo, durante a simulação, e antes de os veículos se aproximarem do sinal luminoso S2, existe um desvio a nível vertical, indicando se estes irão mudar de direcção ou continuar na mesma via, como ilustra a Figura 67.

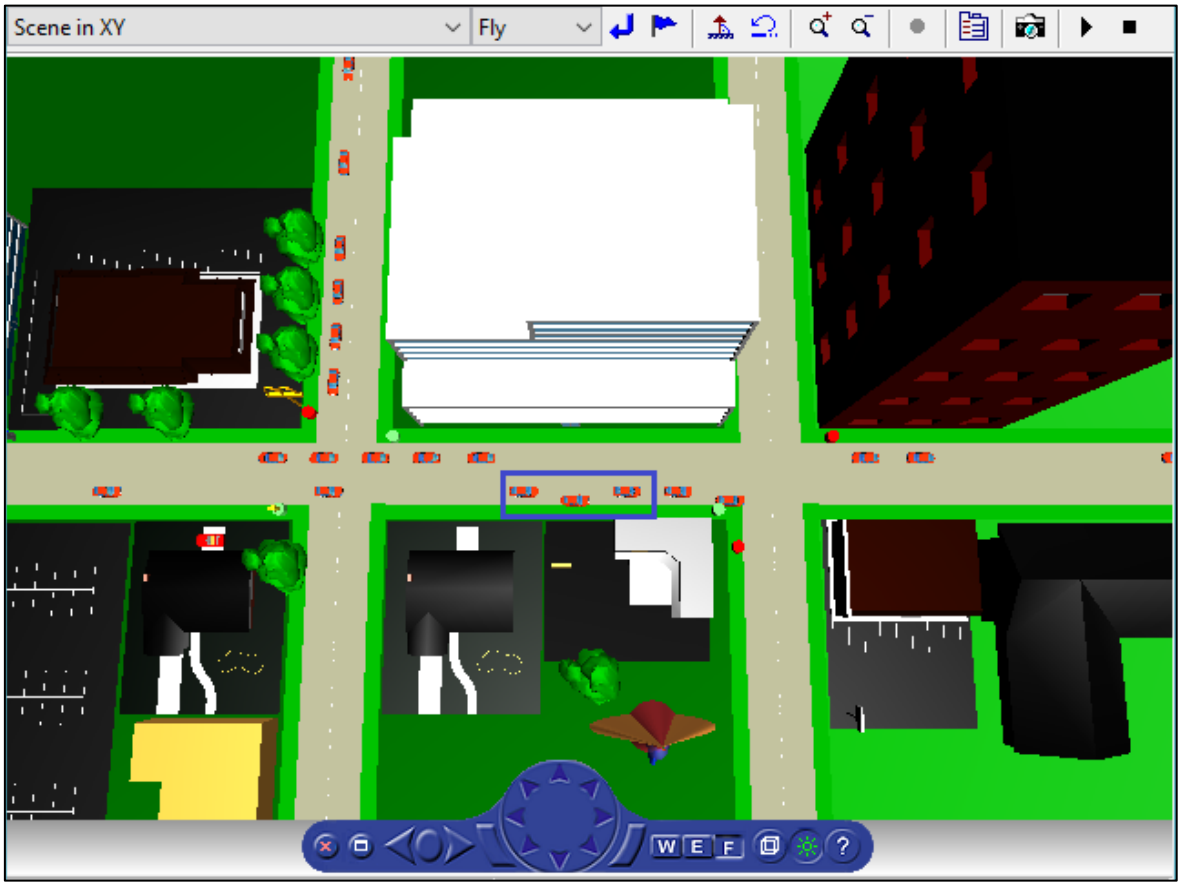


Figura 67 Antecipação da mudança de direcção

Deste modo, a velocidade de saída é ligeiramente alterada com o intuito de fornecer uma alteração à posição, após a velocidade ser integrada fora deste bloco de lógica (objecto de estado *AlteracaoPosicao*).

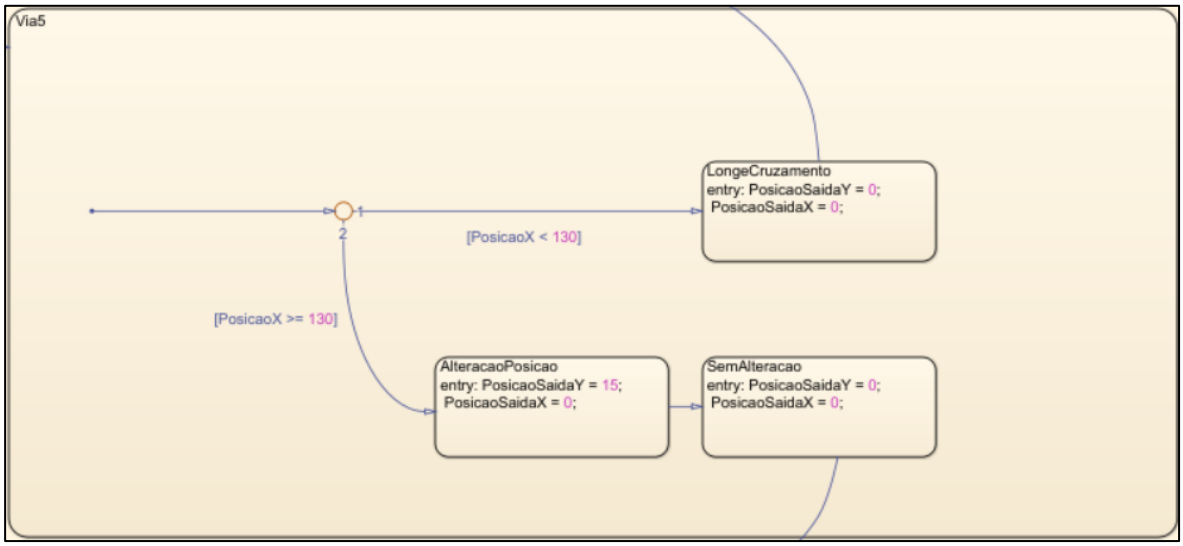


Figura 68 *Stateflow* – Via 5

- No caso do veículo optar por fazer uma mudança de direcção à esquerda, dirigindo-se para a via 7, o comportamento é similar até após o cruzamento. Consequentemente, na mudança de direcção, é alterada a velocidade vertical (objecto de estado *MudancaDirecao*), seguido por uma velocidade constante, enquanto o veículo se move na vertical na via 7.

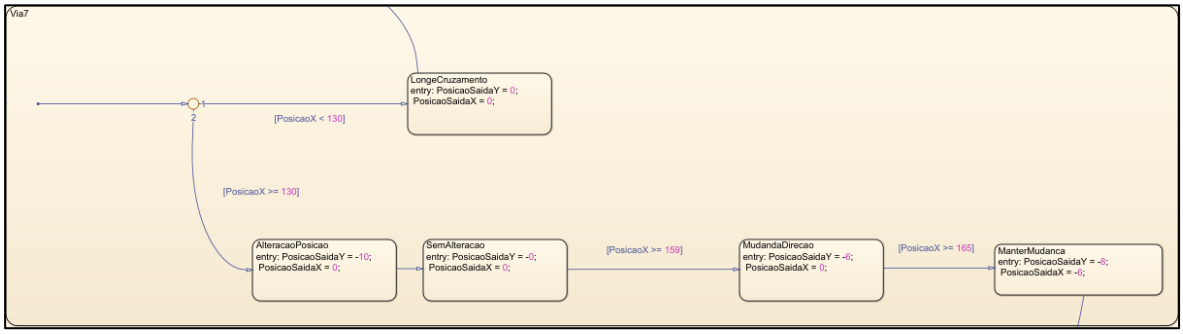


Figura 69 *Stateflow* – Via 7

De forma análoga à anterior, quando o veículo efectua a mudança de direcção à direita, este comporta-se de forma semelhante, contudo os valores inferidos à velocidade e, consequentemente, à posição, são opostos.

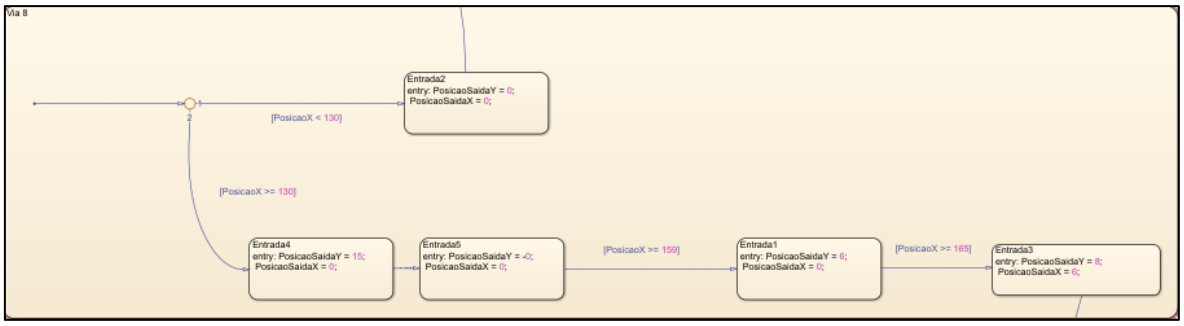


Figura 70 *Stateflow* – Via 8

De notar a necessidade de sempre retornar uma ligação no final do fluxo à junção inicial. Caso contrário, quando esse veículo fosse removido e, consequentemente, substituído por outro mais recente, a lógica associada à via poderia não ser realizada.

A entrada 1 (velocidade), sinal de saída da Figura 64, e representante da posição X do veículo C, é adicionada com o sinal de saída do diagrama do *Stateflow*, sendo integrada, resultando na posição final do veículo.

Após isso, foi adicionada uma função MATLAB à saída do integrador de forma a assegurar o restabelecimento da posição original na base de dados quando um veículo é removido e outro o substitui. Se esta função não fosse desenvolvida, a posição inicial do novo veículo teria como valores a posição final do veículo anterior, o que levaria o novo a iniciar fora do modelo 3D.

Por fim, as posições actualizadas são concatenadas e inseridas na base de dados, com recurso a um *Data Store Write*, como já analisado na Figura 60.

4.4. SISTEMA 3D ANIMATION

O subsistema da modelação virtual encontra-se dividido em sub-blocos, cada um representando uma gama de veículos (C, D, E e F) e um outro para os sinais luminosos (S1 e S2). A função destes sistemas é a de receber a actualização dos valores nas base de dados e inseri-las no modelo virtual, tal como se demonstra na Figura 71.

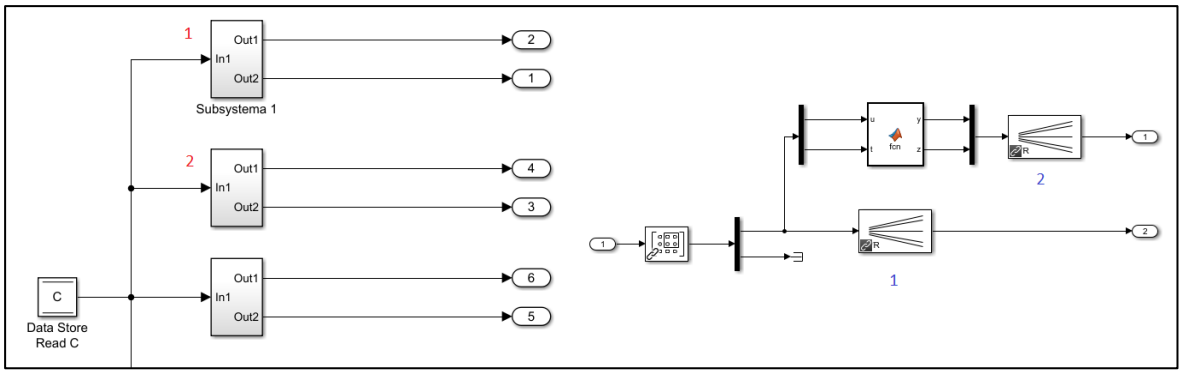


Figura 71 Sub Sistema 3D – Leitura da base de dados e envio de sinal

Nesta imagem podemos analisar a leitura da base de dados C. Cada linha, correspondente a um veículo é introduzida num Subsistema (identificado à esquerda). Cada subsistema contém os mesmos blocos constituintes como uma submatrix, dois blocos *VR Expander* e uma função MATLAB, como identificado à direita:

- O objectivo da submatrix é o de seleccionar os elementos pretendidos da base de dados C. Neste caso, é configurada para retirar a primeira linha, significando o primeiro veículo. No subsistema 2 (identificado a vermelho), a submatrix estará configurada para conter os dados da segunda linha da base de dados.

- Os *VR Expander* são utilizados para agrupar a informação recebida. No VR Expander 1, identificado a azul, é agrupada a posição no eixo dos xx e no eixo dos yy, utilizada para actualizar o modelo 3D. O *VR Expander* 2 é utilizado para associar os valores de rotação do veículo, recebidos a partir da função MATLAB que a antecede.

- No caso da função MATLAB, tem como dados de entrada as posições absolutas dos veículos e, consoante estas ascendam a determinado valor, alteram a rotação do veículo. Esta função tem como finalidade a alteração da rotação do veiculo nos casos em que este transita com uma mudança de direcção à esquerda ou à direita, durante o cruzamento 2.

Na saída deste subsistema, o utilizador dispõe das novas posições e rotação a ser consideradas e a actualizar no modelo 3D, tal como demonstra a Figura 72.

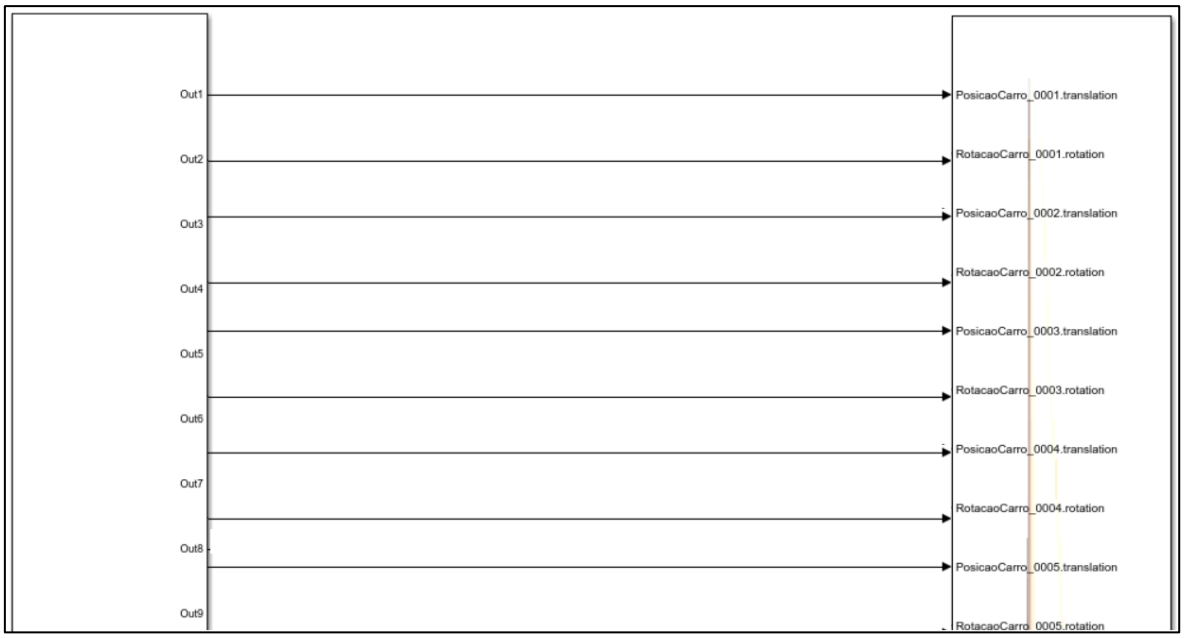


Figura 72 Ligação entre as actuais posições e o VR Sink (modelo 3D)

Do lado esquerdo encontra-se o subsistema previamente descrito e do lado direito o um excerto do bloco VR Sink, permitindo o interface entre o *Simulink* e o modelo.

Do mesmo modo que se encontra desenvolvido um subsistema para os veículos C, existem subsistemas similares para as restantes base de dados de veículos D, E e F.

Resta contudo associar a comutação de sinais luminosos da base de dados à mudança de ambiente no modelo 3D, conforme ilustra a Figura 73. De forma análoga ao anteriormente descrito, o valor da variável é lido da base de dados S1 e S2 e, com o auxilio de uma função MATLAB (Anexo F), associada à cor respectiva dos sinais para o modelo. O parâmetro cor, associado ao objecto *Semaforo* no *Simulink 3D Animation* contém três variáveis, sendo estas configuráveis e alteráveis a partir do *Simulink*. O conceito desenvolvido baseia-se em ler a variável global S1 e S2, actualizadas a partir do controlador no sistema discreto na Figura 55 e Figura 59 e alterar a saída consoante a cor pretendida, onde os vectores:

- [1 0 0] – corresponde à cor vermelha.
- [0.4 1 0.4] – corresponde à cor amarela.
- [1 1 0] – corresponde à cor verde.

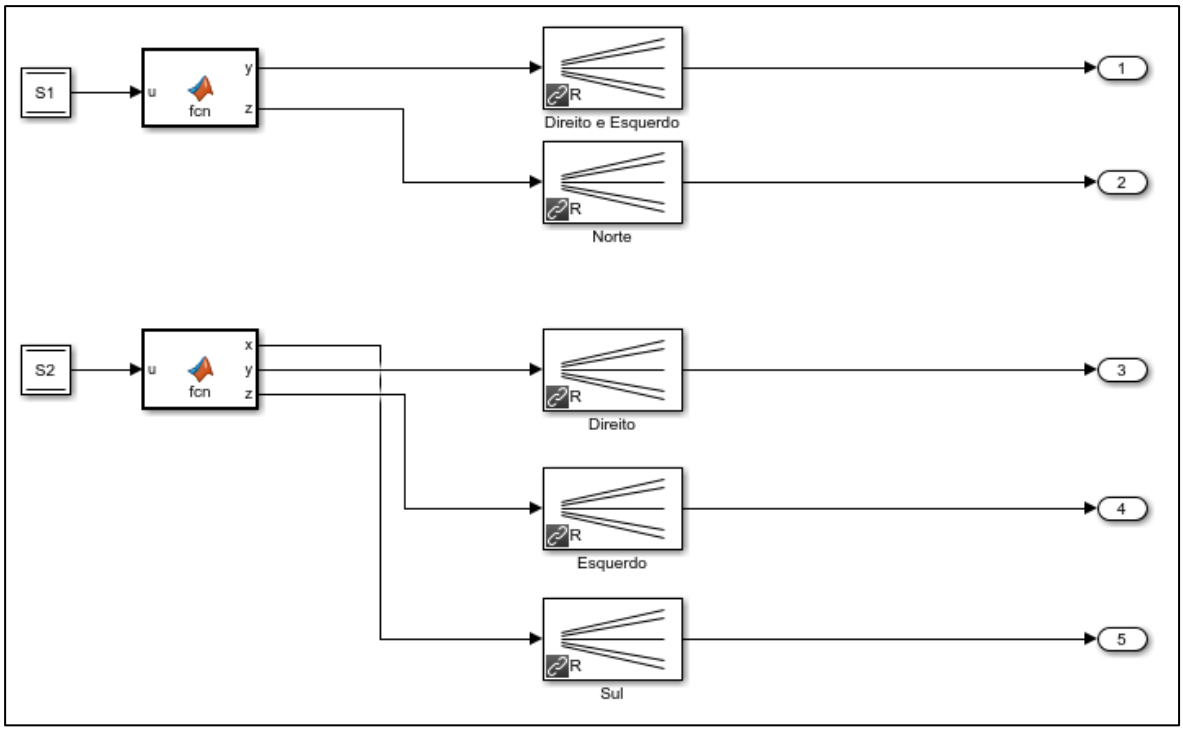


Figura 73 Sistema 3D Animation – Comutação de sinais luminosos

4.5. DESENVOLVIMENTO DO MODELO 3D

Durante o desenvolvimento do projecto recorreu-se à criação de um modelo virtual de forma a comprovar e testar todo o sistema. Neste modelo, as figuras geométricas foram construídas recorrendo a diversos objectos em formato .STL, importadas para o modelo.

De seguida irá ser feito um breve resumo das partes mais relevantes na criação deste modelo. Após isso, serão analisadas individualmente e ordenadas de forma similar durante à construção do modelo.

A divergir do nó principal existem vários nós secundários desde veículos, edifícios, objectos, árvores, estradas, entre outos.

- Um modelo STL de um veículo foi importado diversas vezes de forma a criar todos os veículos necessários ao modelo. Este objecto contém objectos de translação e rotação que serão utilizados a partir do *Simulink*.

- Foram criados múltiplos *viewpoints*, com o objectivo de permitir ao utilizador a visualização em diversos pontos do sistema, incluindo o interior de um veículo.

- Os vários edifícios circundantes, assim como a estrada, os semáforos e todos os restantes objectos servem o propósito de tornar o modelo mais real, não interagindo com os veículos.

4.5.1. VEÍCULO

Tendo por base que a toolbox *Simulink 3D Animation*, não permite a reutilização de um só nó para várias entradas, foi importado um modelo de veículo para cada entidade gerada a partir do sistema. Isto obriga o utilizador a conectar todos os elementos das base de dados a todos os objectos no bloco *VR Sink*.

Um sistema constituído por vários objectos formará o objecto final: o veículo. Sempre que seja necessário um objecto no sistema depender de outro, é imprescindível estarem no mesmo nó. Exemplificando, as rodas do veículo necessitam de estar ligadas a este. Desta forma, terão de ser criadas debaixo do mesmo nó.

De igual forma, a translação e rotação do carro são sinais introduzidos no sistema que dependem hierarquicamente do veículo.

Como pode ser comprovado a partir da Figura 74, o objecto carro é constituído por vários sub nós, entre estes o *SistemaVeículo*, que inclui a translação, a *RotacaoVeiculo* e as *RodasVeiculo*, como assinalados a cor azul.

De forma ao carro ser colocado num extremo da Via 1, é necessário proceder à sua configuração prévia, nomeadamente no parâmetro de translação, indicado a verde. Neste caso, para os veículos C, a translação caracteriza-se por uma deslocação no eixo dos zz no valor de 124, 0.1 no eixo dos zz e 3 no eixo dos yy.

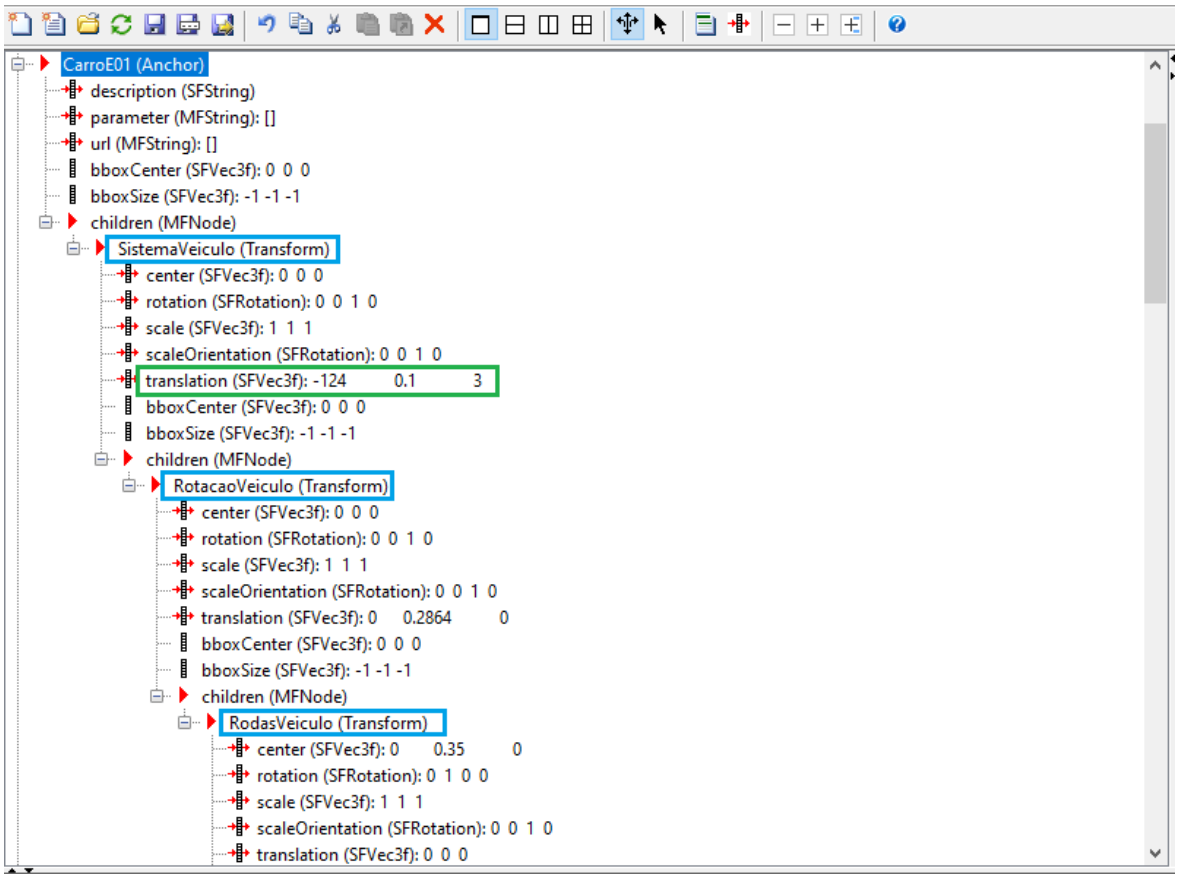


Figura 74 *Simulink* 3D – Exemplo de árvore do nó pai Carro

4.5.2. VIEWPOINTS

Foram criadas diversas vistas do utilizador para o sistema global e para o veículo:

- *Vista em XY* – representa o sistema visto de cima. Caracteriza-se por uma translação no eixo vertical no valor de 150. Como não se encontra associado a nenhum objecto encontra-se criada debaixo do nó principal do modelo.

As vistas seguintes encontram-se associadas ao veículo, visto dependerem deste. Quando existe uma translação no nó principal, nesta situação no veículo, as vistas sofrem uma translação de igual forma. O modelo hierárquico, no seguimento da Figura 74, pode ser analisado na Figura 75.

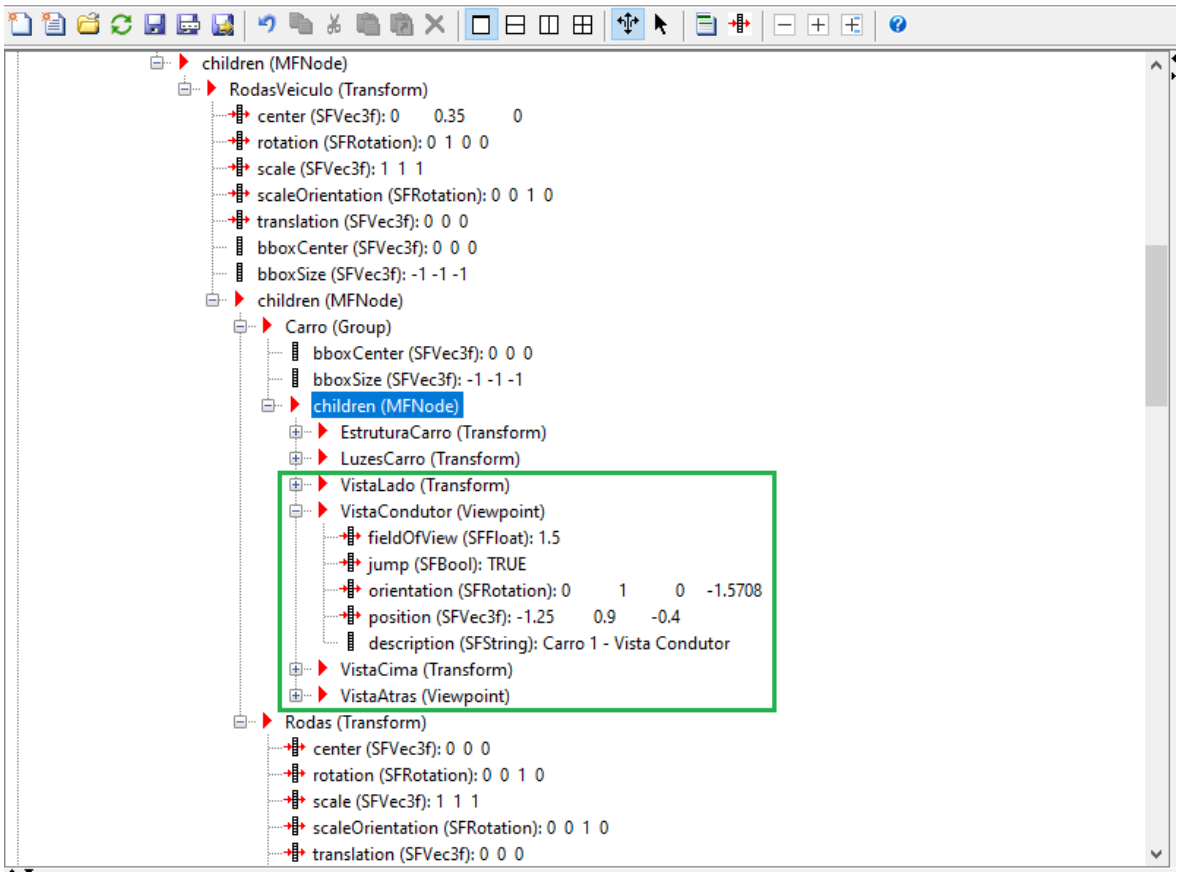


Figura 75 *Simulink* 3D – Listagem de vistas do veículo

- *Vista Lado*, *Vista Cima*, *Vista Condutor*, *Vista Atrás* – Caracterizam-se por serem *Transforms* criadas dentro do nó principal que representa o veículo. As posições nos três eixos são ajustadas de forma a dar o efeito pretendido ao ponto de vista.

Como demonstrado de seguida, a criação de diferentes pontos de vista permite ao utilizador ter um ângulo de acção dos planos que desejar para o seu projecto.

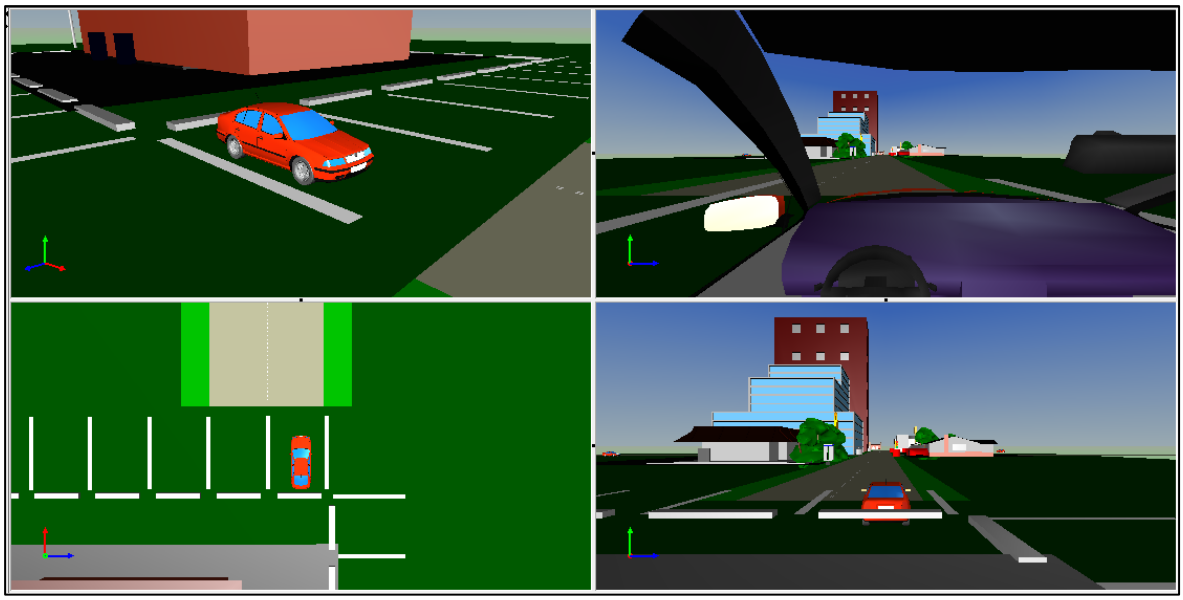


Figura 76 *Simulink* 3D – Diferentes tipos de vista (Lado, Condutor, Cima, Traseira)

De forma a representar algo sólido, onde todos os outros objectos fossem capazes de se basear, decidiu-se adicionar o solo do modelo. Este é representado por um ponto de coordenadas (0,0,0) e marca o ponto de referência do modelo. Todos os objectos adicionados posteriormente terão de sofrer uma translação, nos eixos pretendidos, de forma a ficarem posicionados na posição correcta, como já analisado na subsecção 4.5.1 para os veículos.

O valor *scale*, parâmetro configurável de todos os objectos foi ajustado individualmente de forma a manter a coerência no modelo, visto que os objectos diferem na sua construção. Deste modo, todos os objectos não interactivos foram adicionados ao modelo, transladados e parametrizados seguindo as descrições anteriores.

5. RESULTADOS

Este capítulo tem como objectivo comprovar a eficácia de um controlador difuso comparativamente a um controlador temporal. De tal forma, irão ser apresentados resultados de diversas simulações, recorrendo ao tempo médio de espera dos veículos, em cada via onde estes são gerados. Também terá como objectivo comprovar o funcionamento do sistema, assim como a lógica inerente à comutação dos sinais luminosos, escolha das vias por parte dos veículos e modelo 3D.

Estas simulações tem por base todo o processo descrito previamente no capítulo 0, recorrendo aos blocos do *SimEvents* e do *Simulink 3D Animation*.

Desta forma, irão ser descritos os resultados de dois tipos de simulações distintas:

- Simulação recorrendo a dois sinais luminosos (S1 e S2) controlados via lógica difusa.
- Simulação recorrendo a dois sinais luminosos (S1 e S2) controlados via lógica temporal.

De forma a simular o mesmo comportamento, ambas as simulações terão como base 300 segundos de simulação e uma aproximação do número de veículos gerados.

5.1. CONTROLADOR DIFUSO

Tendo sido gerados mais de 50 nas vias C e D, em nenhum espaço temporal da simulação existiu uma acumulação de mais de seis veículos. De igual forma, como se pode comprovar na Figura 77, os veículos E tiveram um período médio de espera de 7 segundos de simulação enquanto as vias C e D foi de 36 e 34, respectivamente.

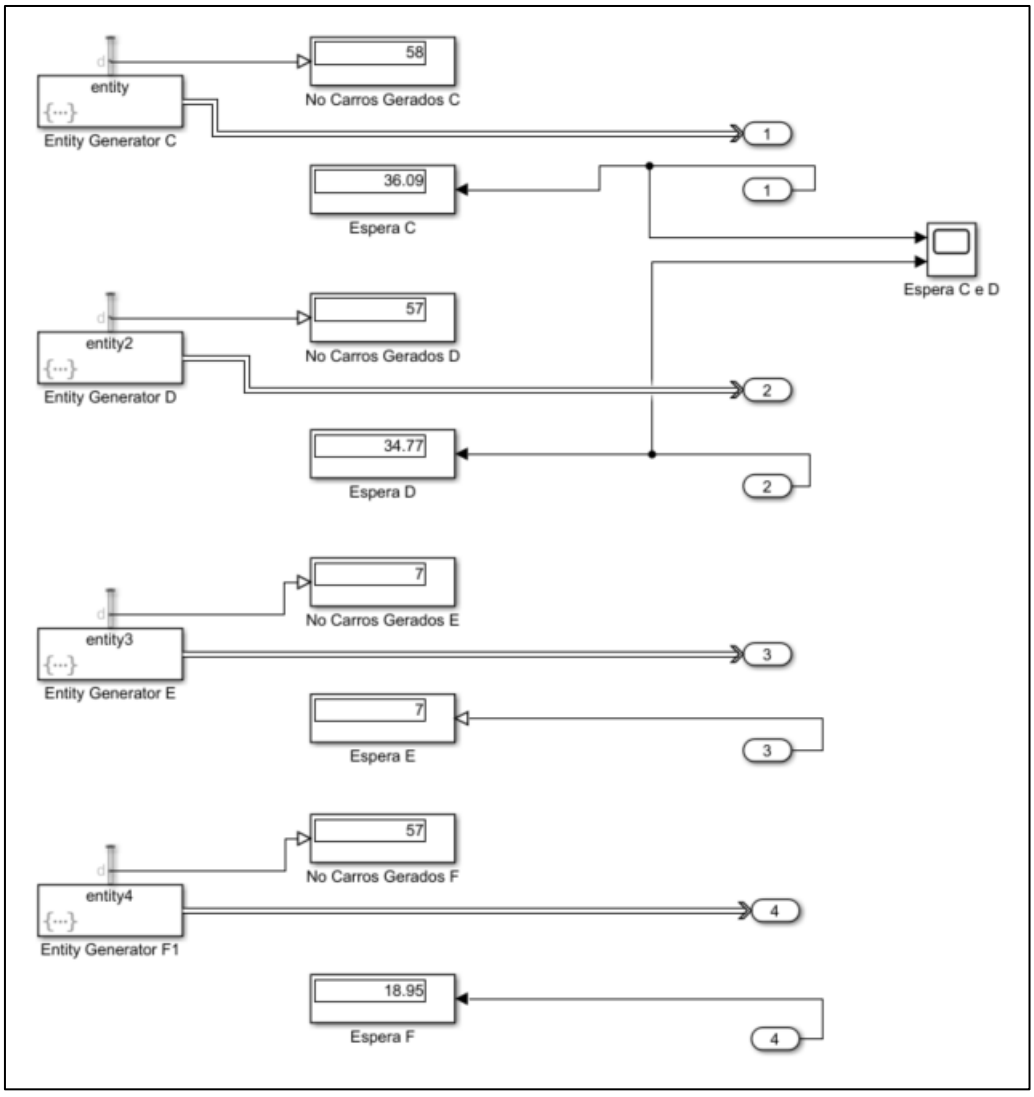


Figura 77 *SimEvents* – Resultado da simulação para controlador difuso

De igual forma, o trânsito decorreu sem grandes congestionamentos e, segundo a Figura 78, pode comprovar-se os veículos da via 1 com as mudanças de direcção efectuadas á esquerda e direita, como assinalado.

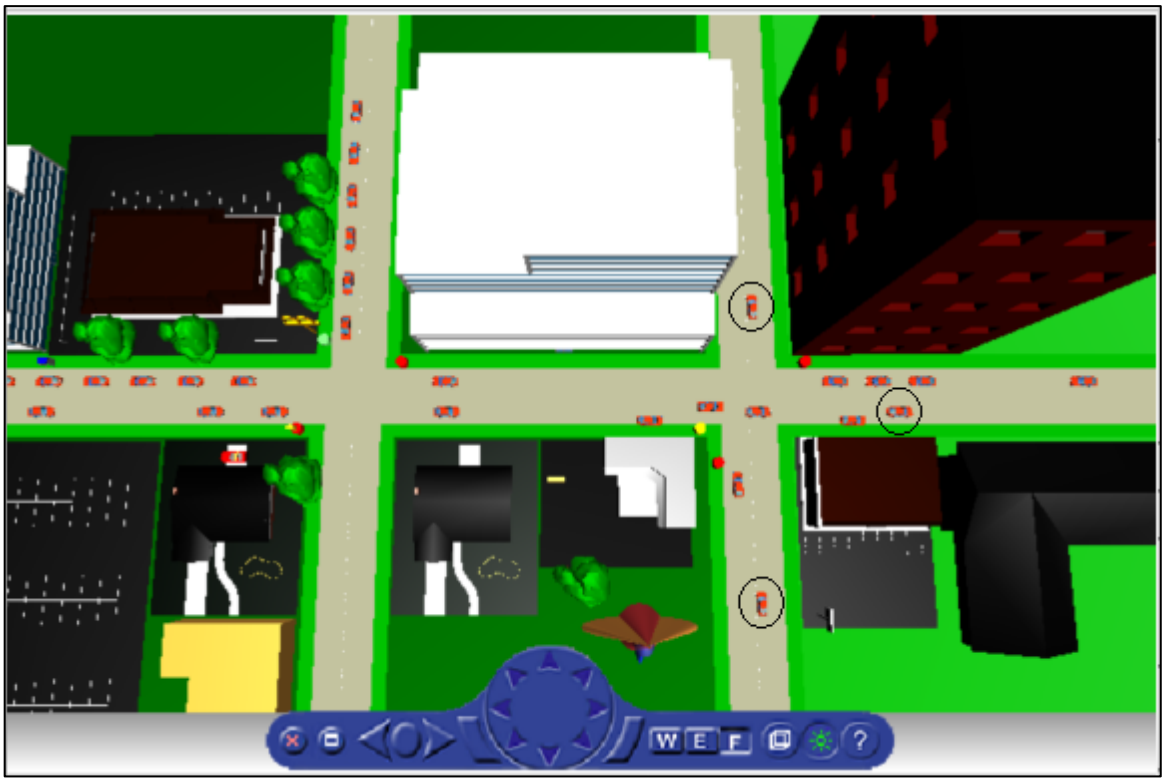


Figura 78 *Simulink 3D Animation* – Mudança de direcção de veículos C

Na Figura 79 pode analisar-se o tempo médio de espera, em segundos, desde que o veículo inicia a marcha na faixa respectiva até que é removido da entidade. Neste exemplo, das vias C e D, pode comprovar-se que após um período inicial, o tempo médio de espera aumenta devido à criação de veículos e consequentemente congestionamento das faixas contrárias. Contudo, após o sistema entrar em equilíbrio, as variações são inexistentes e pode considerar-se um resultado final sólido.

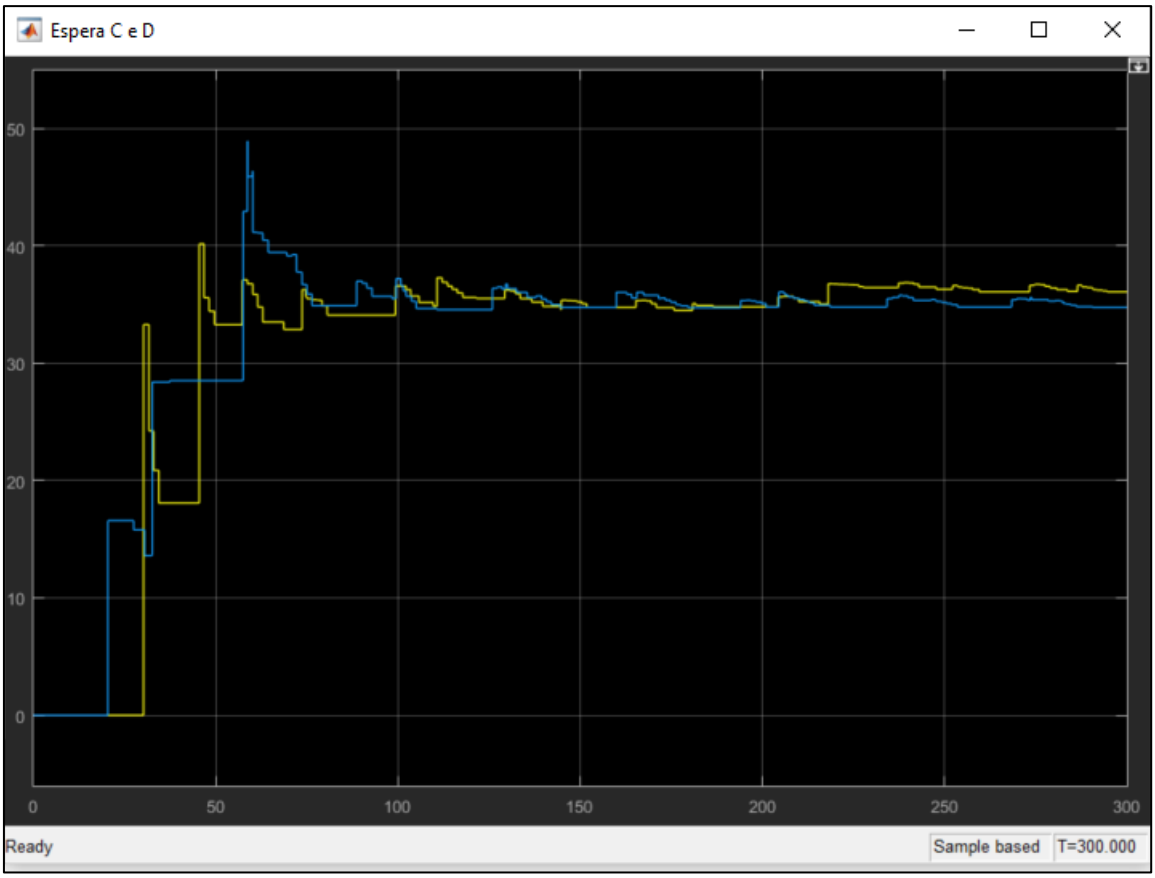


Figura 79 Tempo médio de espera dos veículos da via C e D – difuso

A título de resumo, a Tabela 2 demonstra os tempos de espera de cada faixa no final de uma simulação de 300 segundos.

Tabela 2 Tempo de espera com recurso a controlador difuso

Veículos	Tempo médio de espera
C	36,09s
D	34,77s
E	7s
F	18,95s

5.2. CONTROLADOR TEMPORAL

Como pode ser comprovado a partir da Figura 80, o número de entidades geradas aproxima-se da simulação anterior, o que comprova a eficiente criação de entidades com o *SimEvents*. De igual forma, como esperado, o tempo médio de espera das vias C e D sofreu um aumento em cerca de 50%, quando comparado com a optimização dos controladores difusos. De notar também que a via E, que engloba veículos com prioridade máxima, não consegue dar suporte a esta necessidade, tendo estes um tempo de espera aumentado em 300%.

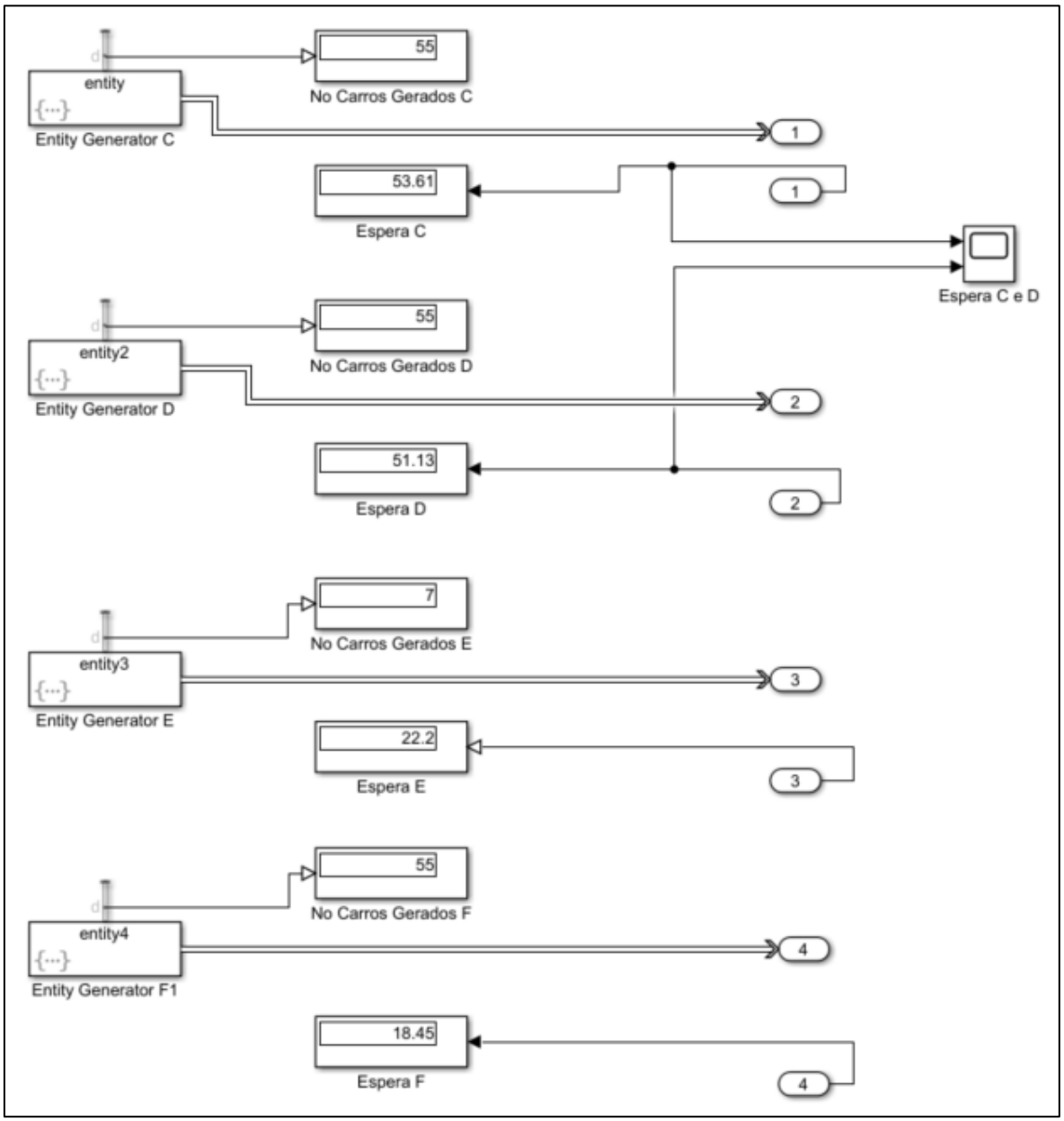


Figura 80 *SimEvents* – Resultados da simulação para controlador temporal

De forma análoga à subsecção anterior, na Figura 81 demonstra-se um excerto da simulação onde existe um maior número de tráfego de veículos. Este não só ultrapassa os resultados

anteriormente obtidos, como prejudica também os veículos que ultrapassam o cruzamento verticalmente, visto excederem o limite físico das vias centrais, como assinalado.

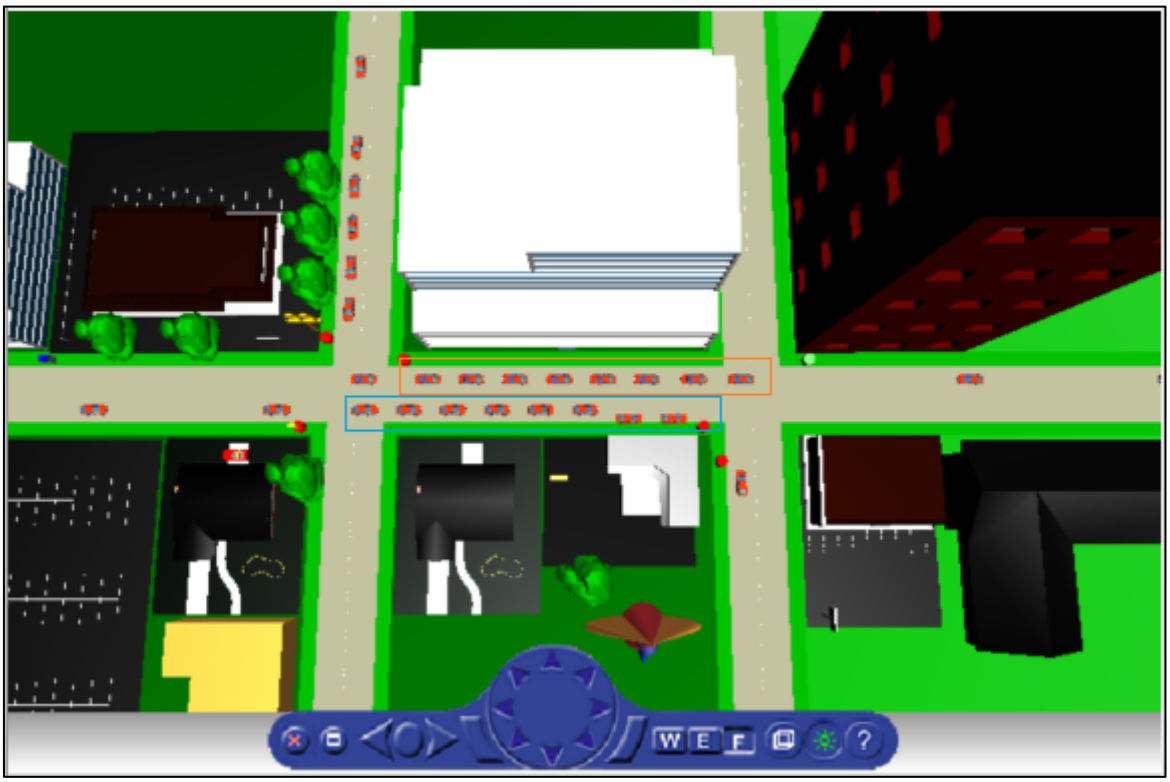


Figura 81 *Simulink 3D Animation* – Tráfego de veículos na via central

Contrariamente ao sistema estável criado a partir do controlador difuso, na Figura 82 pode analisar-se um sistema que, mesmo ao fim de 300 segundos de simulação, não estabiliza os tempos médios de espera, continuando até em ligeira subida perante os valores anteriormente mencionados.

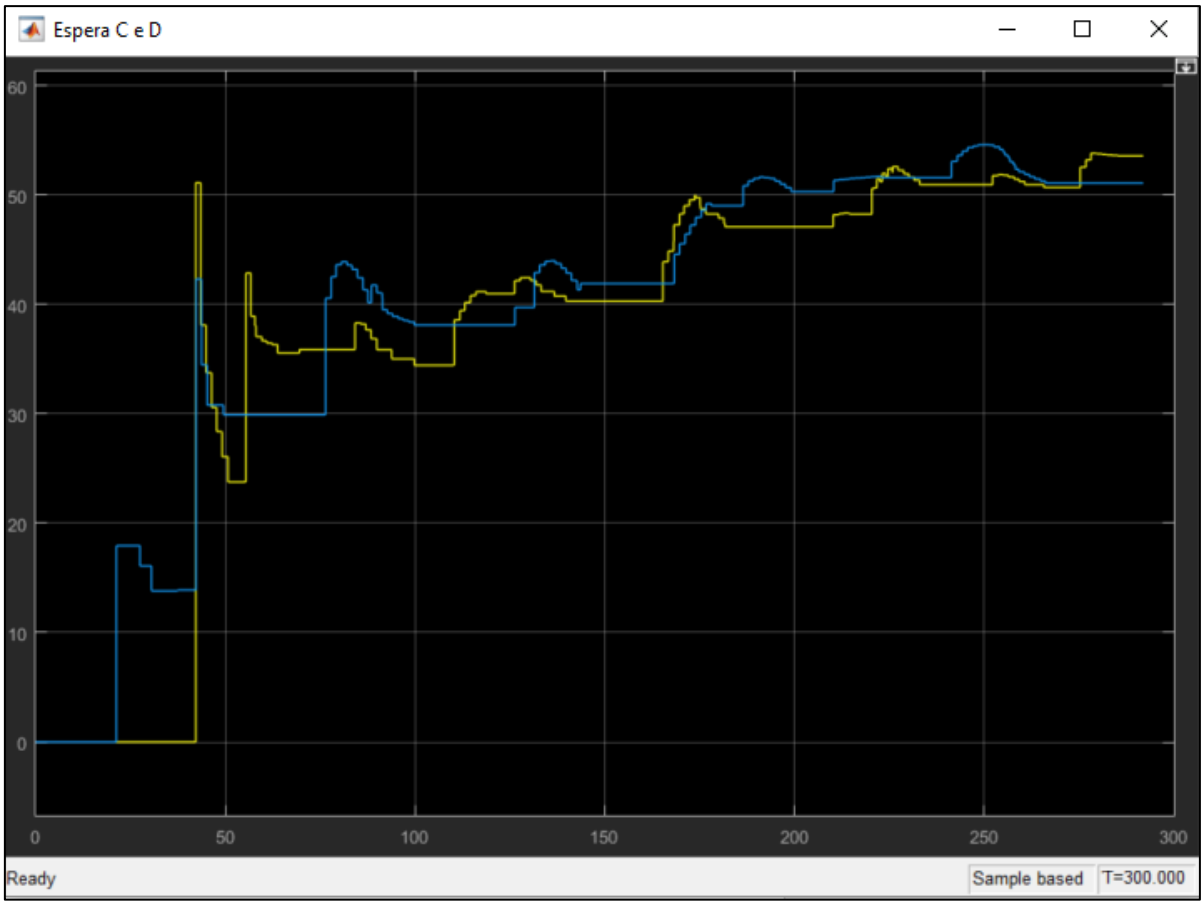


Figura 82 Tempo médio de espera dos veículos da via C e D – temporal

De forma similar ao controlador difuso, apresenta-se de seguida a Tabela 3, representando o tempo de espera dos veículos da faixa C, D, E e F controlados com recurso a um tempo fixo.

Tabela 3 Tempo de espera com recurso a controladores temporais

Veículos	Tempo médio de espera
C	53,61s
D	51,13s
E	22,2s
F	18,45s

6. CONCLUSÕES

Ao longo deste documento foram sendo tecidas opiniões e conclusões que sustentam o contínuo trabalho neste projecto. Projecto este que estuda um dos *softwares* mais utilizados na área de engenharia, comprovando a sua utilidade, eficácia e capacidade na modelação de um sistema complexo, mesmo sendo necessária a integração das mais variadas ferramentas para o efeito. Com este estudo de alguns dos componentes, conseguiu-se ganhar competências de grande relevo a nível investigatório, pois só com esta capacidade de pesquisa básica, tentativa e erro e persistência, foi possível arranjar soluções para os diversos problemas que foram sendo encontrados ao longo do projecto.

Inicialmente, tendo o projecto como base a ferramenta do *Simulink*, foi-se graduando a complexidade, assim como o desafio, tendo em vista os objectivos propostos, convertendo-se o projecto numa multiplicidade de ferramentas, incluindo as ferramentas de *Fuzzy Logic*, o *SimEvents, Stateflow* e *Simulink 3D Animation*. Desta forma, tornou-se clara a capacidade de adaptação e desenvolvimento que uma ferramenta destas proporciona ao mundo académico e profissional. Na área empresarial, é comum empresas recorrerem a simulações complexas previamente a desenvolverem modelos físicos e reais, onde despendem grande parte do orçamento. Sem esta capacidade dos softwares de simulação, estas ficariam, com certeza, comprometidas e impedidas de inovar.

De igual modo, a visualização em 3D recorrendo ao *Simulink 3D Animation*, permitindo a simulação de sistemas discretos ou contínuos, observando como estes se comportam, é de extrema importância. Assim, torna-se clara a importância da integração e relacionamentos entre os diversos *softwares* e diferentes ferramentas do MATLAB.

Contudo, o foco deste documento também incide sobre o controlo difuso, quando comparado com controladores temporais. O desenvolvimento e a implementação destes no *software*, com o auxilio da ferramenta *Fuzzy Logic*, torna-se de fácil compreensão, sendo auxiliada pelos variados documentos existentes sobre o assunto.

Relativamente aos resultados obtidos, como comprovados no Capítulo 5, existe uma clara diferença entre os tempos obtidos na simulação com recurso ao controlo difuso e controlador temporal, na ordem dos 30%. Mesmo existindo uma variação desta percentagem nas diferentes vias, é notório o aumento de desempenho que a larga maioria das infra-estruturas pode ter quando adopte este sistema. Apesar dos resultados visíveis desta implementação poderem ser visualizados no número de veículos a cada momento nas intersecções, a qualidade, conforto e custo de vida reflectem-se na cidade e nas pessoas que a habitam e transitam.

Com base nos estudos teóricos efectuados inicialmente no Capítulo 2, era expectável uma melhoria no fluxo automóvel com base nos 15%. Contudo, e mesmo sendo isto dependente dos parâmetros utilizados no controlador, características da via ou dos utentes, pode ser afirmado que os resultados foram conseguidos.

Contudo, e apesar dos resultados obtidos, estas ferramentas não estão preparadas originalmente para tal integração. Neste documento foram já descritas as diferenças entre os sistemas discretos e sistemas contínuos, ambos utilizados neste projecto, mas com recurso a técnicas que não deveriam estar presentes, como a necessidade de colocar informação na base de dados para ser lida, constantemente, pelo sistema continuo. De lembrar também que os veículos criados inicialmente em ambiente discreto não contém coordenadas, existindo a necessidade de iniciar e automatizar, com recurso ao *Simulink*, as coordenadas em todos os eixos dos veículos.

Como descrito anteriormente, este *software* tem a capacidade de desenvolver soluções especificas para certos desafios, como é o caso. Contudo, na eventual necessidade de se perspectivarem melhorias futuras para este projecto, teria de ter como base toda esta solução,

visto que qualquer alteração, causaria impacto no restante projecto. Desta forma, a existirem melhorias, existe a possibilidade de criar uma aumento de zonas rodoviárias e um fluxo maior de tráfego. De notar, contudo, que traria uma diminuição de performance à plataforma de simulação.

Em suma, este foi um projecto particular, desenvolvido a pensar numa solução final que foi atingida, comportando-se como um sistema com dinâmica real, autónoma e aleatória.

Referências Documentais

[1] Haack On *Fuzzy Logic*,
http://www.bu.edu/wcp/Papers/Logi/LogiGrun.htm,
Acedido em Novembro de 2017

[2] Schmöcker J.-D., Bell M. G. H (2010) Traffic control: current systems and future vision of cities. International Journal of Intelligent Transportation Systems

[3] Dunn Engineering Associates (2005). *Traffic Control Systems Handbook*. Westhampton Beach, NY, USA;

[4] Madhavan Nair B., Cai J. A (2007) - *Fuzzy Logic* controller for isolated signalized intersection with traffic abnormality considered. Proceedings of the IEEE Intelligent Vehicles Symposium (IV '07);

[5] Zoriktuev V. T., Ts S. G., Mesyagutov J. F. (2007) Methodology represented and derivation of knowledge in control systems of the mechatronic machine systems. *STIN Systems*

[6] Dimitrakopoulos, George, and George Bravos (2016) - Current Technologies in Vehicular Communication. Springer

[7] Shahraki AA, Shahraki MN, Mosavi MR (2013) - Design and Simulation of a Fuzzy Controller for a Busy Intersection. IEEE: 1-6

[8] Aleksandar Stevanovic, Peter T. Martin, (2008) - Split-Cycle Offset Optimization Technique and Coordinated Actuated Traffic Control Evaluated Through Microsimulation

[9] Institute for Transport Studies, University of Leeds, Urban traffic control systems: Evidence on performance",
http://www.konsult.leeds.ac.uk/private/level2/instruments/instrument014/l2_014c.htm

[10] Yu XH, Recker WW (2006) - Stochastic Adaptive Control Model for Traffic Signal Systems. Transportation Research

[11] The MathWorks. MATLAB Documentation Center ,[07/12/2015],
Vector Concatenate

[12] MATLAB Central – Real-Time Pacer for *Simulink*, [08/12/2015]

[13] http://www.mathworks.com/matlabcentral/fileexchange/29107-real-time-pacer-for-*Simulink*

[14] http://thecityfix.com/blog/naked-streets-without-traffic-lights-improve-flow-and-safety/

[15] Mathworks, Message in Charts
https://www.mathworks.com/help/*Stateflow*/ug/message-syntax-in-charts.html,
Acedido em Junho de 2018

[16] Mathworks, Message in *Stateflow*

https://www.mathworks.com/help/*Stateflow*/ug/define-messages-in-a-*Stateflow*-chart.html,
Acedido em Junho de 2018

[17] Mathworks, Message Properties
https://www.mathworks.com/help/*Stateflow*/ug/set-message-properties.html,
Acedido em Junho de 2018

[18] Mathworks, Modeling Errors
https://www.mathworks.com/help/*Stateflow*/ug/common-modeling-errors-the-debugger-can-detect.html,
Acedido em Julho de 2018

[19] Mathworks, *SimEvents* Documentation
https://www.mathworks.com/help/*SimEvents*/index.html,
Acedido em Fevereiro de 2018

Anexo A. Função IdentificarCarros()

```
function y = fcn(FaixaCarro)
global NumeroCarrosC;
global C;
global Lane;
NumeroCarrosC = NumeroCarrosC + 1;
P = find(C == inf);
C(P(1)) = NumeroCarrosC;
C(P(1)-60,3) = FaixaCarro;
E = find(Lane == inf);
Lane(E(1)) = NumeroCarrosC;
Lane(E(1),2) = FaixaCarro;
y = NumeroCarrosC;
```

Anexo B. Função EnableGate()

```
function y = fcn
%#codegen

global S1;
global C;
double cont;

if S1 == 1 || S1==2
    count = length(C(C(:,2)>=82 & C(:,2)<83));
    if count >= 1
        y = 1;
    else
        y = 0;
    end
else
    y = 0;
end
end
```

Anexo C. Função RemoverEntidade()

```
function fcn(ID,Data)

global C;
global Lane;
double Posicao;
single Data;
if Data == 500
        P = find(C == ID);
        C(P(1)) = [inf];
end
end
```

Anexo D. Função ResetVeiculo()

```
function [y,z]= fcn(a,b)

double temp;
double temp2;
if b == inf
    temp = 0;
    temp2 = 1;
else
    temp = a;
    temp2 = 0;
end
y = temp;
z = temp2;
```

Anexo E. Función Rotacao()

```
function [y,z]= fcn(u,t)
if t>159
    if u<0 && u>=-1
        y=1;
        z=0.3;
    elseif u<-1 && u>=-2
        y=1;
        z=0.5;
    elseif u<-2 && u>=-4
        y=1;
        z=0.7;
    elseif u<-4 && u>=-6
        y=1;
        z=1;
    elseif u<-6 && u>=-8
        y=1;
        z=1.2;
    elseif u<-8
        y=1;
        z=1.6;
    else
        y=0;
        z=0;
    end
else
        y=0;
        z=0;
end
end
```

Anexo F. Função ComutacaoSinal()

```
function [y,z] = fcn(u)

y=[0 0 0];
z=[0 0 0];
if u==0
    y=[1 0 0];
    z=[0.4 1 0.4];
elseif u==1
    y=[0.4 1 0.4];
    z=[1 0 0];
elseif u==2
    y=[1 1 0];
    z=[1 0 0];
elseif u==3
    y=[1 0 0];
    z=[1 1 0];
elseif u==4
    y=[1 0 0];
    z=[1 0 0];
end
```