Universidade de Aveiro
2014
Departamento de Electrónica, Telecomunicações e Informática

Carlos Manuel Carvalho Magalhães

Modelização do Integrador do Agente CAMBADA

Universidade de Aveiro
2014
Departamento de Electrónica, Telecomunicações e Informática

Carlos Manuel Carvalho Magalhães

Modelização do Integrador do Agente CAMBADA

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica dos Doutores José Nuno Panelas Nunes Lau e António José Ribeiro Neves, Professores Auxiliares do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

Dedico este trabalho aos meus pais que que sempre apoiaram o meu projecto de vida, que nestes últimos anos pervagou por alcançar o grau de Mestre em Engenharia de Computadores e Telemática da Universidade de Aveiro.

o júri

Presidente	Doutor. Luís Filipe de Seabra Lopes professor associado da Universidade de Aveiro
Vogais	Doutor. Armando Jorge Miranda de Sousa professor auxiliar da Universidade do Porto - Faculdade de Engenharia
	Doutor. José Nuno Panelas Nunes Lau professor auxiliar da Universidade de Aveiro

agradecimentos

Agradeço aos Doutores que orientaram e apoiaram o desenvolvimento desta tese de mestrado, pelo seu bom trabalho, paciência e dedicação.
Agradeço também aos meus pais que me proporcionaram a oportunidade de engrenar na vida académica, apoiando sempre as minhas decisões.

palavras-chave robótica, robô móvel e autónomo, modelização, localização, seguimento

resumo

No âmbito do projecto CAMBADA, do qual o DETI/UA (Departamento de Electrónica, Telemática e Informática da Universidade de Aveiro) é parceiro, surge a necessidade de analisar e produzir uma estrutura modelar para o integrador que incorpora o agente CAMBADA. O principal objectivo desta dissertação é dotar o integrador de um modelo fácil de compreender, de rápida evolução e simples implementação no desenvolvimento de novos conceitos.

A equipa CAMBADA já possui uma estrutura bem definida para o agente que controla os seus robôs, de tal forma que a necessidade de o manter funcional e munido das suas capacidades é algo inevitável. Assim sendo, o integrador deste agente deixou de ser um módulo extenso, interdependente nas suas responsabilidades e de edição complexa, para se tornar num módulo que obedece a um modelo simples e de fácil edição.

O módulo integrador é um dos importantes componentes que integra o agente CAMBADA, já que é o módulo com a responsabilidade de integrar toda a informação referente ao robô e ao estado do mundo que o rodeia. Um módulo complexo, mas fundamental para o bom funcionamento do robô e para o sucesso nas tomadas de decisão durante um jogo.

O projecto de modelizar o integrador do agente CAMBADA incide no trabalho subjacente a esta dissertação, que consistiu essencialmente numa primeira análise laboriosa do integrador existente e posteriormente na criação de uma estrutura completamente nova e modelada, que é suportada por um modelo alicerçado em interfaces que dividem a complexidade da integração nas diversas áreas sensoriais que o robô CAMBADA incorpora.

Nesse sentido, o novo modelo é constituído por módulos independentes para a integração da informação relativa à bola e ao robô, já que a informação relativa aos obstáculos, ao estado do jogo e ao baixo nível é integrada como anteriormente no módulo principal de integração. Foram exploradas formas de filtragem assentes no uso de filtros de Kalman e de Partículas, para as observações referentes às amostras usadas no cálculo da posição e velocidade da bola de jogo.

O objectivo principal deste trabalho foi cumprido, já que foi evidente a melhoria que o novo modelo trouxe ao módulo de integração do agente CAMBADA. Quanto à implementação dos novos métodos de filtragem segundo o uso dos filtros estudados, não produziram o resultado esperado, uma vez que não trouxeram uma melhoria significativa à precisão dos dados obtidos do mundo. Contudo, serviram para demonstrar o quão importante é o novo modelo, porque possibilitou a integração de novas implementações sem a necessidade de alterar o integrador.

keywords robotics, mobile autonomous robot, modeling, location tracking

abstract

In the context of the CAMBADA project, in which the DETI/UA (Department of Electronics, Telematics and Informatics, University of Aveiro) is a partner, there is the need to analyze and produce an improved model structure for the integrator that incorporates the CAMBADA agent. The objective of this dissertation is to provide an integrator model that is easy to understand, allows rapid evolution and where it is simple to implement new concepts of development.

The CAMBADA team already has a well-defined structure for the agent that controls the robots, hence, they need to maintain functional and equipped their capabilities. The integrator of this agent has changed from an extensive, interdependent in their responsibilities and complex module to become a module that follows a simple model and is easy to edit.

The integrator module is one of the important components that integrates the agent CAMBADA, since it is the module with the responsibility to integrate all of the information on the robot and on the state of the world that surrounds it. A complex module, but it is essential for the proper functioning of the robot and the success in decision-making during a game.

The new model of the CAMBADA agent integrator was specified and implemented in this dissertation work. First, a laborious analysis of the existing integrator was performed and subsequently the creation of a completely new and modeled structure which houses grounded model interfaces which divide the complexity of integrating the various sensory areas that incorporates CAMBADA robot was achieved.

In this sense, the new model consists of independent modules for the integration of information on the ball and the robot. Information concerning obstacles and the state of the game at the lower level is integrated as before in the main module integration. Forms of filtering based on the use of Kalman and Particle filters, for observations relating to the samples used to calculate the position and velocity of the game ball were explored.

The main objective of this work was completed. There is an evident improvement with the new model brought to the integration of CAMBADA agent module. The implementation of new methods of filtering using the filter according to the study did not produce the expected results, since it did not bring a significant improvement in the accuracy of the data obtained from the world. However it served to demonstrate how important the new model is, because it allowed the new implementations of integration, without the need to change the integrator.

Índice

Lista de Figuras

Lista de Diagramas

Lista de Gráficos

Capítulo 1 - Introdução

O primeiro capítulo desta dissertação de mestrado fala da motivação que esteve na origem do trabalho executado, enquadrando-nos na história do conceito de desenvolvimento do mesmo. É também neste capítulo delineada a estrutura que suporta o documento.

1.1. Motivação e enquadramento

O conceito de "robô" fez-se público pela primeira vez em 1921 através de uma obra de ficção do escritor *Karel Čapek*, que contou com a presença de uma entidade artificial autónoma, um autómato com forma humana e capaz de fazer tudo o um ser humano é capaz [1]. Nos anos 70 observou-se uma procura mais significativa dos robôs para desempenho de tarefas árduas, desagradáveis ou mesmo perigosas para humanos, sendo na sua maioria trabalhos mecânicos repetitivos. Já nos dias que correm os robôs não são utilizados exclusivamente para o trabalho industrial, visto que nos últimos tempos fez-se notar uma disseminação gradual por outras áreas, incluindo ambientes de competição, domésticos, etc [2].

O *RoboCup*, descrito no Capítulo 2, é um evento à escala mundial que proporciona aos seus participantes um ambiente de competição internacional dividida por várias áreas da robótica, cujo objectivo é promover o seu desenvolvimento tecnológico e científico. Entre as várias áreas, destaco as ligas de futebol robótico onde competem diferentes tipos de robôs consoante as suas características. Contudo, partilham um conceito em comum: todos os robôs participantes têm de ser autónomos, capazes de se movimentar e comunicar remotamente sem intervenção humana [3].

É neste conceito comum que se enquadra esta dissertação, pois foi motivada pelo desejo de fomentar uma maior capacidade de obter a informação do mundo que rodeia o robô e também pela necessidade de agilizar a forma como o desenvolvimento e a implementação das novas ideias são consumados.

1.2. Objectivos principais

Centrados no módulo relativo ao Agente CAMBADA, os objectivos desta dissertação enquadram-se numa análise detalhada deste, de forma a criar uma nova estrutura do seu integrador e assim explorar novas formas de integração. Isto é, uma organização modelar dotada de uma estrutura de fácil percepção, cuja hierarquia e conceitos associados tornem o desenvolvimento de novas ideias eficazes e eficientes. Dessa forma, o novo integrador deve possuir as três características específicas que distingo abaixo.

1.2.1. Modelização

O código relativo à integração dos vários sistemas do robô deve estar dividido por módulos bem específicos, para que a sua localização seja rápida, a implementação seja concisa e a criação de novas ideias evolua de forma estruturada.

1.2.2. Historicidade

O novo integrador deve ser capaz de manter as valências e capacidades do anterior, para que estando estas distribuídas pela nova estrutura seja possível alternar entre as várias formas de integração implementadas. Possibilitando assim uma análise fácil e comparativa do comportamento resultante do uso das diferentes versões ou conceitos de integração.

1.2.3. Multiplicidade

De forma a suportar uma gama de condições variada, o novo integrador deve ser dotado de diferentes implementações que efectuam o tratamento da informação. Contudo, devem respeitar interfaces de acessos únicos, para que seja rápido e fácil o desenvolvimento ou

alteração de uma dada implementação sem que seja necessário proceder a alteração dos restantes módulos.

Descritos os objectivos principais desta dissertação, com particular cuidado na modelização, historicidade e multiplicidade do integrador do agente CAMBADA, resta referir que este trabalho também recaiu na análise e melhoria da implementação existente. Sendo que, para isso, o estudo de paradigmas associados à velocidade e posição de objectos do mundo real foi inevitavelmente necessária.

1.3. Estrutura da Dissertação

A estrutura deste documento encontra-se dividida em cinco capítulos descritos abaixo, sendo que cada um deles foi cuidadosamente escolhido de forma a proporcionar uma fácil procura dos conceitos que estão na base e desenvolvimento desta Dissertação de Mestrado, bem como uma boa percepção do trabalho desenvolvido.

No primeiro capítulo encontra-se a descrição da motivação para o exercício implícito nesta Dissertação, bem como os principais objectivos e enquadramento que resumem toda a dissertação.

O segundo capítulo transmite o estudo realizado do estado do mundo em relação aos principais eixos que suportam esta Dissertação de Mestrado, assim como os conceitos investigados para a resolução dos problemas detectados.

O terceiro capítulo descreve o trabalho desenvolvido no âmbito desta Dissertação de Mestrado, expondo de forma explícita as alterações executadas na estrutura existente e apresentando o novo modelo para a implementação de novas ideias de integração de modo rápido, fácil e estruturado.

O quarto capítulo apresenta os diferentes testes realizados ao novo modelo e melhorias associadas, terminando com a devida avaliação executada para cada um destes. É de uma forma gráfica, que neste capítulo, os resultados conseguidos pela implementação de novas ideias são visíveis.

O quinto e último capítulo, destina-se à discussão dos vários resultados obtidos, sem esquecer a referência ao trabalho futuro que é relevante efectuar para que o novo modelo e conceitos introduzidas tenham a devida produtividade.

Capítulo 2 – Estado da Arte

Este segundo capítulo tem como objectivo demonstrar o estudo efectuado ao estado da arte, relacionado com os conceitos abrangidos por esta dissertação. Inicialmente abordando a competição de um modo geral e em particular a liga MSL, o estudo seguiu-se pelo projecto CAMBADA, terminando nos modelos existentes para a filtragem de dados relativos à observação de objectos do mundo real.

2.1. A competição RoboCup

RoboCup é uma competição robótica a nível internacional que se realiza anualmente desde 1997, cujo nome surge de três conceitos distintos: *Robot*, *Soccer* e *World Cup*. Dado que a base desta competição é o estudo e o desenvolvimento da Robótica associada à inteligência Artificial, fornece várias competições onde podem ser aplicadas diversas tecnologias e metodologias que combinadas têm atingido bons resultados [3].

Dr. Hiroaki Kitano, presente na Figura 1, investigador na área da inteligência artificial e natural de Tóquio, foi o responsável pela iniciativa que fez surgir a *RoboCup Federation* da qual foi presidente e fundador. O seu principal objectivo sempre foi a dinamização da Inteligência Artificial com especial cuidado para Sistemas Autónomos Multiagente [4].

De forma a atingir uma maior panóplia de realidades, esta competição do *RoboCup* divide-se em quatro grandes grupos: *RoboCupSoccer*, *RoboCupRescue*, *RoboCup@Home* e *RoboCupJunior*.

Figura 1 - Dr. Hiroaki Kitano

A divisão adjacente a cada uma destas competições está assente nas várias ligas:

RoboCup Soccer:

- Standard Platform League
- Small Size League
- Middle Size League
- SimulationLeague (2D e 3D Soccer Simulation)
- Humanoid League

RoboCup Rescue

- Recue Robot League
- Rescue Simulation League

RoboCup@Home

RoboCup Junior

- Soccer Challenge
- Dance Challenge
- Rescue Challenge
- General

De notar que o RoboCup não se limita à competição, uma vez que possui uma vertente relativa ao simpósio, onde são apresentados e discutidos trabalhos científicos das diversas áreas decorrendo, em geral, após a competição. Com esta divulgação dos novos conceitos e

ideias presentes na concepção final do trabalho desenvolvido, as equipas incentivam a continuidade e evolução dos adversários. Por outro lado, estimulam a criação de novas equipas que vão surgindo já com uma qualidade razoável na competição [5].

2.1.1. Liga Middle Size League

A *Middle Size League* (MSL) é a liga de Futebol Robótico Médio, uma das ligas oficiais do *RoboCup* cujas equipas participantes são constituídas por cinco robôs que se movem e, inevitavelmente, são autónomos. Com uma altura máxima de 80 centímetros e uma largura máxima de 50 centímetros, cada robô não pode pesar mais do que quarenta quilogramas e deve ser dotado de sensores que lhe proporcionem toda a informação relativa ao ambiente em que se encontram [6].

Cada um dos robôs deve ser capaz de se comunicar com a restante equipa através de uma rede sem fios, à qual também se encontram ligadas as *Basestation* de cada equipa, que em geral, são interfaces gráficas que mostram informação partilhada pelos robôs da equipa. Por sua vez, a *Basestation* de cada equipa está ligada por uma rede cablada ao *RefereeBox*, um computador que sinaliza a marcação das várias situações de jogo, já que não é permitida qualquer intervenção humana durante o jogo, exceptuando a remoção de robôs e reposição da bola por parte do árbitro [6].

É certo que o futebol foi a grande motivação para o nascimento da MSL, já que se trata de um desporto popular em todo o mundo e ao nível científico apresenta desafios muito interessantes, porque estamos a falar de um jogo em equipa que abrange decisões colectivas e individuais. Denota-se, por isso, a necessidade do robô se localizar individualmente, assim como localizar e identificar objectos relevantes para si e para a equipa, como o exemplo das balizas, a bola e outros.

Por outro lado, a necessidade colectiva de possuir uma estratégia para que todos funcionem em equipa é também relevante, dada a existência de regras bem específicas que envolvem necessidades, como a troca de bola ou tempos de espera associados à marcação de uma falta. Situações que acontecem devido à existência de um ambiente totalmente dinâmico e à realidade de que o jogo é entre duas equipas diferentes.

Assim como nos mostra a Figura 2, relativa à situação inicial do jogo da liga MSL no *IranOpen* 2014, em que as equipas *MRL* e *CAMBADA* estão na sua formação inicial de jogo.

Figura 2 – Posicionamento inicial de duas equipas num jogo da liga MSL

2.1.2. Regras da liga Middle Size League

O futebol robótico da liga *MSL*, assim com o futebol humano, é dotado de regras que servem de critério para a marcação de faltas e outras situações no decorrer de um jogo. Para isso, existe pelo menos um árbitro que circula nas laterais do campo, sendo o responsável por sinalizar as faltas e repor a bola em campo, como nos mostra a figura 3, em que o árbitro depois de assinalar um golo coloca a bola nomeio campo.

Há também um outro árbitro, que é responsável por reproduzir a marcação de faltas na *RefereeBox*, e desta forma, a *basestation* de cada equipa é alertada para a falta que ocorreu, que por sua vez informa todos os robôs da situação de jogo assinalada.

Figura 3 – Reposição da bola em jogo pelo árbitro

Um jogo da liga *MSL* é dividido em duas partes com quinze minutos cada, limitadas por um intervalo de dez minutos entre elas. Em cada uma das partes, podem ser sinalizadas pelo árbitro as seguintes situações de jogo:

- *Kick Off*: semelhante ao pontapé de saída de um jogo de futebol da *FIFA*, a bola deve estar no centro e tem de existir um passe entre os dois robôs que iniciam o jogo.

- *Free Kick*: marcação de uma falta, sendo que a bola é posicionada pelo árbitro do jogo no local da marcação da falta e assim como na situação anterior, deve existir um passe entre dois robôs da mesma equipa.

- *GoalKick*: situação análoga ao pontapé de baliza de um jogo da *FIFA*, em que a bola é colocada pelo árbitro num dos pontos negros mais próximos da baliza.

- *Throwin*: estado de jogo parecido com um lançamento de um jogo da *FIFA*, porém a bola é colocada em campo pelo árbitro junto da linha de fundo onde saiu.

- *Corner*: em tudo igual à marcação de canto de um jogo da *FIFA*, mas como a situação anterior é o árbitro que coloca a bola no arco relativo ao canto.

- *Penalty*: a bola é colocada na marca de grande penalidade, sendo que o robô apenas se desloca até à bola e chuta.

Ainda deve ser referido que em qualquer uma das situações descritas, os robôs adversários devem salvaguardar uma distância mínima, com um raio de três metros, podendo avançar passados dez segundo após ser assinalada a situação de jogo, ou imediatamente após ser iniciada uma jogada pela equipa contrária.
O campo oficial para a realização de jogos da liga *MSL*, possui um aspecto semelhante ao da Figura 4, dotado de marcas e zonas que obedecem a medidas oficiais.

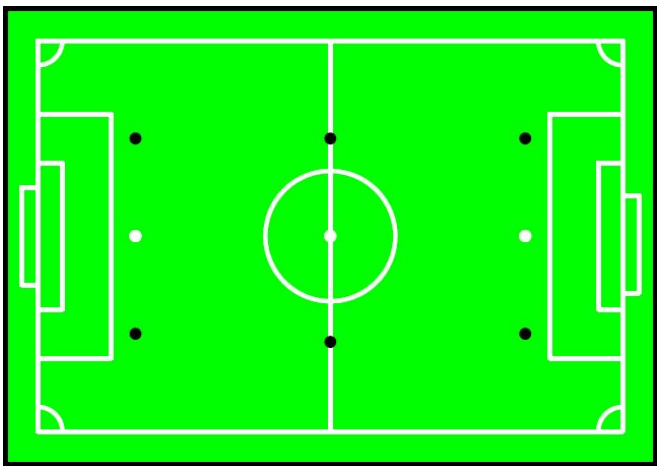

Figura 4 – Representação gráfica do campo oficial da liga MSL

Quatro linhas brancas delimitam o campo verde que deve possuir uma largura de doze metros e um comprimento de dezoito metros que é dividido em duas metades por uma linha branca central, com os quatro cantos dotados de um arco branco com um metro de raio no interior. Já o centro do campo é assinalado com um ponto branco, alinhado com a linha central e acompanhado por um círculo com dois metros de raio e dois pontos negros que se situam entre o centro e as linhas limitadoras [6].

Cada uma das partes do campo deve possuir uma baliza com 2 metros de largura e 1 metro de altura, alinhada com o centro do campo mas situada fora dele, estando assim perfilada com a linha delimitadora. Deve haver também em ambas as partes, uma marca branca de penalti que centrada com a baliza está a 3 metros de distância e é acompanhada com duas marcas negras que alinhadas a esta, estão a metade da distância entre o centro e as linhas delimitadoras.

Integram também ambas as partes do campo duas áreas, uma nomeada de pequena área e a outra de grande área, ambas ligadas à linha de baliza e alinhadas ao centro desta. O que difere entre as duas áreas, além do nome, é o comprimento e a largura destas, sendo que para a grande área o comprimento é de 3 metros e a largura de 8 metros, já para a pequena área, apenas 75 centímetros de comprimento e 3 metros e meio de largura [6].

2.2. Projecto CAMBADA

O nome CAMBADA é o acrónimo de *Cooperative Autonomous Mobile roBots with Advanced Distributed Architecture*, um projecto que teve início oficial em Outubro de 2003 e deu origem a uma equipa preparada para a competição da liga *MSL* do *RoboCup*. Os elementos que a constituíam vinculados à Universidade de Aveiro, eram pertencentes ao IEETA (Instituto de Electrónica e Engenharia Telemática de Aveiro) e ao DETI (Departamento de Electrónica, Telecomunicações e Informática) [7].

Com um vasto leque de participação em competições de futebol robótico e outras de carácter semelhante, destaca-se o primeiro lugar obtido no *RoboCup World Championship* em 2008, tornando-se assim a primeira competição em que esta equipa portuguesa ganha um campeonato internacional de futebol robótico. A Figura 5, ilustra o aspecto físico actual dos robôs da equipa, maioritariamente, pensado e desenvolvido pelos elementos da equipa.

Figura 5 – Aspecto de um robô CAMBADA

2.2.1. Arquitectura do robô

A arquitectura de software adoptada pela equipa para os seus robôs, é centralizada num processo principal de alto nível que é responsável pela coordenação dos vários módulos. Isto é, um processo principal que processa a informação obtida dos vários sensores que constituem o robô e também da comunicação externa com os restantes robôs da equipa,

determinando de forma controlada a caracterização do estado do robô e da realidade que o rodeia [8].

O suporte computacional de base é um sistema *Linux*, sendo que a comunicação entre os robôs usa um protocolo de transmissão *TDMA* adaptativo [9], que funciona sobre o protocolo *802.11b* do *IEEE*, de forma a reduzir a latência de comunicação entre eles e dessa forma minimizar a probabilidade de colisão entre as diversas transmissões de informação. Relativamente ao sistema de visão, é usada uma câmara e um espelho hiperbólico para que sejam obtidas um conjunto de imagens que aplicadas a algoritmos elaborados identifiquem e localizem objectos de interesse, como a bola, linhas do campo, entre outros [10].

O sistema sensorial é um sistema inteligente de interpretação e coordenação dos vários módulos, que se divide em três fases distintas: a fusão de sensores, comportamentos e cooperação. Como nos mostra o Diagrama 1, serve de suporte a este sistema inteligente, um sistema de baixo nível, constituído por um *IMU* e um conjunto de microcontroladores interligados por meio de um barramento, onde estão presentes as quatro funções principais do controlo: a odometria, o controlo de movimento, o sistema de chuto e a monitorização do sistema.

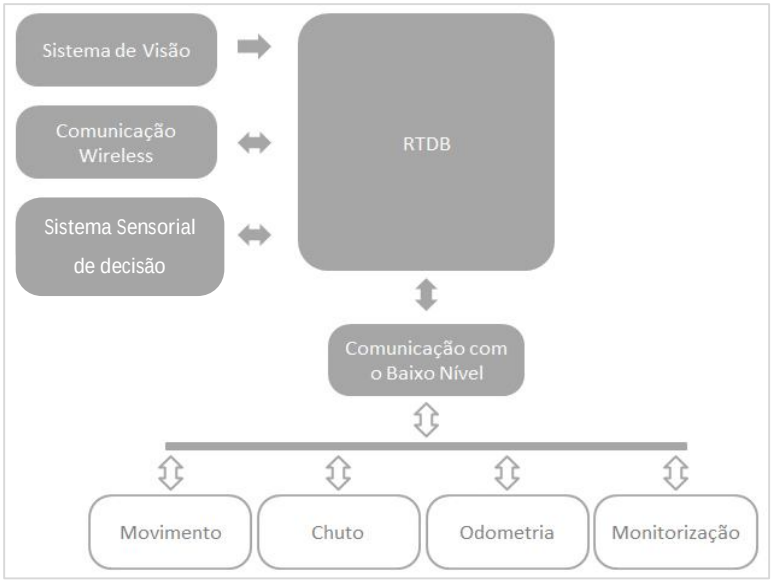

Diagrama 1 – Módulos intrínsecos a um robô CAMBADA

É através da comunicação com o sistema de baixo nível, que o sistema sensorial e de decisão processa os valores provenientes da odometria e da monitorização do sistema, actualizando assim a base de dados *RTDB* com os resultados do processamento. Este

sistema principal e inteligente, usa o mesmo meio de comunicação para partilhar os comportamentos deliberados para execução de baixo nível, isto é, partilha relativa às ordens de chuto ou de movimentação [11].

2.2.2. Equipas da liga Middle Size League

A equipa CAMBADA no seu percurso competitivo tem conhecido diferentes equipas que respeitando as normas exigidas na liga MSL do *RoboCup*, apresentam robôs com diferentes formatos. As diferenças não são notórias apenas na aparência mas também no conjunto de sistemas que os constituem.

A equipa que está presente na Figura 6, teve a sua origem no centro de pesquisa nomeado de *Mechatronics Research Laboratory* (MRL), ligado à Universidade Islâmica *Azad* de *Qazvin*. Nomeada de equipa MRL, usufrui de robôs que são o fruto do trabalho de investigação de grande parte dos alunos do curso de Mestrado em Mecatrónica e, surgiram com o objectivo de criar um robô equipado com um algoritmo optimizado para o controlo de visão, de planeamento de trajectória e estratégias de jogo, de acordo com uma avançada perícia mecatrónica [12].

Figura 6 – Robôs da equipa MRL

A equipa *Tech United Eindhoven*, presente na figura 7, é associada à Universidade tecnológica de Eindhoven. É desde 2005 constituída na sua maioria por alunos de Mestrado e Licenciatura das áreas ligadas à robótica. O objectivo inicial foi apenas a

participação na liga *MSL* do *RoboCup*, mas seis anos depois conceberam o robô *AMIGO* que participa agora na liga *RoboCup@Home*. Entre outros fatores, os conhecimentos adquiridos na construção dos seus robôs futebolistas foram a base para a participação na liga *Humanoid League* em 2012 [13].

Figura 7 – Equipa Tech United Eindhoven

A equipa *Carpe Noctem*, criada em 2005 no grupo distribuído de sistemas da Universidade de *Kassel*, viu a sua reputação crescer internacionalmente devido ao sucesso nas suas pesquisas na área das plataformas *Middleware* e administração de sistemas distribuídos.
O projecto de pesquisa *Carpe Noctem* é focado na procura de técnicas e soluções para a cooperação autónoma de sistemas móveis, fazendo parte do laboratório *Distributed Autonomous Systems* [14].

Figura 8 - Equipa Carpe Noctem

Refiro também dois nomes importantes, a equipa Watter que foram recentemente campeões, e a equipa Tribots que deixaram uma importante marca na liga MSL.

2.2.3. Classificações em competição

A listagem que segue abaixo, diz respeito aos resultados obtidos pela equipa CAMBADA em competições relacionadas com futebol robótico:

Portuguese Robotics Open 2014: 2º lugar;
Iran Robotics Open 2014: 2º lugar;
Portuguese Robotics Open 2013: 2º lugar;
RoboCup World Championship 2013:
- 3º lugar no torneio; 1º lugar no desafio técnico; 3º lugar no desafio científico;

RoboCup World Championship 2012:
- 4º lugar no torneio; 1º lugar no desafio técnico; 2º lugar no desafio científico;

DutchOpen 2012: 3º lugar;
Portuguese Robotics Open 2012: 1º lugar;
RoboCup World Championship 2011:
- 3º lugar no torneio; 1º lugar no desafio livre;

Portuguese Robotics Open 2011: 1º lugar.
RoboCup World Championship 2010:
- 3º lugar no torneio; 4º lugar no desafio técnico;

GermanOpen 2010: 2º lugar;
Portuguese Robotics Open 2010: 1º lugar;
RoboCup World Championship 2009:
- 3º lugar no torneio; 1º lugar no desafio técnico;

Portuguese Robotics Open 2009: 1º lugar;
RoboCup World Championship 2008:
- 1º lugar no torneio; 2º lugar no desafio técnico;

Portuguese Robotics Open 2008: 1º lugar.
RoboCup World Championship 2007:
- 5º lugar no torneio;

Portuguese Robotics Open 2007: 1º lugar.

2.3. Modelos do Mundo

A realidade que conhecemos do nosso mundo contém fenómenos naturais, que envolvem comportamentos e movimentos que a ciência tem conseguido definir através de fórmulas físicas e matemáticas. Contudo, ainda estamos muito longe de abranger a totalidade destes prodígios naturais que assistimos normalmente durante a nossa vida.

A identificação de objectos e movimento destes ao nível da percepção de um robô, apenas é executável pela existência do conceito de Visão Computacional, que é sem dúvida uma área em expansão nestes últimos anos pela diversidade de aplicações. Neste contexto, a detecção de objectos recorre a algoritmos elaborados que são nutridos de imagens obtidas do amplo ambiente em que os robôs estão envolvidos, contudo a análise do movimento é mais elaborada e divide-se pelas fases de detecção, seguimento e reconhecimento [15].

Para cada uma das etapas descritas na detecção de movimento, existem dificuldades inerentes, associadas à complexidade dos objectos e ao ambiente em que estão inseridos. Alguns exemplos desta complexidade são o aparecimento de novos objectos, o desaparecimento total ou parcial, a detecção em simultâneo de vários objectos [16], entre outros. Tais exemplos estão associados a problemas de oclusão, de ruído e alteração das condições de iluminação. Por outro lado, não existem modelos computacionais perfeitos e mesmo quando são adaptados a certos meios baseiam-se em aproximações à realidade, possuidoras de perturbações que não podem ser controladas pelas suas características não determinísticas [17].

De forma a contornar o problema, os modelos estocásticos são uma boa alternativa para abordagens determinísticas, dado que se mostram adequados ao manuseamento de dados multivariados e processos não-lineares ou não-Gaussianos, melhorando de forma significativa os resultados obtidos pelas técnicas mais tradicionais [18].

Sob uma perspectiva bayesiana, o seguimento consiste no cálculo recursivo do grau de certeza associado a cada estado em determinado instante, tendo em consideração os dados obtidos até esse momento. Para tal, é assumido que as técnicas de modelação produziram um sistema descritivo, sob a forma de uma equação diferencial estocástica para descrever a propagação de estados – o modelo do sistema [28]. Porém, numa perspectiva Discreta, o modelo de resume-se a medidas discretas no tempo que são corrompidas por ruído.

De forma a obter estimativas óptimas dos estados do sistema, combinam-se as medições estatísticas das incertezas com as medições obtidas no modelo do sistema, que de um modo geral torna a optimização da estimativa dependente do critério utilizado [17].

Foram estudados dois filtros no âmbito desta dissertação, Filtro de *Kalman* e de Partículas, formados pelos conceitos presentes na percepção do seguimento de objectos, a previsão e a correcção. A previsão, utiliza o modelo do sistema referido para antecipar a função densidade de probabilidade do estado no instante seguinte. Por outro lado, a correcção utiliza a medição mais recente de forma a alterar a função densidade de probabilidade prevista [17].

Foi também estudada uma modelação estatística, a Regressão Linear, com o objectivo de facultar uma comparação entre o uso de métodos probabilísticos e modelos estocásticos para o seguimento de objectos móveis.

2.3.1. Filtro de Kalman

O filtro de *Kalman* assume que a função densidade de probabilidade do estado de um sistema em cada instante de tempo segue uma distribuição Gaussiana, permitindo assim uma estimativa do estado minimizando o quadrado da média do erro [19]. É uma solução óptima para o seguimento se respeitar algumas restrições: o ruído deve ter uma distribuição Gaussiana de parâmetros conhecidos e a transição de estados representada pelo modelo do sistema tem de ser linear [19,20].

Este tipo de filtro assume que a dependência do estado actual para com o estado anterior é verdade de acordo com:

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

Onde, F_k é o modelo de transição de estado aplicado ao estado anterior x_{k-1} e B_k é o modelo de controlo da entrada aplicada ao vector de controlo u_k. Já w_k é o ruído do processo que é adoptado a partir de uma distribuição normal multivariada com média zero e covariância Q_k [21].

Apesar do filtro ser descrito apenas pela equação anterior, é caracterizado por ser um estimador recursivo e subdividir-se em duas fases, a fase de previsão e a fase de

actualização. Normalmente, estas fases são alternadas, sendo que a previsão é responsável por fazer avançar o estado e a actualização é integrada com a observação.

A fase de previsão é calculada à priori e divide-se em duas equações distintas, uma onde é estimado o estado $\hat{x}_{k|k-1}$ e outra onde é estimada a covariância $P_{k|k-1}$, dadas pelas funções que se seguem:

$$\hat{x}_{k|k-1} = F_k \hat{x}_{k-1|k-1} + B_k u_k$$

$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_k$$

Já a fase de actualização é processada à posteriori e reparte-se em duas equações, que de certa forma completam as anteriores, pois procedem actualização do estado $\hat{x}_{k|k}$ e da covariância $P_{k|k}$ através das seguintes funções:

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \tilde{y}_k$$

$$P_{k|k} = (I - K_k H_k) P_{k|k-1}$$

Tendo em conta as equações descritas, só é possível obter de forma quantitativa o valor das mesmas com o cálculo prévio do valor residual $\tilde{y}_k$, a evolução da covariância S_k e o ganho do filtro K_k, através das equações seguintes:

$$\tilde{y}_k = z_k - H_k \hat{x}_{k|k-1}$$

$$S_k = H_k P_{k|k-1} H_k^T + R_k$$

$$K_k = P_{k|k-1} H_k^T S_k^{-1}$$

Todavia, tal situação não se verifica se, por algum motivo, a observação não estiver disponível e para este caso a actualização é ignorada, sendo apenas realizadas as múltiplas etapas de previsão. Da mesma forma, se estiverem disponíveis múltiplas observações

independentes, ao mesmo tempo, são realizadas as várias etapas de actualização correspondentes [22].

2.3.2. Filtro de Partículas

Esta metodologia encontrava-se limitada a algumas aplicações, devido à sua complexidade computacional relativamente elevada. Porém, com a maior disponibilidade de recursos computacionais na última década, a filtragem por partículas tem-se tornado uma área de pesquisa muito promissora [23].

O objectivo de um filtro de partículas é estimar a densidade posterior das variáveis de estado, quando conhecidas as variáveis de observação. É um filtro projectado a partir de um modelo oculto de *Markov*, definido por um sistema composto por variáveis observáveis e variáveis ocultas. Isto é, as variáveis observáveis fazem parte do processo de observação e estão relacionadas com as variáveis ocultas, que fazem parte do processo relativo ao estado por meio de uma forma probabilística conhecida [24].

Um filtro de partículas genérico, assim como o filtro de *Kalman* estudado na secção anterior, estima a distribuição posterior dos estados ocultos utilizando para isso o processo de avaliação da observação, só assim consegue estimar os valores dos estados ocultos *X*, quando são conhecidos apenas os valores do processo de observação *Y*.

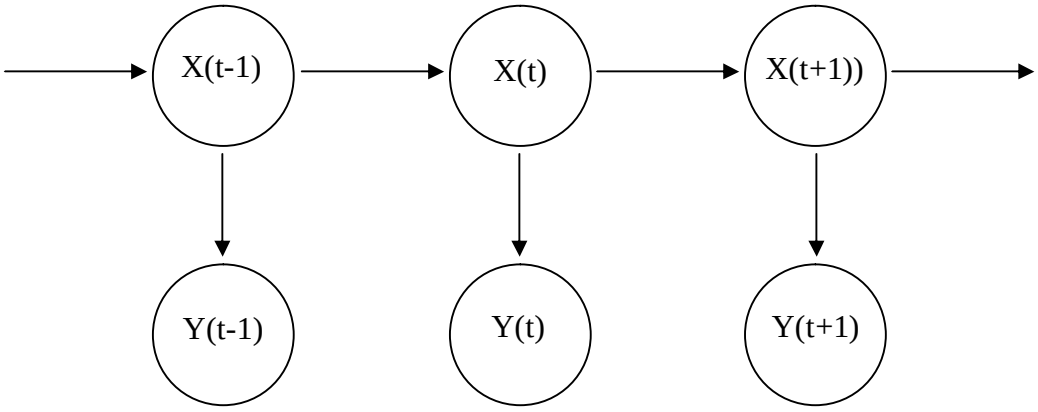

Diagrama 2 – Visão geral dos estados de um filtro de partículas

Por outras palavras, um filtro de partículas tem como principal função estimar a sequência de parâmetros escondidos, x_k com base apenas no valor de y_k, dados observados para k inicialmente zero e incrementado com uma unidade posteriormente.

É facto, que todas as estimativas Bayesianas de x_k seguem uma distribuição posterior $p(x_k \mid \mathrm{y}_0, y_1, ..., \mathrm{y}_\mathrm{k})$ [25].

O algoritmo de filtragem das partículas desenvolvido por Gordon em 1993, nomeado de *Sequential Importance Resampling (SIR)* [26], é um algoritmo que se aproxima de uma distribuição de filtragem $p(x_k \mid \mathrm{y}_0, y_1, ..., \mathrm{y}_\mathrm{k})$ por um conjunto ponderado de *N* partículas. O algoritmo segue 4 etapas distintas, sendo a primeira a inicialização do filtro que parte do conjunto anterior $p(x_0)$ de forma a atingir:

$$\{(\mathrm{x}_0^{*(\mathrm{i})}, \omega_0^{(\mathrm{i})}), \mathrm{i} = 1, ..., \mathrm{N}\}$$

A segunda etapa é a da avaliação, onde podem ser calculados os pesos das várias partículas e posteriormente normalizados, segundo as formulas:

$$\omega_{t-1}^{(\mathrm{i})} \propto \omega_{t-2}^{(i)} p\left(y_{t-1} \mid x_{t-1}^{*(i)}\right),\ i = 1, ..., N$$

$$\tilde{\omega}_{t-1}^{(\mathrm{i})} = \frac{\omega_{t-1}^{(\mathrm{i})}}{\sum_{i=1}^{N} \omega_{t-1}^{i}},\ i = 1, ..., N$$

Segue-se a etapa da reamostragem, ou por outras palavras, a etapa da selecção, onde são reamostradas as partículas $x_t^{(\mathrm{i})}$ com probabilidade $\tilde{\omega}_{t-1}^{(\mathrm{i})}$, obtendo-se assim N partículas aleatórias independentes e identicamente distribuídas conforme $p(x_t \mid \mathrm{y}_{l:t})$. Através da fórmula:

$$\tilde{\omega}_t^{(\mathrm{i})} = \frac{l}{N},\ l = 1, ..., N$$

Por fim, a etapa da previsão, ou evolução, que faz avançar os estados no tempo $t-1$ para t, usando a seguinte equação da evolução dos estados:

$$x_t^{*(\mathrm{i})} = p(x_t \mid x_{t-1}^{(i)}),\ i = 1, \ldots, N$$

Este método tornou-se uma alternativa importante para o filtro de Kalman descrito anteriormente, já que nesta metodologia as distribuições contínuas são aproximadas por medidas aleatórias discretas e não envolvem uma linearização em torno das estimativas actuais, mas sim aproximações na representação das distribuições desejadas.

2.3.3. Regressão Linear

A regressão linear em estatística é um método usado para estimar o valor esperado de uma variável, sendo conhecidos os valores de outras variáveis . O seu nome "linear" deve-se ao facto de ser considerada a relação da resposta às variáveis, como uma função linear de alguns parâmetros, porém, os métodos que não o verificam são chamados de não-lineares.
Usada em amplas aplicações práticas, a regressão linear é uma das primeiras formas de análise recursiva a ser rigorosamente estudada. Isto porque, os parâmetros são desconhecidos mas o modelo depende linearmente deles, contudo, são mais fáceis de ajustar que os modelos não-lineares dos seus, pois as propriedades estatísticas dos estimadores resultantes são fáceis de determinar [27].
A regressão linear é caracterizada pelo seguinte modelo:

$$Y_i = \alpha + \beta X_i + \varepsilon_i$$

Onde Y_i é o valor que pretendemos atingir, α representa a intercepção da recta com o eixo vertical, βX_i o valor do coeficiente angular aplicado à variável independente que representa o factor explicativo na equação e por fim ε_i, a variável que inclui todos os factores residuais e erros de medição.

Mas para que o modelo acima descrita seja aplicável é necessário que os erros associados se encaixem em uma das duas suposições: os erros são variáveis normais com a mesma variância ou então são independentes da variável explicativa X. Se os erros associados à medição cumprirem uma destas suposições, com a aplicação da fórmula da regressão linear, serão exibidos os coeficientes de regressão da equação linear que envolve uma ou mais variáveis independentes, prevendo assim o melhor valor da variável dependente.

Capítulo 3 – Trabalho desenvolvido

Neste capítulo descreve-se o trabalho desenvolvido no âmbito desta dissertação de mestrado, sendo abordado o antigo integrador do agente CAMBADA e posteriormente as alterações executadas neste. Torna assim possível uma análise mais cuidada em relação às melhorias que são indicadas com a finalidade de as compreender bem.

3.1. Agente CAMBADA

O agente CAMBADA é constituído por um sistema assente em regras, comportamentos, controladores, estruturas de dados e um integrador responsável pela integração da informação. A forma de funcionamento, de um modo geral, tem em conta o estado do mundo aos olhos do robô e através de uma estratégia bem definida, são adoptadas regras pelo agente consoante as suas tomadas de decisão em cada instante, estando estas associadas à ocorrência de eventos de jogo e outros.

O diagrama de classes do agente CAMBADA é definido pelas classes presentes no Diagrama 3, onde são também perceptíveis as dependências existentes entre elas. De notar que a árvore de classes exibida, parte da raiz *Cambada*, sendo que os seus ramos estão expostos de uma forma generalizada, devido ao seu número e dependências associadas.

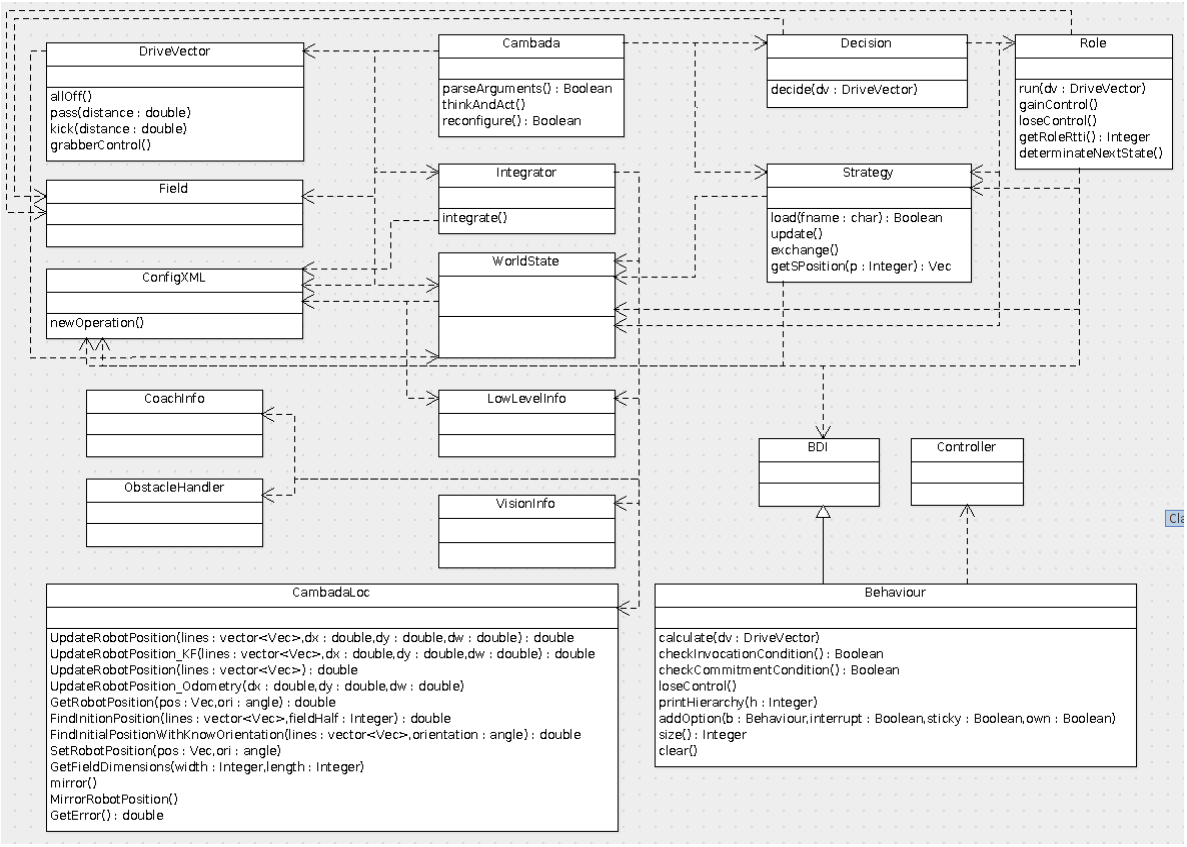

Diagrama 3 – Diagrama de classes do agente CAMBADA

No sentido de explicar o funcionamento do agente, passo a explicar resumidamente qual o papel de cada uma das classes que estão relacionadas a ele:

- *Cambada*: classe responsável pelo fio de execução do agente, constituída por métodos públicos que manipulam o ciclo de vida do agente.

- *DriveVector*: definida por métodos associados à actuação do sistema de chuto e velocidades associadas à deslocação física do robô, a esta classe compete a execução do passe, chuto e movimento, sendo que está implícita a necessidade de agarrar a bola nos primeiros casos.

- S*trategy*: está reflectida no acesso a esta classe a estratégia de jogo do agente, isto é, estão acessíveis através desta classe as informações relativas à formação que a equipa deve assumir nas diversas situações de jogo.

- *Decision*: definida apenas com um método público, é nesta classe que se ponderam as tomadas de decisão do agente. É eleita uma regra que tem em conta as diversas situações do estado do mundo, assim como a estratégia da equipa.

- *Role*: é através desta classe que se definem ordens lógicas, que através do um conjunto de comportamentos necessários, respondem às diversas situações do jogo. Por exemplo, numa situação de penalti, o robô deve executar no mínimo três comportamentos distintos: deslocar-se para a marca de penalti, agarrar a bola e chutar, está sequência lógica é uma regra para a situação de jogo relativa a marcação de uma grande penalidade.

- *Behaviour*: os vários comportamentos que estão definidos para a movimentação e atitude do robô durante um jogo, são acessíveis através desta classe. Sendo estes usados e associados em conjuntos de forma lógica, pela classe *Role* nas suas implementações.

- *BDI*: classe responsável pela implementação da lista de prioridades imposta por uma *Role*, sendo por isso uma generalização da classe *Behaviour*.

- C*ontroller*: classe que torna disponíveis os controladores de movimento simples, isto é, o movimento rectilíneo ou em arco do robô de um ponto para o outro.

- *WorldState*: o estado do mundo mantido pelo agente, encontra-se espelhado nesta classe. Sendo que as principais estruturas existentes nesta, são referentes à bola, ao robot e aos obstáculos.

- *integrator*: classe responsável pela integração da informação proveniente dos diversos meios de observação do robô e por conseguinte, pela actualização do estado do mundo usado pelo agente.

- *CambadaLoc*: os métodos disponibilizados por esta classe, demonstram que a sua competência passa pela localização do agente, perante as diferentes fontes de informação.

- *LowLevelInfo*: classe que cuida da comunicação com o baixo nível, isto é, troca de informação relativa ao IMU, odometria, sistema de chuto e movimento, entre outros. Mantendo assim a estrutura que reflecte os dados de baixo nível sempre actualizada.

- *VisionInfo*: os valores relativos ao sistema de visão são estruturados nesta classe, que por sua vez, fornece métodos de acesso simples à informação observada.

- *CoachInfo*: informação relativa à equipa de jogo está acessível através dos métodos desta classe, usada principalmente na fase de integração.

- *ObstacleHandler*: a esta classe está associada a informação relativa aos obstáculos que estão presentes no campo de jogo, uma lista que cresce e diminui mediante o que é observado.

Apesar de cada uma das classes descritas ter um papel importante para o agente, serão abordadas agora as que são consideradas fundamentais no ciclo de vida do agente a partir da sua raiz de execução, que reside na classe *Cambada*. Repetidamente, a função *thinkAndAct* faz o agente pensar e agir, conceitos que se resumem a etapas distintas. Inicialmente, acontece a integração da informação por parte do *Integrator*, seguida de uma decisão que determina a *Role* que se adequa ao estado do jogo, terminando por fim com a aplicação desta, através da execução de *Behaviours*. Naturalmente, todas as classes que incorporam o integrador estão directamente envolvidas na execução do processo descrito, consoante a dependência que têm entre si.

3.2. Integrador do agente CAMBADA

O integrador do agente CAMBADA é um módulo inteligente, responsável pela integração da informação proveniente dos vários sensores que incorporam o robô, isto é, a informação originária do sistema de visão, *encoders* dos motores, tensão das baterias e condensador do *kicker*, sem esquecer a informação partilhada pelos restantes robôs.

O processo de integração é realizado em cada iteração do agente, como referido anteriormente, sendo organizado em várias etapas de integração. Inicia com uma filtragem do conhecimento adquirido, seguindo-se as etapas que empregam toda a actualização da informação relativa ao estado do mundo.
A fim de aprofundar um pouco mais as fontes usadas para cada um dos casos de uso do integrador, passo a descrever as classes que albergam as estruturas de dados usadas para guardar a informação usada pelo integrador nas várias etapas de integração:

- *LowLevel*: classe que realiza através dos seus métodos a ponte com o baixo nível, munido de uma estrutura de dados referente à informação directa que é obtida do *IMU* e da monitorização do sistema.

- *Compass*: possui uma estrutura de dados que contém a informação tratada do *LowLevel* referente à odometria e IMU com métodos específicos de leitura e actualização, estando assim acessível a orientação do robô de acordo com a cor da equipa.

- *Coach*: como o próprio nome indica, esta classe alberga a informação referente à gestão da equipa, isto é, dados relativos à estratégia seleccionada, à cor da equipa, à cor da baliza, entre outras.

- *Vision*: é através dos métodos existentes nesta classe, que a estrutura interna é preenchida com os valores devidamente estudados do sistema de visão. Isto é, através do uso de algoritmos implementados nesta classe, a informação originária do sistema de visão fica acessível de forma simples (pontos das linhas de campo, posição das bolas e pontos pretos).

- *EgoMotion*: a necessidade de estimar as velocidades implícitas à movimentação do robô surge esta classe, composta por alguns métodos que alimentam o algoritmo interno com a localização do robô e outros que fornecem a estimação do mesmo.

- *WorldState*: o nome não deixa dúvidas, já que esta classe disponibiliza a estrutura de dados que abriga toda a informação referente ao mundo que rodeia o robô. Por outro lado, a classe contém uma numerosa gama de métodos que servem de apoio aos restantes módulos do agente, uma vez que a estrutura que contém é acessível praticamente em todos os módulos.

- *BallKalmanFilter*: a descrição do movimento da bola de jogo é complexa, por isso, com o objectivo de minimizar o ruido da observação e dar precisão aos valores calculados para a velocidade e posição da bola. Esta classe possui uma implementação de um filtro de *Kalman*, e métodos que servem para alimentar o filtro e disponibilizar as valores calculado por este.

- *DB*: a troca e salvaguarda de informação proveniente dos vários agentes são asseguradas por esta classe, que faz jus ao seu nome e mantém uma estrutura que acomoda a informação de todos os agentes.

3.3. Alterações no integrador do agente CAMBADA

Um dos objectivos principais desta dissertação passou pela modelização do integrador do agente CAMBADA, que se revelou complexo na sua implementação depois de um estudo cuidado. Realidade que associada à análise do estado da arte, referido no capítulo 3, sustentou a motivação para a criação da nova estrutura modular que sustém o novo integrador.

O estudo executado ao integrador anterior mostrou que este se resumia a uma interface simples, pois possuía apenas o método público nomeado de *integrate()*. Contudo, detinha todos os casos de uso ilustrados no diagrama 4, sendo que alguns estavam implementados em métodos internos do integrador e outros directamente na função que contava com 475 linhas de código. Para ser mais fácil a percepção dos casos de uso envolvidos, passo a descrever individualmente cada um:

- *Update_LowLevel*: etapa em que é executada a actualização da informação do baixo nível, isto é, dados do *IMU* e da monitorização do sistema;

- *Update_PlayerInfo*: fase onde são actualizados os dados do robô relativos à equipa, isto é, dados de estratégia, cores da equipa e da baliza, entre outros.

- *Update_PlayerLocalization*: nesta etapa é calculada a localização do robô que é executada através do objecto *CambadaLoc*, alimentado pelos valores obtidos do sistema sensorial.

- *Update_PlayerVelocity*: dependente da execução da anterior, esta tarefa usa o objecto *EgoMotion* para calcular a velocidade do robô, de forma a actualizar as estruturas que salvaguardam estas informações.

- *Update_OthersRobotsInfo*: este procedimento é responsável pela actualização de um campo da base de dados, com a finalidade de informar os restantes agentes que está em execução.

- *Update_Obstacles*: a informação referente aos obstáculos é actualizada nesta fase, possuindo como base a lista existente e a detecção de novos obstáculos pelo sistema de visão, bem como os obstáculos partilhados.

- *Update_BallData*: os dados relativos à posição e velocidade da bola de jogo, é actualizada nesta etapa do integrador recorrendo ao uso do filtro de Kalman, que é alimentado pelos valores obtidos do sistema de visão.

- *Update_GameState*: após as etapas anteriores estarem concluídas, o resultado é a alteração do estado do robô e do mundo que o rodeia. Dessa forma, esta fase é destinada a actualizar o estado de jogo, que tem em conta o estado anterior e os novos valores.

- *Predict*: a última etapa diz respeito à previsão de situações de jogo em que o robô tem de tomar decisões que fogem do âmbito normal, como é o exemplo da colisão com a bola.

De notar que a ordem pelos qual foi referido cada caso de uso, reflecte a ordem de execução dos mesmos, visto que existem dependências a serem respeitadas para que os dados alusivos ao estado do robô, da bola, dos obstáculos e do mundo que os rodeia sejam correctamente calculados. Todos os casos de uso encontravam-se implementados no módulo *Integrator*, directamente na função *integrate()* ou através de métodos internos a este. Assim sendo, o módulo reunia toda a complexidade associada à implementação dos passos essenciais da integração, que se estendia por 1976 linhas de código, motivo que levou à criação de um modelo que assegura agora uma divisão por conceitos.
O diagrama 4 ajuda também a perceber esta divisão, sendo que os casos de uso presentes nele, distribuem-se pelos conceitos: *LowLevel*, *Player*, *Ball* e *GameState*.

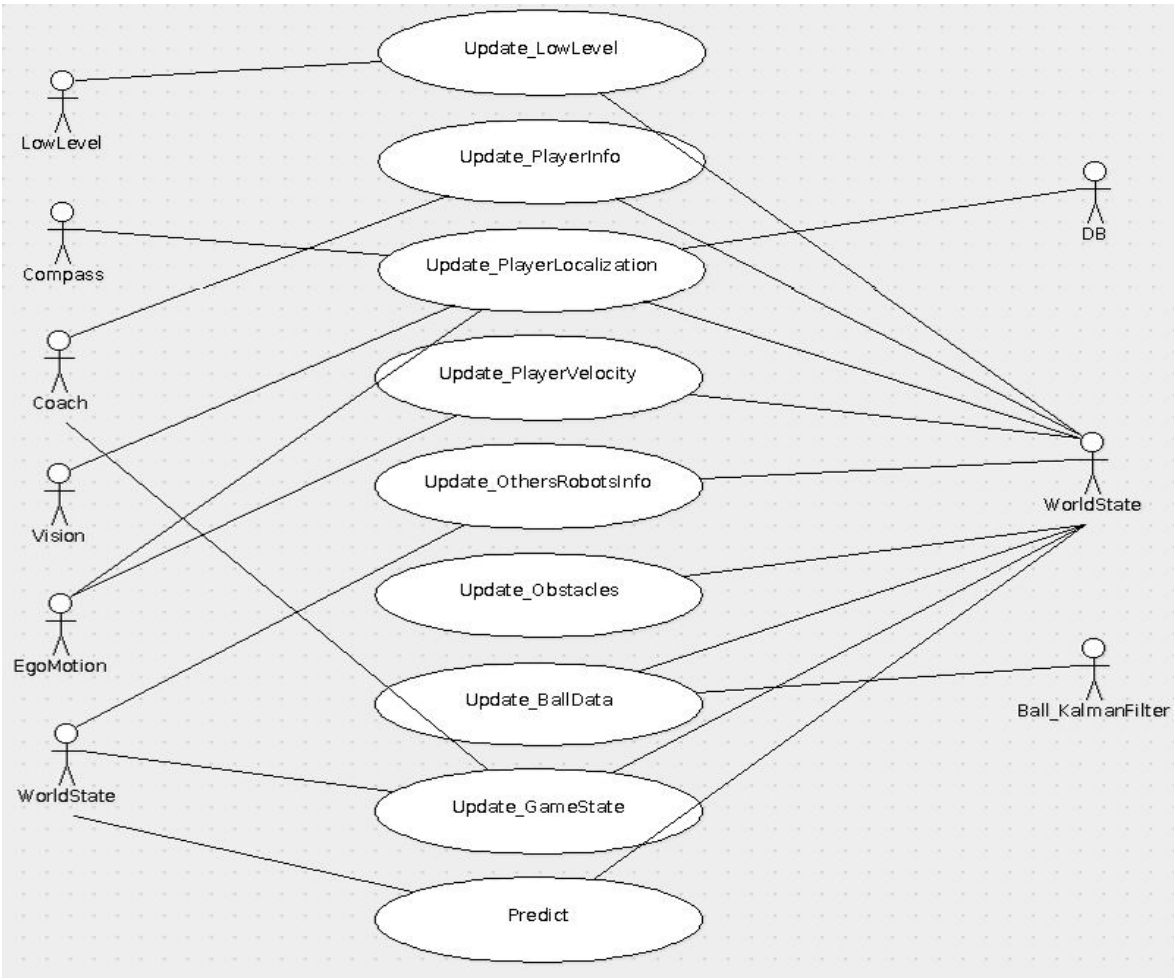

Diagrama 4 – Casos de uso relativos ao integrador do agente

3.3.1. Interfaces Filter e Localization

O estado da arte veio revelar que existe uma panóplia de soluções relativas à caracterização da velocidade e posição de objectos móveis, assim sendo, com a finalidade de tornar o novo integrador adaptável às várias soluções existentes, ou novas que possam surgir, foram definidas duas interfaces com métodos adequados às necessidades.

A primeira interface criada, tem o objectivo criar uma abstracção entre o uso de filtros e a sua implementação, isto é, o desenvolvimento de melhorias associadas à integração, ou outras, não necessitam de conhecer a implementação da filtragem, apenas usá-la, sabendo de antemão que os métodos acessíveis são os da interface *Filter*, ilustrada no diagrama 5.

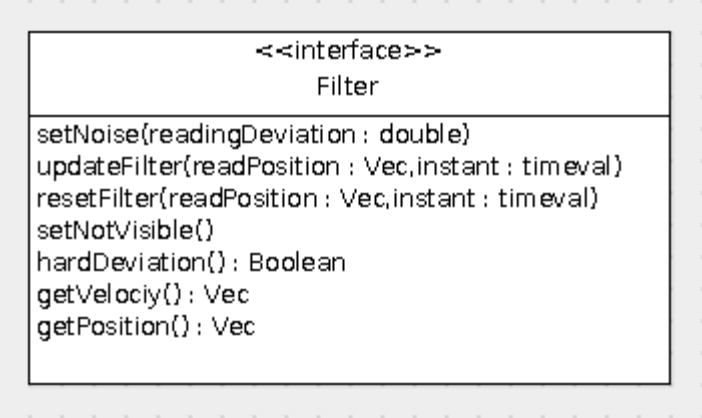

Diagrama 5 – Interface Filter

Os métodos que formam esta interface, foram definidos após uma análise cuidada das necessidades subentendidas na implementação dos conceitos de filtragem abordados no segundo capítulo, disponíveis num pacote *Filters* criado para o efeito no módulo integrador. Assim sendo, os métodos acessíveis nesta interface são:

- *setNoise*: método que de um modo geral é usado para injectar ruído do processo no filtro, pois como vimos na secção três do segundo capítulo, o ruído está implícito na forma de laboração dos filtros servindo como que uma margem de erro para as medidas filtradas.

- *updateFilter*: como o próprio nome indica, os métodos assim nomeados têm a finalidade actualizar o filtro, um método tem em conta que o filtro é alimentado com a nova posição e o instante em que esta foi observada, sendo que o outro apenas actualiza o filtro sem que exista uma nova posição.

- *resetFilter*: a necessidade de reiniciar o filtro ao longo de um processo de filtragem, é a razão para este método ser incluído na interface, uma vez que a sua utilidade é exactamente essa, reiniciar os valores internos do filtro.

- *setNotVisible*: uma alternativa ao método anterior é este, que se destina a declarar a ausência do objecto na observação, sem que para isso seja necessário o recomeço do filtro. Isto é, actualizar o filtro sem introduzir uma nova posição.

- *hardDeviation*: este método surge da necessidade de, em qualquer momento, saber se ocorreu um desencaminhamento nos valores do filtro. Este deslocamento é considerado para 3 repetições de desfasamentos na ordem dos 0,15 metros mais o ruído associado ao filtro.

- *getVelocity*: o método que devolve a velocidade do objecto calculada pelo filtro na sua última iteração.

- *getPosition*: semelhante ao anterior, este método declara a posição do objecto calculada na última iteração.

De notar que a criação de novas implementações para este conceito de filtragem devem respeitar esta interface, visto que actualmente a implementação do integrador apenas está preparada para tal, por outro lado, o uso desta interface associado a melhorias no agente ou novas utilizações na implementação do mesmo, deve ser possível mantendo sempre o seu princípio de funcionamento.
Assim como a interface anterior, a interface *Localization* que está ilustrada no diagrama 6, tem como objectivo criar uma abstracção entre o seu uso da localização do robô e as diferentes formas de implementar os conceitos associados a esta, isto é, atribuir a

complexidade das várias soluções à implementação sem que para isso tenha de afectar o desenvolvimento de melhorias ou novas tarefas que necessitem usar a localização do robô.

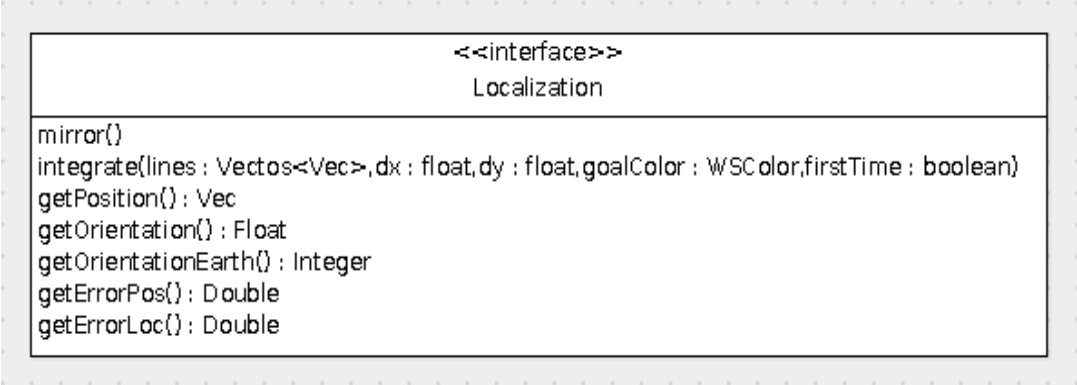

Diagrama 6 – Interface Localization

A criação desta interface é fundada pela noção de evolução, que está presente no nosso dia-a-dia com o progresso constantes na área da robótica. Os robôs CAMBADA são alvo de alterações esporádicas, para assim acompanharem este progresso tecnológico, da mesma forma, a implementação do seu agente deve também estar preparada para esta evolução.
Por outro lado, esta interface é uma generalização da classe *CambadaLoc*, uma vez que possui praticamente a mesma função, acrescentando as noções de odometria e cor da baliza. Assim sendo, são propostos por esta interface os seguintes métodos:

- *mirror*: método que efectua a inversão dos valores relativos à posição e orientação do robô.

- *integrate*: método onde são efectuados todos os cálculos da localização em cada observação, isto é, a integração dos dados internos relativos à localização é executada aqui, sejam eles provenientes do *IMU* ou de outra fonte que venha a ser usada no futuro.

- *getPosition*: método que devolve a última posição calculada pela implementação.

- *getOrientation*: método usado para obter a orientação do robô determinada em relação ao ponto de referência que em geral é a baliza adversária.

- *getOrientationEarth*: método que retorna a orientação do robô computada em relação à terra.

-*getErrorLoc*: método que retorna a estimativa do erro relacionado ao processo de localização.

3.3.2. Classes IntegratePlayer e IntegrateBall

Inicialmente, a linha de pensamento que foi seguida para a modelação do novo integrador, baseou-se em identificar e dividir a integração dos vários conceitos existentes neste, com o objectivo de repartir a complexidade existente no integrador anterior. Por outro lado, o ponto de partida foi a análise feita na Secção 2 deste capítulo, uma vez que foram identificadas as etapas do integrador e facilmente divididas por áreas: baixo nível, robô, obstáculos, bola e estado de jogo.

Foi no acto de efectivar esta divisão que surgiram duas novas classes, nomeadas de *IntegratePlayer* e *IntegrateBall*, classes que albergam a integração relativa ao robô e à bola respectivamente. Desta forma, a complexidade do integrador ficou reduzida às restantes áreas de integração referidas, pois a sua implementação não se revelou tão complexa ao ponto de existir a necessidade de formarem uma nova classe, isto é, já se encontravam devidamente divididas por métodos internos e por classes de apoio.

É visível no Diagrama 7 que a nova implementação do integrador do agente CAMBADA, nesta primeira versão, passa a usufruir de dois novos objectos internos, capazes de tratar a integração dos dados observados relativamente ao robô e à bola, respectivamente o *integrate_player* e *integrate_ball*.

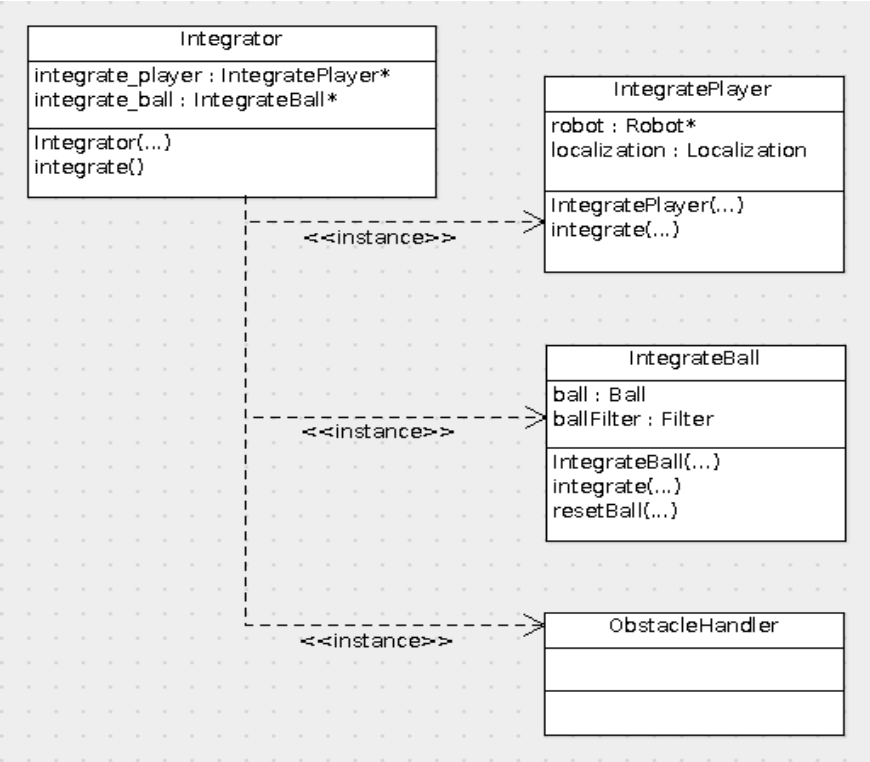

Diagrama 7 – Diagrama de classes do novo integrador (1ª versão)

Uma análise mais pormenorizada às classes introduzidas dá-nos a indicação de que a classe *IntegratePlayer* é composta por uma referência nomeada *robot* e aponta para a estrutura *Robot*, que abarca com as informações relativas ao estado do robô. Também possui um objecto *localization* do tipo *Localization*, interface que já foi abordada na secção anterior. Nesta fase inicial da modelização de todo o integrador, surgem duas implementações associadas à interface *Locatization*, visíveis no Diagrama 7, apenas diferem do uso ou não da odometria.

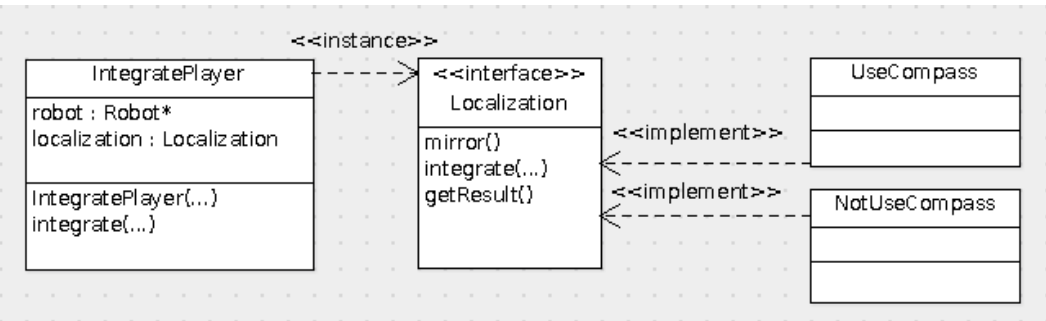

Diagrama 8 – Diagrama de classes da estrutura associada ao IntegratePlayer

Desta forma, o novo integrador fica capaz de adquirir novas implementações relativas à localização do robô, estando para isso dependente da alteração do *Hardware* utilizado e da introdução de novas ideias.

A classe *IntegrateBall* visível no Diagrama 8, é também composta por elementos internos importantes, porque servem de apoio à integração dos dados relativos à bola. O objecto *ball* do tipo *Ball* é a estrutura que serve para guardar todos os dados obtidos pela computação dos valores de observação e outros executados nesta integração, já o objecto *ballFilter* é do tipo *Filter*, uma interface que como foi referido na secção anterior, é dotada de implementações de métodos de filtragem, para que os valores computados sejam os mais próximos da realidade possível.

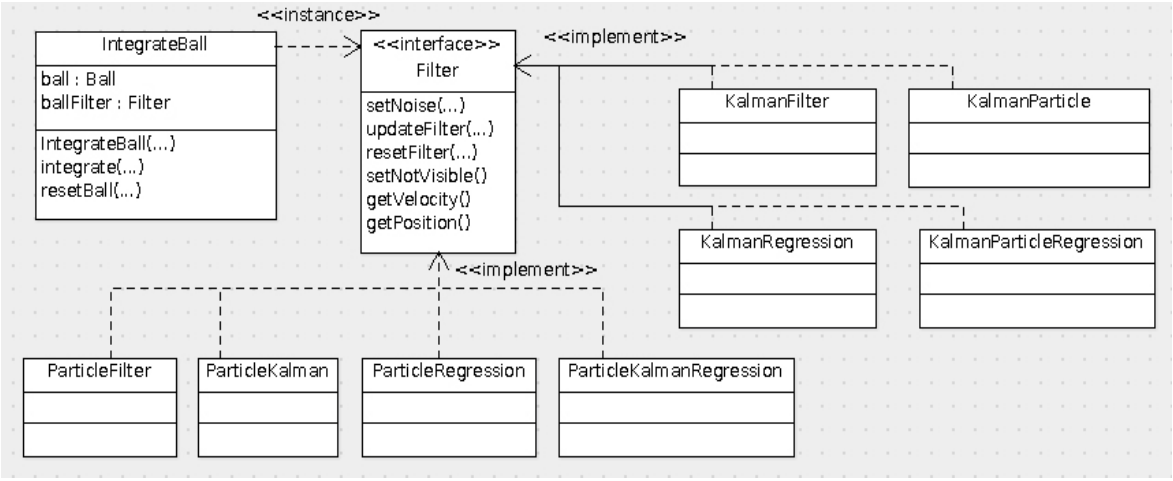

Diagrama 9 – Diagrama de classes da estrutura do IntegrateBall

O novo integrador conta também com oito implementações para a interface *Filter*, usada na classe *IntegrateBall*. Duas delas, referentes ao *KalmanFilter* e ao *ParticleFilter*, são implementações directas dos algoritmos relativos aos filtros de *Kalman* e de Partículas respectivamente. As restantes implementações, passam pelo cruzamento entre estes mesmos dois filtros e uma regressão linear:

- *KalmanParticle*: implementação da interface *Filter* que possui internamente dois filtros, um filtro de Kalman que é alimentado pelas posições da bola observadas e um filtro de Partículas que é sustentado pelas posições que resultam da filtragem do filtro de *Kalman*. Desta forma o resultado deste cruzamento é a filtragem resultante do filtro de Partículas, sendo este o responsável pelo cálculo final das posições e velocidade da bola.

- KalmanRegression: esta implementação é nutrida apenas por um filtro de *Kalman* que é actualizado com as posições da bola observadas, que por sua vez introduz os valores filtrados por si numa regressão linear. A regressão linear, aplica o modelo que a define e quando solicitado, devolve o valor da velocidade da bola, competindo ao filtro devolver o valor da posição.

- KalmanParticleRegression: uma implementação que junta a ideia existentes nas duas anteriores, visto que é composta por dois filtros, um de *Kalman* e um de Partículas, seguidos de uma regressão linear. Assim como na primeira implementação, o filtro de *Kalman* é alimentado pelas posições da bola observadas, sendo este que sustenta o filtro de Partículas que lhe sucede. Por fim, é o filtro de Partículas que insere os seus resultados na regressão linear, ficando esta responsável por determinar os valores referentes à velocidade da bola, e o filtro de Partículas a posição.

- ParticleKalman: implementação que contém dois filtros, o de Partículas que é sustentado pelos valores obtidos na observação da bola, que posteriormente à execução da filtragem que lhe compete, alimenta o filtro de *Kalman*. Desta forma, é o filtro de *Kalman* o responsável por indicar os valores relativos à posição e velocidade da bola.

- ParticleRegression: esta implementação contém um filtro de Partículas que executa a filtragem através dos valores que vai obtendo da observação da bola, inserindo os resultados dessa filtragem numa regressão linear. Assim, a regressão linear com o seu modelo interno é capaz de devolver o valor da velocidade da bola quando necessário, sendo o filtro o responsável por conceder a posição da bola.

- ParticleKalmanRegression: esta última implementação contém dois filtros, um filtro de Partículas, alimentado pelas observações da bola que após a filtragem que lhe pertence insere os resultados num filtro de *Kalman*. Fica assim o filtro de Kalman responsabilizado pela determinação das posições filtradas da bola, encarregando a determinação da velocidade da bola à regressão linear que alimenta.

É perceptível que nas novas classes de integração e na classe integrador, o método *integrate* é comum. Apesar de ser implementado de forma diferente em todas as classes, possui a mesma finalidade que é integrar a informação referente ao conceito incutido a classe onde se encontra.

3.3.3. Estrutura e implementação

A divisão ponderada para o novo integrador nas diferentes áreas de integração, motivaram a criação de novas classes e por consequência a concepção de novas zonas para o desenvolvimento e avanço das mesmas. Detendo a modelação como objectivo, a restruturação do módulo integrador foi inevitável, criando assim as áreas representativas às realidades de integração que distinguimos neste capítulo.
A estrutura do novo integrador ganhou assim dois novos pacotes, nomeados de *filter* e *localization*, que albergam as diferentes implementações relativas às interfaces da localização do robô e à filtragem dos dados observados da bola.
Podemos observar no diagrama 10 a estrutura de classes que constituem o novo integrador, sendo que se distribuem pelos pacotes da seguinte forma:

- Pacote *integrator*: é a raiz do integrador, portanto as classes nele presentes são: *Integrator*, *IntegratePlayer*, *IntegrateBall*, *ObstacleHandler* e *ObstaclePositionKalman*. Também as interfaces *Filter* e *Localization* estão acessíveis aqui.

- Pacote *Filter*: contém as implementações da interface *Filter* abordadas na secção anterior: *KalmanFilter*, *KalmanRegression*, *KalmanParticle*, *KalmanParticleRegression*, *ParticleFilter*, *ParticleKalman*, *ParticleRegression* e *ParticleKalmanRegression*.

- Pacote *Localization*: aloja as implementações da interface *Localization* que são: *UseCompass* e *NotUseCompass*.

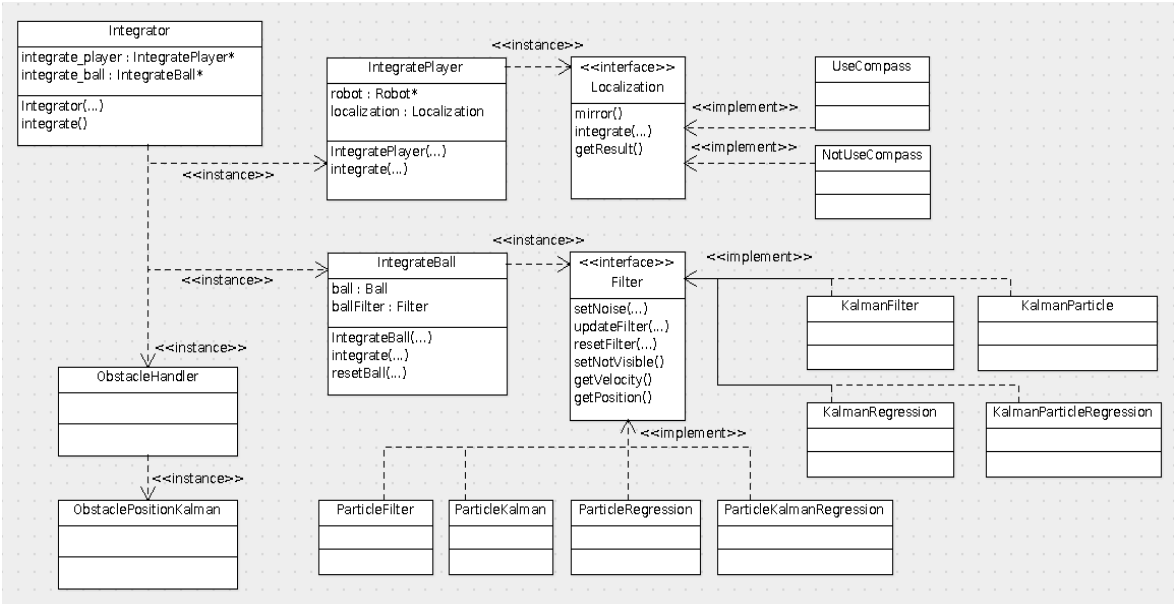

Diagrama 10 – Diagrama de classes completo do novo integrador

No que diz respeito à implementação, o novo integrador viu a sua função *integrate* reduzida no número de linhas, isto é, passou das 475 linhas para as 179 actuais. De notar foi também a redução de linhas do módulo integrator, que passou das suas 1976 linhas para as actuais 1183. A redução fez-se notar principalmente no número de métodos privados, pois passou dos seus 22 métodos para apenas 9 existentes agora.

Um proveito mais importante ainda para o módulo *integrator*, foi o facto de ter ganho uma estrutura nova e sustentada por um modelo simples, que divide a complexidade da integração pelos vários conceitos de integração existentes, sendo ao mesmo tempo capaz de disponibilizar o uso de filtros distintos e a possibilidade de evoluir facilmente com o desenvolvimento de novas ideias.

Capítulo 4 – Resultados

Neste capítulo são abordados os testes efectuados ao novo integrador e os resultados obtidos, de forma a verificar o bom funcionamento da nova estrutura e qual a melhor implementação para a filtragem dos valores observado em relação à bola de jogo.

4.1. Testes

O novo integrador foi já testado em duas competições pela equipa CAMBADA, no *Iran Robotics Open 2014* e no *Portuguese Robotics Open 2014*. Contudo, os vários testes efectuados ao novo integrador não se resumiram ao uso do mesmo em competição, mas também à tentativa de o tornar mais eficiente relativamente à detecção da posição e velocidade da bola de jogo.

Dessa forma, os resultados presentes na secção dois deste capítulo dizem respeito aos testes executados à modelização do novo integrador, tendo em conta as diversas implementações para a filtragem das observações da bola. Foram praticados dois tipos de testes, que contaram com uma bola parada em diferentes posições estratégicas e com um robô que possuiu movimentos distintos.

Cada um dos testes foi repetido para cada uma das implementações da interface *Filter* abordada na secção três do terceiro capítulo, isto é, as implementações referentes ao *KalmanFilter, KalmanParticle, KalmanRegression, KalmanParticleRegression, ParticleFilter, ParticleKalman, ParticleRegression e ParticleKalmanRegression.* De notar que o número de partículas usadas nas implementações que usaram o filtro de partículas foi 10, número de particular por defeito existente no filtro.

4.1.1. Robô em rotação

Neste primeiro teste foram elaboradas cinco experiências para cada implementação, em que o robô está posicionado junto da linha da grande área da baliza, como mostra a imagem 9, executando um movimento repetitivo de rotação sobre si mesmo. A bola foi colocada a uma distância inicial de um metro do robô, posteriormente deslocada um metro para a frente em cada experiência, sendo a deslocação executada paralelamente à linha da grande área.

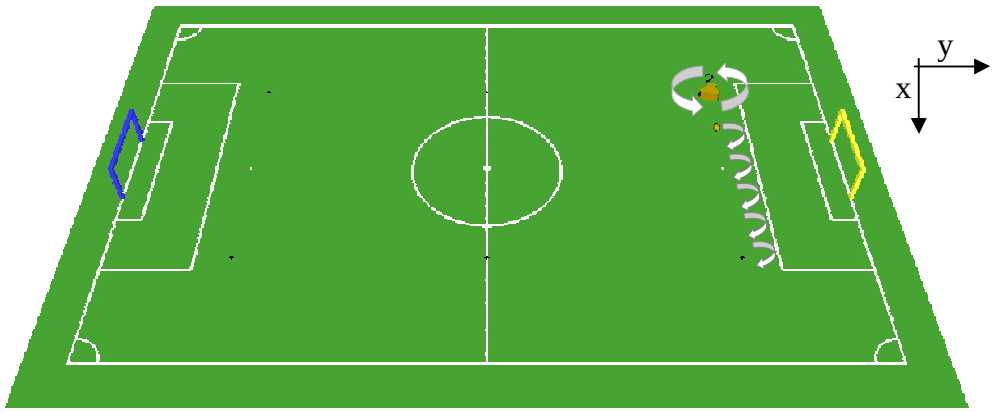

Figura 9 – Representação gráfica do primeiro teste efectuado

Cada experiência contou com três velocidades de rotação distintas para o robô com uma amostragem de mil observações: inicialmente parado, depois uma rotação de 90 rotações por minuto e por fim, uma rotação de 180 rotações por minuto.

4.1.2. Robô em translação

Para o segundo teste foram também executadas cinco experiências para cada implementação, contudo o robô excuta agora um movimento de translação paralelamente à linha da grande área da baliza, como mostra a imagem 10. Como no teste anterior, a bola foi colocada a uma distância inicial de um metro e posteriormente deslocada um metro para a frente em cada uma das experiências, mas em relação ao ponto negro do lado direito da baliza.

Neste teste foi usada sempre a mesma velocidade para o robô, sendo que, de forma semelhante à anterior, a amostragem de cada experiência contou com mil observações. De notar que o robô em cada uma das extremidades executava um movimento de rotação sobre si próprio, para que o movimento fosse executado com orientações diferentes.

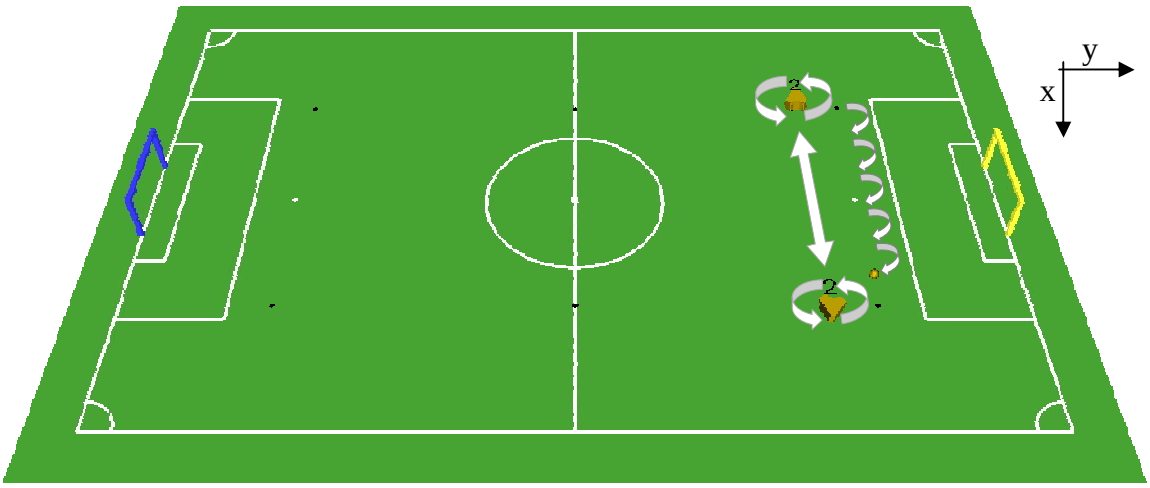

Figura 10 – Representação gráfica do segundo teste efectuado

4.2. Resultados

Nesta secção são ilustrados graficamente os resultados para ambos os testes, diferenciando para cada um deles o resultado obtido, com o uso das diferentes implementações para a filtragem da bola. Todos os testes aqui presentes foram executados no simulador que integra o projecto CAMBADA e dessa forma a imprecisão na colocação da bola teve a sua influência no resultado final obtido.

Os gráficos referentes ao primeiro teste, identificam as posições da bola obtida em relação ao robô, sendo identificadas por cores diferentes as várias experiências que pretendiam diferir apenas em 1 metro da distância. Deve ser referido contudo, que as 5 posições que a bola possuiu foram únicas, isto é, para cada uma das posições foram executadas as observações sem que a posição da bola e do robô fossem alterados.

Por outro lado, os gráficos relativos ao segundo teste, mostram as posições da bola em relação ao ponto negro como foi referido na secção anterior deste capítulo. Foram também usadas cores distintas para distinguir as diferentes experiências, que assim como para o teste anterior, pretendiam diferenciar-se apenas em 1 metro de distância entre elas.

A escala usada pelos diversos gráficos é diferente, bem como a relação entre os eixos *x* e *y*, pela simples razão de que existe uma grande discrepância entre as grandezas associadas. Pretende-se assim, ilustrar com mais precisão os valores associados ao erro de filtragem.

4.2.1. Resultado para o primeiro teste

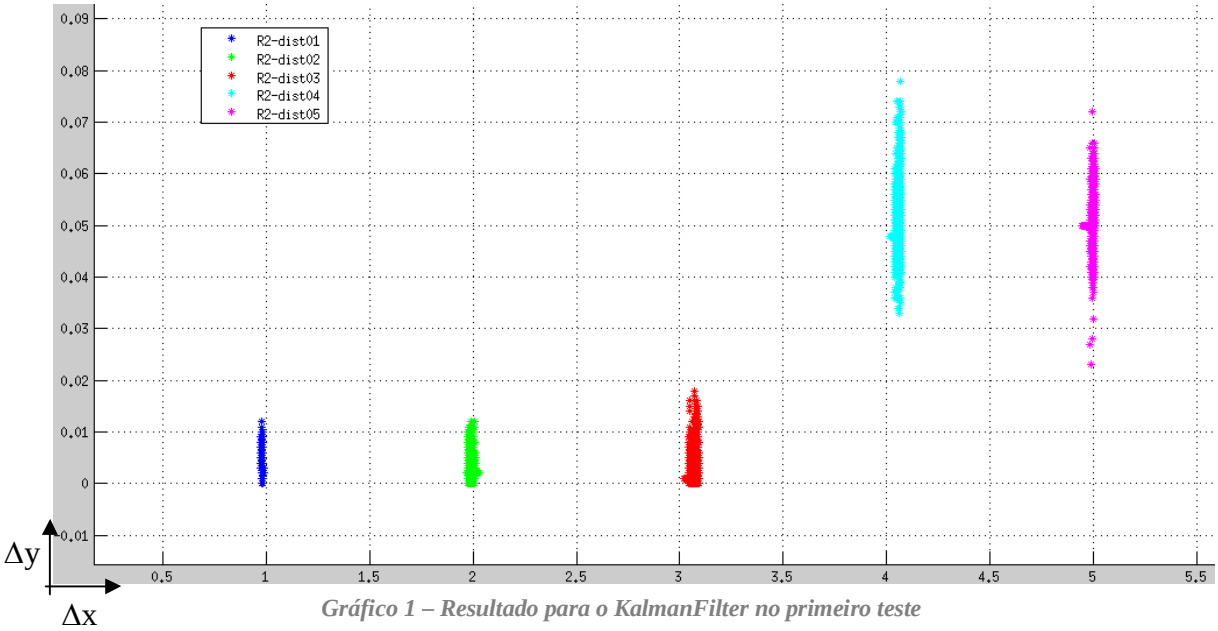

Gráfico 1 – Resultado para o KalmanFilter no primeiro teste

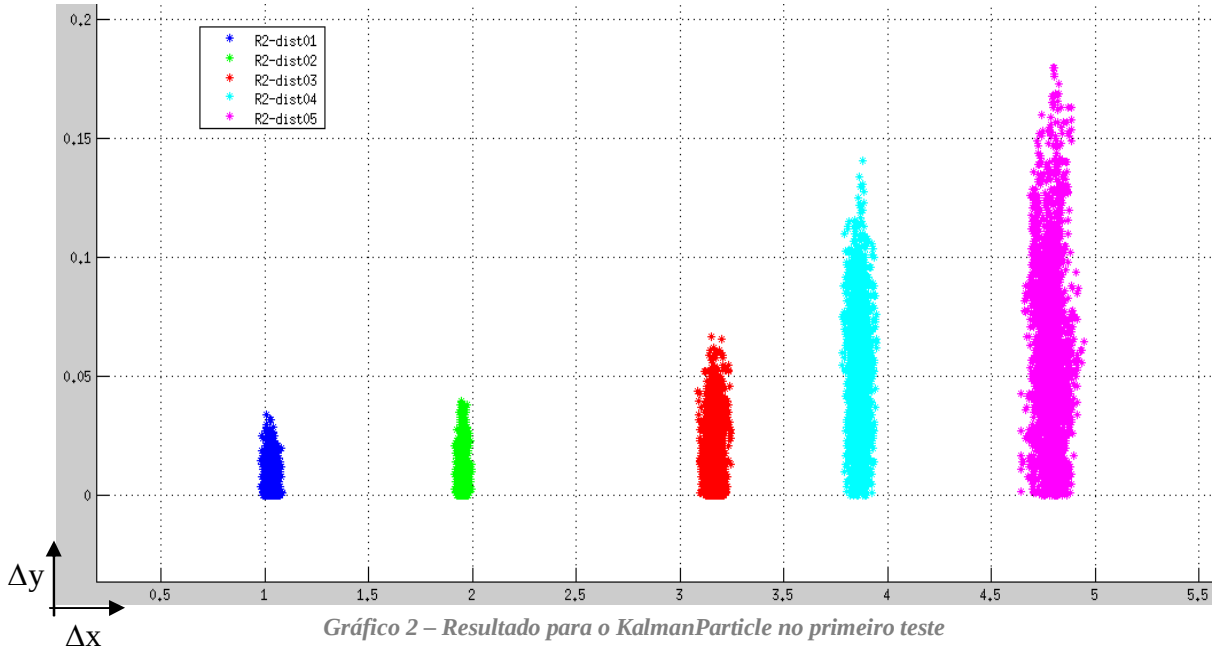

Gráfico 2 – Resultado para o KalmanParticle no primeiro teste

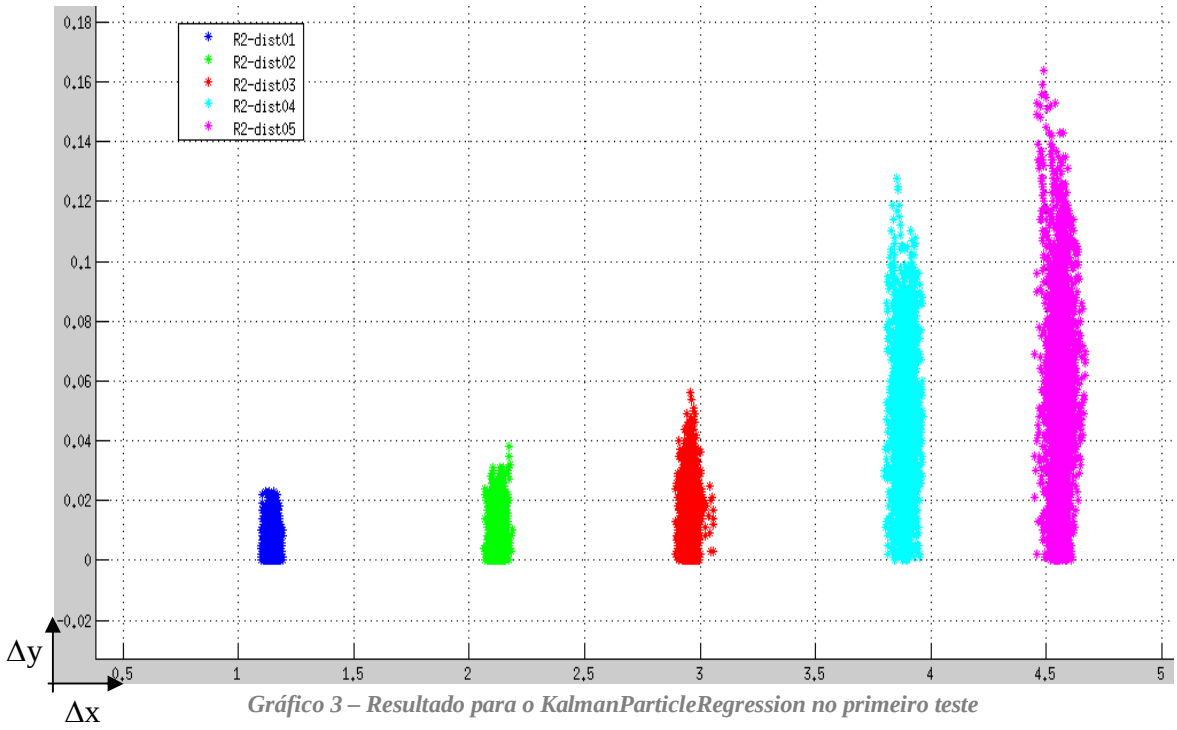

Gráfico 3 – Resultado para o KalmanParticleRegression no primeiro teste

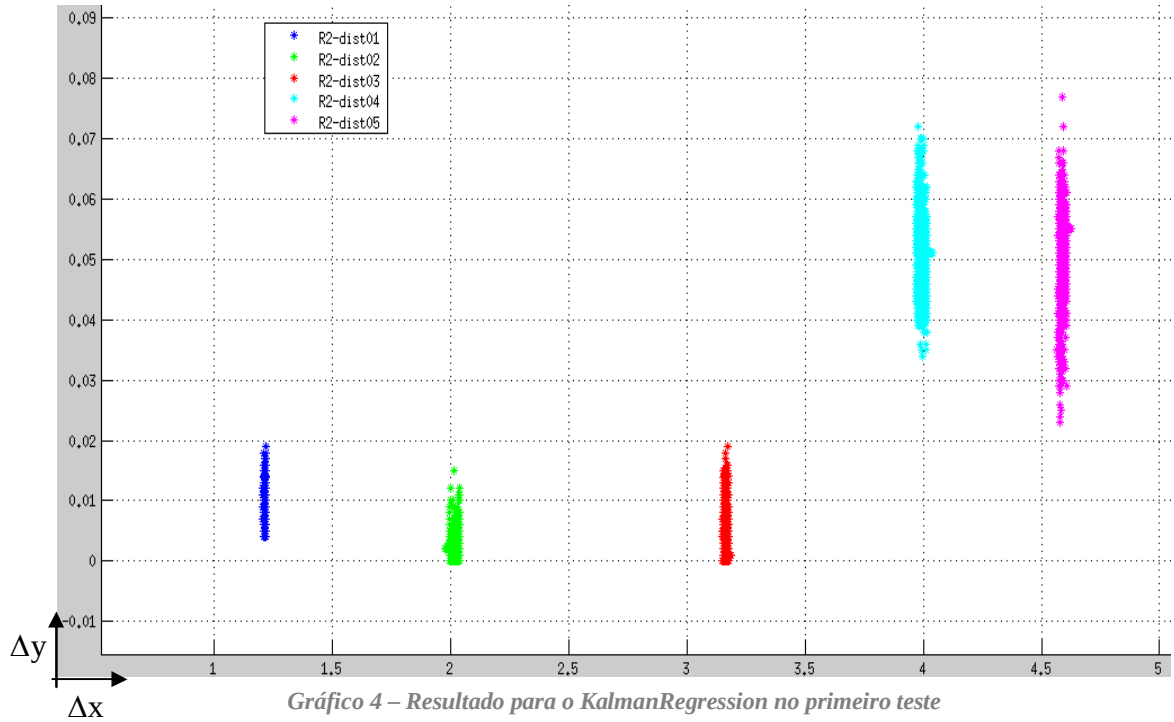

Gráfico 4 – Resultado para o KalmanRegression no primeiro teste

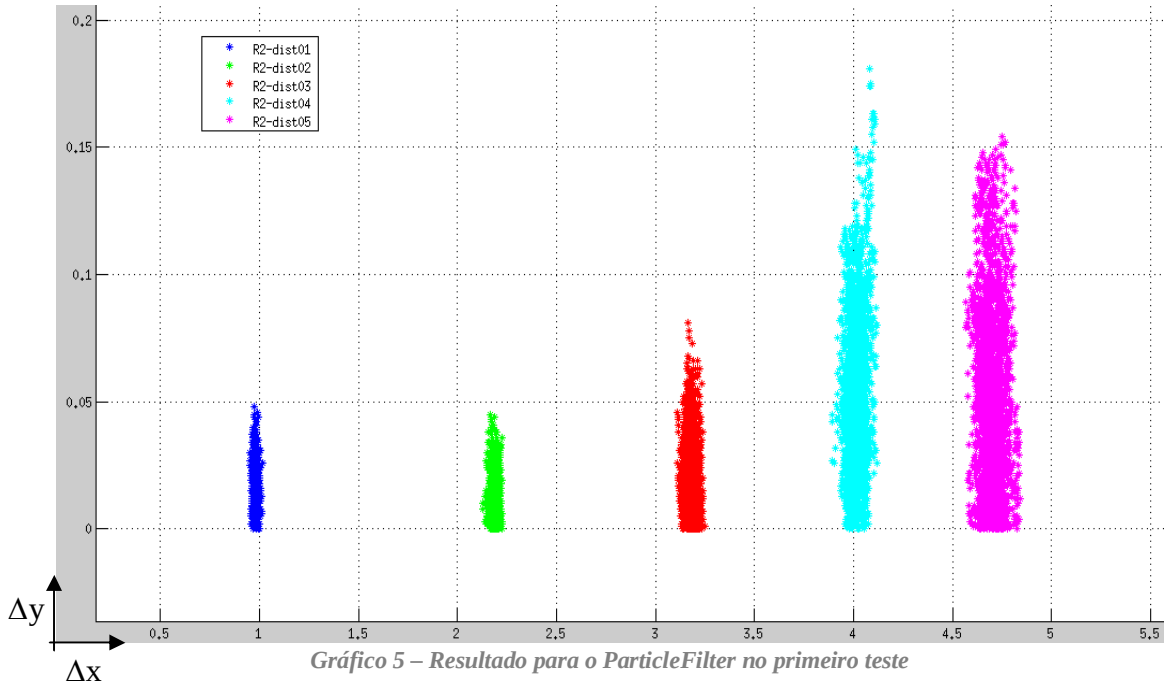

Gráfico 5 – Resultado para o ParticleFilter no primeiro teste

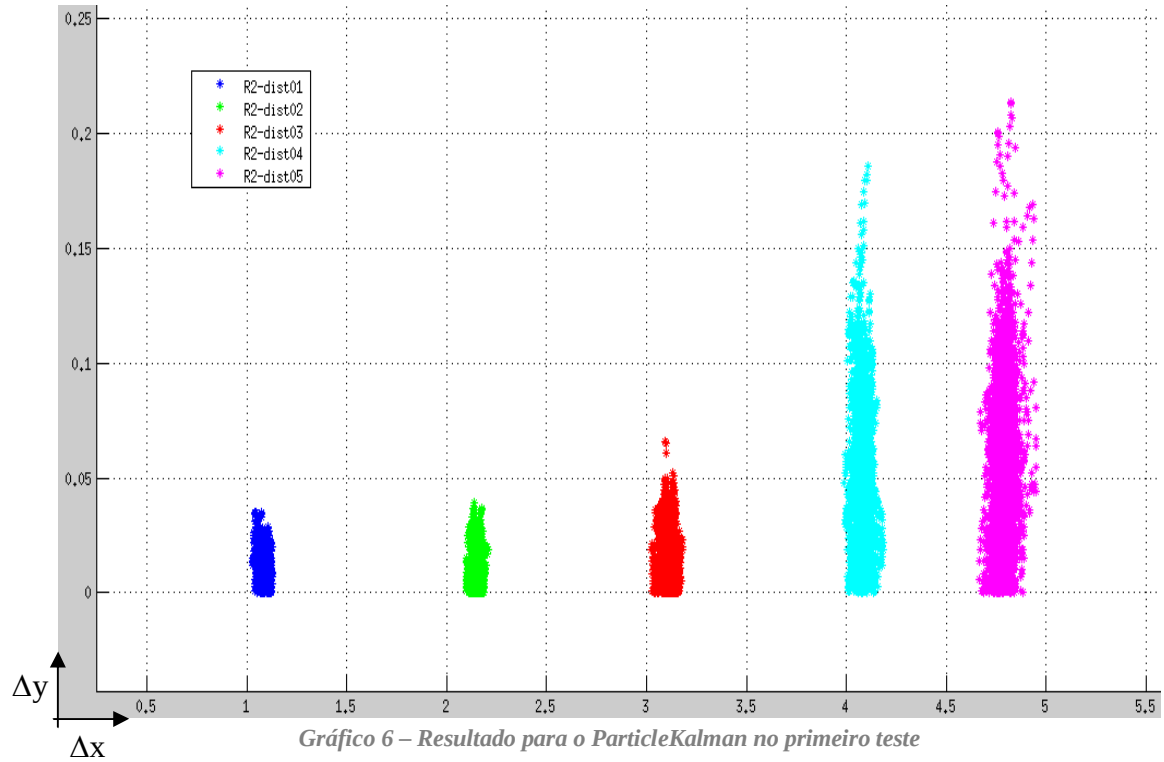

Gráfico 6 – Resultado para o ParticleKalman no primeiro teste

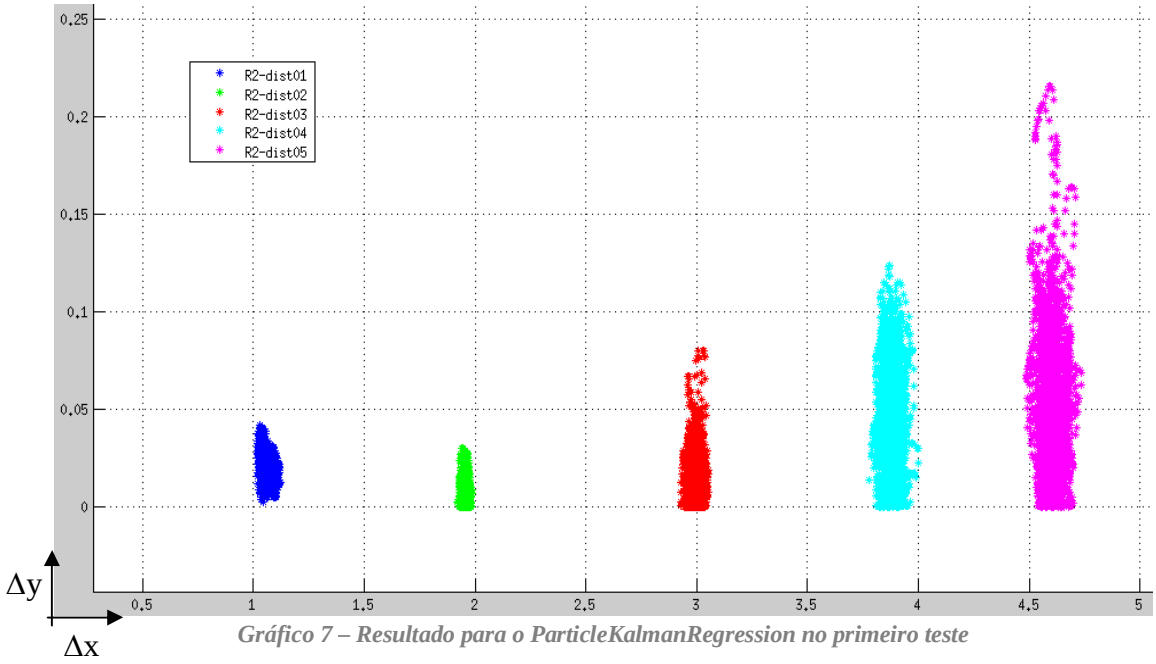

Gráfico 7 – Resultado para o ParticleKalmanRegression no primeiro teste

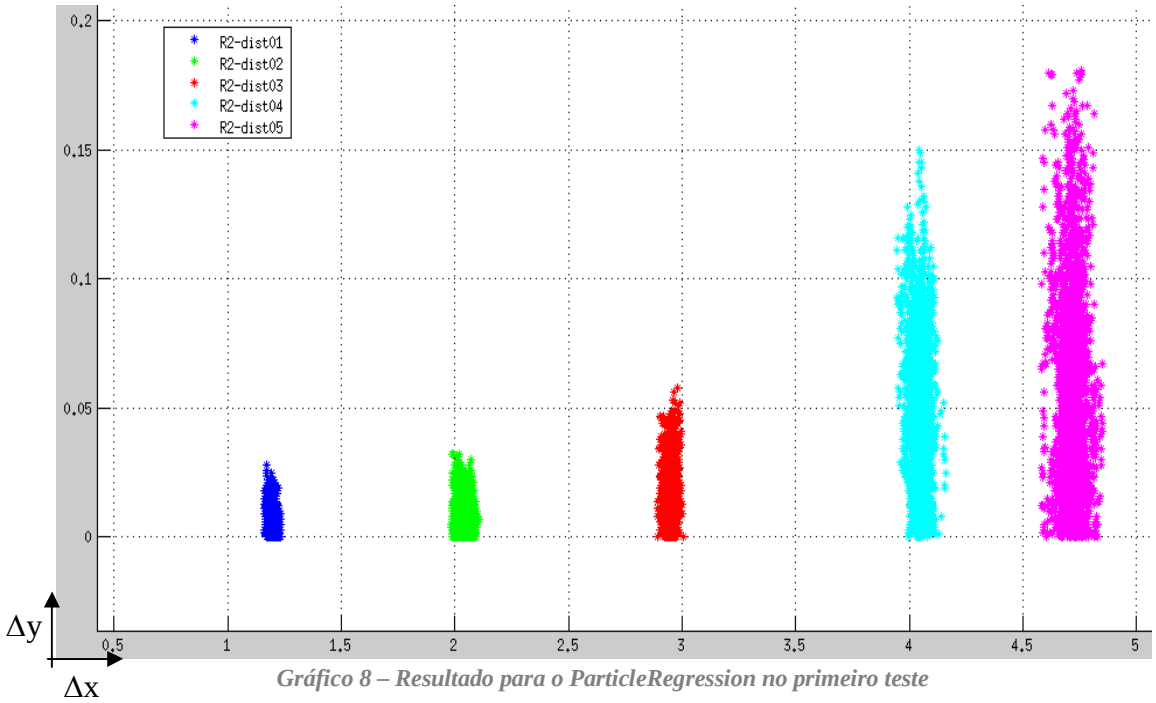

Gráfico 8 – Resultado para o ParticleRegression no primeiro teste

4.2.2. Resultado para o segundo teste

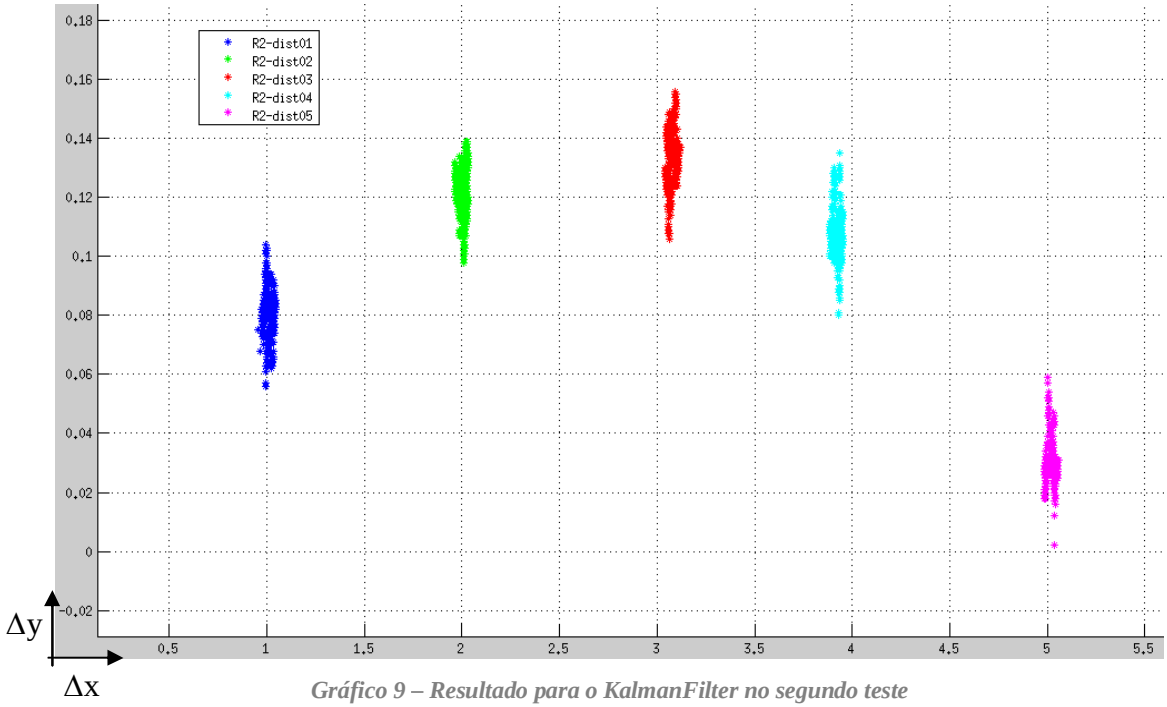

Gráfico 9 – Resultado para o KalmanFilter no segundo teste

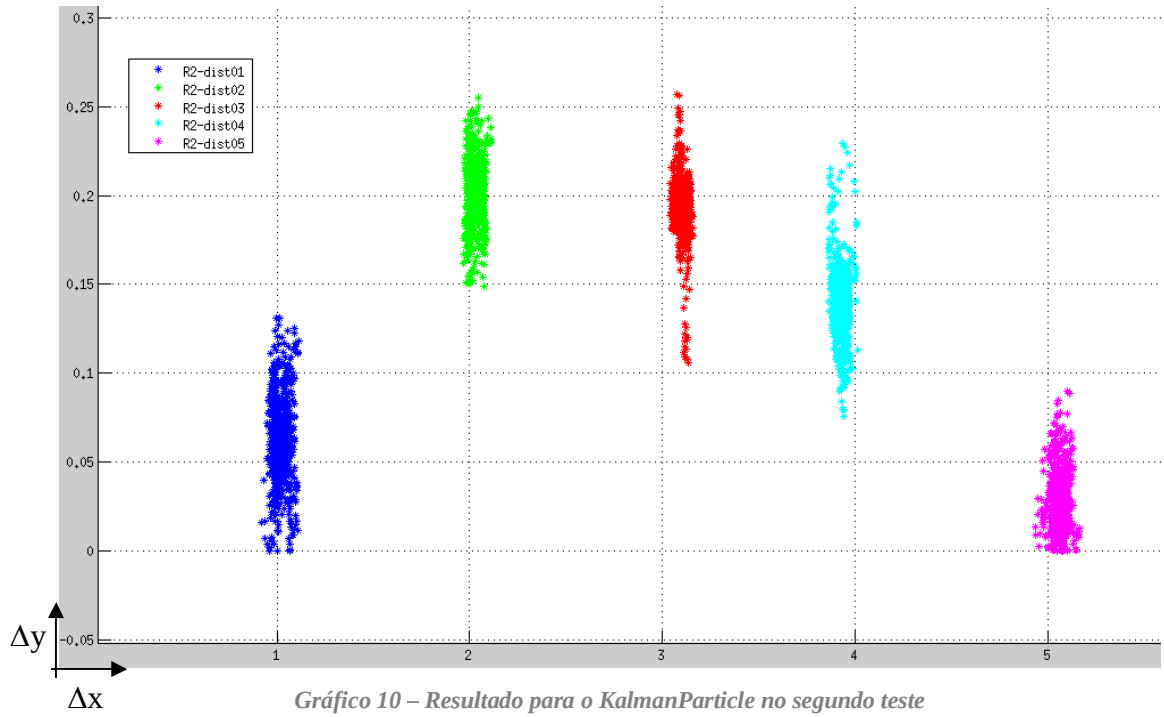

Gráfico 10 – Resultado para o KalmanParticle no segundo teste

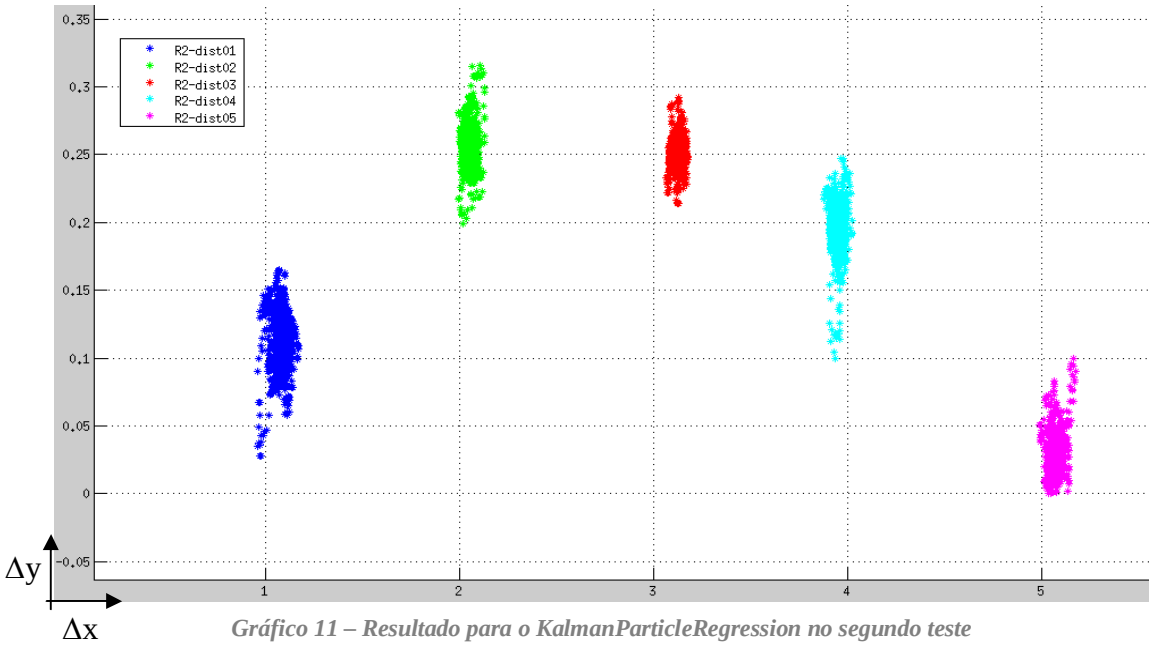

Gráfico 11 – Resultado para o KalmanParticleRegression no segundo teste

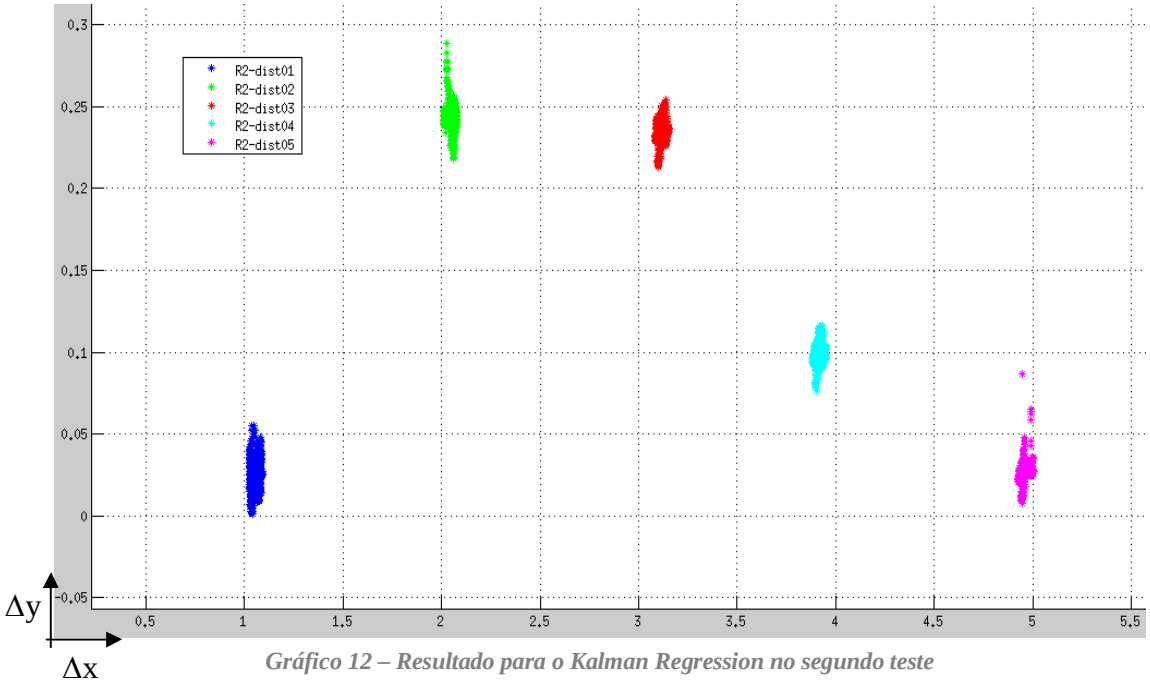

Gráfico 12 – Resultado para o Kalman Regression no segundo teste

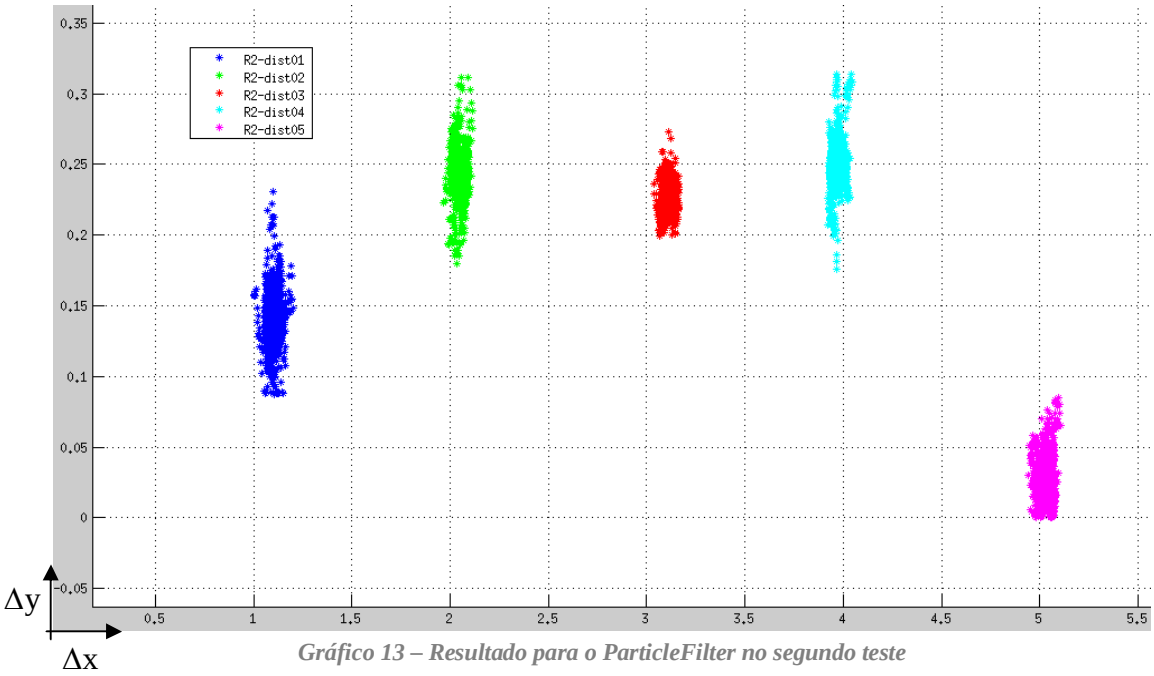

Gráfico 13 – Resultado para o ParticleFilter no segundo teste

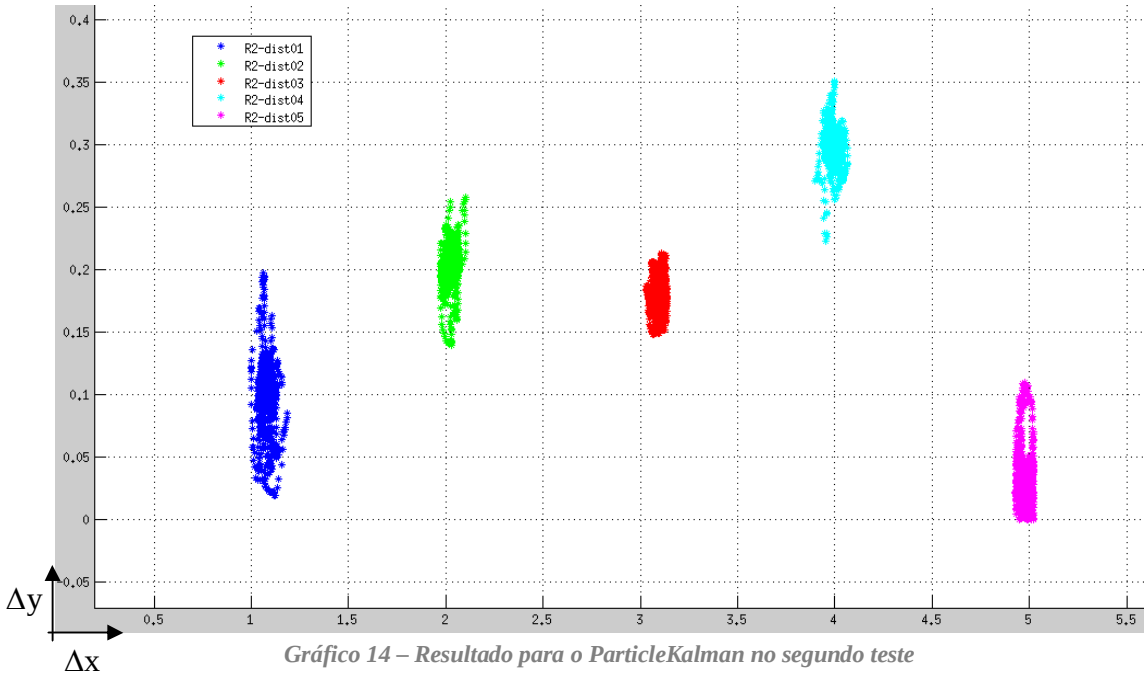

Gráfico 14 – Resultado para o ParticleKalman no segundo teste

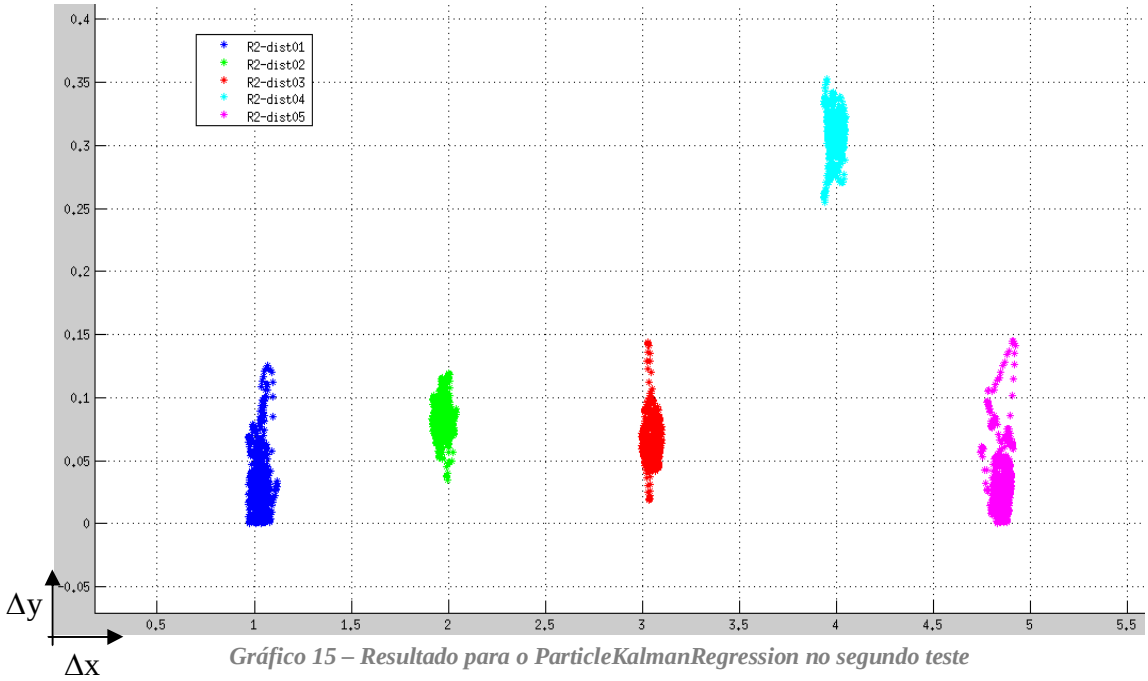

Gráfico 15 – Resultado para o ParticleKalmanRegression no segundo teste

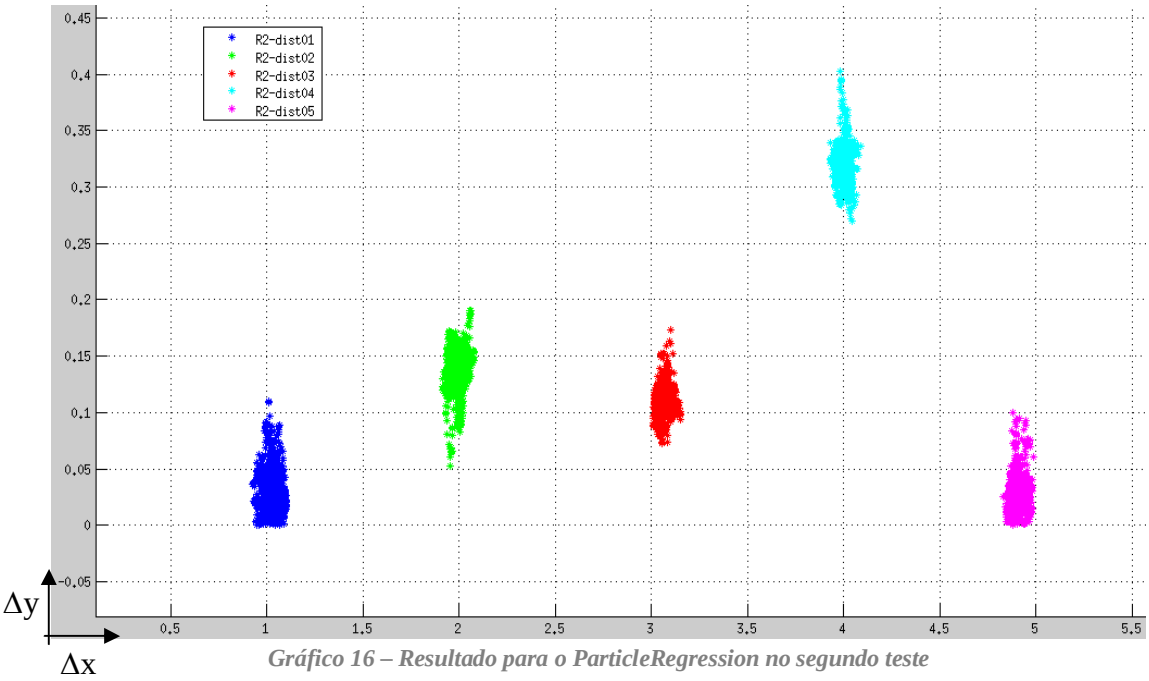

Gráfico 16 – Resultado para o ParticleRegression no segundo teste

4.3. Avaliação dos resultados

Analisando os resultados do primeiro teste, é perceptível que o erro de filtragem se situa aproximadamente entre os 5 e os 20 centímetros para a coordenada do comprimento e entre os 10 e os 40 centímetros para a coordenada da largura. De notar que os menores erros se apresentaram nas implementações *KalmanFilter* e *KalmanRegression*, sendo que os maiores erros se verificam para as implementações *ParticleKalmanRegression e ParticleKalman*. O que nos leva a concluir, que para o primeiro teste, a aplicação do filtro de *Kalman* seguido de uma regressão linear é a melhor opção entre as que foram estudadas. Já a pior situação é o uso de um filtro de partículas associado a um filtro de *Kalman* com uma regressão linear no final.

Para o segundo teste, os erros associados à filtragem são ligeiramente maiores, situando-se aproximadamente entre os 5 e os 30 centímetros para a coordenada do cumprimentos e entre os 10 e os 50 centímetros para a coordenada da largura. Também para este segundo teste, os erros mais curtos dizem respeito às implementações *KalmanFilter* e *KalmanRegression*, sendo que as implementações com erros mais significativos foram o *ParticleKalman* e o *ParticleRegression*. Concluímos assim, que este segundo teste nos demonstrou que como no anterior, a aplicação de um filtro de *kalman* seguido de uma regressão linear é a melhor opção de filtragem entre as estudadas, sendo que se deve evitar o uso do filtro de Partículas seguido de uma regressão linear sob a pena de obter um erro mais elevado na filtragem dos valores obtidos pela observação.

Nas competições referidas no capítulo anterior, foi usado o filtro *KalmanRegression* no integrador que integrou a agente de cada robô, que se revelou capaz dada a prestação e o comportamento que a equipa assumiu ao longo dos jogos de ambas as competições.

Tendo em conta as conclusões anteriores, podemos terminar esta avaliação certos de que as implementações que usam o filtro de *Kalman* foram as que obtiveram melhores resultados, sendo que, a implementação que usa o filtro seguido de uma regressão linear foi a melhor para as situações estudadas. Estamos certos também, de que que as implementações que recorreram ao filtro de Partículas obtiveram os piores resultados para os testes efectuados, apesar da aplicação quer de um filtro de *Kalman*, quer de um filtro de Partículas.

Capítulo 5 – Conclusão

Neste capítulo final, são discutidos os resultados apresentados no capítulo anterior, tendo em conta o estudo efectuado e as alterações realizadas no integrador do agente CAMBADA, concluindo com apresentação do trabalho futuro necessário para que a nova estrutura produza os frutos desejados.

5.1. Discussão de resultados

O resultado obtido com a nova estrutura do integrador do agente CAMBADA, apesar de oferecer melhorias não muito significativas em relação ao anterior no campo da filtragem, cumpre muito bem o seu objectivo principal que é o de possuir uma estrutura modelada para o integrador.

O novo integrador outorgou a ideia de que está bem implementado, nas competições do *Iran Robotics Open 2014* e no *Portuguese Robotics Open 2014*, onde foi usado. Foi notória a facilidade de configuração e alteração no seu módulo de integração, assim como a possibilidade de manter um funcionamento análogo ao que possuía anteriormente.

Os resultados para a filtragem continuam a não ser os melhores, dada a precisão que oferecem para a determinação da posição da bola, afectando dessa forma o cálculo da sua velocidade. Contudo, o módulo integrador oferece agora uma forma mais fácil para implementar e experimentar novas ideias no campo da filtragem, sendo para isso apenas necessário criar uma nova implementação da interface *Filter*.

Também na área da localização do robô, a modelização executada no integrador deu o seu fruto, tornando-a mais simples de compreender e aberta a novas ideias que depois de

implementadas e testadas, deixaram o agente CAMBADA preparado para uma vasta gama de situações onde a sua localização não será o problema.

Desta forma se conclui que o trabalho desenvolvido no âmbito desta Dissertação de Mestrado não foi em vão, visto que, criou um novo modelo para o integrador do agente CAMBADA, tornando assim a compreensão deste mais fácil para novos membros, dividindo a complexidade da implementação pelos vários conceitos que a ele estão associados.

5.2. Trabalho futuro

O trabalho futuro passa por estudar novos modelos e formas de filtrar os valores obtidos pelo sistema de visão e sensorial, de forma a minimizar ao máximo o ruido presente nos mesmos, porque só assim os dados calculados e usados pelo agente serão os mais próximos da realidade possível. Contudo, não basta procurar novos modelos de filtragem, mas também explorar novas tecnologias para observar os dados da realidade, porque quanto mais precisas forem as observações, mais concreto será o resultado da filtragem e integração.

Num futuro próximo, as alterações introduzidas no projecto CAMBADA serão mais uma vez postas à prova quando a equipa participar no RoboCup 2014 que se realiza no Brasil de 19 a 25 de Julho do ano corrente, onde é esperada uma boa prestação da equipa.

Referências

1. Horakova, J.; Kelemen, J. (2011) “Some Impacts of Karel Capek's Concept of Robots as Artificial Organisms”, Emerging Trends in Engineering and Technology (ICETET), 2011 4th International Conference on, pp.49-54, IEEE , ISBN 978-1457718472.

2. Pereira, J. (2003) "Avaliação e Correção do Modelo Cinemáticio de Robôs Móveis Visando a Redução de Erros no Seguimento de Trajetórias", PhD thesis, Universidade do Estado de Santa Catarina.

3. Kitano, H. (1998) “RoboCup-97: Robot Soccer World Cup I”, Springer, ISBN 978-3540697893.

4. Kitano, H.; Asada, M.; Kuniyoshi, Y.; Noda, I.; Osawa, E. (1997) “RoboCup: The Robot World Cup – Initiative”, Proceedings of the first international conference on Autonomous agents", pp 340-347, Marina del Rey, CA, USA, ISBN 978-0897918770.

5. Veloso, Manuela, Pagello, Enrico, Kitano, Hiroaki (Eds.). (2000) “RoboCup-99: Robot Soccer World Cup III”, Springer, ISBN 978-3540453277.

6. MSL Technical Committee (2014) “Middle Size Robot League Rules and Regulations for 2014”, Version 17.1 20140123

7. Almeida, L.; Azevedo, P.; Bartolomeu, E.; Brito, B.; Cunha, P.; Figueiredo, P.; Fonseca, C.; Lima, R.; Marau, L.; Pedreiras, A.; Pereira, A.; Pinho, F.; Santos, L.; Lopes, S.; Vieira, J. (2004) “CAMBADA 2004: Team Description Paper”, CD proceedings of RoboCup Symposium.

8. Silva, V.; Marau, R.; Almeida, L.; Ferreira, J.; Calha, M.; Pedreiras, P.; Fonseca, J. (2005) "Implementing a distributed sensing and actuation system: The CAMBADA robots case study", IEEE ETFA 2005, Catania, Italy.

9. Santos, F.; Currente, G.; Almeida, L.; Lau, N.; Lopes, L.; (2007) “Selfconfiguration of an Adaptive TDMA wireless communication protocol for teams of mobile robots”, Proc. of the 13th Portuguese Conference on Artificial Intelligence, EPIA 2007, Guimarães, Portugal.

10. Trifan, A.; Neves, A.; Cunha, B. (2013) “Evaluation of Color Spaces for User-supervised Color Classification in Robotic Vision”, 17th International Conference on Image Processing, Computer Vision & Pattern Recognition, Las Vegas, Nevada, USA.

11. Bartolomeu, P.; Lopes, L.; Lau, N.; Pinho, A.; Almeida, L. (2005) "Integração de Informação na Equipa de Futebol Robótico CAMBADA", Electrónica e Telecomunicações, vol. 4 (4), Universidade de Aveiro.

12. Rostami, V.; Ebrahimijam, S.; Khajehpoor, P.; Mirzaei, P.; Yousefiazar,M. (2007) “Cooperative Multi Agent Soccer Robot Team”, International Journal of Electrical, Vol1, No.9, ISBN 978-4431669425.

13. Tech United Eindhoven (2013) “Tech United Eindhoven Team Description 2013 Middle Size League”, CD proceedings of RoboCup Symposium

14. Amma, T.; Beifu, J.; Bozic, Z.; Bui, M.; Gawora, F.; Haque, T.; Geihs, K.; Jakob, S.; Kirchner, D.; Kubitza, N.; Liebscher, K.; Opfer, S.; Saur, D.; Schaake, T.; Schluter, T.; Triller, S.; Witsch, A. (2013) “Carpe Noctem 2013”, CD proceedings of RoboCup Symposium

15. Aggarwal, J.; Cai, Q. (1999) “Human Motion Analysis: A Review”, Computer Vision and Image Understanding, vol. 73. Issue 3, pp. 428-440, ISSN 1077-3142

16. Moeslund, T.; Granum, E.; (2001) “A Survey of Computer Vision-Based Human Motion Capture”, Computer Vision and Image Understanding, vol. 81. Issue 3, pp. 231-268, ISSN 1077-3142,

17. Maybeck, P. (1979) “Stochastic Models, Estimation, and Control”, Mathematics In Science and Engineering, vol. 141. Academic Press

18. Isard, M.; Blake, A. (1998) “Condensation - Conditional Density Propagation of Visual Tracking”, International Journal on Computer Vision. Vol. 29, Nr.1, pp 5-28, Kluwer Academic Publishers

19. Welch, G.; Bishop, G. (1995) “An Introduction to Kalman Filter”, PhD thesis, University of North Carolina at Chapel Hill.

20. Arulampalam, M.; Maskell, S.; Gordon, N.; Clapp, T. (2002) “A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking”, IEEE Transactions on Signal Processing, Vol. 50, N. 2, pp174-188, IEEE.

21. Kalman, R. (1960) "A New Approach to Linear Filtering and Prediction Problems", Journal of Basic Engineering, N. 82, Issue 1, pp 167 – 179, Wiley-IEEE Press.

22. Stratonovich, R. (1959) “On the theory of optimal non-linear filtering of random functions”, Theory of Probability and its Applications, N. 4, pp 223–225.

23. Candy, J. (2009) “Bayesian signal processing : classical, modern, and particle filtering methods”, Willey, ISBN 978-0470180945.

24. Haug, A. (2005). "A Tutorial on Bayesian Estimation and Tracking Techniques Applicable to Nonlinear and Non-Gaussian Processes", The MITRE Technical Report 05W0000004.

25. Doucet, A.; Godsill, S.; Andrieu, C.; (2000). "On sequential Monte Carlo sampling methods for Bayesian filtering", Statistics and Computing N. 10, Issue 3, pp 197-208, Kluwer Academic Publishers.

26. Gordon, N. J.; Salmond, D. J. and Smith, A. F. M. (1993). "Novel approach to nonlinear/non-Gaussian Bayesian state estimation". IEEE Radar and Signal Processing, Proceedings F. Vol. 140, Inssue 2, pp 107 – 113.

27. Ferreira, D. (2009) “Estatística Básica 2º Edição Revisada”. UFLA, ISBN 9788587692719.

28. Pinho. R, Tavares. J, Correia. M, (2005) “Seguimento de objectos em visão computacional usando métodos estocásticos” , Congreso de Métodos Numéricos en Ingeniería, SEMNI.