**UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO**

**CENTRO TECNOLÓGICO**

**PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA**

AVELINO FORECHI SILVA

# SISTEMA DE NAVEGAÇÃO ROBÓTICA POR IMAGENS DE PONTOS DE INTERESSE

VITÓRIA

2012

AVELINO FORECHI SILVA

# SISTEMA DE NAVEGAÇÃO ROBÓTICA POR IMAGENS DE PONTOS DE INTERESSE

Dissertação apresentada ao Programa de Pós-Graduação em Informática do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Mestre em Informática.

VITÓRIA

2012

AVELINO FORECHI SILVA

# SISTEMA DE NAVEGAÇÃO ROBÓTICA POR IMAGENS DE PONTOS DE INTERESSE

COMISSÃO EXAMINADORA

---

Prof. Dr. Alberto Ferreira de Souza

Universidade Federal do Espírito Santo

Orientador

---

Profa. Dra. Claudine Santos Badue Gonçalves

Universidade Federal do Espírito Santo

---

Prof. Dr. Jun Okamoto Junior

Escola Politécnica da Universidade de São Paulo

Vitória, 29 de Agosto de 2012.

# DEDICATÓRIA

A minha família e amigos.

# AGRADECIMENTOS

A Deus, em primeiro lugar, minha fonte de esperança e fé. Aos meus pais, Avelino e Zeni, que me incentivaram a estudar. A minha amada esposa Karolini Herzog, por estar sempre ao meu lado, cuidando e me amando. Ao professor Alberto Ferreira De Souza, pela acolhida e inspiração. Aos meus queridos amigos do LCAD, Bruna Colnago, Filipe Mutz, Jorcy Oliveira, Lauro Lyrio, Lucas Veronese, Michael Gonçalves, Rômulo Ramos, Tiago Alves e Vitor Barbirato, que muito me ajudaram no desenvolvimento deste trabalho e, especialmente, a professora Mariella Berger pela pré-disposição de revisá-lo. Tenho muito a agradecer ao Franco Machado pela oportunidade de trabalhar com soluções inovadoras na Mogai. E à FAPES e FINEP, por apoiarem financeiramente alguns dos projetos de pesquisa dos quais participei.

# EPÍGRAFE

*"Vision is the act of knowing what is where by looking."*

Aristóteles

# RESUMO

As imagens projetadas dentro de nossos olhos mudam o tempo todo por conta do movimento dos olhos ou do nosso corpo como um todo. Contudo, percebemos o mundo retratado nas imagens capturadas pelos olhos como estável. Muito embora as imagens projetadas nas retinas humanas sejam bidimensionais, o cérebro é capaz de sintetizar uma representação tridimensional estável a partir delas, com informações sobre cor, forma e profundidade a respeito dos objetos no ambiente ao nosso redor, viabilizando nossa navegação no mundo por meio do sentido da visão.

Neste trabalho investigamos modelos matemático-computacionais de cognição visual - compreensão do mundo e das ideias por meio da visão - aplicados ao problema de navegação de veículos autônomos. Nós empregamos modelos probabilísticos de navegação exploratória e de localização e mapeamento simultâneos (*Simultaneous Localization And Mapping* - SLAM) e desenvolvemos modelos dos movimentos oculares de busca visual e vergência baseados em Redes Neurais Sem Peso (*Virtual Generalizing* RAM - VG-RAM) para navegar um robô autônomo através de uma caminho definido por uma sequência de pontos de interesse especificados na forma de simples fotos 2D do ambiente.

Nossos resultados mostraram que é possível navegar um robô autônomo de um ponto de origem até um ponto de destino fornecendo apenas imagens não contiguas do caminho.

# ABSTRACT

The projected images into our eyes change all the time due to the movement of the eyes or the body as a whole. However, we perceive a stable world depicted in the images captured by the eye. Although the images are projected onto two-dimensional human retinas, the brain is able to synthesize a stable three-dimensional representation from them with information about color, shape and depth about the objects in the environment around us, enabling our navigation in the world via the sense of vision.

In this study we investigate mathematical-computational models of visual cognition - understanding of the world and the ideas through vision - applied to the problem of autonomous vehicle navigation. We employ probabilistic models of exploratory navigation and simultaneous localization and mapping (Simultaneous Localization And Mapping - SLAM) and develop models of eye movements for visual search and vergence based on Weightless Neural Network (Virtual Generalizing RAM - VG-RAM) to navigate an autonomous robot through a path defined by a sequence of points of interest specified as simple 2D images of the environment.

Our results showed that it is possible to navigate a robot from an origin point to a destination point by providing only non-contiguous images of the way.

# SUMÁRIO

# LISTA DE FIGURAS

# LISTA DE TABELAS

# 1 INTRODUÇÃO

As imagens projetadas dentro de nossos olhos mudam o tempo todo por conta do movimento dos olhos ou do nosso corpo como um todo. Contudo, em um aparente paradoxo, percebemos o mundo retratado nas imagens capturadas pelos olhos como estável. Além disso, as imagens projetadas nas retinas humanas são bidimensionais; entretanto, o cérebro é capaz de sintetizar uma representação tridimensional estável a partir delas, com informações sobre cor, forma e profundidade a respeito dos objetos no ambiente ao nosso redor, eliminando os efeitos dos movimentos dos olhos e do corpo.

O sistema visual biológico viabiliza a nossa movimentação (navegação) através do ambiente 3D de forma precisa. Sendo assim, a modelagem das funcionalidades do sistema visual biológico pode contribuir para o desenvolvimento de sistemas de localização e mapeamento simultâneos (*Simultaneous Localization And Mapping* – SLAM [THR05]), e navegação de veículos autônomos. SLAM é provavelmente o problema mais fundamental da robótica autônoma. Veículos robóticos autônomos necessitam saber onde estão em sua área de atuação para navegar nela e realizar atividades de interesse. Atualmente, as abordagens probabilísticas têm se mostrado as mais apropriadas para resolver os problemas de SLAM [THR05] e navegação.

A visão constitui nossa principal fonte de informação a respeito do mundo para usarmos na integração do caminho (*Path Integration*) [GOL99] que estamos seguindo (navegando) de forma a conseguirmos estimar nossa localização em relação a pontos já visitados no percurso e também em relação aos próximos pontos pertencentes ao planejamento global do caminho conhecido. Contudo, no mundo moderno, o ser humano tem se utilizado cada vez mais de dispositivos, tais como GPS (*Global Positioning System*), para realizar navegação global ao longo de um trajeto, seja ele conhecido ou não. Estes dispositivos se utilizam de mapas previamente conhecidos da região onde estamos e computam nossa posição georeferenciada neles a cada instante. Contudo, antes da existência desses equipamentos, ou nos casos de indisponibilidade dos mesmos, o ser humano é plenamente capaz de se deslocar de um ponto ao outro se guiando apenas por pontos de referência (fotos, monumentos, acidentes geográficos, constelações, etc.) que se encontram ao longo do caminho a ser percorrido.

Inspirados por essas habilidades humanas de navegação e capacidades cognitivas visuais associadas, desenvolvemos e validamos algoritmos que possibilitam a veículos autônomos a realização de navegação robótica exploratória em condições semelhantes às enfrentadas pelos seres humanos na ausência de mapas prévios ou de qualquer informação de orientação e posicionamento.

Para alcançarmos esse objetivo, implementamos um sistema de navegação robótica por imagens de pontos de interesse capaz de navegar autonomamente um robô de um ponto de interesse a outro fornecendo como entrada para o sistema apenas uma sequência de imagens do caminho que o robô deve reconhecer e seguir sem colidir com os obstáculos. Nosso sistema de navegação se subdivide em: um sistema de SLAM, que fornece um mapa e a posição do robô atualizados em tempo real; um sistema de planejamento, que define rotas para regiões não exploradas; um sistema de navegação, que conduz o robô em segurança pela rota planejada; um sistema de busca visual, que reconhece as fotos do caminho por onde o robô deve passar, buscando correspondências entre os pontos mais salientes da foto no caminho e os das imagens capturadas pela câmera; e um sistema de vergência, que calcula coordenadas tridimensionais dos pontos salientes presentes nas imagens reconhecidas pelo sistema de busca visual. Essas coordenadas são fornecidas para o sistema de navegação guiar o robô até as proximidades desses locais.

## 1.1 Motivação

Neste trabalho investigamos modelos matemático-computacionais de cognição visual – compreensão do mundo e das ideias por meio da visão – aplicados ao problema de navegação de veículos autônomos. Nós empregamos SLAM e navegação robótica exploratória probabilística e desenvolvemos modelos dos movimentos oculares de busca visual e vergência para navegar um robô autônomo através de uma caminho definido por uma sequência de pontos de interesse especificados na forma de simples fotos 2D do ambiente.

A cognição visual é viabilizada no cérebro humano por uma quantidade enorme de neurônios (dezenas de bilhões de neurônios [KAN00]). Trata-se de uma capacidade extremamente complexa com inúmeras facetas. Contudo, implementações de modelos

bastante limitados de algumas destas facetas têm se mostrado valiosas na solução dos problemas de SLAM e navegação de veículos autônomos [THR06]. Uma equipe da *Stanford University* liderada por S. Thrun [THR06] desenvolveu um veículo autônomo, denominado Stanley, que empregava câmeras para inferir a região de interesse mais distante da estrada à sua frente. Stanley foi capaz de se deslocar autonomamente por 142 milhas (aproximadamente 229km) dentro do contexto de uma competição promovida em 2005 pela *Defense Advanced Research Projects Agency* (DARPA) dos USA denominada DARPA *Grand Challenge* (http://archive.darpa.mil/grandchallenge05/).

A primeira DARPA *Grand Challenge* ocorreu em 2004 e possuía um percurso composto basicamente por estradas não pavimentadas. Nesta edição (2004), nenhum dos veículos autônomos conseguiu completar o percurso. Apenas um ano depois, Stanley venceu esta competição (mesmo tipo de percurso) e cinco outros veículos autônomos conseguiram também completar o percurso de 142 milhas, quatro deles dentro do tempo limite estabelecido (10 horas) [STA05, RED05a, RED05b, GRE05, TER05]. Em 2007, a DARPA promoveu uma *Grand Challenge* em percurso urbano (http://archive.darpa.mil/grandchallenge/). Nesta, os veículos autônomos precisavam respeitar regras de trânsito e coabitar com outros veículos não participantes da corrida. Seis veículos autônomos conseguiram completar o percurso, sendo que o primeiro, novamente de Stanford (Junior), o completou em cerca de 4 horas [STA07] (Junior acabou ficando em segundo lugar, muito embora tenha terminado a corrida em menor tempo, devido a outros critérios de pontuação http://archive.darpa.mil/grandchallenge/).

Quase a totalidade dos veículos que participaram das DARPA *Grand Challenge* empregaram câmeras e sensores laser (*Laser Range Scan*, ou *Light Detection And Ranging* – LIDAR) para examinar o terreno à frente e realizar, juntamente com sensores GPS, sua localização e mapeamento simultâneos (SLAM) e, a partir dos mapas, evitar obstáculos e perseguir os objetivos das provas (navegação). Contudo, nenhum deles empregava modelos matemático-computacionais baseados na biologia do sistema visual humano para realizar estas tarefas. Na verdade, os veículos que foram capazes de completar as DARPA *Grand Challenge* implementaram seus algoritmos de SLAM usando fortemente como entrada a informação capturada por sensores LIDAR (ver "*Semi-finalist Technical Papers*" em http://archive.darpa.mil/grandchallenge/resources.asp). Sensores LIDAR empregam laser para medir a distância de pontos ao longo de uma linha à frente usando um mecanismo

mecânico/ótico de varredura – tal mecanismo não encontra paralelo na biologia. Além disso, sensores LIDAR são fortemente afetados por chuva, entre outras condições meteorológicas naturais, o que limita sua aplicabilidade em casos que requeiram operação ao ar livre, e são facilmente detectáveis à distância (por causa do laser), o que limita sua aplicabilidade militar.

Neste trabalho aprofundamos nossos estudos sobre modelos matemático-computacionais da visão humana. Nossa principal motivação é buscar compreender como nosso sistema visual viabiliza a cognição visual em suas diversas facetas (reconhecimento de objetos, percepção de profundidade, movimento e forma, etc.) e, em particular, nossa capacidade de fazer SLAM e navegação – estudos com ratos e humanos demonstraram a existência de estruturas neurais que codificam nossa localização e o mapeamento do ambiente [KEE71, HAF05, FYH08, MOS08, DOE10]. Isso é relevante no contexto de veículos autônomos porque, com apenas nossos dois olhos e nosso cérebro, somos capazes de dirigir um automóvel em velocidades muito superiores às observadas nas DARPA *Grand Challenges* e em condições muito mais desafiadoras. Assim, avanços no desenvolvimento de modelos matemático-computacionais da visão humana propiciarão a simplificação das implementações de veículos autônomos e a ampliação das possibilidades de sua utilização.

## 1.1 Objetivos

O principal objetivo deste trabalho foi a integração de modelos matemático-computacionais do sistema visual humano com modelos de robótica probabilística de forma a possibilitar a um robô desempenhar missões de busca por pontos de interesse no ambiente a sua volta a partir de uma sequência de imagens não contíguas desse mesmo ambiente. Esta integração permitiu a validação com imagens dinâmicas de modelos matemático-computacionais inicialmente desenvolvidos para tratar imagens estáticas.

## 1.2 Contribuições

As principais contribuições deste trabalho foram:

1. Implementação do algoritmo de SLAM FastSLAM baseado em mapas *Occupancy Grid* [ELF89] e filtros de partículas [THR05]. Nossa implementação diferencia-se da original [THR05] devido à utilização do método de amostragem *Improved Proposal Distribution* empregado no FastSLAM 2.0 baseado em mapas de *Landmark* [MON03].

2. Desenvolvimento de um algoritmo de exploração MDP (*Markov Decision Process*es) aproximativo por distribuição de partículas sobre o espaço de estados que o robô pode assumir no mapa de *Grid* [THR05].

3. Implementação de um modelo matemático-computacional baseado em Redes Neurais Sem Peso VG-RAM (*Virtual Generalizing* RAM - VG-RAM) que emula os movimentos oculares de sacada observados na busca visual [NET12].

4. Implementação de um novo modelo matemático-computacional baseado em VG-RAM que emula os movimentos oculares de vergência. Este modelo é um aprimoramento do inicialmente proposto por Komati em [KOM02].

5. Desenvolvimento de um sistema integrado de navegação robótica e exploração visual baseado em sequências não contíguas de fotos do caminho de interesse.

## 1.3 Estrutura do trabalho

No Capítulo 1 apresentamos uma breve introdução sobre os modelos biológicos da visão humana examinados neste trabalho e suas capacidades cognitivas associadas de navegação pelo ambiente que inspiraram o desenvolvimento do Sistema de Navegação Robótica por Imagens de Pontos de Interesse, tema deste trabalho e cuja arquitetura é discutida em detalhes no Capítulo 7. No Capítulo 2 abordamos a estrutura e o funcionamento de partes do sistema

visual humano importantes para explicar os modelos matemático-computacionais descritos no Capítulo 3. Neste capítulo detalhamos os sistemas propostos de movimento dos nossos olhos para busca visual e vergência. Nos capítulos 4 e 5 apresentamos os modelos probabilísticos de SLAM e navegação exploratória utilizados para permitir que o robô se desloque pelo ambiente com segurança. No Capítulo 8 apresentamos a metodologia empregada na implementação do Sistema de Navegação Robótica por Imagens de Pontos de Interesse, enquanto que, no Capítulo 9, apresentamos os resultados dos experimentos realizados para sua avaliação. No Capítulo 10 apresentamos uma discussão sobre trabalhos correlatos e uma análise crítica deste trabalho de pesquisa. Por fim, no Capítulo 11, apresentamos nossas conclusões e possibilidades de trabalhos futuros.

# 2 SISTEMA VISUAL HUMANO

Neste capítulo é descrito sumariamente o sistema visual humano. Ele apresenta conceitos e termos que são essenciais para compreender as contribuições deste trabalho. Seu conteúdo foi, fundamentalmente, extraído de [OLI05].

## 2.1 O Olho

O globo ocular com cerca de 25 milímetros de diâmetro é o responsável pela captação da luz refletida pelos objetos. Anatomicamente, o globo ocular fica alojado em uma cavidade formada por vários ossos chamada órbita e é constituído por três túnicas (camadas): túnica fibrosa externa, túnica intermédia vascular pigmentada e túnica interna nervosa.

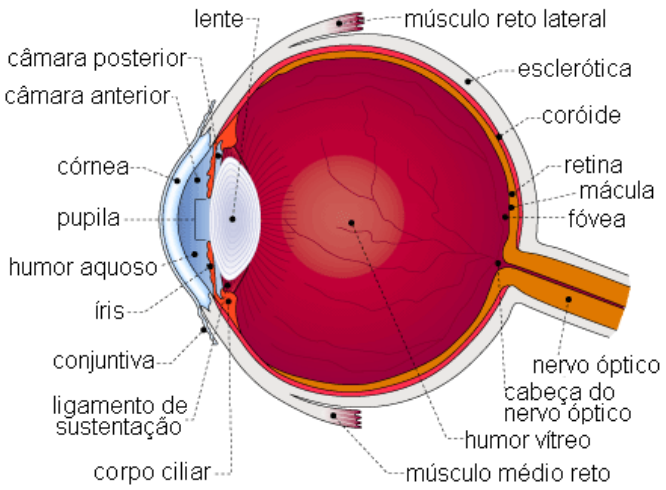



**Figura 1: Anatomia do olho humano. Corte medial horizontal do olho direito visto de cima. Figura retirada de http://www.escolavesper.com.br/olho_humano.htm.**

Na Figura 1 estão representadas todas as partes que formam as túnicas fibrosa externa, intermédia vascular pigmentada e a túnica interna nervosa. A túnica fibrosa externa ou esclerótica, também chamada de "branco do olho", tem uma função protetora. É resistente, de tecido fibroso e elástico, e envolve externamente o olho (globo ocular). A maior parte da esclerótica é opaca e chama-se esclera. A ela estão conectados os músculos extraoculares que

movem os globos oculares, dirigindo-os ao seu objetivo visual. A parte anterior da esclerótica chama-se córnea, que é transparente e atua como uma lente convergente.

A túnica intermédia vascular pigmentada ou úvea compreende a coroide, o corpo ciliar e a íris. A coroide está situada abaixo da esclerótica e é bastante pigmentada para absorver a luz que chega a retina, evitando sua reflexão dentro do olho. Ela é intensamente vascularizada e tem também como função nutrir a retina. A íris é uma estrutura muscular de cor variável (parte circular que dá cor aos olhos), é opaca e tem uma abertura central, chamada pupila, por onde a luz passa. O diâmetro da pupila varia, aproximadamente de 2mm a 8mm, de acordo com a intensidade luminosa do ambiente. Em ambientes claros a pupila se estreita, diminuindo a passagem de luz, evitando a saturação das células detectoras de luz da retina. No escuro a pupila se dilata, aumentando a passagem de luz e sua captação pela retina. A luz que passa pela pupila atinge imediatamente o cristalino, uma lente gelatinosa que focaliza os raios luminosos sobre a retina.

O corpo ciliar é uma estrutura formada por musculatura lisa e que envolve o cristalino (a lente do olho). Ele é capaz de mudar a forma do cristalino permitindo assim ajustar a visão para objetos próximos ou distantes. Este processo é conhecido como acomodação visual. A convergência correta do cristalino faz com que a imagem seja projetada nitidamente na retina. Se a imagem for maior ou menor que a necessária, fica fora de foco. Se o cristalino está ajustado para certa distância de um objeto, e este objeto se aproxima, a imagem perde a nitidez. Para recuperá-la, o corpo ciliar aumenta a convergência do cristalino, acomodando-o, diminuindo a distância focal. Caso o objeto se afaste, ocorre o processo inverso. Este processo está ilustrado na Figura 2.

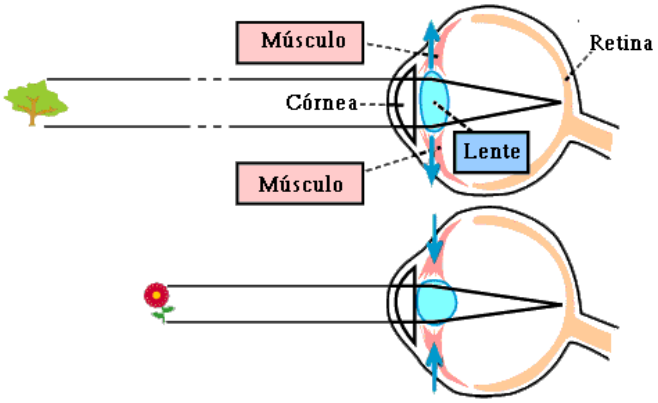



**Figura 2: Figura mostrando a acomodação visual do cristalino. Figura retirada de http://www.portalsaofrancisco.com.br/alfa/conhecimentos-gerais/por-que-temos-que-usar-oculos.php.**

A túnica interna nervosa é a retina. É na retina que se formam as imagens visualizadas. A imagem projetada na retina é invertida, mas isto não causa nenhum problema já que o cérebro se adapta a isto desde o nascimento. Para que a imagem seja projetada na retina, a luz percorre o seguinte caminho: primeiramente a luz atinge a córnea, que conforme visto anteriormente é um tecido transparente, passa pela pupila, que é a abertura situada na íris que regula a intensidade de luz que entra no olho, atravessa o cristalino, que é a lente gelatinosa e que tem a função de focalizar a imagem na retina, atravessa um fluido viscoso chamado humor vítreo, que preenche a região entre o cristalino e a retina, e finalmente atinge a retina.

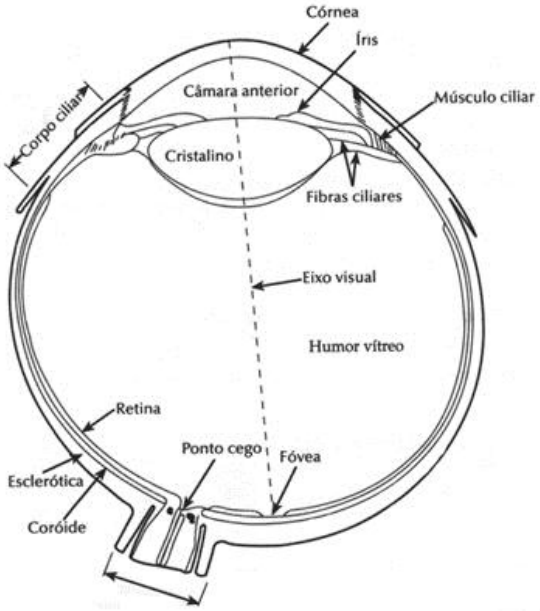



**Figura 3: Eixo visual. Figura retirada de http://www.on.br/glossario/alfabeto/o/olho_humano.html.**

A retina é composta por mais de 100 milhões de células fotossensíveis: cerca de 7 milhões de cones e entre 75 milhões e 150 milhões de bastonetes. Estas células, quando excitadas pela energia luminosa, estimulam as células nervosas adjacentes, gerando um impulso nervoso que é propagado pelo nervo óptico. A imagem fornecida pelos cones é mais nítida e rica em detalhes. Os cones são sensíveis às cores. Há três tipos de cones: um que se excita com luz vermelha, outro com luz verde e outro com luz azul. Os bastonetes não têm poder de resolução visual tão bom nem conseguem detectar cores, mas são mais sensíveis à luz. Em situações de pouca luminosidade a visão passa a depender exclusivamente dos bastonetes.

As imagens dos objetos visualizados diretamente são projetadas normalmente numa região da retina chamada *fovea centralis* ou simplesmente fóvea com cerca de 1,5 mm de diâmetro e que fica na direção da linha (eixo visual) que passa pela córnea, pupila e pelo centro do cristalino (Figura 3). Os cones são encontrados na retina central, em um raio de aproximadamente 10 graus a partir da fóvea. Os bastonetes estão localizados principalmente na retina periférica. O local da retina de onde sai o nervo óptico não possui cones nem bastonetes. Este local é chamado de ponto cego porque uma imagem que se forme sobre este ponto, não é vista.

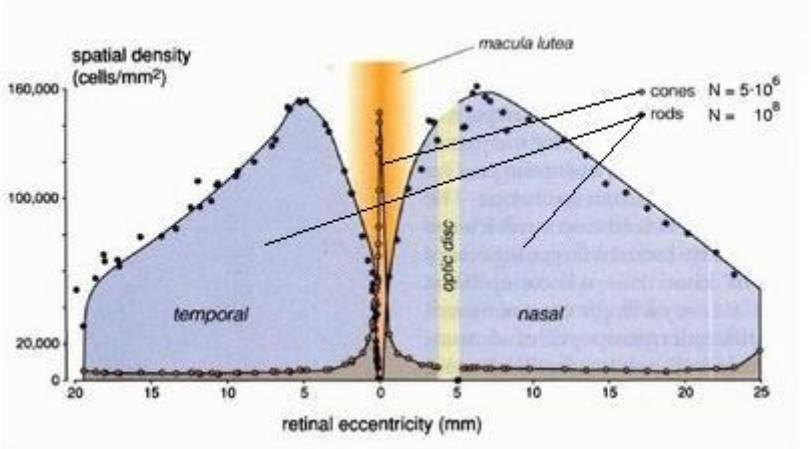



**Figura 4: Distribuição de cones e bastonetes (*rods*) na retina. A *macula lutea* (região da fóvea e vizinhanças) possui alta densidade de cones, enquanto que os bastonetes se concentram na periferia. O ponto cego fica na região do disco óptico (*optic disc*). Figura retirada de http://www.brainworks.uni-freiburg.de/group/wac.**

Os neurônios de saída da retina são as células ganglionares que projetam seus axônios através do nervo óptico, levando a informação visual para o cérebro. Entre as células

ganglionares e as células fotorreceptoras (cones e bastonetes), existem três classes de interneurônios: células bipolares, horizontais e amácrinas. Estas células transmitem os sinais dos fotorreceptores para as células ganglionares combinando sinais de vários fotorreceptores e fazendo com que a saída de cada célula ganglionar da retina dependa de um padrão preciso de luz e de como este padrão muda com o tempo [KAN00].

Cada célula ganglionar recebe informações de um conjunto de células fotorreceptoras vizinhas em uma área circunscrita na retina que é o seu **campo receptivo**. O campo receptivo de uma célula ganglionar é a área da retina na qual esta célula controla. Duas características importantes podem ser percebidas nos campos receptivos das células ganglionares. Primeiro, são aproximadamente circulares. Segundo, são divididos em duas partes: um círculo central e um anel periférico.

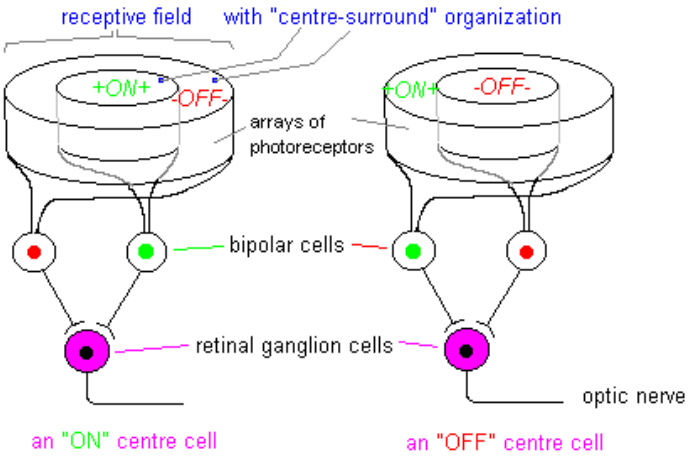



**Figura 5: Campo receptivo das células ganglionares. Figura retirada de http://www.cf.ac.uk/biosi/staff/jacob/teaching/sensory/vision.html.**

As células ganglionares respondem bem a uma iluminação diferencial entre o centro e a periferia dos seus campos receptivos. Assim, é possível identificar dois tipos de células ganglionares: *on center* e *off center*. Células ganglionares com campos receptivos *on center* ficam excitadas quando a luz estimula o centro e ficam inibidas quando a luz estimula o contorno do campo receptivo. Células *off center*, funcionam ao contrário, ficam excitadas quando a luz estimula a periferia e ficam inibidas quando a luz estimula o centro do campo receptivo. Os sinais visuais de intensidade luminosa (na verdade, de contraste), depois das transformações feitas na retina, são levados até o cérebro pelo nervo ótico.

## 2.2 Fluxo de informações visuais

Nesta seção será descrito o fluxo de informações visuais em dois estágios: primeiro, a informação visual saindo da retina e indo para o mesencéfalo e tálamo (Figura 8), e depois, a informação saindo do tálamo para o córtex visual primário, conforme indicado na Figura 6. Para tanto, serão definidos alguns conceitos a seguir.

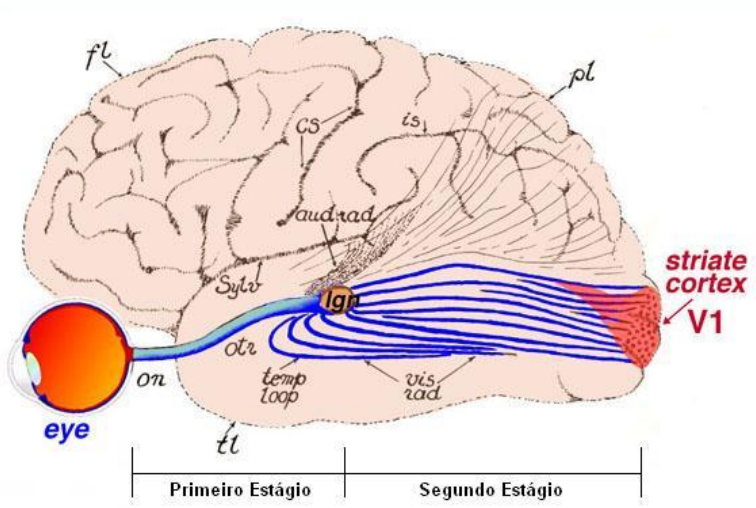



**Figura 6: Fluxo das informações visuais. Figura retirada de http://webvision.med.utah.edu/VisualCortex.html e alterada com iserção dos estágios.**

A retina pode ser dividida em duas partes: hemirretina nasal e hemirretina temporal, cuja separação é uma linha imaginária que corta o olho de cima a baixo passando pela fóvea. Numa situação em que as fóveas de ambos os olhos estão fixas num ponto do espaço situado em linha reta com o nariz, é possível dividir o campo visual em *left hemifield* (campo visual esquerdo) à esquerda do ponto fixo no espaço e *right hemifield* (campo visual direito) à direita do ponto fixo no espaço. O *left hemifield* é projetado na hemirretina nasal do olho esquerdo e na hemirretina temporal do olho direito. O *right hemifield* é projetado na hemirretina nasal do olho direito e na hemirretina temporal do olho esquerdo, conforme mostrado na Figura 7.

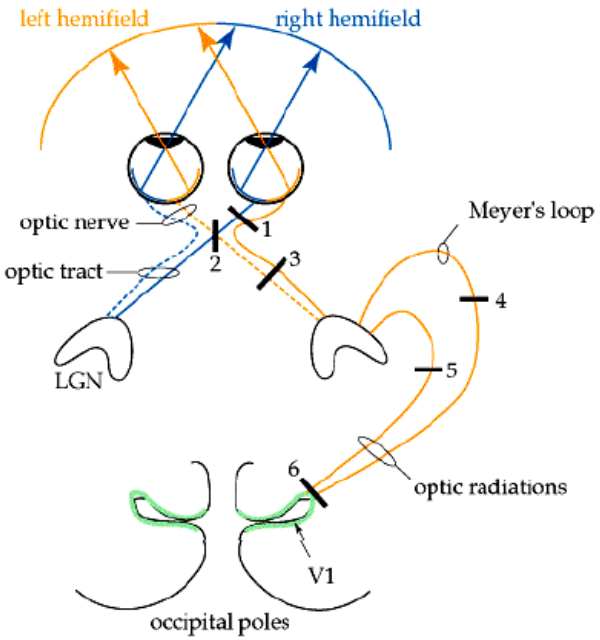



**Figura 7: Campo visual. 1) Nervo óptico, 2) Quiasma óptico, 3) Trato óptico. Figura retirada de http://thalamus.wustl.edu/course/basvis.html.**

O nervo óptico de cada olho projeta-se para o quiasma óptico que é a região onde é feita a separação das fibras de cada olho, em tratos ópticos, destinadas para um mesmo lado do cérebro. Os tratos ópticos se projetam cada uma para três áreas subcorticais simétricas (que existem nos dois lados de cérebro): região pretectal ou *pretectum*, o *superior culliculus* do mesencéfalo e o *lateral geniculate nucleus* (LGN) do tálamo conforme mostra a Figura 8.

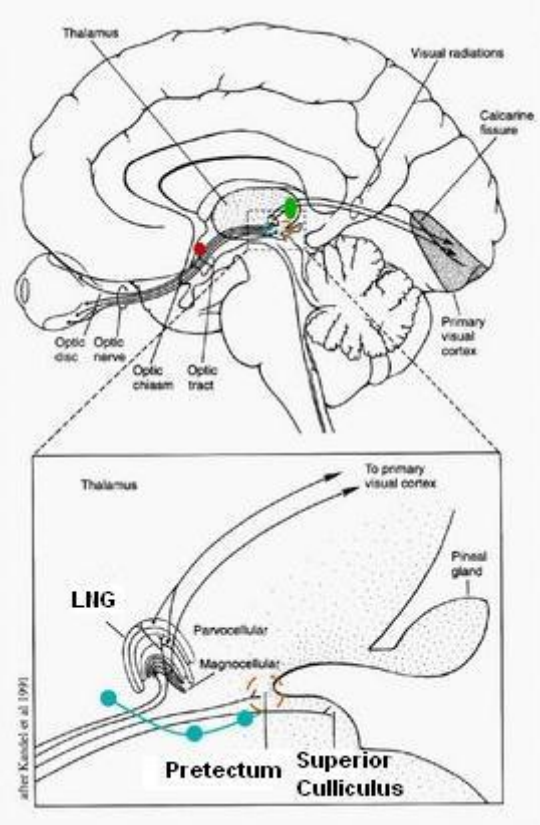



**Figura 8: Projeções da retina no mesencéfalo. Figura retirada de [http://www.brainworks.uni-freiburg.de/group/wachtler/VisualSystem/](http://www.brainworks.uni-freiburg.de/group/wachtler/VisualSystem/).**

A região pretectal do mesencéfalo possui células que se projetam bilateralmente para os neurônios do sistema simpático/parassimpático que controla os reflexos pupilares, contraindo e dilatando a pupila de acordo com a quantidade de luz que incide nos olhos. O *superior culliculus* é uma estrutura de camadas alternantes cinzentas e brancas localizada no teto do mesencéfalo. As células das camadas superficiais projetam-se para uma vasta área do córtex cerebral, formando uma via indireta da retina para o córtex cerebral. As camadas superficiais também recebem sinais provenientes do córtex visual enquanto que as camadas mais profundas recebem projeções de várias outras áreas do córtex ligadas a outros sentidos. O *superior culliculus* possui um mapeamento visual além de responder a estímulos auditivos e somatossensórios.

Células das camadas mais profundas do *superior culliculus* respondem positivamente antes dos movimentos sacádicos dos olhos, no qual os olhos trocam rapidamente de um ponto de fixação para outro numa cena. Estas células formam um mapa de movimento sacádico

ordenado com o mapa visual. Para controlar os movimentos sacádicos, o *superior culliculus* recebe informações não só da retina, mas também do córtex cerebral.

O *lateral geniculate nucleus* (LGN) é o ponto de retransmissão das informações visuais provenientes da retina para o córtex visual. Cerca de 90% dos axônios da retina chegam até o LGN, que possui uma representação retinotópica da metade contralateral (lado oposto) do campo visual.

A razão entre uma área do LGN e uma área correspondente da retina que representa um grau do campo visual é chamada de fator de magnificação daquela área do LGN. A fóvea possui uma representação relativamente maior que a retina periférica no LGN, ou seja, as regiões do LGN que monitoram a fóvea possuem um maior fator de magnificação.

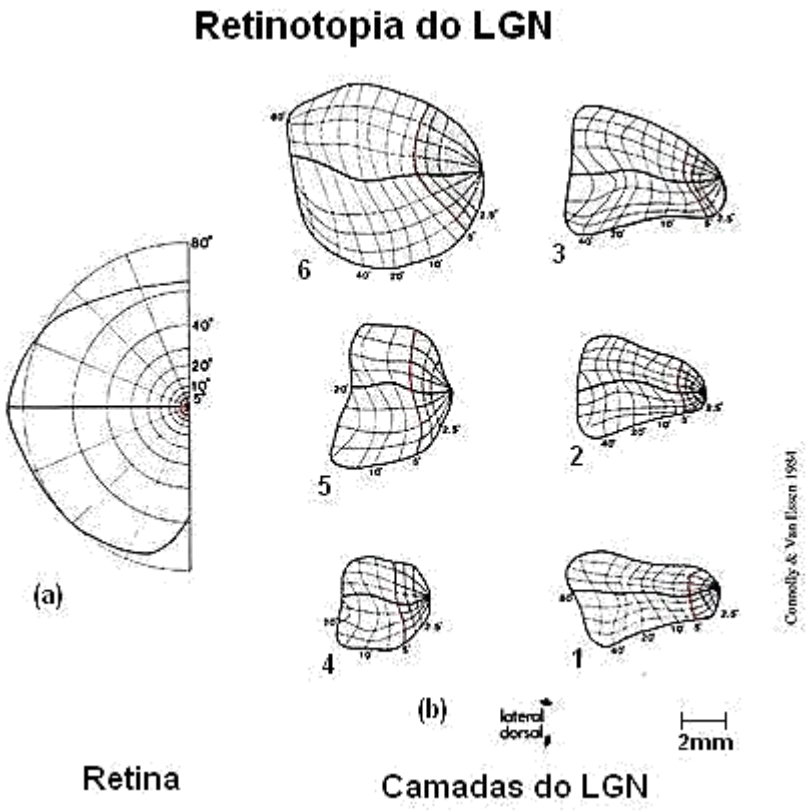



**Figura 9: Retinotopia do LGN. (a) Mapeamento da retina. (b) Mapeamento da retina nas camadas 1 à 6 do LGN. Figura retirada e adaptada de http://www.brainworks.uni-freiburg.de/group/wachtler/VisualSystem/.**

Nos primatas, incluindo os humanos, o LGN é formado por seis camadas numeradas de 1 (ventral) a 6 (dorsal). As duas camadas mais ventrais (camadas 1 e 2) contém células

relativamente grandes que recebem conexões das células ganglionares M da retina e são conhecidas como camadas magnocelulares enquanto que as outras 4 camadas dorsais (camadas 3, 4, 5 e 6) contém células que recebem conexões das células ganglionares P da retina e são conhecidas como camadas parvocelulares. A Figura 9 mostra de forma esquemática o mapeamento da retina nas diversas camadas do LGN. Nela é possível ver como o fator de magnificação varia da fóvea para a periferia no LGN.

Todas as camadas do LGN possuem células com campo receptivo *on center* e *off center*, sendo que cada camada recebe sinais somente de um olho. As fibras da hemirretina nasal contralateral (outro lado) são projetadas nas camadas 1, 4 e 6, enquanto que as fibras da hemirretina temporal ipsilateral (mesmo lado) são projetadas nas camadas 2, 3 e 5 conforme mostrado na Figura 10.

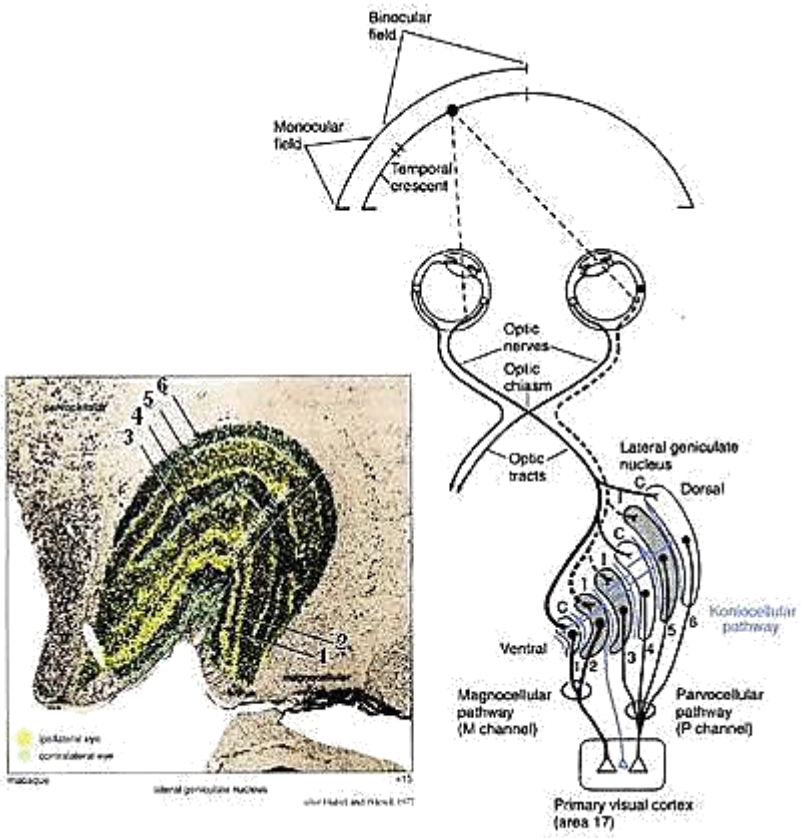



**Figura 10: LGN. Figura retirada de [http://www.brainworks.uni-freiburg.de/group/wachtler/VisualSystem/](http://www.brainworks.uni-freiburg.de/group/wachtler/VisualSystem/).**

As células das camadas magnocelulares e parvocelulares do LGN projetam-se para o córtex visual formando duas vias independentes (vias M e P) que se estendem desde a retina

até o córtex visual primário. A via P é essencial para a visão de cores e sensível a estímulos de alta frequência espacial e baixa frequência temporal da imagem na retina. A via M é mais sensível a estímulos de baixa frequência espacial e alta frequência temporal.

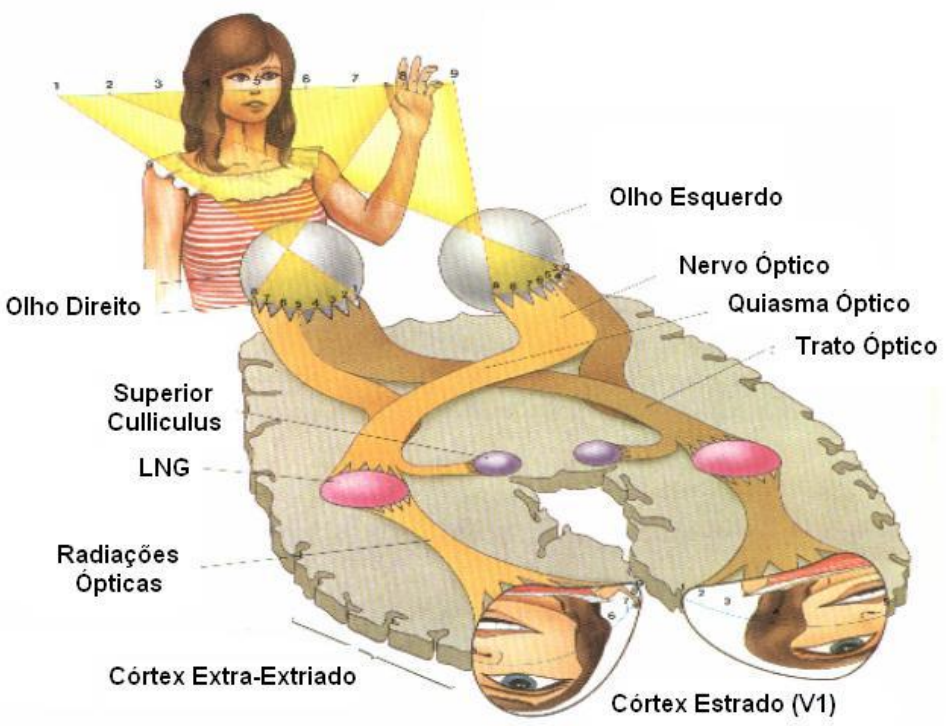



**Figura 11: Exemplo ilustrando o fluxo de informações visuais da retina ao córtex estriado. Figura retirada de http://www.brainworks.uni-freiburg.de/group/wachtler/VisualSystem/.**

## 2.3 Organização do córtex visual

A informação visual é processada em diversas áreas corticais, sendo que cada uma delas contribui diferencialmente para o processamento da percepção de movimento, profundidade, forma e cor. Aqui nós descreveremos brevemente cinco áreas corticais visuais mais diretamente ligadas a este trabalho: V1, V2, V3, V4 e MT (também conhecida como V5). Na Figura 12-A é apresentada uma vista lateral de um hemisfério do cérebro de um macaco e na Figura 12-B é mostrado este hemisfério estendido de modo a formar um plano, onde são indicadas com uma tonalidade mais escura as áreas corticais visuais e são apontadas as áreas V1, V2, V3, V4 e MT. Como é possível observar na Figura 12-A, as áreas corticais visuais ocupam aproximadamente metade do córtex de um macaco.

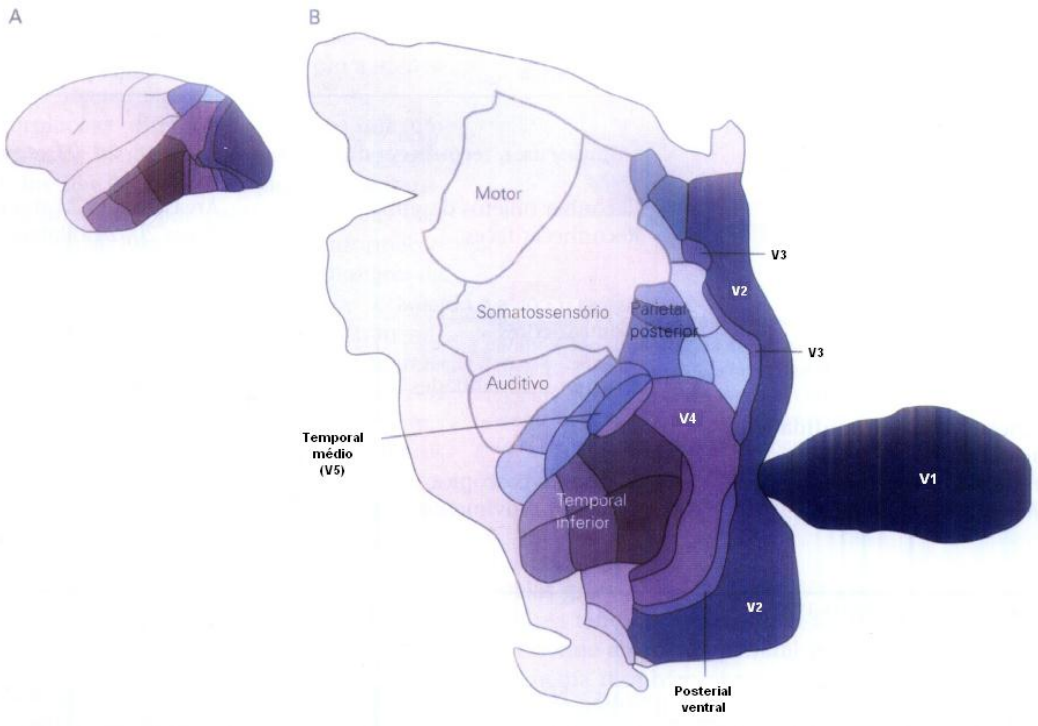



**Figura 12: Áreas corticais. Figura retirada de [KAN00].**

### 2.3.1 V1

Quase toda informação visual vinda da retina entra no córtex via a área V1 que, devido a sua aparência estriada, também é conhecida como córtex estriado. As outras áreas são conhecidas como córtex extra-estriado. V1 também é chamada de córtex visual primário e de área 17 de Brodmann [KAN00]. Nos humanos o córtex visual primário possui cerca de 2mm de espessura e é dividido em 6 camadas numeradas de 1 à 6 (Figura 13). A camada 4 é a que recebe a maioria das projeções dos axônios do LGN e pode ser dividida em 4 subcamadas: 4A, 4B, 4Cα e 4Cβ. Os axônios de células das camadas parvocelulares (P) do LGN terminam principalmente na camada 4Cβ com algumas poucas projeções para 4A e 1, enquanto que os axônios de células das camadas magnocelulares (M) terminam principalmente na camada 4Cα.

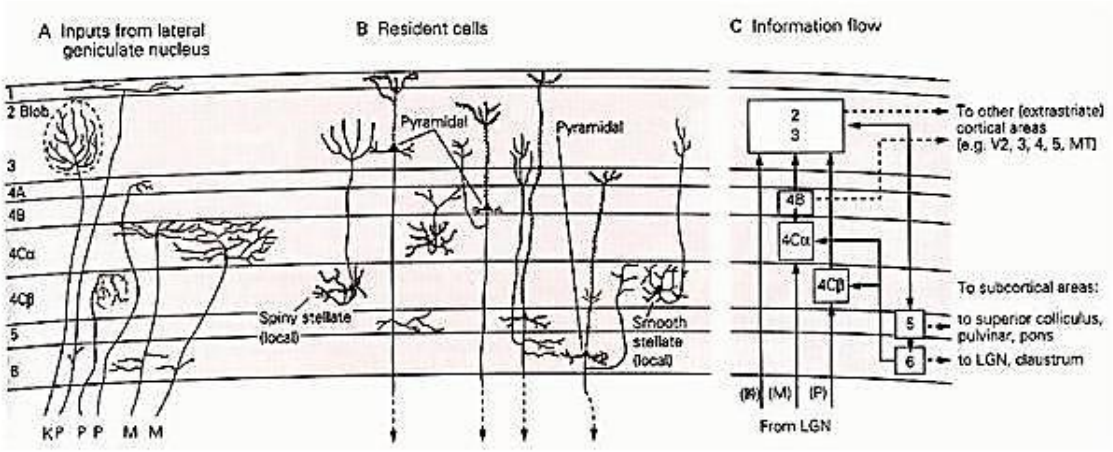



**Figura 13: Organização de V1. A) Os axônios dos neurônios P e M do LGN terminam na camada 4; B) Células de V1; C) Concepção do fluxo de informação em V1. Figura retirada de http://www.brainworks.uni-freiburg.de/group/wachtler/VisualSystem/.**

Através de estudos feitos em macacos verificou-se que o córtex estriado, assim como LGN, possui um mapa retinotópico, isto é, áreas do campo visual vizinhas da retina são também vizinhas em V1, do campo visual contralateral [TOO82]. O aspecto mais importante deste mapa é que cerca da metade das projeções da retina sobre o córtex visual primário são provenientes da fóvea e regiões circunvizinhas. Esta área apresenta a maior acuidade visual.

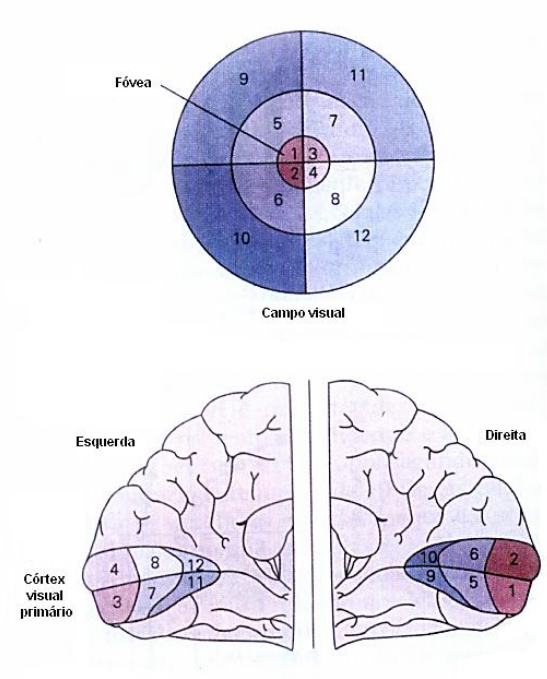



**Figura 14: Campo visual representado no córtex visual primário humano. Figura retirada de [KAN00].**

O formato do campo receptivo das células de V1 é diferente do formato dos campos receptivos das células da retina e do LGN que são circulares. Em V1, os campos receptivos das células são alongados e, consequentemente, respondem melhor a estímulos alongados do que a estímulos pontuais. Hubel e Wiesel [HUB62] classificaram as células de V1 de acordo com a complexidade de sua resposta, dividindo-as em dois grupos chamados simples e complexos.

As células simples também possuem campo receptivo com regiões excitatórias e inibitórias, contudo estas regiões têm seu formato alongado. Estas células respondem melhor a estímulos na forma de barras com uma orientação específica. Uma célula simples que responde melhor a um estímulo vertical não responderá bem a um estímulo horizontal ou oblíquo, e vice-versa.

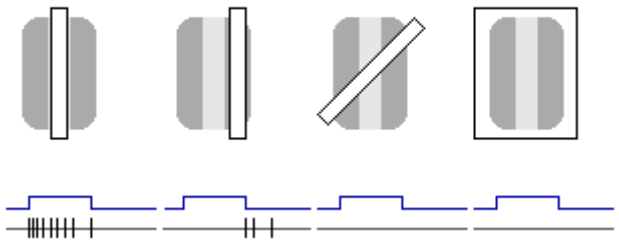


**Figura 15: Resposta de uma célula simples em função da projeção de um estímulo em forma de barra. Figura retirada de [MAT12]**

Na Figura 15 é apresentado de forma esquemática o comportamento de uma célula simples quando um estímulo em forma de barra é projetado em seu campo receptivo. Para produzir os resultados mostrados na Figura 15 o pesquisador, tipicamente, monitora a tensão no interior da célula através de um microeletrodo (na verdade, uma micropipeta que perfura a parede celular), ao mesmo tempo em que o animal cuja célula está sendo monitorada observa um estímulo. Na Figura 15, quatro estímulos na forma de barra são mostrados posicionados sobre uma representação do campo receptivo da célula; os três primeiros, da esquerda para a direita, são barras brancas estreitas e de mesma largura, e o quarto é uma barra branca larga que cobre todo o campo receptivo. O campo receptivo em questão possui orientação vertical, sendo que a parte do mesmo que excita a célula é central e as que inibem ficam nas laterais esquerda e direita. Abaixo de cada conjunto estímulo-campo receptivo são mostrados dois gráficos. O primeiro, imediatamente abaixo de cada conjunto estímulo-campo receptivo,

mostra o momento em que o estímulo está desligado ou ligado (trata-se do comportamento de um sinal elétrico ao longo do tempo que, no nível baixo, indica que o estímulo está inativo e, no nível alto, indica que o estímulo está ativo). O segundo representa o sinal capturado pelo microeletrodo conectado à célula.

Como o primeiro par de gráficos Figura 15 (o mais a esquerda) mostra, a célula simples responde fortemente (emitindo vários pulsos pouco afastados no tempo, que é o modo como as células do córtex sinalizam sua ativação) quando o estímulo é ligado estando corretamente orientado e posicionado sobre a parte central do campo receptivo. A resposta da célula é mais vigorosa imediatamente após o acionamento do estimulo, o que mostra um aspecto temporal da resposta da célula. Na verdade, permanecendo o estímulo por muito tempo (de dezenas de segundos a alguns minutos), a resposta da célula desapareceria totalmente, por um processo conhecido como acomodação. Mas um estímulo constante por muito tempo não ocorre naturalmente, uma vez que movemos os olhos continuamente.

Como o segundo par de gráficos da Figura 15 mostra, quando o estímulo é posicionado sobre a parte do campo receptivo que inibe a célula ela não responde no momento em que o estímulo é ligado, embora responda, fracamente, imediatamente após o estímulo ser desligado (também uma evidência do aspecto temporal da resposta da célula). Nos outros casos a célula não responde.

As células complexas são mais numerosas em V1 do que as células simples e, assim como as células simples, respondem bem apenas para um estímulo com uma orientação específica. Porém, diferentemente das células simples, a resposta das células complexas não é seletiva à posição espacial do estímulo, ou seja, não varia com a posição do estímulo dentro do seu campo receptivo. Muitas células complexas são sensíveis ao sentido e direção do movimento do estímulo dentro do seu campo receptivo, respondendo somente quando este estímulo se move numa determinada direção e sentido.

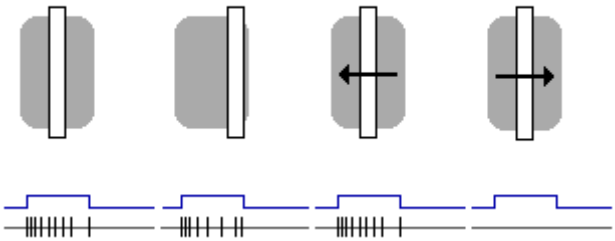


**Figura 16: Resposta de uma célula complexa em função da projeção de um estímulo em forma de barra. Figura retirada de [MAT12]**

Na Figura 16 é apresentado o comportamento de uma célula complexa quando um estímulo em forma de barra é projetado no seu campo receptivo. A célula complexa responde independente da posição do estímulo no campo receptivo, diferentemente da célula simples. As células complexas também são sensíveis à movimentação do estímulo dentro de seu campo receptivo. Caso o estímulo se mova no mesmo sentido em que o campo receptivo esteja sintonizado, a célula continua respondendo, caso contrário para de responder ao estímulo.

Hubel e Wiesel foram os primeiros a descobrir que as células de V1 são arranjadas e organizadas de uma forma precisa em relação à sensibilidade à orientação. Ao longo da superfície de V1, a sensibilidade à orientação varia gradualmente, mas permanece constante ao longo dos 2mm de córtex (de uma coluna do córtex). Hubel e Wiesel também descobriram que a resposta das células varia de acordo com o olho estimulado. Muitas células de V1 respondem de forma aproximadamente equivalente a estímulos provenientes de ambos os olhos, mas a maioria das células de V1 respondem preferencialmente à estímulos provenientes de um determinado olho. Esta característica, chamada de dominância ocular, é organizada no córtex visual primário de uma forma que não varia verticalmente (em colunas), mas alterna ao longo da superfície de V1.

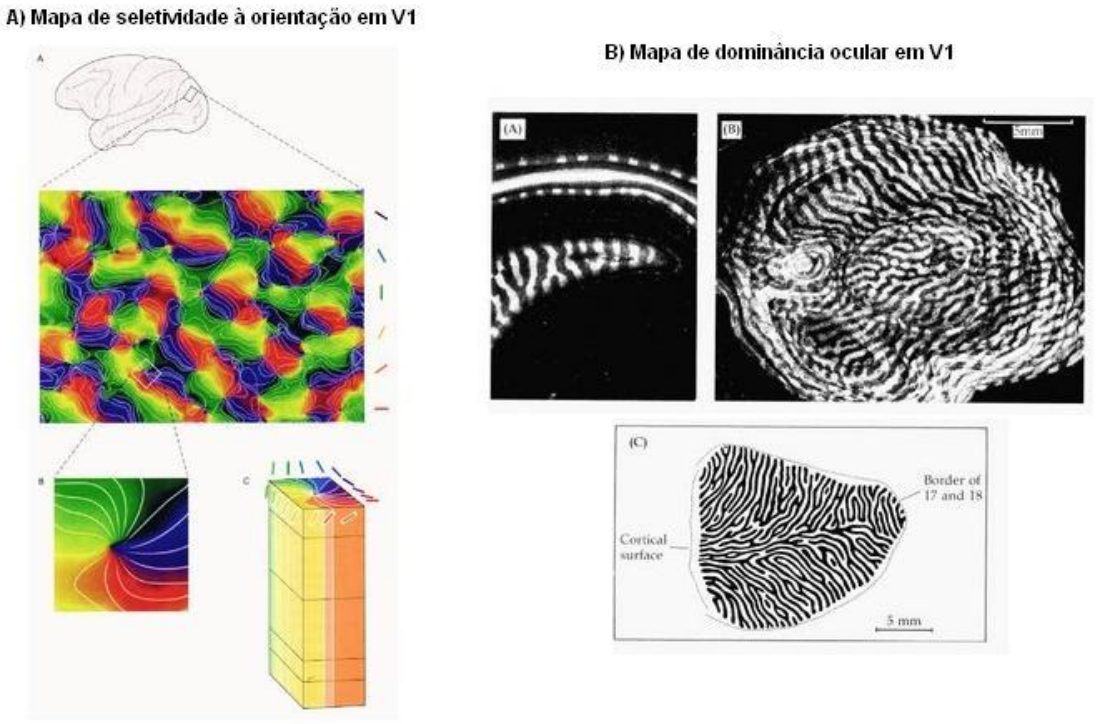



**Figura 17: A seletividade à orientação (a) e a dominância ocular (b), variam ao longo da superfície de V1, porém não se alteram numa mesma coluna. Figuras retiradas de http://www.brainworks.uni-freiburg.de/group/wachtler/VisualSystem/.**

Um grupo de colunas que respondem a linhas (estímulos) com todas as orientações numa região particular do campo visual foi denominado de hipercoluna por Hubel e Wiesel. Essas hipercolunas aparecem repetidas regularmente e precisamente sobre a superfície de V1, ocupando cada uma cerca de $1mm^2$. Essa organização sugere uma modularização do córtex cerebral, na qual cada módulo de uma área do córtex processaria todas as variantes locais da informação visual tratada naquela área cortical. Assim, se uma determinada área processa orientações do estímulo visual, uma hipercoluna dela codifica todas as orientações da região do campo visual monitorada pela hipercoluna. O mesmo ocorrendo para áreas corticais responsáveis por processar. profundidade, movimento, etc.

### 2.3.2 V2

A área V2 possui uma extensa fronteira com a área V1. Esta área apresenta regiões chamadas de faixas grossas e faixas finas separadas por regiões chamadas de interfaixas (vide Figura 18). As faixas finas e interfaixas recebem projeções da via parvocelular que vêm das

camadas 2 e 3 da área V1 enquanto que as faixas grossas recebem projeções da via magnocelular que vêm das camadas 4B, 4Cα e 4Cβ. As faixas finas e as interfaixas se projetam para a área V4, enquanto que as faixas grossas se projetam para área temporal média (MT ou V5). Estes caminhos não são totalmente separados conforme descrito, pois existem conexões entre as faixas finas e grossas e também projeções da área V4 de volta para as faixas finas de V2. Também existem conexões das faixas grossas com a área V3.

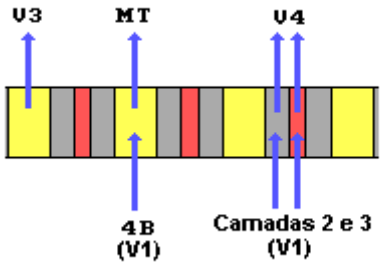



**Figura 18: Esquemático de V2. Figura retirada de [MAT12]**

As células de V2, assim como as células de V1, são sensíveis à orientação, à cor e à profundidade dos estímulos, ou seja, estas células continuam a análise iniciada em V1. A resposta das células de V2 para contornos reais e ilusórios foram testados juntamente com as células de V1 em alguns experimentos. Um exemplo de percepção de contornos ilusórios pode ser visto na Figura 19. Na figura, apesar de não existir um quadrado desenhado, é possível facilmente "enxergar" um contorno ilusório de um quadrado.

Muitas células de V2 responderam aos contornos ilusórios exatamente como responderam às bordas, enquanto que poucas células de V1 responderam aos mesmos contornos ilusórios da mesma forma como responderam às bordas [HEY84]. Essas observações sugerem que na área V2 é feito um processamento de contornos num nível acima do processamento que ocorre em V1, constituindo assim uma evidência da análise progressiva que ocorre no córtex visual.

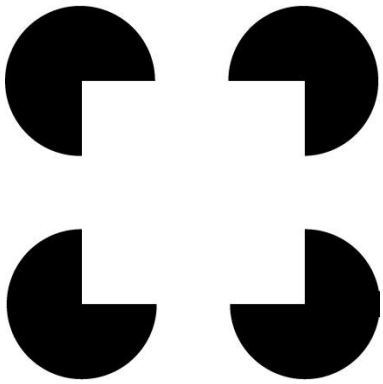


**Figura 19: Contorno ilusório de Kanizsa. É possível "visualizar" um quadrado branco na figura, apesar de não existir um quadrado desenhado explicitamente. Figura retirada de [http://selfpace.uconn.edu/paper/ClarkAusHppp.html].**

### 2.3.3 V3

Pouco se sabe sobre as propriedades funcionais dos neurônios da área extra-estriada V3. Esta área recebe informações das faixas grossas da área V2 e da camada 4B da área V1, e faz projeções tanto para a área temporal média (MT ou V5) quanto para a área V4. Grande parte das células de V3 são seletivas em relação à orientação e direção do estímulo visual, sendo que algumas células são seletivas a cores, o que sugere que em V3 ocorre uma interação entre o processamento de cor e movimento [GEG97].

### 2.3.4 V4

A área extra-estriada V4 foi estudada em profundidade inicialmente por Semir Zeki [ZEK73]. Esta área recebe projeções principalmente das faixas finas e interfaixas de V2, provenientes do caminho parvocelular que vêm do LGN e retina, mas também recebe projeções das áreas V1 e V3. Inicialmente pensava-se que as células de V4 fossem exclusivamente dedicadas ao processamento de cores, porém estudos posteriores mostraram que as células de V4 são sensíveis a combinações de cores e formas. As células de V4 se projetam principalmente para o córtex temporal inferior, onde é feito o reconhecimento de faces e de outras formas complexas.

### 2.3.5 V5 ou MT

A área V5, também conhecida como área temporal média ou MT, recebe projeções da camada 4B do córtex visual primário (V1) e das faixas grossas de V2. Estas conexões são provenientes do caminho magnocelular que parte das células M da retina, passa pelas camadas magnocelulares do LGN e chega ao córtex MT. Assim como a área V1, MT possui um mapa retinotópico do campo visual contralateral, porém os campos receptivos das células de MT são bem maiores que os campos receptivos das células de V1.

O processamento de movimento começa de forma rudimentar em V1 atingindo formas bem mais abstratas em MT numa abstração sucessiva, ou seja, em etapas. Anthony Movshon *et al*. [MOV85] testou a hipótese de que o movimento é processado em 2 etapas, registrando a resposta de células em V1 e MT para um padrão de linhas cruzadas em forma de xadrez em movimento. As células de V1 responderam ao movimento dos elementos isolados do padrão, que são as linhas, enquanto que as células em MT responderam ao movimento do padrão em forma de xadrez por completo.

A percepção de profundidade também é processada em MT. Embora células sensíveis à disparidade binocular sejam encontradas em várias áreas corticais, como V1, V2 e V3, as células de MT respondem melhor a estímulos em distâncias específicas do plano de fixação (mais próximo, ou mais distante do ponto de fixação). Esse processamento da disparidade binocular pode ser utilizado tanto para a percepção de profundidade quanto para o controle do movimento de vergência dos olhos.

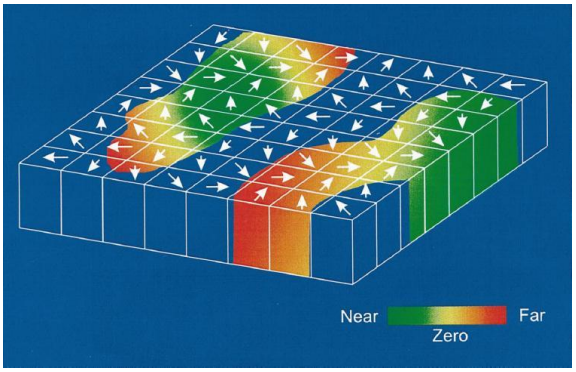



**Figura 20: Modelo esquemático da arquitetura funcional de MT. Figura retirada de [DEA99].**

A Figura 20 apresenta um modelo esquemático da arquitetura funcional de MT que mostra que esta área processa tanto informações de movimento (direção) quanto de profundidade. Na figura, as setas indicam a direção preferencial dos neurônios numa coluna. Estas direções preferenciais variam suavemente através da superfície de MT. A percepção de disparidade é representada pela faixa colorida que varia de *near* (estimulo afastado do ponto de fixação, mas perto do observador), em verde, até *far* (estimulo afastado do ponto de fixação, mas longe do observador), em vermelho, passando pela disparidade zero (estímulo à mesma distância do observador do que o ponto de fixação), codificada em amarelo. As regiões do modelo que estão em azul são regiões da área MT que aparentemente têm pouca seletividade para disparidade.

## 2.4 Vias paralelas

As informações visuais vindas da retina são conduzidas através de vias paralelas que se iniciam na retina, passam pelo LGN, chegam em V1, e depois continuam até os córtices parietal posterior (via dorsal) e temporal inferior (via ventral), como mostra a Figura 21. As células P da retina se projetam para as camadas parvocelulares (camadas 3, 4, 5 e 6) do LGN e seguem para o córtex visual primário, recebendo o nome de via P ou via parvocelular. A partir de V1, esta via se projeta para as faixas finas e interfaixas de V2, que depois seguem para a área V4, formando assim a via ventral que alcança o córtex temporal inferior (Figura 21). Os neurônios que fazem parte da via ventral são mais sensíveis em relação ao contorno das imagens, sua orientação e bordas. Outro aspecto importante que é processado nesta via é a percepção de cores. Estas células possuem alta resolução espacial, baixa resolução temporal e alta sensibilidade a cores e bordas, o que proporciona a este sistema a capacidade de analisar “o quê” é visto. Lesões no lobo temporal inferior causam deficiências relacionadas ao reconhecimento de objetos complexos, inclusive o reconhecimento de faces.

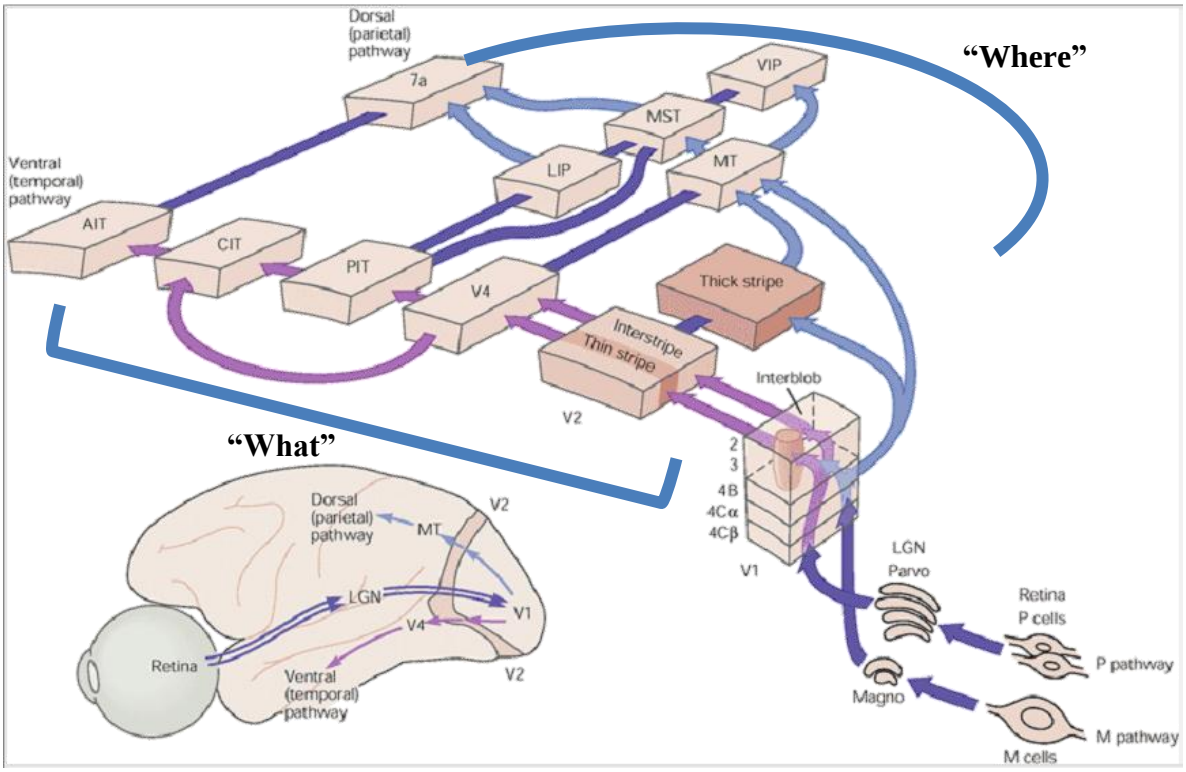



**Figura 21: As vias paralelas M e P projetam-se para o córtex visual passando pelo LGN. Figura retirada de [KAN00] e alterada com a inserção dos nomes dos caminhos "What" e "Where".**

As células M da retina se projetam para as camadas magnocelulares (camadas 1 e 2) do LGN e também seguem para o córtex visual primário, recebendo o nome de via M ou via magnocelular. A via M se estende de V1 até as faixas grossas de V2, que depois se projetam para a área temporal média (MT) formando a via dorsal que se estende até o córtex parietal posterior (Figura 21). Conforme visto anteriormente, o MT (também chamado de V5) está relacionado ao processamento do movimento e profundidade. Os neurônios que formam este sistema são poucos sensíveis a cor e objetos parados, diferentemente dos neurônios associados à via ventral, mas possuem alta resolução temporal e sensibilidade a disparidade binocular, o que faz com que este sistema tenha capacidade de analisar onde ("Where") estão os objetos vistos. Lesões na via dorsal causam deficiência na percepção de movimentos e nos movimentos dos olhos dirigidos a alvos em movimento (movimento de perseguição suave).

A via dorsal, responsável pela análise do quê ("What") é visualizado, continua até terminar numa região do córtex pré-frontal especializada na memória de trabalho visual espacial enquanto que a via ventral, responsável pela análise de onde ("Where") estão os objetos visualizados, continua até terminar numa outra região também do córtex pré-frontal

especializada na memória de trabalho de cognição. Esta análise mostra que o sistema visual está organizado em vias paralelas bem definidas, com uma organização sequencial e hierárquica em cada uma delas.

## 2.5 Sistema óculo-motor

O ângulo total de visão humana possui arco de cerca de 200 graus. A melhor definição fica na fóvea, que tem pouco mais de 1 mm de diâmetro e representa cerca de 2 graus de arco no centro do campo de visão (aproximadamente o tamanho da unha do polegar à distância do braço estendido). Cerca de metade de V1 é devotada inteiramente às fóveas e esta concentração de recursos neurais, que vem da retina e persiste nas outras áreas corticais visuais além de V1, resulta em uma percepção visual muito melhor das imagens que estão sobre as fóveas. Por essa razão, a visão um sistema bastante elaborado para a movimentação rápida e precisa dos olhos.

Os movimentos dos olhos se dão em 3 eixos de rotação (Figura 22): vertical (eixo X, movimento para baixo e para cima - *depression* e *elevation*), horizontal (eixo Y, movimento de lado para outro, *abduction* e *adduction*) e torsional (eixo Z, *extorsions* e *intorsions*)

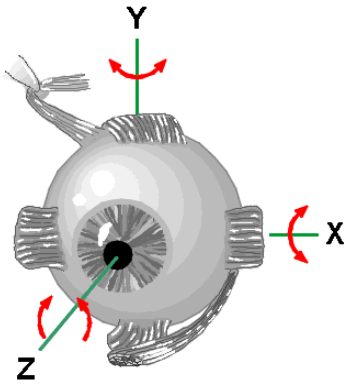



**Figura 22: Movimentos oculares. Figura retirada de http://www.auto.ucl.ac.be/EYELAB/Welcome.html com alterações para inclusão dos eixos X, Y e Z.**

Cada um dos olhos possui 3 pares de músculos extra-oculares, que operam antagonicamente (Figura 23): *Medial Rectus (adduction)* e *Lateral Rectus (abduction)*;

*Superior Rectus (elevation)* e *Inferior Rectus (depression)*; *Superior Oblique (extorsion)* e *Inferior Oblique (introsion)* [KAN00].

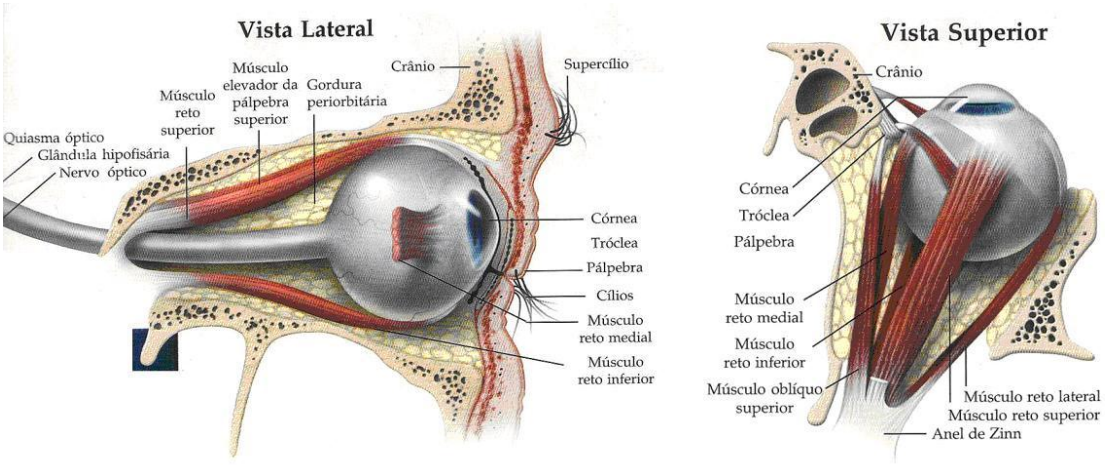



**Figura 23: Músculos oculares. A) Vista Lateral; B) Vista Superior. Figura retirada de http://www20.brinkster.com/tonho/olho/olhohumano.html.**

Olhar de forma exploratória em busca de um ponto de interesse requer mover os olhos rapidamente de modo que a imagem dos objetos seja projetada sobre nossas fóveas. Uma vez localizado o ponto de interesse, contudo, precisamos estabilizar sua imagem na retina, mesmo que a cabeça se movimente. O sistema óculo-motor tem, então, duas grandes funções:

1. Posicionar a imagem do ponto de interesse – o alvo – na parte da retina com maior acuidade, a fóvea;

2. Manter a imagem estacionária na fóvea, independente de movimentos do alvo ou da cabeça.

Por volta de 1902, Raymond Dodge descreveu 5 sistemas separados de controle da posição dos olhos [DOD03]. Estes 5 sistemas podem ser divididos em dois grupos, segundo as duas grandes funções descritas do sistema óculo-motor, como mostrado na Tabela 1.

Os primeiros 4 movimentos são conjugados, cada olho se move na mesma direção e na mesma quantidade. O último não é conjugado: os olhos se movem em direções diferentes e, muitas vezes, de diferentes quantidades.

**Tabela 1: Movimentos Oculares**

| **Movimento** | **Função** |
|---|---|
| | *Movimentos que estabilizam o olho quando a cabeça se move* |
| **Vestíbulo-ocular** | Mantém as imagens estáveis na retina durante rápidas rotações da cabeça |
| **Optokinético** | Mantém as imagens estáveis na retina durante rotação lenta e contínua da cabeça |
| | *Movimentos que mantém a fóvea no alvo* |
| **Sacada** | Trás novos pontos de interesse para a fóvea |
| **Perseguição suave** | Mantém a imagem de um alvo em movimento na fóvea |
| **Vergência** | Ajusta os olhos para que o mesmo ponto seja levado a ambas as fóveas |

Existem outros tipos de movimentos que têm com principal característica amplitudes muito pequenas. Estes movimentos são involuntários e ocorrem quando se está observando um objeto fixo. Estes movimentos são chamados de movimentos de *sustaining* e possuem como principal função manter o foco sobre o objeto que está sendo observado, e produzir variações constantes, mesmo que pequenas, da imagem na retina. Sem estas variações a imagem "desapareceria" devido à acomodação dos neurônios do sistema visual.

## 2.6 Visão binocular

Uma das principais funções do sistema visual é transformar as imagens bidimensionais projetadas na retina numa imagem tridimensional. Estudos indicam que esta transformação é baseada tanto em pistas monoculares para a profundidade como o tamanho familiar dos objetos (sabendo previamente o tamanho aproximado de um objeto, é possível julgar a distância deste objeto), oclusão (caso um objeto A esconda parcialmente um objeto B, assume-se que o objeto A está mais próximo que o objeto B), entre outras pistas, quanto em pistas estereoscópicas oriundas da disparidade binocular causada pela leve diferença das imagens projetadas nas retinas.

Quando os objetos observados estão a distâncias maiores do que cerca de 30 metros, as imagens projetadas nas retinas são praticamente idênticas, eliminando quase que totalmente a disparidade binocular, fazendo com que a percepção de profundidade seja baseada principalmente nas pistas monoculares. Entretanto, a percepção de profundidade a distâncias substancialmente menores do que 30 metros é mediada preferencialmente pela *stereopsis* (originário do grego '*stereo*', que significa sólido, referenciando a imagem em 3D), que é a percepção de profundidade visual baseada na disparidade binocular. A visão estereoscópica é possível porque os dois olhos estão separados horizontalmente, a uma distância média de 65mm em adultos, sendo que cada olho observa o mundo através de uma posição ligeiramente diferente. Desta forma, objetos a distâncias diferentes produzem imagens com afastamento relativo (disparidade) diferente nas retinas como mostra a Figura 24.

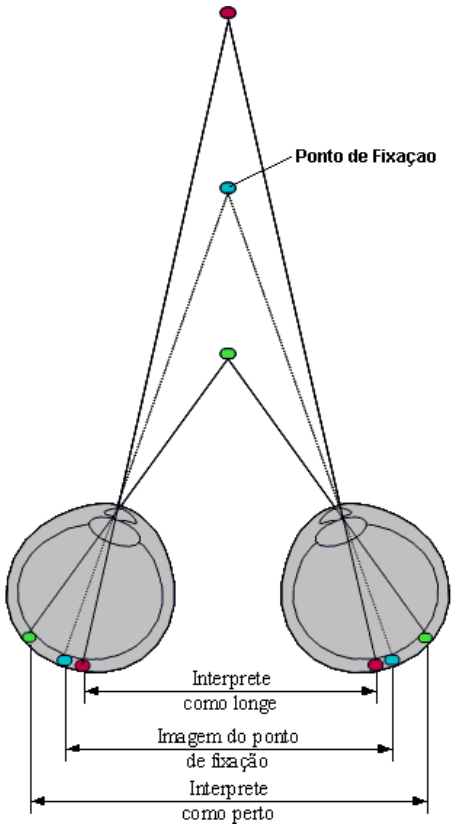



**Figura 24: Visão Binocular. Figura retirada de [KAN00] com alterações.**

Ao olhar um ponto no espaço, a imagem deste ponto é projetada em pontos correspondentes na retina de cada olho, mais precisamente na fóvea de cada olho. Este ponto no espaço é chamado de ponto de fixação e o plano vertical de pontos no espaço onde se localiza o ponto de fixação é chamado de plano de fixação. Pontos no espaço situados antes

do plano de fixação, ou seja, pontos que estão mais próximos do que o ponto de fixação, são projetados em pontos na retina que possuem disparidade positiva (um ponto no olho esquerdo tem seu correspondente no olho direito mais a direita com relação à fóvea), enquanto que os pontos mais distantes do que o ponto de fixação possuem uma disparidade negativa (um ponto no olho esquerdo tem seu correspondente no olho direito mais a esquerda). Pontos que estão a mesma distância possuem disparidade zero (um ponto no olho esquerdo tem seu correspondente no olho direito na mesma posição com relação à fóvea).

### 2.6.1 Geometria binocular

A vergência constitui um dos movimentos oculares mais importantes e é imprescindível para a percepção de distância em relação aos objetos do mundo no campo de visão. Ela permite direcionar os olhos para um determinado ponto, ou seja, fixar as projeções do ponto em questão sobre as fóveas (Figura 24). Em paralelo, o sistema visual é realimentado com informações sobre o posicionamento dos olhos, fornecidas pelo sistema óculo-motor e as interpreta como distância. Desta forma, o observador é capaz de estimar a distância até um ponto em particular do mundo. O processo através do qual se realiza a vergência será abordado na seção 3.3. Nesta seção será discutida a formulação utilizada para calcular a posição de um ponto no espaço a partir das coordenadas deste ponto projetada nas retinas, que neste caso são as câmeras. Para maiores detalhes sobre o processo de vergência e sua relação com mapa de disparidades ver em [OLI05].

Considerando a vergência num ponto qualquer, tem-se a projeção deste ponto nas imagens e, partir daí, localizá-lo no espaço é relativamente simples, a principal dificuldade para viabilizar tal processo é conseguir realizar uma vergência razoavelmente precisa no ponto em questão. A Figura 25 ilustra de forma sucinta a formulação desenvolvida para determinar as coordenadas de um ponto no mundo (espaço 3D).

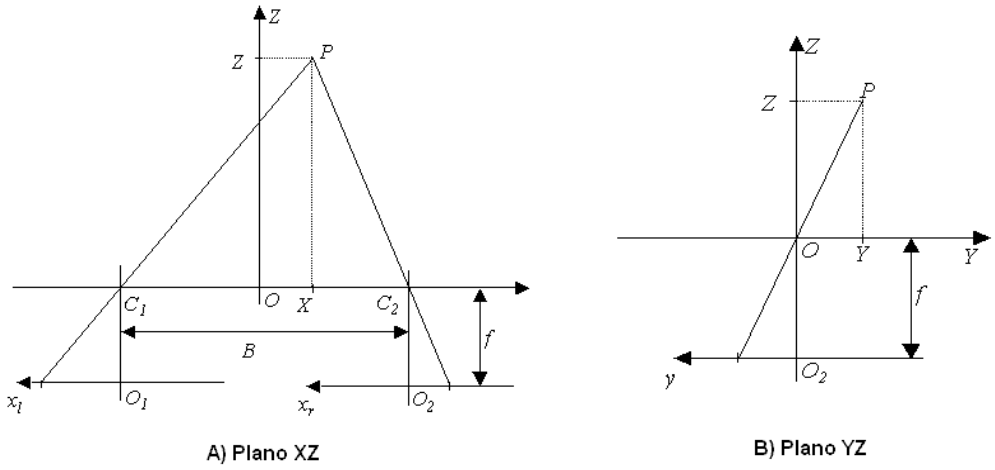



**Figura 25: Geometria do cálculo de um ponto no espaço a partir da projeção nas câmeras. Figura retirada de [OLI05].**

Seja *f* a distância focal das câmeras medida em pixels e *B* a separação entre elas em metros (Figura 25). Por semelhança de triângulos, a coordenada *Z* do ponto no mundo em relação ao referencial *O* (Figura 25-B) é dada por:

$$Z = \frac{f \cdot B}{x_l - x_r}, \qquad \textbf{Equação 1}$$

onde $x_R$ e $x_L$ são as abscissas das projeções do ponto nas imagens direita e esquerda, respectivamente. E $y = y_l = y_r$.

De forma análoga, as coordenadas *X* e *Y* do ponto *P* do mundo são dadas por:

$$X = \frac{B}{2} \cdot \frac{(x_l + x_r)}{(x_l - x_r)} \qquad \textbf{Equação 2}$$

$$Y = \frac{y \cdot B}{(x_l - x_r)} \qquad \textbf{Equação 3}$$

## 2.7 O *Superior Cullicullus*

Esta seção apresenta uma modelagem matemático-computacional proposta em [NET12] do sistema de sacada, ou busca visual, que nos seres humanos fica localizado na região do *superior culliculus* (que está indicado em roxo na Figura 11).

A busca visual é o mecanismo por meio do qual nos permite encontrar um ponto de interesse presente no campo visual a partir de conhecimento prévio da imagem desse ponto de interesse. A região do mesencéfalo conhecida como *superior culliculus* é a responsável pela realização do movimento sacádico dos olhos (o movimento rápido dos olhos), que realiza esse tipo de busca visual nos seres humanos.

Células das camadas mais profundas *do superior culliculus* respondem positivamente antes dos movimentos sacádicos dos olhos, no qual estes trocam rapidamente de um ponto de fixação para outro numa cena [KAN00]. Estas células formam um mapa de movimento sacádico ordenado com o mapa visual que é a imagem oriunda das retinas projetada em V1.

A realização de um procedimento de Busca Visual pelo computador a partir de um conjunto de imagens do mundo externo, que busque similaridade com o sistema biológico, requer uma modelagem das transformações na informação visual que ocorrem durante o caminho que ela faz, saindo da retina, passando pelo núcleo lateral geniculado [KAN00] até o córtex V1 [GRE98, KAN00] e do processamento ocorrido na região do *superior culliculus*, envolvida diretamente no controle dos movimentos de sacada, além do sistema de controle óculo-motor [EBE01] responsável pela movimentação das órbitas oculares.

### 2.7.1 Mapeamento retinotópico retina-V1 (mapeamento log-polar)

Num dado momento, apenas uma fração da informação disponível da cena visual que recai na retina de nossos olhos pode ser processada. Esta fração é representada principalmente

pelas fóveas [KAN00] que, por serem as regiões das retinas que possuem maior quantidade de fotorreceptores, possuem uma maior representação no córtex visual primário.

Através de estudos feitos em primatas, foi verificado como acontece este mapeamento da retina em V1. A Figura 26 mostra a representação de uma imagem contendo círculos concêntricos no córtex V1. Esta representação foi obtida por Tootell [TOO82] através da aplicação de uma substância (2-deoxyglucose) radioativa em um dos olhos de um macaco e do controle da fixação da fóvea [KAN00] deste olho no centro da imagem. Após 45 minutos, o macaco foi sacrificado e a parte correspondente a V1 do hemisfério esquerdo do cérebro do macaco foi recortada, aplainada e posicionada sobre um filme fotográfico sensível à radiação que, quando revelado, mostrou a imagem representada na Figura 26.

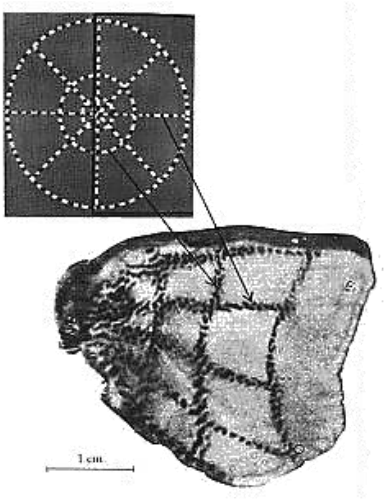



**Figura 26: Representação de uma imagem de círculos concêntricos projetada na retina em V1. Figura retirada de [TOO82].**

Como a Figura 26 mostra, os círculos concêntricos na imagem (à esquerda na figura) se tornam aproximadamente retas em V1, e as regiões circunscritas pelos círculos mais internos, de menor área na imagem, ocupam área muito maior em V1. Este mapeamento, que ocorre na passagem das informações visuais no caminho entre a retina e V1, pode ser modelado matematicamente utilizando uma transformação log-polar da imagem projetada na retina para o córtex visual primário. Esta transformação realiza o mapeamento dos pontos do plano cartesiano (coordenadas x e y) para pontos no plano log-polar (coordenadas $\rho$ e $\theta$) de acordo como mostrado na Figura 27 segundo a Equação 4 e Equação 5.

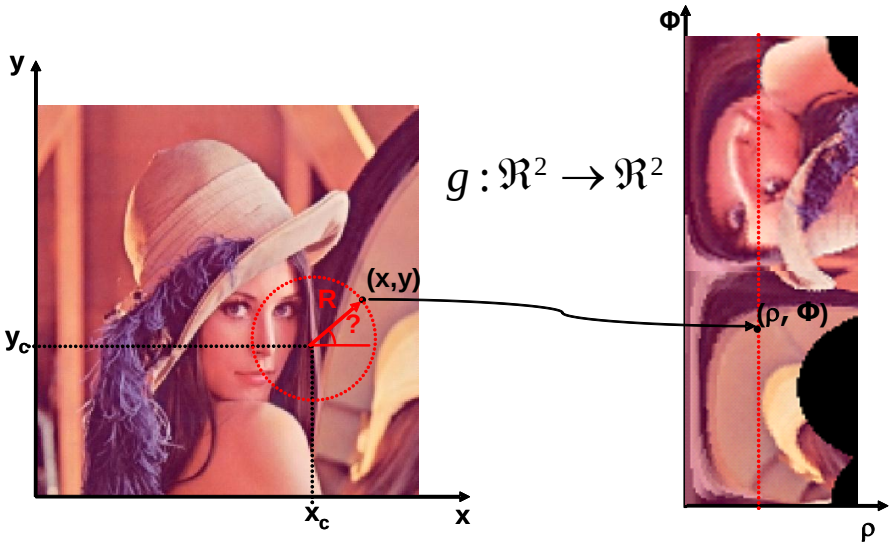



**Figura 27: Transformada log-polar. Figura retirada de [NET12].**

$$R = \sqrt{(x - x_c)^2 + (y - y_c)^2} \rightarrow \rho \, \alpha \, \log(R) \qquad \textbf{Equação 4}$$

$$\theta = \arctan\left( {(y - y_c)} \middle/ {(x - x_c)} \right) \rightarrow \phi \, \alpha \, \theta \qquad \textbf{Equação 5}$$

Uma característica importante desta transformação é que, devido ao comportamento logarítmico da variável ρ, que é diretamente proporcional ao logaritmo da distância R da origem ao ponto (Xc, Yc), Equação 4, as regiões mais próximas ao centro das coordenadas cartesianas possuem uma maior representação, e, portanto uma melhor definição, no plano log-polar, enquanto que as regiões mais afastadas do centro possuem uma menor representação, e, portanto uma pior definição no plano log-polar, semelhante ao que ocorre no sistema visual humano. Este comportamento pode claramente ser visto na Figura 28.

Nesta figura, apresentamos um exemplo da aplicação da transformação log-polar numa imagem seguido do mapeamento inverso. Uma imagem (Figura 28a) de 128x128 pixels sofre uma transformação log-polar com uma amostragem de 64x32 (ρ e θ respectivamente) representada na Figura 28b. Realizando a transformação inversa (Figura 28c) é possível verificar facilmente a perda de definição a medida em que se afasta do centro.


**Figura 28: Aplicação da transformação log-polar seguida do mapeamento inverso. Figura retirada de http://omni.isr.ist.utl.pt/~alex/Projects/TemplateTracking/logpolar.htm.**

Na Figura 27, a transformada log-polar, centrada no ponto (Xc, Yc), da imagem na parte esquerda da figura é como mostrado na imagem da parte direita da figura. Note que o círculo na imagem da esquerda se torna uma reta na imagem da direita. As Equação 4 e Equação 5 são a modelagem matemática da transformada log-polar comumente usada na literatura.

Neste trabalho não empregamos a transformada log-polar da forma como mostramos acima, mas sim uma variante que emula mais precisamente o mapeamento retina-V1 [FAR03]. O resultado da aplicação desta variante da transformada log-polar é mostrado na Figura 29. Como pode ser visto na figura, vizinhanças na imagem ao redor do ponto de fixação, ou seja, a fóvea de nosso modelo (o centro do círculo), são respeitadas na projeção log-polar (retinotopia), como ocorre no cérebro (por meio do corpo caloso [HUB95]). Isso não ocorre na transformada mostrada na Figura 27.

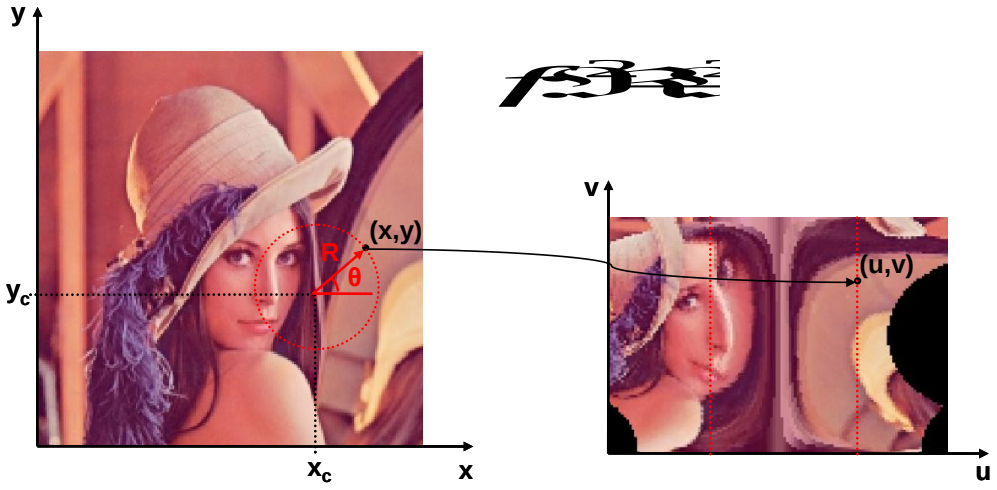



**Figura 29: Transformada log-polar empregada pelo Grupo de Pesquisa em Cognição Visual do LCAD-UFES. Figura retirada de [NET12].**

A Figura 30, abaixo, deixa claro a similaridade entre os resultados obtidos com o modelo aqui empregado e aquele observado no sistema visual biológico. Nesta figura, temos em (a) a imagem projetada na retina do modelo empregado, em (b) a transformada log-polar da imagem projetada na retina e em (c) o mapeamento retina-V1 do córtex de um primata obtido por Tootell em [TOO82].

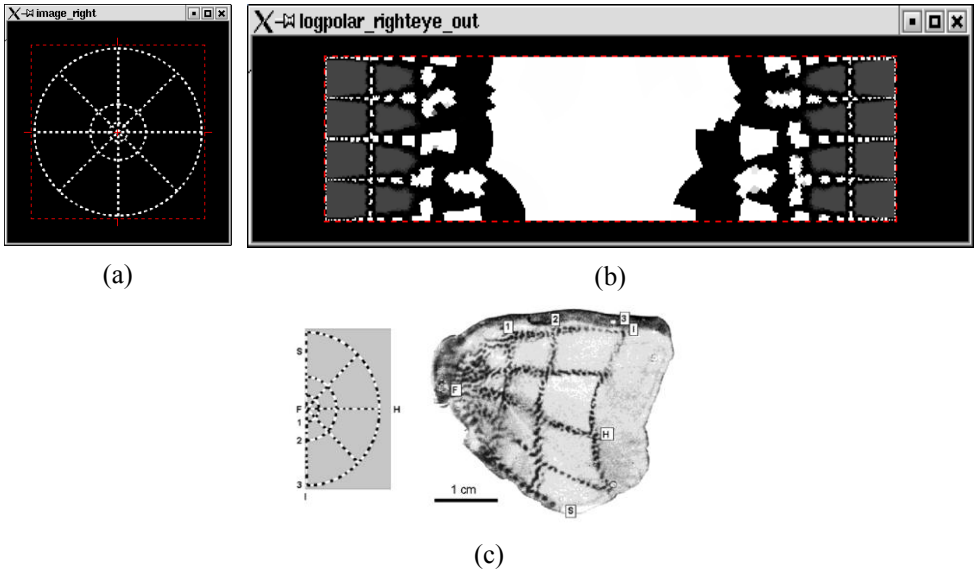



**Figura 30: Similaridade entre o modelo log-polar empregado e o sistema visual biológico. Figura (c) retirada de [TOO82].**

Como pode se ver na Figura 30, o modelo de log-polar empregado guarda enorme semelhança com o sistema visual biológico encontrado em primatas, em que regiões centrais da retina possuem maior acuidade visual e à medida que se afastam do centro perdem resolução. Para manter a projeção de pontos de interesse nessa região central de maior resolução, vimos na Seção 2.5, que nossos olhos realizam diferentes tipos de movimentos com o objetivo de manter um ponto de interesse centrado na fóvea. Na Figura 31 é representado em (a) um ponto de interesse projetado na região periférica da retina e em (b) o movimento ocular realizado para trazê-lo para região central da retina (fóvea).

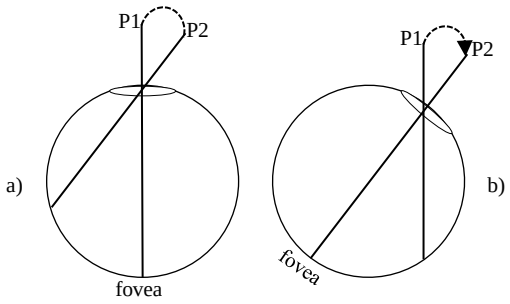



**Figura 31: Movimentos oculares para projeção de pontos de interesse na fóvea. Em (a) um ponto de interesse P2 está projetado na região periférica da retina e se deseja centralizá-lo na fóvea onde há maior acuidade visual. Em (b), está ilustrado o movimento ocular de sacada que leva a projeção do ponto de interesse P2 para a fóvea. Observe que o cruzamento dos raios das projeções de P1 e P2 permanece inalterado. Apenas o olho rotacionou para acomodar o ponto de interesse P2 sobre a fóvea.**

Assumindo a retina como uma superfície esférica e côncava, e o eixo de rotação do olho centrado no centro da esfera, uma consequência importante do uso da modelagem log-polar no tratamento de imagens de câmeras é que ela permite substituir o movimento da câmera na direção de um ponto de interesse pelo simples posicionamento do ponto central da transformada log-polar na imagem deste ponto. Tal propriedade é de grande valia no modelo de Busca Visual implementado (maiores detalhes no Capítulo 3.2).

# 3 SISTEMAS DE BUSCA VISUAL E VERGÊNCIA BASEADOS EM VG-RAM

Neste capítulo apresentamos duas propostas de arquiteturas neurais biologicamente plausíveis que visam reproduzir os movimentos de sacada e vergência dos nossos olhos. Começamos com uma introdução às Redes Neurais Sem Peso, empregada em ambas as arquiteturas.

## 3.1 Redes neurais sem peso

Redes Neurais Sem Peso (RNSP) são baseadas em *Random Access Memories* (RAM) e não armazenam conhecimento em suas conexões, mas em memórias do tipo RAM dentro dos nodos da rede, ou neurônios [ALE66]. Estes neurônios operam com valores de entrada binários e usam unidades RAM [ERC98] para implementar funções baseadas em tabelas-verdade (*lookup tables*) [LUD99, ERC98].

Desta forma, as sinapses de cada neurônio coletam um vetor de bits da entrada da rede que é usado como o endereço da RAM e o valor armazenado neste endereço é a saída do neurônio. O treinamento pode ser feito em um único passo e consiste basicamente em armazenar a saída desejada no endereço associado com o vetor de entrada do neurônio.

Apesar da sua notável simplicidade, RNSP são muito efetivas como ferramentas de reconhecimento de padrões, oferecendo treinamento e teste rápidos e fácil implementação [ALE98]. No entanto, se a entrada da rede for muito grande, o tamanho da memória dos neurônios da RNSP torna-se proibitivo, dado que tem de ser igual a $2^n$, onde n é o tamanho da entrada. As RNSP do tipo *Virtual Generalizing* RAM (VG-RAM) [LUD99] são redes neurais baseadas em RAM que somente requerem capacidade de memória para armazenar os dados relacionados ao conjunto de treinamento.

Os neurônios VG-RAM armazenam os pares entrada-saída observados durante o treinamento, ao invés de apenas a saída. Na fase de teste, as memórias dos neurônios VG-RAM são pesquisadas mediante a comparação entre a entrada apresentada à rede e todas as entradas dos pares entrada-saída aprendidos. A saída de cada neurônio VG-RAM é determinada pela saída do par cuja entrada é a mais próxima da entrada apresentada – a função de distância adotada pelos neurônios VG-RAM é a distância de *Hamming*, ou seja, o número de bits diferentes entre dois vetores de bits de igual tamanho. Se existir mais do que um par na mesma distância mínima da entrada apresentada, a saída do neurônio é escolhida aleatoriamente entre esses pares.

A Tabela 2 ilustra a tabela-verdade (*lookup-table*) de um neurônio VG-RAM com três sinapses ($w_1$, $w_2$ e $w_3$). Esta tabela-verdade contém três entradas (pares entrada-saída) que foram armazenadas durante a fase de treinamento (*entrada* #1, *entrada* #2 e *entrada* #3). Durante a fase de teste, quando um vetor de entrada é apresentado à rede, o algoritmo de teste VG-RAM calcula a distância entre este vetor de entrada e cada entrada dos pares entrada-saída armazenados na tabela-verdade.

No exemplo da Tabela 2, a distância de *Hamming* entre o vetor de nova entrada  e a entrada #1 é dois, porque ambos os bits $w_2$ e $w_3$ não são semelhantes aos bits $w_2$ e $w_3$ do vetor de nova entrada. A distância da entrada #2 é um, porque $w_1$ é o único bit diferente. A distância da entrada #3 é três, podendo ser verificada manualmente neste caso. Portanto, para este vetor de nova entrada, o algoritmo avalia a saída do neurônio, N, como "*saída* 2", pois é o valor de saída armazenado na entrada #2.

**Tabela 2: Tabela-verdade de um neurônio da RNSP VG-RAM**

| *Tabela verdade* | $w_1$ | $w_2$ | $w_3$ | **N** |
|---|---|---|---|---|
| ***entrada* #1** | **1** | **1** | **0** | ***saída* 1** |
| ***entrada* #2** | **0** | **0** | **1** | ***saída* 2** |
| ***entrada* #3** | **0** | **1** | **0** | ***saída* 3** |
| | ↑ | ↑ | ↑ | ↓ |
| ***nova entrada*** | **1** | **0** | **1** | ***saída* 2** |

As RNSP VG-RAM já foram empregadas com sucesso por pesquisadores do LCAD em diversos problemas relacionados à Ciência da Cognição e Aprendizado de Máquina, como o controle de vergência em sistemas de visão artificial [KOM02], reconhecimento de faces [DES08b] e categorização automática de texto [DES08a, DES09, DES07, BAD08].

Esses exemplos citados anteriormente têm em comum o modelo de memória distribuída (individual) dos seus neurônios. Contudo, a Busca Visual, outro exemplo de aplicação de RNSP VG-RAM que veremos em detalhes a seguir, emprega o modelo de memória compartilhada (ou coletiva) dos neurônios.

No caso de memória compartilhada, durante a fase de teste, cada neurônio deve calcular a distância de *Hamming* entre a sua entrada e as entradas armazenadas por todos os neurônios da rede. Neste caso, a saída também é a associada à entrada da tabela-verdade que possuir a menor distância de *Hamming*, mas podendo ter sido memorizada por qualquer neurônio da rede.

Vale ressaltar que o modelo de memória compartilhada implica em maior gasto de tempo para se computar por completo a saída da rede neural. No modelo de memória distribuída tradicional, este tempo é diretamente proporcional ao número de entradas armazenadas durante a fase de treinamento, enquanto que com memória compartilhada, este passa a ser proporcional ao número de entradas de treinamento vezes o número de neurônios, uma vez que a memória é consultada de forma coletiva.

## 3.2 Arquitetura neural do sistema de busca visual

Nesta subseção apresentamos detalhes sobre a arquitetura do sistema de Busca Visual implementado com Redes Neurais Sem Peso. Inicialmente, fornecemos uma visão geral da arquitetura, detalhando as camadas que a compõem, como estas operam e como são conectadas entre si. Em seguida, explicamos o funcionamento do padrão de interconexão sináptico tipo log-polar. E, por fim, apresentamos os métodos considerados para o tratamento da saída da camada neural. Também discutimos sua plausibilidade biológica e de que forma estes métodos influenciam a convergência e o desempenho do sistema de Busca Visual Neural.

### 3.2.1 Visão geral da arquitetura neural

No modelo de busca visual implementado, empregamos uma arquitetura de RNSP VG-RAM com uma camada bidimensional de neurônios, N, com m×n neurônios. A Figura 32 mostra a arquitetura neural, que tem como componente principal os neurônios N e na sua base a imagem de entrada I de dimensões $\xi$ x $\eta$.

Antes de chegar à entrada da rede neural, $\Phi$ (de dimensões $2\xi$ x $2\eta$), a imagem I sofre uma translação, um ajuste de escala e uma suavização gaussiana. A translação é aplicada de modo que o ponto de interesse da busca visual, o ponto de fixação dos olhos, esteja posicionado no pixel central da camada de entrada $\Phi$, onde se conectam as sinapses dos neurônios. O ajuste de escala e a suavização gaussiana são aplicados de modo que a imagem na camada de entrada $\Phi$ da rede seja o mais semelhante à incidente no olho humano (aproximação do objeto de interesse e ajuste de foco ocular) [HUB95].

Sinapses de RNSP coletam apenas um bit (0 ou 1) da entrada. Para permitir seu uso com entradas que podem assumir valores não binários, usamos *minchinton cells* [MIT98]. Na arquitetura neural empregada, cada sinapse, $w_t$, forma uma *minchinton cell* com a próxima, $w_{t+1}$ ($w_{|W|}$ forma uma *minchinton cell* com $w_1$). Cada uma destas *minchinton cells* retorna 1 se a sinapse $w_t$ está conectada a um elemento da camada de entrada cujo valor é maior que o

valor do elemento ao qual a sinapse $w_{t+1}$ está conectada; caso contrário a *minchinton cells* irá retornar 0. Existem várias formas de comparar os elementos da entrada Φ. Neste trabalho, essa comparação é feita por meio da sinapse de cor seletiva composta [NET12].

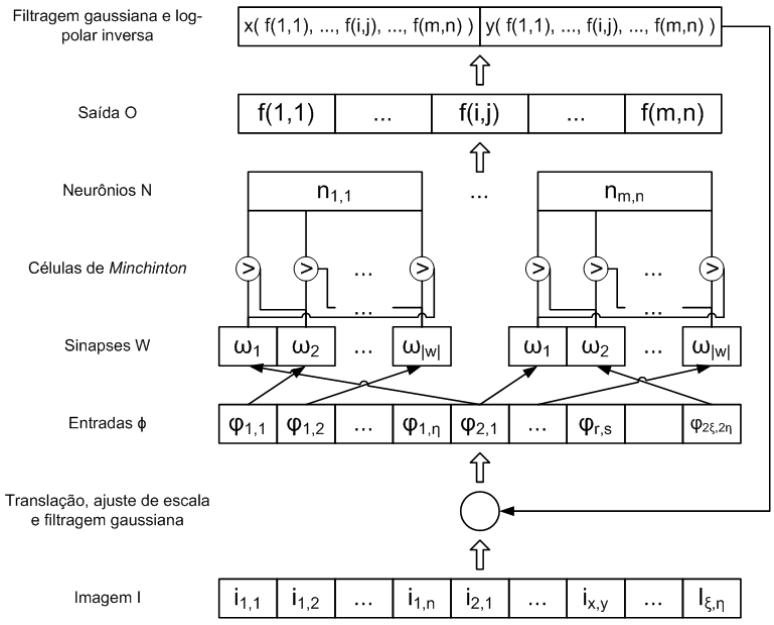



**Figura 32: Arquitetura neural usada na aplicação de busca visual.**

### 3.2.2 Padrão de interconexão sináptico log-polar

Cada neurônio, $n_{i,j}$, possui um conjunto de sinapses, $W = \{w_1, ..., w_{|W|}\}$. Estas sinapses são conectadas à entrada bidimensional da rede, Φ, composta de $2\xi$ x $2\eta$ elementos de entrada, $\varphi_{r,s}$, segundo um padrão de interconexão $\Omega_{i,j}(W)$. Este padrão de interconexão sináptica segue uma distribuição aleatória bidimensional com PDF (*Probabilistic Density Function*) Normal centrada no pixel $\varphi_{\mu i,\mu j}$, onde as coordenadas $\mu_i$ e $\mu_j$ dos elementos de Φ são determinados pela transformação log-polar inversa das coordenadas i, j da camada neural N, como mostrado na Figura 33, segundo a Equação 6, Equação 7, Equação 8 e Equação 9.

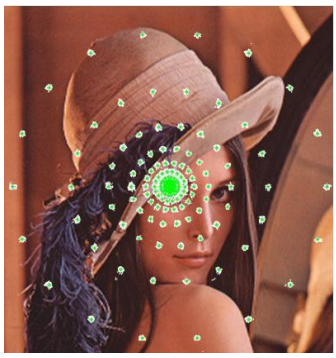


**Figura 33: Mapeamento da posição central dos neurônios no plano da imagem de entrada. Figura retirada de [NET12].**

$$d = \xi \cdot \left( 2^{\left( \frac{|i - m/2|}{m/2} - 1 \right)} - \frac{1}{2} \right) \qquad \textbf{Equação 6}$$

$$\theta = \begin{cases} \pi \cdot \left( \frac{3n}{2} - \frac{j}{n} \right) + \frac{\pi}{2n} \; ; \; \text{se } i < \frac{m}{2} \\ \pi \cdot \left( \frac{3n}{2} + \frac{j}{n} \right) + \frac{\pi}{2n} \; ; \text{se } i > \frac{m}{2} \end{cases} \qquad \textbf{Equação 7}$$

$$\mu_i = \frac{\xi}{2} + d \cdot \cos(\theta) \qquad \textbf{Equação 8}$$

$$\mu_j = \frac{\eta}{2} + d \cdot sen(\theta) \qquad \textbf{Equação 9}$$

,

onde m denota o número de linhas da camada neural N e n denota o número de colunas da camada neural N.

As coordenadas r e s dos elementos de Φ aos quais $n_{i,j}$ se conecta via W seguem as PDFs da Equação 10 e Equação 11, onde o desvio padrão σ da distribuição gaussiana é um parâmetro da arquitetura neural.

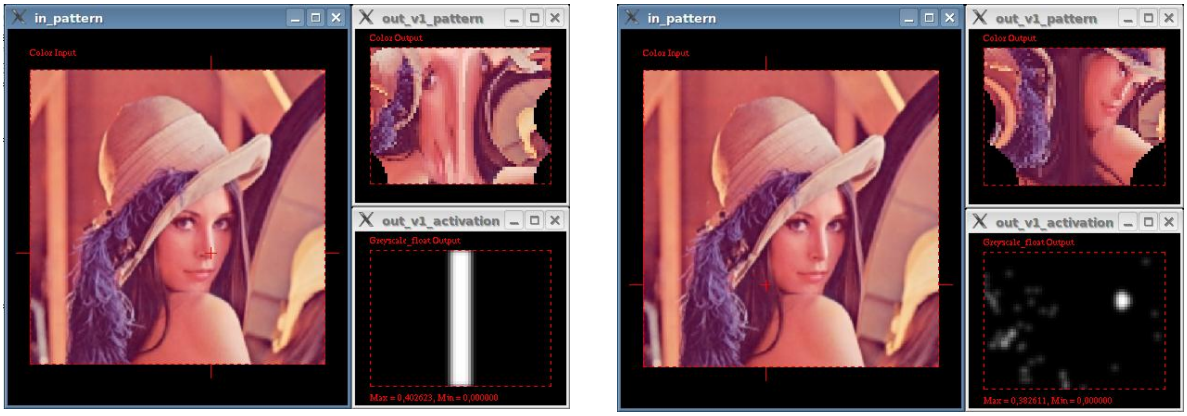
$$\varphi_{\mu_i, \sigma^2}(r) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(k - \mu_i)^2}{2\sigma^2}} \qquad \textbf{Equação 10}$$

$$\varphi_{\mu_j, \sigma^2}(s) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(l - \mu_j)^2}{2\sigma^2}} \qquad \textbf{Equação 11}$$

Este padrão de interconexão sináptica é comum em neurônios biológicos [KAN00]. Entretanto, uma diferença do padrão de interconexão sináptica adotado na aplicação de busca visual em relação ao empregado em outras aplicações com RNSP VG-RAM é que, embora aleatório, é o mesmo para todos os neurônios, além destes possuírem memória comum como citado anteriormente.

### 3.2.3 Operação do sistema de busca visual

Dada a arquitetura neural descrita anteriormente, quando a rede é treinada, cada neurônio armazenará um padrão de bits condizente com a região da imagem que suas sinapses monitoram e associado a ele uma saída cujo valor é dado pela distribuição normal truncada com média m/2 e desvio padrão parametrizável, que denominamos padrão de ativação em forma de "barra". No teste da rede o padrão de bits monitorado pelas sinapses de cada neurônio será comparado com os padrões de bits aprendidos por todos os neurônios – a isto denominamos memória coletiva dos neurônios. A Figura 34, abaixo, mostra exemplos de situações de treinamento e uso de nossa arquitetura de RNSP para busca visual.




(a) (b)

**Figura 34: Treinamento e uso do sistema de busca visual. (a) Fase de Treinamento (b) Fase de Teste (Realização da Busca Visual). Figura retirada de [NET12].**

A Figura 34(a) mostra uma situação de treinamento onde a rede é treinada para buscar o nariz da modelo na imagem. Na imagem da esquerda da Figura 34(a), a cruz vermelha é o centro da log-polar; enquanto que, na imagem superior direita da Figura 34(a) apresentamos o

mapeamento log-polar da imagem na camada de neurônios; por fim, na imagem inferior direita da Figura 34(a), a saída da matriz de neurônios após o treinamento. Como esta última figura mostra, os neurônios com centro do campo receptivo *(receptive fields* – RFs) [KAN00] sobre ou próximos ao ponto de interesse são treinados para gerar saídas com valores altos (branco ou tons de cinza), enquanto que aqueles com RFs distantes do ponto de interesse são treinados para gerar saída zero (preto). O RF de um neurônio é dado pelos pixels monitorados pelo neurônio por meio de suas sinapses.

A Figura 34(b) mostra uma situação de busca visual onde os neurônios da rede, treinados para buscar a imagem do nariz da modelo, geram suas saídas de acordo com a região da imagem monitorada por seus RFs. Como a imagem inferior direita da Figura 34(b) mostra, os neurônios com RF sobre o nariz da modelo geram saída mais alta.

Na busca visual é feita após uma filtragem gaussiana da saída da camada neural e computado um vetor de deslocamento sacádico [LEE88, WAL05], usando essa saída filtrada. Em seguida é realizado o deslocamento do centro da log-polar e a saída dos neurônios é computada novamente. Quando esta movimentação do centro da log-polar (movimento sacádico [KAN00]) levar ao mesmo ponto, ou seja, quando uma transformação sobre a saída da camada neural acusar deslocamento igual à zero, foi encontrado um ponto de interesse, caso contrário, um novo ciclo de movimento sacádico e avaliação da saída da rede é efetuado até um limite máximo, parametrizado, de vezes consecutivas.

O grau de certeza da rede quando da convergência neste ponto pode ser medido comparando-se a imagem final na saída da camada neural com aquela usada no treinamento (imagem inferior direita da Figura 34(a)). No exemplo da Figura 34(b), com apenas um passo de busca visual o ponto de interesse é encontrado com grau de certeza muito próximo ao máximo.

### 3.2.4 Tratamento da saída da camada neural e movimento sacádico

Walton [WAL05] sugere em seus trabalhos relacionados à controle visio-motor de movimentos sacádicos que o movimento de sacada é determinado por uma soma vetorial ponderada, segundo o padrão de ativação dos neurônios do *superior culliculus*. Tal hipótese

possui embasamento em experimentos com primatas envolvendo movimentos sacádicos sujeitos a múltiplos alvos e consequentemente a ativação de múltiplas localidades do *superior culliculus* [POR03].

Desta forma, o centro da log-polar seria deslocado segundo um vetor $\vec{D}$ dado pela Equação 12, onde $\vec{V}(i, j)$ representa o vetor de deslocamento sacádico codificado pelo mapeamento da posição central do neurônio $n_{i,j}$ no plano da imagem segundo a transformada log-polar inversa e $f(i, j)$ o valor de saída retornado por este neurônio após a filtragem gaussiana na saída.

$$\vec{D}(i, j) = \frac{\sum_{i,j} \vec{V}(i, j) \cdot f(i, j)}{\sum_{i,j} f(i, j)} \qquad \textbf{Equação 12}$$

Muito embora seja de grande plausibilidade biológica, tal abordagem torna os movimentos sacádicos do sistema de Busca Visual que implementamos muito sujeitos à influência de ruídos, ou à presença de regiões de pouca relevância na camada de entrada Φ.

Isto ocorre, pois, mesmo para alguns neurônios cuja posição central no plano de Φ esteja sobre um alvo indesejado, este pode apresentar uma ativação diferente de zero, ainda que baixa. Desta forma, o vetor sacádico $\vec{V}(i, j)$ do neurônio $n_{i,j}$ influencia de forma indesejável esta soma vetorial e pode fazer com que o sistema de Busca Visual se desvie do ponto de interesse. Estes desvios podem ser vistos como uma consequência direta do uso da geometria log-polar no modelo que implementamos se comparado à modelos de geração de movimentos sacádicos com geometria planar [WAL05].

Logo, uma forma de se evitar este tipo de perda mantendo-se a plausibilidade biológica do modelo é calcular a função inversa da log-polar nas coordenadas do neurônio com maior valor de saída, como mostrado na Equação 13.

$$\vec{D}(i, j) = \vec{V}\left( \arg\max_{i,j} (f(i, j)) \right) \qquad \textbf{Equação 13}$$

Com este procedimento é possível realizar movimentos sacádicos mais precisos e menos sujeitos à ruídos, uma vez que os neurônios cujas sinapses sensibilizam uma região de interesse na imagem tendem a apresentar níveis de ativação muito mais elevados que os demais neurônios, logo, fazendo com que a busca convirja para uma região de máxima certeza [WAL05].

## 3.3 Arquitetura neural do sistema de vergência

Nesta subseção apresentamos detalhes sobre a arquitetura do sistema de vergência implementado com Redes Neurais Sem Peso. Inicialmente, fornecemos uma visão geral da arquitetura, detalhando as camadas que a compõem, como estas operam e como são conectadas entre si. Nossa arquitetura difere substancialmente daquelas propostas inicialmente em [KOM02]. Em seguida, explicamos o funcionamento do padrão de interconexão sináptico tipo log-polar. E, por fim, apresentamos os métodos considerados para o tratamento da informação codificada no padrão de ativação dos neurônios da rede. Também discutimos sua plausibilidade biológica e de que forma estes métodos influenciam a convergência e o desempenho do sistema de vergência neural.

### 3.3.1 Visão geral da arquitetura neural

No modelo de vergência implementado, empregamos uma arquitetura de RNSP VG-RAM com uma camada bidimensional de neurônios, N, com m×n neurônios. A Figura 35 mostra a arquitetura neural, que tem como componente principal os neurônios N e na sua base a imagem de entrada I de dimensões $\xi$ x $\eta$.

Antes de chegar à entrada da rede neural, $\Phi$ (de dimensões $\xi$ x $\eta$), a imagem I sofre uma suavização gaussiana, de modo que a imagem na camada de entrada $\Phi$ da rede seja o mais semelhante à incidente no olho humano (ajuste de foco ocular) [HUB95].

Como dito anteriormente, sinapses de RNSP coletam apenas um bit (0 ou 1) da entrada. Portanto, assim como na Busca Visual, utilizamos *minchinton cells* [MIT98] para permitir

entradas com valores não binários, como por exemplo imagens. Mas diferente da Busca Visual, a comparação entre os elementos da entrada é feita por meio da sinapse de cor RGB [OLI05].

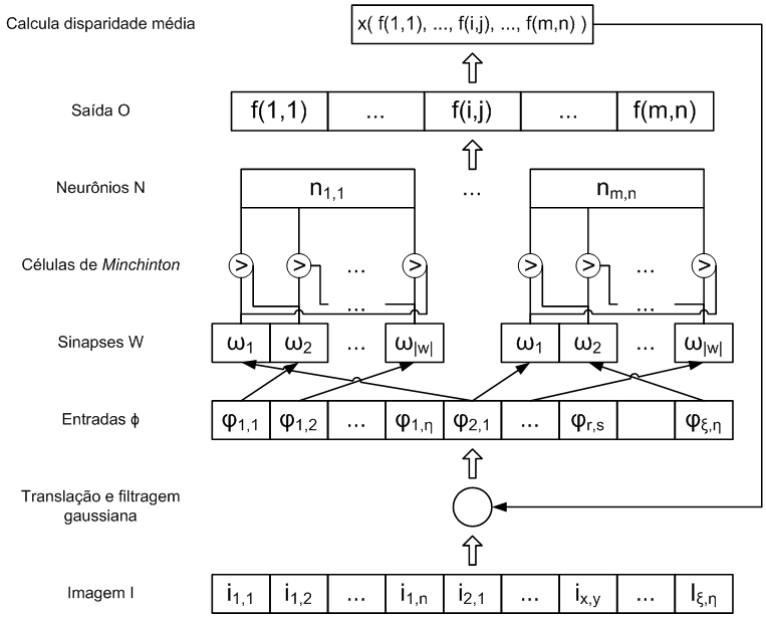



**Figura 35: Arquitetura neural usada no sistema de Vergência.**

### 3.3.2 Padrão de interconexão sináptico log-polar

Cada neurônio, $n_{i,j}$, possui um conjunto de sinapses, $W = \{w_1, ..., w_{|W|}\}$. Estas sinapses são conectadas à entrada bidimensional da rede, $\Phi$, composta de $\xi$ x $\eta$ elementos de entrada, $\varphi_{r,s}$, segundo um padrão de interconexão $\Omega_{i,j}(W)$. Este padrão de interconexão sináptica segue uma distribuição aleatória bidimensional com PDF (*Probabilistic Density Function*) Normal centrada no pixel $\varphi_{\mu i,\mu j}$, onde a posição central dos neurônios $\mu_i$ e $\mu_j$ no plano da imagem é determinada pela transformação log-polar inversa da posição i,j dos neurônios no plano da imagem, como mostrado na Figura 33, segundo as Equação 6, Equação 7, Equação 8 e Equação 9.

Desta forma, as coordenadas r e s dos elementos de $\Phi$ aos quais $n_{i,j}$ se conecta via W seguem as PDFs da Equação 10 e Equação 11, onde o desvio padrão $\sigma$ da distribuição gaussiana é um parâmetro da arquitetura neural, e $\mu_i$ e $\mu_j$ representa a posição central do neurônio $n_{i,j}$ no plano da camada de entrada $\Phi$.

Este padrão de interconexão sináptica é comum em neurônios biológicos [KAN00]. Entretanto, uma diferença do padrão de interconexão sináptica adotado no sistema de vergência em relação ao empregado em outras aplicações com RNSP VG-RAM é que, embora aleatório, é o mesmo para todos os neurônios, além destes possuírem memória individual, diferente da busca visual.

### 3.3.3 Treinamento do sistema de vergência

Considerando a arquitetura neural descrita anteriormente, quando a rede é treinada, é como se fossem criadas camadas de neurônios treinadas com um valor de disparidade específico para cada posição monitorada na imagem de entrada, conforme ilustrado na Figura 36. O treinamento da RNSP para vergência é realizado para um determinado número máximo de disparidades parametrizado n, que determina quão próximo se deseja procurar pelo ponto de interesse de vergência, lembrando que a disparidade é inversamente proporcional à distância. Sendo assim, o treinamento consiste em definir o ponto de interesse de vergência na imagem direita como o centro da log-polar para em seguida deslocar a imagem direita iterativamente de [-n;+n] disparidades gravando na camada neural a cada passo a disparidade em questão (-n).

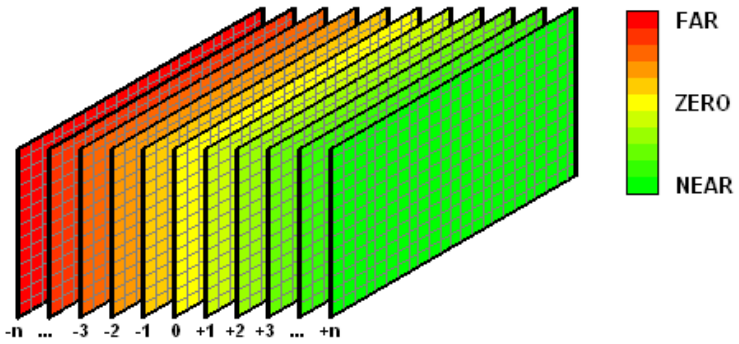



**Figura 36: Treinamento do sistema de vergência. Figura retirada de [OLI05].**

### 3.3.4 Tratamento da saída da camada neural e controle de vergência

A Figura 37 mostra exemplos de situações de uso de nossa arquitetura de RNSP para vergência.

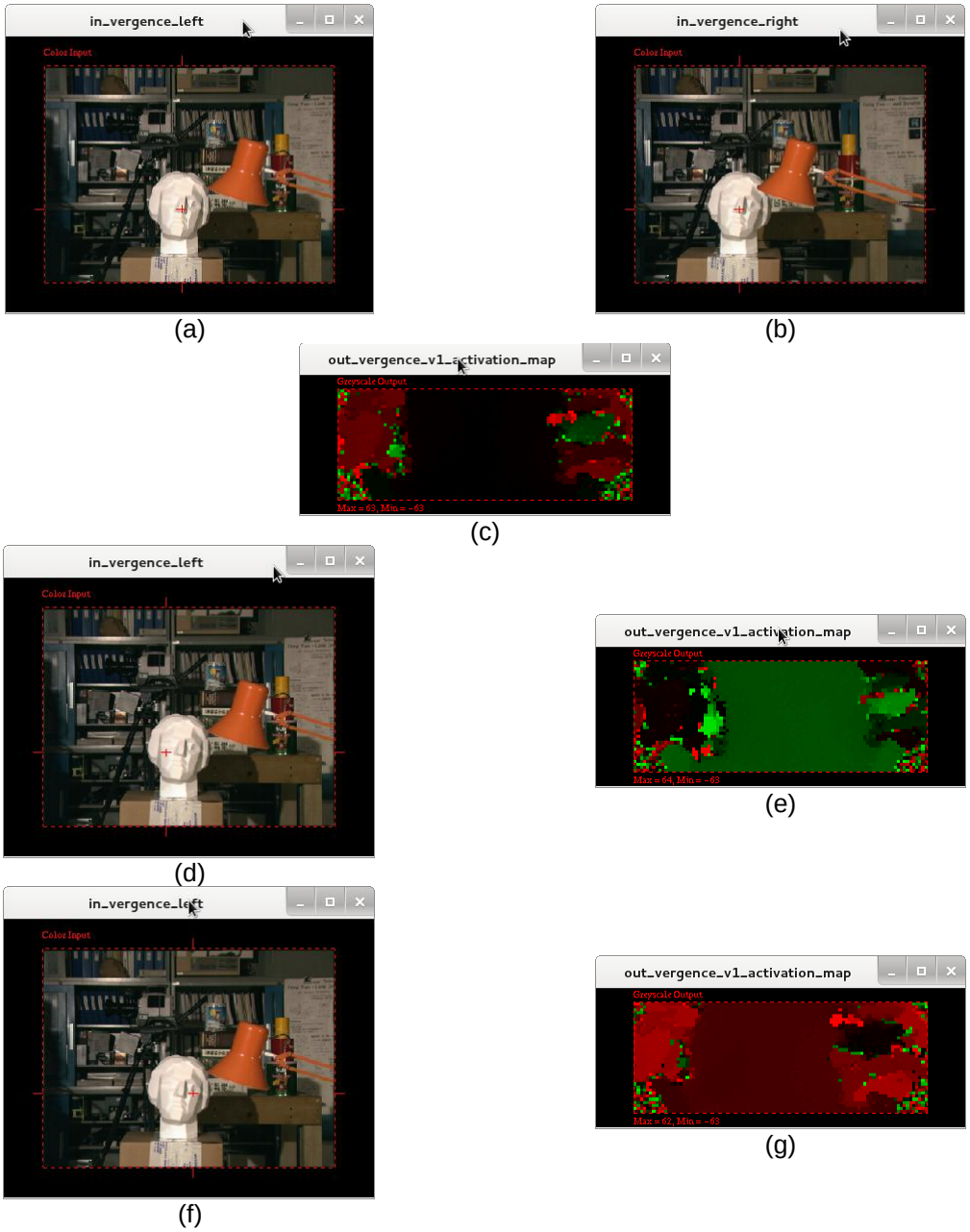



**Figura 37: Funcionamento do sistema de vergência. (a) Imagem direita usada para treino. (b) Imagem esquerda após vergência final. (c) Mapa de disparidade final. (d) Imagem esquerda com ponto de atenção aquém do ponto de vergência. (e) Mapa de disparidade com ponto de atenção aquém do ponto de vergência. (f) Imagem esquerda com ponto de atenção além do ponto de vergência. (g) Mapa de disparidade com ponto de atenção além do ponto de vergência.**

A Figura 37 (a), (b), (d) e (f) mostra a imagem Tsukuba (http://vision.middlebury.edu/stereo/) com uma cruz vermelha representando o ponto de atenção central do mapeamento log-polar da RNSP. A Figura 37 (b) demonstra uma situação de treinamento, em que a rede é treinada para vergir o ponto de atenção para o nariz da escultura visto na imagem direita. A Figura 37 (a) demonstra o ponto de atenção posicionado exatamente sobre o nariz da escultura na imagem esquerda após a vergência final. Na Figura 37 (c) apresentamos o mapa de ativação log-polar (ou mapa de disparidade codificado em cores que representam disparidades positivas em verde, negativas em vermelho e zero em preto) obtido como resposta dos neurônios após a vergência. O significado da resposta dos neurônios é a disparidade que cada um diz estar do ponto de atenção. Sendo assim, para fazer a vergência simplesmente calculamos a média de todas as respostas (disparidades) dos neurônios. O centro da log-polar seria deslocado segundo um valor positivo ou negativo de disparidade $d$ que tenderia a disparidade média da saída dos neurônios para zero, como mostrado na Equação 14.

$$d = \arg\min_{i,j}\left(\frac{\sum f(i,j)}{m \cdot n}\right) \qquad \textbf{Equação 14}$$

Em seguida, é realizado o deslocamento que foi calculado e a saída dos neurônios é atualizada novamente. Quando esta movimentação do centro da log-polar, dito movimento sacádico [KAN00], levar ao mesmo ponto, a disparidade média será próxima de zero e o ponto de vergência estará bem próximo do desejado. Caso contrário, um novo ciclo de movimento sacádico e uma nova avaliação da saída da rede é efetuada até um limite máximo, parametrizado, de vezes consecutivas. Na Figura 37 (d) e (e) é possível visualizar o ponto de atenção à esquerda do ponto de vergência desejado (nariz da estátua) e, consequentemente, a região central da camada de saída dos neurônios indicando em verde uma disparidade positiva a ser tomada. Já a Figura 37 (f) e (g) apresenta um caso ilustrativo em que o ponto de atenção está à direita do ponto de vergência desejado (nariz da estátua) e, então, a região central da camada de saída dos neurônios indicando que há uma disparidade negativa (codificada em vermelho) a ser realizada até o ponto de vergência. Observe novamente na Figura 37 (c) que a região central da camada de saída dos neurônios quando o ponto de atenção está sobre o ponto de vergência é zero (codificada em preto).

# 4 LOCALIZAÇÃO E MAPEAMENTO SIMULTÂNEOS - SLAM

O problema de localização e mapeamento simultâneos (SLAM) [THR05] é provavelmente o mais fundamental da robótica autônoma. Veículos robóticos autônomos necessitam saber onde estão (*Localizing*) em sua área de atuação (*Mapping*) para navegar nela e realizar atividades de interesse. Esse consiste em construir de forma incremental um mapa condizente do ambiente e, simultaneamente, localizar o robô nesse mapa. O problema de SLAM já foi formulado e resolvido para diferentes tipos de aplicações, desde ambientes fechados (*indoor*) a ambientes abertos (*outdoor*) terrestres, aéreos e submarinos.

Cada tipo de aplicação pode variar o emprego de sensores, mapas e estratégias de SLAM que forem mais apropriadas ao contexto. Por exemplo, quase a totalidade dos veículos que participaram das DARPA *Grand Challenges* empregou câmeras e sensores laser (*Laser Range Scan*, ou *Light Detection And Ranging* - LIDAR) para examinar o terreno à frente e realizar, juntamente com sensores GPS, SLAM e, a partir dos mapas, evitar obstáculos e perseguir os objetivos das provas por meio da navegação.

## 4.1 Formulação probabilística do problema de SLAM

Considere um robô móvel que se desloca através de um ambiente qualquer e que utiliza um sensor para fazer observações sobre uma série de pontos de referência (*landmarks)* desconhecidos, como ilustrado na Figura 38. Neste contexto, pode-se definir, para um instante de tempo *k*, as seguintes variáveis:

- $\mathbf{x}_k$: um vetor de estado que descreve a localização e orientação (pose) do robô no tempo *k*.
- $\mathbf{u}_k$: um vetor de controle, empregado entre o tempo *k*-1 e o tempo *k*, para deslocar o robô do estado $\mathbf{x}_{k-1}$ para o estado $\mathbf{x}_k$ no tempo *k*.
- $\mathbf{m}_i$: um vetor que descreve a localização estática de uma i-ésima *landmark*.

- $\mathbf{z}_{k,i}$: uma observação, realizada pelo robô, da i-ésima *landmark* no tempo *k*.

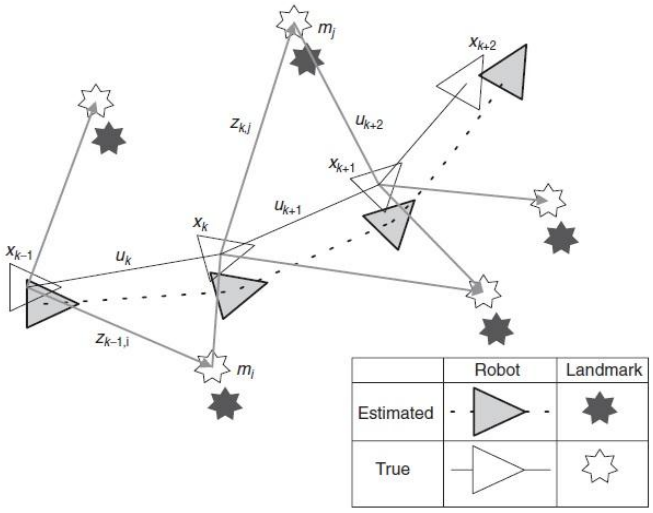



**Figura 38: O problema de SLAM. As verdadeiras posições nunca são conhecidas ou medidas diretamente. Observações são feitas entre o robô real e as *landmarks*. Figura retirada de [DUR06].**

Além disso, os seguintes conjuntos podem também ser definidos:

- $\mathbf{X}_{0:k} = \{\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_k\} = \{\mathbf{X}_{0:k-1}, \mathbf{x}_k\}$: histórico de poses do robô.
- $\mathbf{U}_{0:k} = \{\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_k\} = \{\mathbf{U}_{0:k-1}, \mathbf{u}_k\}$: histórico de controles empregados.
- $\mathbf{m} = \{\mathbf{m}_1, \mathbf{m}_2, \cdots, \mathbf{m}_n\}$: *landmarks.*
- $\mathbf{Z}_{0:k} = \{\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_k\} = \{\mathbf{Z}_{0:k-1}, \mathbf{z}_k\}$: observações de *landmarks*.

A solução do problema de SLAM, modelado de forma probabilística, requer que a distribuição de probabilidade da Equação 15 seja computada para todos os instantes *k*. Essa distribuição define a probabilidade do robô estar em uma determinada pose (estado $\mathbf{x}_k$) em um mapa de *landmarks* **m**, dado um conjunto de observações $\mathbf{Z}_{0:k}$**,** controles $\mathbf{U}_{0:k}$ e pose inicial $\mathbf{x}_0$.

$$P(\mathbf{x}_k, \mathbf{m} | \mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{x}_0) \qquad \textbf{Equação 15}$$

Começando com uma estimativa para o estado inicial na Equação 15 no momento *k*-1, após um controle de entrada $\mathbf{u}_k$ e uma observação de *landmark* $\mathbf{z}_k$, calcula-se a probabilidade *P* utilizando o teorema de Bayes [THR05]. Este cálculo exige que um modelo de transição de estado e um modelo de observação sejam definidos, descrevendo o efeito do controle aplicado

ao robô na estimativa da sua localização e da observação realizada pelo robô para corrigir a sua localização com base no mapa de *landmarks* obtido.

O modelo de observação descreve a probabilidade de se realizar uma observação $\mathbf{z}_k$, dado que a pose do robô e seu mapa de *landmarks* sejam conhecidos, e é geralmente descrito como a probabilidade *P* da Equação 16.

$$P(\mathbf{z}_k|\mathbf{x}_k, \mathbf{m}) \qquad \textbf{Equação 16}$$

O modelo de transição de estados descreve a probabilidade do robô estar no estado $x_k$ dado o estado anterior $x_{k-1}$ e o comando $u_k$ associado ao movimento, que pode ser descrito em termos da distribuição de probabilidade da Equação 17.

$$P(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{u}_k) \qquad \textbf{Equação 17}$$

Assume-se a transição de estados como um processo de Markov [THR05], no qual o próximo estado $\mathbf{x}_k$ depende somente do estado imediatamente anterior $\mathbf{x}_{k-1}$ e do controle aplicado $\mathbf{u}_k$ (Figura 39), ou seja, a probabilidade do robô estar em determinada posição no momento *k* é calculada de acordo com sua pose anterior e o comando enviado ao robô. É importante ressaltar que o modelo de transição de estados é independente das observações realizadas e do mapa de *landmarks* do robô e o modelo de observação é dependente da pose do robô e do seu mapa.

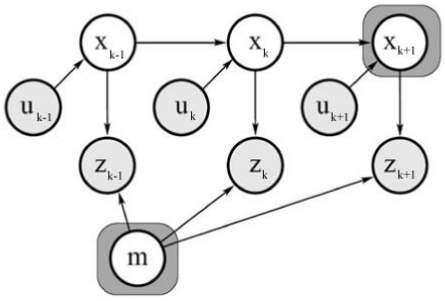



**Figura 39: Cadeia de *Markov* para o processo de transição de estados. Figura retirada de [THR05].**

Uma vez definidos os modelos de transição de estado e de observação, o problema de SLAM é resolvido de maneira incremental, realizando predições (atualização do estado) e correções (inovação de medidas) sobre a localização do robô. Para cada instante de tempo *k* é

computada uma estimativa para a posição do robô. Devido às imprecisões no controle e erros acumulados na posição $\mathbf{x}_{k-1}$ anterior, tal estimativa do estado é errônea e, por isso, é realizado um passo de correção, que faz uso das observações de *landmarks* tomadas pelo robô para corrigir a predição de estado feita anteriormente. De forma matemática, a Equação 18 (atualização de estado) e Equação 19 (inovação das medidas) são executadas para todos os instantes *k* ao longo do tempo.

$$P(\mathbf{x}_k, \mathbf{m}|\mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{x}_0) = \int P(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{u}_k) \times P(\mathbf{x}_{k-1}, \mathbf{m}|\mathbf{Z}_{0:k-1}, \mathbf{U}_{0:k-1}, \mathbf{x}_0)\mathrm{d}\mathbf{x}_{k-1} \quad \textbf{Equação 18}$$

$$P(\mathbf{x}_k, \mathbf{m}|\mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{x}_0) = \frac{P(\mathbf{z}_k|\mathbf{x}_k, \mathbf{m})P(\mathbf{x}_k, \mathbf{m}|\mathbf{Z}_{0:k-1}, \mathbf{U}_{0:k}, \mathbf{x}_0)}{P(\boldsymbol{z}_k|\boldsymbol{Z}_{0:k-1}, \mathbf{U}_{0:k})} \quad \textbf{Equação 19}$$

Note que o problema de construção de mapas pode ser formulado como o cálculo da probabilidade condicional $P(\mathbf{m}|\mathbf{X}_{0:k}, \mathbf{Z}_{0:k}, \mathbf{U}_{0:k})$, pressupondo que a posição do robô $\mathbf{x}_k$ seja conhecida. Logo, um mapa **m** pode ser construído pela fusão de diversas observações de *landmarks* tomadas de diferentes locais do ambiente, de modo que o mapa final seja globalmente consistente. Por outro lado, o problema de localização pode ser formulado como a computação da distribuição de probabilidade $P(\mathbf{x}_\mathrm{k} \mid \mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{m})$, pressupondo que as localizações das *landmarks* (mapa) são conhecidas e, dessa forma, o objetivo é calcular uma estimativa da localização do robô com relação ao mapa dado.

## 4.2 Mapas de ocupação

Mapas de ocupação (*Occupancy grid map*) constituem uma representação volumétrica para ambientes robóticos. Ou seja, um mapa de ocupação representa o mapa do espaço como um conjunto de variáveis, que estão distribuídas uniformemente em um espaço reticulado bidimensional ou tridimensional, sendo cada variável binária correspondente ao estado de ocupação (ocupado ou não ocupado) da posição espacial que ela representa.

Na construção de *grids* bidimensionais, tipicamente a informação (medição ou observação) obtida a partir dos sensores do robô é integrada ao longo do tempo. Dessa forma, a cada nova medição oriunda dos sensores, as posições do mapa (células do *grid*) que estão no campo de visão do sensor são atualizadas. Se a medição informa que a posição é não ocupada, a probabilidade de ocupação da célula do *grid* é decrementada, e, caso contrário, a probabilidade de ocupação da célula do *grid* é decrementada.

## 4.3 Grid-based FastSLAM

Grid-based FastSLAM é um algoritmo de SLAM que utiliza filtro de partículas para estimar o estado da trajetória seguida pelo robô e o estado dos mapas de ocupação do ambiente. Diferentemente dos algoritmos de SLAM baseado em *landmarks*, a representação volumétrica é capaz de modelar tipos de ambientes arbitrários por não dependerem de nenhuma característica (*landmark*) predefinida do ambiente.

Para estender o algoritmo FastSLAM para mapas de *grid*, serão necessárias três condições básicas definidas na Seção 4.1:

- Primeiro, precisamos amostrar uma estimativa da próxima pose $x_t$ por meio da função probabilística $P(\mathbf{x}_t|\mathbf{x}_{t-1}^{[k]}, \mathbf{u}_t)$ que modela o movimento do robô como na Equação 17.

- Segundo, usaremos *occupancy grid mapping* [THR05] como técnica de estimativa do mapa de cada partícula segundo a função probabilística $P(\mathbf{m}^{[k]}|\mathbf{z}_{1:t}, \mathbf{x}_{1:t})$.

- Terceiro, precisamos calcular o peso que mede a importância de cada partícula. Ou seja, precisamos calcular a certeza $P(\mathbf{z}_t|\mathbf{x}_t^k, \mathbf{m}^{[k]})$ da observação $z_t$, dada a pose $x_t$, o mapa $m^{[k]}$, e a medida mais recente do sensor $z_t$.

1: FastSLAM_occupancy_grids($\mathcal{X}_{t-1}$, $u_t$, $z_t$):
2: $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
3: for k=1 to M do
4: $x_t^{[k]}$ = sample_motion_model($u_t$, $x_{t-1}^{[k]}$)
5: $w_t^{[k]}$ = measurement_model_map($z_t$, $x_t^{[k]}$, $m_{t-1}^{[k]}$)
6 $m_t^{[k]}$ = updated_occupancy_grid($z_t$, $x_t^{[k]}$, $m_{t-1}^{[k]}$)
7: $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + < x_t^{[k]}, m_t^{[k]}, w_t^{[k]} >$
8: endfor
9: resample($\bar{\mathcal{X}}_t, \mathcal{X}_t$)
10: return $x_t^{[k]}$ with max $w_t^{[k]}$

**Figura 40: FastSLAM original para mapas de ocupação. Figura retirada de [THR05].**

Como podemos ver na Figura 40, a extensão do FastSLAM para usar mapas de *grid* é bastante direta devido a reutilização de conceitos de Localização baseada em Filtro de Partículas (*Monte Carlo Localization* [THR05]) e Mapeamento (*Occupancy Grid Mapping* [THR05]).

## 4.4 Grid-based FastSLAM (adaptado)

O algoritmo de SLAM implementado é uma extensão do FastSLAM [THR05] para mapas probabilísticos *Occupancy Grid* [ELF89] com filtros de partículas [THR05]. Uma importante diferença foi adotarmos o método de amostragem *Improved Proposal Distribution* empregado no FastSLAM 2.0 [MON03a] para compensar as deficiências do FastSLAM 1.0.

Na Figura 41 a seguir apresentamos o pseudocódigo do algoritmo FastSLAM adaptado.

1: FastSLAM_occupancy_grids_improved_proposal($\mathcal{X}_{t-1}$, $u_t$, $z_t$):
2: $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
3: for k=1 to M do
4: $x_t^{[k]}$ = improved_proposal_distribution($z_t$, $u_t$, $x_{t-1}^{[k]}$, $m_{t-1}^{[k]}$)
5: $w_t^{[k]}$ = measurement_model_map($z_t$, $x_t^{[k]}$, $m_{t-1}^{[k]}$)
6: $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + < x_t^{[k]}, m_t^{[k]}, w_t^{[k]} >$
7: Endfor
8: if $n_{eff}$ < M/2
9: resample($\bar{\mathcal{X}}_t, \mathcal{X}_t$)
10: return $x_t^{[k]}$ with max $w_t^{[k]}$

**Figura 41: Algoritmo FastSLAM com *Improved Proposal* e *Selective Resampling***

### 4.4.1 *Improved proposal distribution*

A principal diferença para o FastSLAM original é a incorporação da medida $\mathbf{z_t}$ no mapa $\mathbf{m_t}$ na amostragem da pose $\mathbf{x_t}$ a partir do modelo de movimento do robô (ver *improved_proposal_distribution* na linha 4 da Figura 41).

```
1:  improved_proposal_distribution(z_t, u_t, x_t-1, m_t-1[k]):
2:  for k=1 to N do
3:          x_t[k] = sample_motion_model(u_t, x_t-1[k])
4:          w_t[k] = measurement_model_map(z_t, x_t[k], m_t-1[k])
5:  endfor
6:  return x_t[k] with max w_t[k]
```

**Figura 42: Algoritmo para *Improved Proposal Distribution***

O modelo de movimento do robô (ver *sample_motion_model* na linha 3 da Figura 41) pode ser de odometria ou velocidade [THR05] para o robô com direção diferencial considerado neste trabalho. Contudo, o FastSLAM que implementamos também suporta veículos autônomos que usam modelo de direção Ackermann [WEI10], o que foge do escopo deste trabalho.

Na Figura 43 fica claro como o algoritmo *Improved Proposal Distribution* descrito na Figura 41 pode melhorar a certeza da pose levando em consideração as informações do sensor e do mapa do ambiente. Na Figura 43(a) é possível ver uma grande incerteza quando o robô se desloca em espaço aberto. No exemplo das Figura 43(b) e Figura 43(c), é possível notar que o espaço de probabilidade para amostragem das poses é moldado pela estrutura física do ambiente. Isto é, as leituras do sensor contendo informações sobre o corredor restringiu a amostragem das poses dadas pelo modelo de movimento.

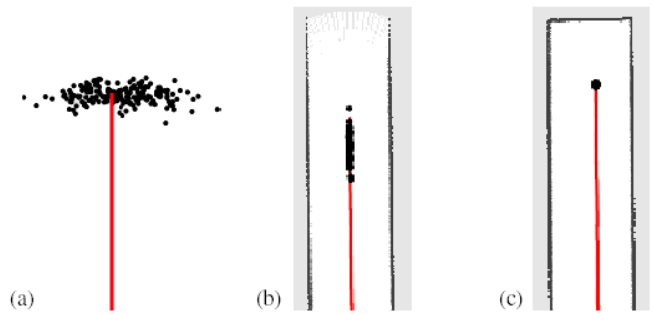



**Figura 43: Exemplos de *Improved Proposal Distribution*. Figura retirada de [GRI05].**

### *4.4.2 Selective resampling*

Um fator que também possui grande influência no desempenho de filtro de partículas é a etapa de *resampling* ou amostragem. Nessa etapa é comum que as partículas com menor peso $w^{[k]}$ sejam substituídas por partículas com maior peso. Poder determinar de forma seletiva (*Selective Resampling*) quando é necessário amostrar as partículas é importante para não descartar sempre as partículas com menor peso, gerando o esgotamento das partículas (*Particle Depletion*) [GRI05]. Isso diminuiria a variância sobre o peso das partículas e, consequentemente, a representatividade de poses do robô no espaço de configurações do mapa para descrever algum novo movimento. Por outro lado, amostrar as partículas é importante para garantir a convergência e estabilização do SLAM de filtro de partículas.

$$n_{eff} = \frac{1}{\sum_k \left(w_t^{[k]}\right)^2} \qquad \textbf{Equação 20}$$

Para resolver esse dilema (*trade-off*) em algoritmos de SLAM com filtro de partículas, é calculado o número efetivo de partículas $\mathbf{n_{eff}}$ apresentado na Equação 20 [GRI05], que representa uma medida de quão boa é a aproximação da distribuição de partículas. Ou seja, este número efetivo de partículas descreve a variância do peso das partículas, que assume valor máximo quando os pesos das partículas são iguais e, assim, se aproximam da distribuição verdadeira (*Proposal Distribution).*

**Tabela 3:** ***Resample particles***

```
1: resample(X̄_t, X_t)
2: for k=1 to M do
3:         draw i with probability proportional to w_t^[i]
4:         add < x_t^[k], m_t^[k] > to X_t
5: Endfor
```

Essa melhoria foi incorporada em nossa implementação do FastSLAM, como pode ser visto na linha 8 da Figura 41, onde $\mathbf{n_{eff}}$ é calculado de acordo com a Equação 20, de forma que, ao cair abaixo de um limite (*threshold*) M/2 do número de partículas, amostramos novamente as partículas seguindo a mesma abordagem descrita em [GRI05]. Esse limite garante um equilíbrio das partículas com maior e menor peso ao longo das iterações do SLAM.

# 5 NAVEGAÇÃO ROBÓTICA PROBABILÍSTICA

O problema de navegação de veículos autônomos em ambientes desconhecidos tem recebido muita atenção da comunidade científica, especialmente nos últimos anos com eventos tais como a DARPA *Grand Challenge* [BUE05] e a DARPA *Urban Challenge* [BUE08a, BUE08b]. Neste problema, o robô deve ser capaz de realizar um planejamento prévio de suas ações com o objetivo de navegar por posições pré-definidas num mapa, conhecidas na literatura como *waypoints*.

Neste capítulo apresentamos um algoritmo de navegação por *waypoints* baseado em programação dinâmica [KON00] e outro com sistema anticolisão, que se complementam para permitir ao robô navegar e evitar obstáculos de forma segura. Tais algoritmos permitem que veículos tomem decisões automaticamente e determinem mudanças de comportamento para alcançarem algum objetivo estabelecido [LAU98]. Estes algoritmos estão disponíveis no *Toolkit* de Navegação Robótica CARMEN [MON03b].

## 5.1 Navegação com objetivo definido

No trabalho descrito por Konolige em [KON00], o espaço de configurações em que o robô está representado (em vermelho) é discretizado em um mapa de *grid* estático semelhante ao da Figura 44. Assumir que o mapa é estático simplifica o desenvolvimento de algoritmos na área de robótica móvel, pois não é necessário verificar e modelar objetos se deslocando ou que não estão presentes no mapa.

**Figura 44: Mapa do asilo de Longwood em Oakmont Pittsburgh, disponível no ambiente de simulação do CARMEN.**

Foi feita uma adaptação no módulo de navegação do CARMEN para suportar mapas de *grid* atualizados em tempo real pelo algoritmo de SLAM apresentado anteriormente. Isso por que a versão original considerava que o mapa do ambiente era previamente conhecido e imutável, o que não é possível na maioria dos ambientes. No cenário atual, o mapa aumenta à medida que o robô se desloca para regiões antes desconhecidas. Isso implica em constantes replanejamentos de caminho para incorporar as novas atualizações no mapa de regiões livres (para o robô trafegar) e de regiões obstruídas (para o robô desviar).

Konolige, em seu trabalho [KON00], se utiliza de uma técnica que consiste em aumentar os obstáculos do mapa de *r* unidades, onde *r* é o raio do robô, para simplificar os cálculos de planejamento do caminho ao considerar o robô como um ponto no espaço, como explicaremos a seguir.

### 5.1.1 Inicialização do mapa de custos

Na etapa de inicialização do algoritmo, cada posição discreta do mapa é representada como uma célula do *grid* no espaço bidimensional e recebe um valor de custo, que é função de sua ocupação: posições ocupadas possuem um custo elevado, enquanto que posições livres (trafegáveis) recebem um custo muito baixo ou nulo.

A modelagem de uma função de custo fornece pistas para o planejamento, pois se a função objetivo é minimizar o custo de um caminho, o planejamento tenderá trafegar por regiões de baixo custo ao invés de trafegar por regiões de custo elevado. Com esta modelagem, por exemplo, um caminho direto entre o robô e o alvo que envolveria atravessar uma parede ou outro obstáculo não deve ser selecionado, pois o custo de outro caminho que contorna o obstáculo tende a ser menor.

### 5.1.2 Construção da função de navegação

Uma vez definido o próximo objetivo ou *waypoint* a ser alcançado pelo robô, é desejável encontrar um caminho entre a posição atual do robô e o objetivo estabelecido. Um caminho pode ser representado como uma sequência de pontos do espaço de configurações discretizado, conforme Equação 21.

$$\mathrm{P} = \{P_1, P_2, \ldots, P_k\}, \qquad \textbf{Equação 21}$$

onde a sequência de pontos é contínua, não existem pontos repetidos e o último ponto $P_k$ da sequência deve ser a posição discreta do objetivo no mapa de *grid*.

Em geral, o custo de um caminho pode ser modelado matematicamente como sendo a soma do custo de seus pontos, adicionado da soma do custo das arestas ligando os pontos consecutivos, conforme Equação 22.

$$custo(P) = \sum_{i=1}^{|P|} custo(P_i) + \sum_{i=1}^{|P|-1} custo(P_i, P_{i+1}), \qquad \textbf{Equação 22}$$

Dessa forma, o custo das células ocupadas será propagado para as células vizinhas, como pode ser percebido pelo efeito degradê do mapa de custos da Figura 45, gerado pelo algoritmo em questão.

**Figura 45: Mapa de custos gerado pelo planejador do CARMEN**

Konolige, em seu trabalho [KON00], define uma função de navegação como sendo a atribuição de um valor de utilidade para cada posição do espaço de configurações, de forma tal que a posição do objetivo é alcançável facilmente a partir de qualquer localidade do *grid*. Esta função de navegação calcula o valor de utilidade $V(P_{i+1})$ em cada célula $P_{i+1}$ baseado nos valores de utilidade e no custo das células vizinhas, conforme Equação 23.

$$V(P_{i+1}) = V(P_i) - custo(P_i, P_{i+1}) \cdot custo(P_{i+1}), \qquad \textbf{Equação 23}$$

onde o $custo(P_{i+1})$ foi calculado na fase de inicialização do mapa de custos e as células $P_i$ e $P_{i+1}$ são células vizinhas. Em um mapa de *grid* bidimensional, cada célula possui exatamente oito células vizinhas. O custo da aresta entre células vizinhas $P_i$ e $P_{i+1}$ depende da direção da vizinhança (horizontal, vertical ou diagonal), e pode ser definido de acordo com a Equação 24.

$$custo(P_i, P_{i+1}) = \begin{cases} \sqrt{2}, & se\ P_i e\ P_{i+1}\ são\ vizinhos\ na\ diagonal \\ 1, & caso\ contrário \end{cases} \qquad \textbf{Equação 24}$$

O ponto de objetivo possui valor máximo de utilidade e é também o ponto de partida, sendo a primeira célula a ser adicionada a uma *lista de pontos ativos*. A cada iteração, as células que estão na *lista de pontos ativos* são removidas da lista e suas células vizinhas têm seu valor de utilidade calculado e são adicionadas a essa lista. Uma célula vizinha é adicionada na *lista de células ativas* para ter sua função de navegação calculada apenas se o

valor de utilidade da célula atual é superior ao custo do caminho até a célula vizinha. Essa heurística de seleção de células para cálculo da função de navegação na prática faz com que as regiões de custo elevado (regiões não trafegáveis) não recebam valores de utilidade (seus valores de utilidade permanecem nulos).

### 5.1.3 Construção do caminho até o objetivo

O caminho do robô ao objetivo é extraído com base nos valores de utilidade calculados pela função de navegação. A célula do *grid* que contém o objetivo possui valor de utilidade máximo e as células vizinhas vão recebendo valores de utilidade gradativamente menores à medida que se afastam do objetivo. Isso por que os valores de utilidades são subtraídos da função não negativa de custos conforme a Equação 23. Sendo assim, o gradiente da função de navegação em cada célula representa a direção de máximo ganho local, como pode ser visto no efeito degrade da Figura 46.

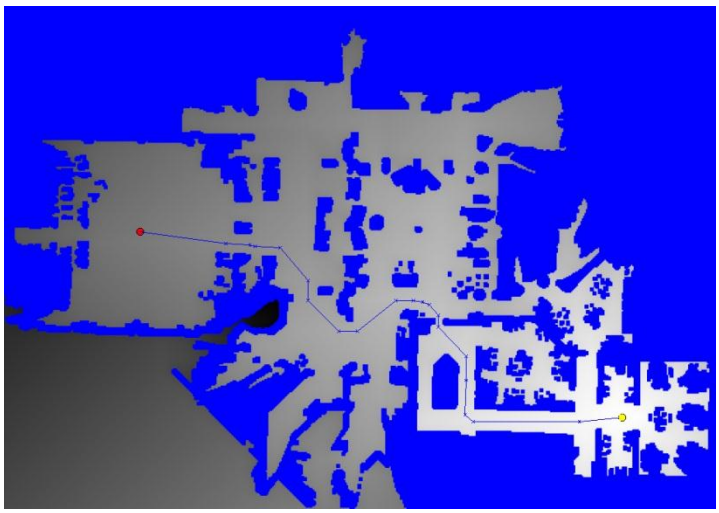


**Figura 46: Mapa de navegação gerado pelo planejador do CARMEN**

Uma vez que o cálculo da função de navegação foi realizado a partir do ponto de objetivo, com os vizinhos sendo adicionados iterativamente, é possível construir um caminho a partir de qualquer localidade $P_i$ que possui um valor de utilidade $V(P_i) > 0$ até o objetivo. Em particular, se a posição atual do robô $P_r$ possui um valor de utilidade $V(P_r) > 0$ associado, é possível encontrar um caminho até o alvo caminhando sempre na direção do gradiente.

### 5.1.4 Análise de complexidade assintótica

Sendo ***m*** e ***n***, respectivamente, as dimensões horizontal e vertical do mapa de *grid* bidimensional que representa o espaço de configurações em que o robô está localizado, o algoritmo proposto por Konolige em [KON00] possui complexidade de tempo ***O(m.n)*** no pior caso, referente ao tempo para se calcular a função de navegação para as células do mapa. O pior caso envolve a necessidade de realizar o cálculo da função de navegação para todas as ***m.n*** células do mapa. A complexidade de tempo do algoritmo é diretamente proporcional à quantidade de região trafegável do mapa, pois quanto menor o número de obstáculos, mais células vizinhas são adicionadas à *lista de pontos ativos* durante a construção da função de navegação para terem seus valores de utilidade calculados.

Se considerarmos que a adição da posição atual do robô à *lista de pontos ativos* é um ótimo critério de parada para o algoritmo de construção da função de navegação, os cenários de maior complexidade de tempo se apresentam quando o robô está fisicamente distante do objetivo ou quando qualquer caminho curto entre o robô e o objetivo envolve um custo elevado (por exemplo, com a existência de obstáculos ao longo deste caminho). Nesses cenários, tipicamente, a função de navegação será calculada para mais regiões do espaço de configurações.

A complexidade de espaço do algoritmo também é da ordem de ***O(n.m)*** no pior caso, dado que, na implementação deste algoritmo no módulo de navegação do CARMEN, a função de navegação aloca memória suficiente para tratar o pior caso, por questões de simplificação.

## 5.2 Navegação com sistema anticolisão

Há diferentes abordagens para prevenção de colisão que compreendem desde o desvio de obstáculos (*obstacle avoidance*) [FOX97] por sistemas de SLAM e planejamento de caminho de alto nível até paradas emergenciais detectadas por sistemas anticolisão de baixo nível, como sonar, *bumper* e laser. As duas formas estão implementadas de forma bem semelhantes no CARMEN, mas em níveis de atuação diferentes: a de baixo nível age no módulo base do

robô para freá-lo caso detecte uma colisão com o sensor *bumper* ou quando a velocidade instantânea o levará a uma potencial colisão com os obstáculos presentes na leitura instantânea do sensor laser; já a de alto nível atualiza o mapa de *grid* com os dados do sensor laser para incorporar os novos obstáculos no planejamento e assim traçar uma rota para desviar deles. Originalmente, a atualização do mapa de *grid* com dados instantâneos de sensores foi necessária para navegar em mapas de *grid* estáticos com localização apenas.

Contudo, essa abordagem de atualização de mapas de *grid* estáticos pode tornar o mapa inconsistente em casos que surjam obstáculos momentâneos e, principalmente, por não haver um tratamento probabilístico considerando a incerteza na leitura de dados do sensor, como é feito na formulação do SLAM apresentado na seção 4 anterior. Portanto, um dos objetivos deste trabalho foi integrar a navegação com o SLAM para minimizar o risco de acidentes.

De acordo com as experiências sobre navegação descritas em [MON03b], desmembrar o problema de navegação em um planejador de alto nível e um sistema anticolisão baixo nível pode conduzir a falhas de navegação em situações em que o sistema anticolisão tente atuar para desviar de obstáculos grandes arbitrários fugindo da rota planejada. Por isso, a estratégia implementada no CARMEN para integrar o planejamento de caminho com sistemas de desvios de obstáculos propicia maior segurança, especialmente em situações em que obstáculos não mapeados entram em rota de colisão com o robô.

# 6 EXPLORAÇÃO ROBÓTICA PROBABILÍSTICA

Exploração robótica, segundo Thrun [THR05], refere-se ao problema de controlar um robô de forma a maximizar seu conhecimento a respeito do mundo externo. Podem existir necessidades de exploração robótica diversas, desde ambientes inóspitos e inacessíveis aos seres humanos até situações em que sejam economicamente inviáveis o envio de pessoas para coleta de informações e o seu regresso em segurança.

Apresentaremos a seguir um algoritmo de exploração baseado em processos de decisão de Markov (*Markov Decision Process* - MDP). Esse algoritmo é capaz de planejar rotas de exploração suficientes para adquirir o máximo de informação a respeito do ambiente sob a forma de mapas de *grid*, seja ele fechado (*indoor*) ou aberto (*outdoor*).

## 6.1 MDP *vs.* POMDP

No paradigma probabilístico de planejamento e controle robótico, geralmente, são considerados dois tipos de incerteza: uma na percepção dos dados de sensores e a outra na seleção das ações a serem tomadas pelo robô para atingir um propósito. Isso significa que as ações do robô podem ter efeitos imprevisíveis no ambiente e também gerar incerteza sobre sua percepção do ambiente e sua localização.

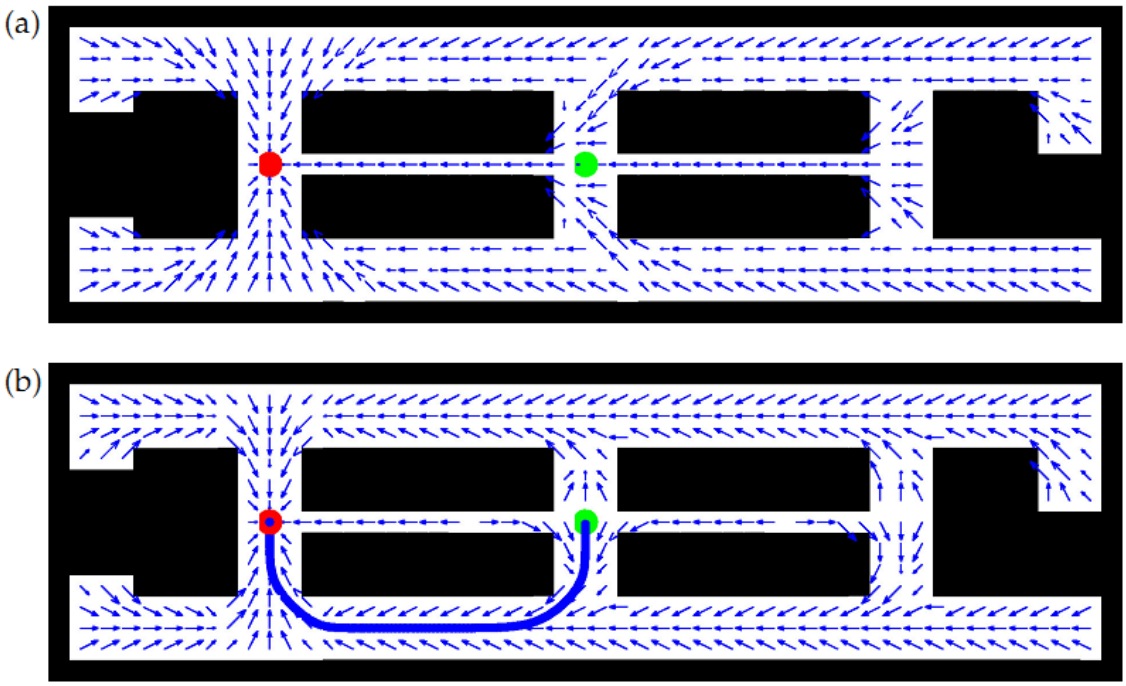



**Figura 47: planejamento e controle em um cenário com efeito (a) determinístico e (b) não determinístico das ações. No modelo determinístico é perfeitamente admissível o robô navegar pelo corredor estreito, mas no caso em que as incertezas são levadas em conta o planejamento prefere desviar do corredor estreito para reduzir o risco de colidir com uma parede. Figura retirada de [THR05].**

No paradigma clássico, as incertezas não são levadas em consideração. Assume-se que as ações quando executadas no mundo físico possuem resultados previsíveis, os quais poderiam ser previamente planejados. Como exemplo, na Figura 47(a) o robô (círculo verde) poderia, simplesmente conhecendo sua posição inicial e a localização do alvo (círculo vermelho), fazer o planejamento *off-line* de uma única sequência de ações (setas azuis) para depois executá-las sem a necessidade de sensoriar o ambiente.

Nesse contexto, um paradigma que trata a incerteza do movimento do robô é denominado como *Markov Decision Processes*, ou MDP. MDP assume que o estado do ambiente pode ser completamente sensoriado (observado) em todos os instantes. Em outras palavras, o modelo de percepção $p(z|x)$ é determinístico e bijetivo. Contudo, MDP aceita os efeitos de ações estocásticas. O modelo de movimento $p(x'|u,x)$ pode ser não determinístico. Como consequência, não é suficiente planejar uma única sequência de ações apenas. Ao contrário, o planejador precisa gerar ações para um conjunto completo de situações possíveis que o robô pode se encontrar devido à imprevisibilidade das ações e da dinâmica do ambiente. Uma

forma de lidar com essa incerteza é criar uma **política de seleção de ações** definida para todos os estados que o robô pode assumir.

O caso mais geral, em que o estado é parcialmente observável, é conhecido como *Partially Observable Markov Decision Processes*, ou POMDP. Nesse caso, além das ações não serem determinísticas, o estado do robô e do mundo a sua volta não são completamente observáveis.

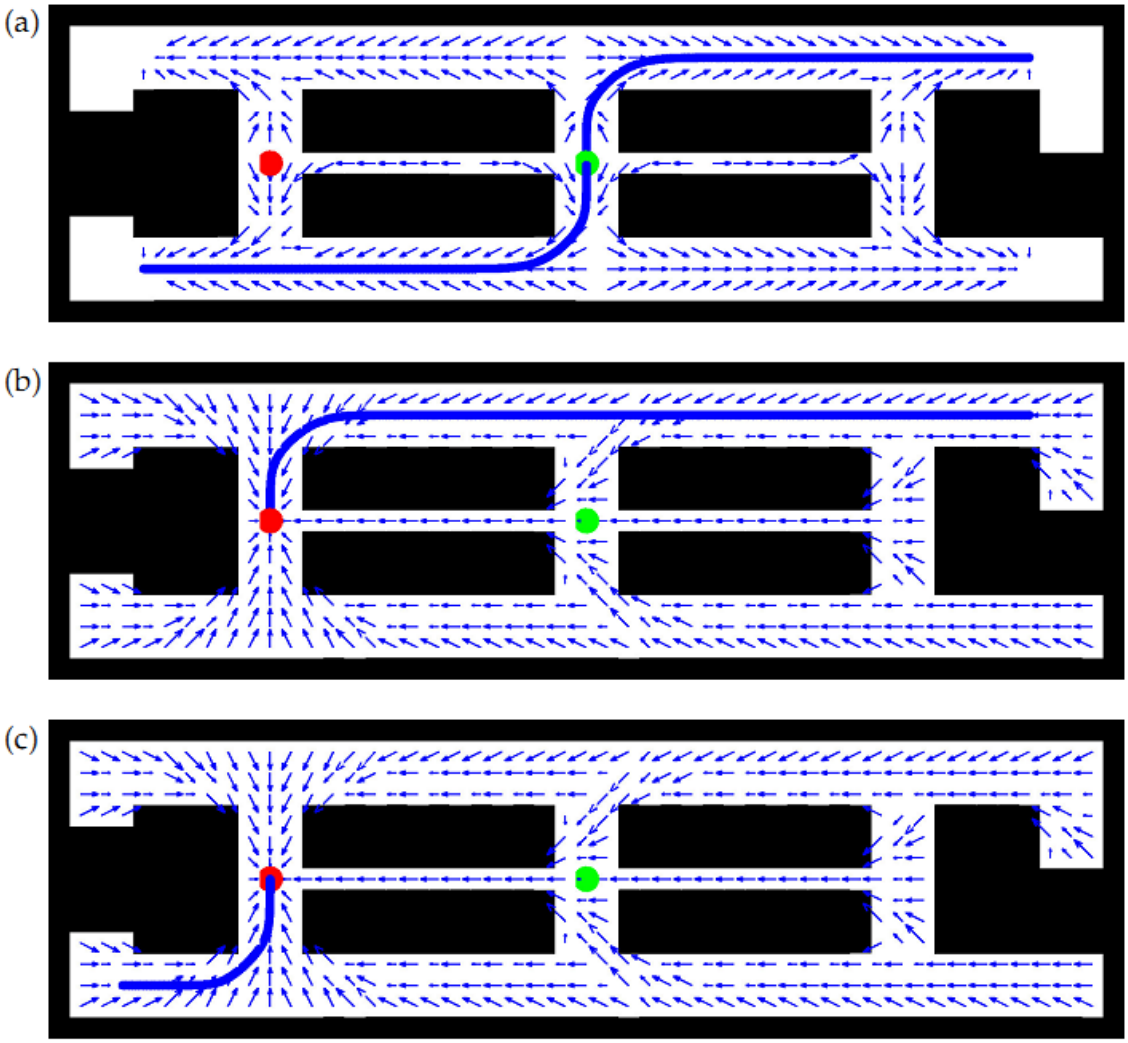



**Figura 48: Ações para aquisição de informação em POMDP. Na situação (a) é apresentada uma política e os caminhos possíveis que o robô pode tomar para remover a ambiguidade sobre sua orientação. Conhecendo sua localização (b) ou (c) o robô poderá navegar em segurança até o objetivo. Figura retirada de [THR05].**

Voltando ao exemplo anterior, mostrado sobre nova perspectiva na Figura 48, assuma agora que o robô sabe sua posição inicial (x, y), mas não sabe se está virado para esquerda ou

direita, ou seja, há incerteza sobre sua orientação Θ. Além disso, ele não possui um sensor para detectar se chegou até o destino (círculo vermelho).Claramente, a simetria do ambiente torna mais difícil tirar a ambiguidade sobre a orientação do robô. Numa primeira tentativa, o robô poderia se mover em linha reta com chance de 50% de errar o alvo e chegar ao lado oposto (direito) em que se encontra o alvo (esquerdo). O plano ótimo com chances superiores a 50% é se mover para qualquer um dos cantos superior ou inferior, esquerdos e direitos do ambiente, que são distintos o suficiente para tirar a ambiguidade sobre a orientação do robô. A política para se mover para esses cantos do ambiente pode ser vista na Figura 48(a). Com base na sua orientação inicial (virado para cima ou para baixo), o robô pode tomar qualquer um dos dois caminhos apresentados na Figura 48(a). Assim que ele chegar a um dos cantos (superior direito ou inferior esquerdo), seus sensores informarão sua orientação e, então, sua posição relativa no ambiente. A partir desse momento, o robô pode estar em uma das duas situações mostradas nas Figura 48(b) e Figura 48(c), de onde ele pode seguramente navegar até o local destino, indicado pelo círculo vermelho.

## 6.2 Planejamento MDP para exploração

Propomos, neste trabalho, uma estratégia de planejamento de rotas para exploração baseada em MDP que, diferente da apresentada por Thrun em [THR05], utiliza apenas uma fração aleatória do espaço de estados para **política de seleção das ações**. Para isso, empregamos uma distribuição uniforme de tamanho M sobre o espaço de configurações desocupadas do mapa de *grid*. Apenas estas M posições aleatórias servirão como entrada para um algoritmo de otimização inspirado no *Value Iteration* [THR05] para planejamento MDP.

### *6.2.1 Value iteration*

Nosso algoritmo de planejamento MDP foi desenvolvido com uma **política de seleção de ações** para beneficiar a exploração, ou seja, com o objetivo de maximizar a informação adquirida a respeito do ambiente sob a forma de probabilidade associada que as células do mapa de *grid*. Regiões desconhecidas ou não exploradas não possuem probabilidade

associada (-1), e as demais regiões desde livre a ocupadas são representadas numa escala crescente de probabilidade entre [0;1].

A Figura 49 apresenta um método para seleção de ações conhecido como *value iteration*. Este método calcula recursivamente um valor de utilidade para cada ação relativa a uma função de recompensa (*Payoff*). A otimização por *value iteration* aproxima iterativamente funções de valor ótimo $V_\infty(x)$ definidas na forma da Equação 25.

$$V_\infty(x) = \gamma \max_u \left[ r(x,u) + \int V_\infty(x') p(x'|u,x) dx' \right] \qquad \textbf{Equação 25}$$

Em geral, nesses tipos de algoritmos de otimização, a escolha de uma ação é orientada por objetivos (*Goals*). Na otimização em questão, estamos interessados em selecionar uma quantidade mínima de ações que levarão ao objetivo pretendido com o menor custo. Isto é, desejamos minimizar a distância percorrida e maximizar o ganho na escolha de ações que levarão a uma região desconhecida (objetivo) com o menor número de manobras.

Conforme provado em [THR05], a otimização por *value iteration* converge para horizontes de tempo T (iterações) infinito com fator de desconto $\gamma<1$ e em casos de horizonte de tempo finito T até para $\gamma=1$. O efeito de aplicar fatores de desconto $0<\gamma<1$ ($\gamma$ pode assumir valores no intervalo [0;1]) é equivalente a dizer que as primeiras recompensas possuem maior importância que as últimas devido ao seu comportamento exponencial. Em nosso problema, assumimos horizonte de tempo finito e, portanto, podemos aplicar $\gamma=1$ e omiti-lo da fórmula.

1: *for i=1 to M do*
2: $\hat{V}(x_i)=r_{min}$
3: *endfor*
4: *repeat until convergence*
5: *for i=1 to M do*
6: $$\hat{V}(x_i) = \gamma \max_u \left[ r(x_i,u) + \sum_{j=1}^{M} \hat{V}(x_j) p(x_j|u,x_i) \right]$$
7: *Endfor*
8: *endrepeat*
9: *return V*

**Figura 49: *MDP_discrete_value_iteration***

Embora pareçam grandezas opostas, é possível unificá-las numa única função, chamada função de recompensa (*payoff function*), para simplificar a formulação e, principalmente,

balancear os objetivos contrapostos de minimização de custo e de maximização das chances de alcançar o alvo. A recompensa, denotada ***r*** na Equação 26, é uma função do estado $\boldsymbol{x}$ e do controle $\boldsymbol{u}$ do robô. Por exemplo, nossa função de recompensa para exploração em ambientes fechados é a seguinte:

$$r(x,u) = \begin{cases} +100, & se\ o\ comando\ u\ leva\ ao\ objetivo \\ -100, & caso\ contrário \end{cases} \qquad \textbf{Equação 26}$$

Nossa função de recompensa premia o robô com +100, se a ação leva a uma área não explorada vizinha, e penaliza o robô com -100, para cada passo do tempo em que a ação não levar a uma área desconhecida. Tal função de recompensa retornará um valor acumulativo máximo se o robô atingir o objetivo em menos manobras. Na linha 2 do algoritmo da Figura 49, $r_{\text{min}}$ representa o valor mínimo imediato de recompensa.

Neste trabalho, assumimos a probabilidade *p* de um comando $u$ levar o robô da pose anterior $x$ a próxima pose $x'$, ou seja, $p(x' \mid u, x) = 1$.

### 6.2.2 Máquina de estados

Como visto na Seção 6.1, o planejamento com base em MDP é, normalmente, executado *off-line* e depois utilizado como guia para navegação em busca de um objetivo. Um objetivo, ou alvo, em nosso caso de exploração, corresponde a uma configuração do mapa de *grid* que ainda não foi tocada pelo sensor e é vizinha a uma área trafegável. Como pretendemos utilizar o planejamento MDP num contexto de exploração em que o mapa muda ao longo do tempo, é necessário utilizar um mecanismo de controle de estados que permita realizarmos uma exploração global por meio de ciclos de planejamentos MDP *off-line*. Para tanto, definimos a máquina de estados da Figura 50 abaixo.

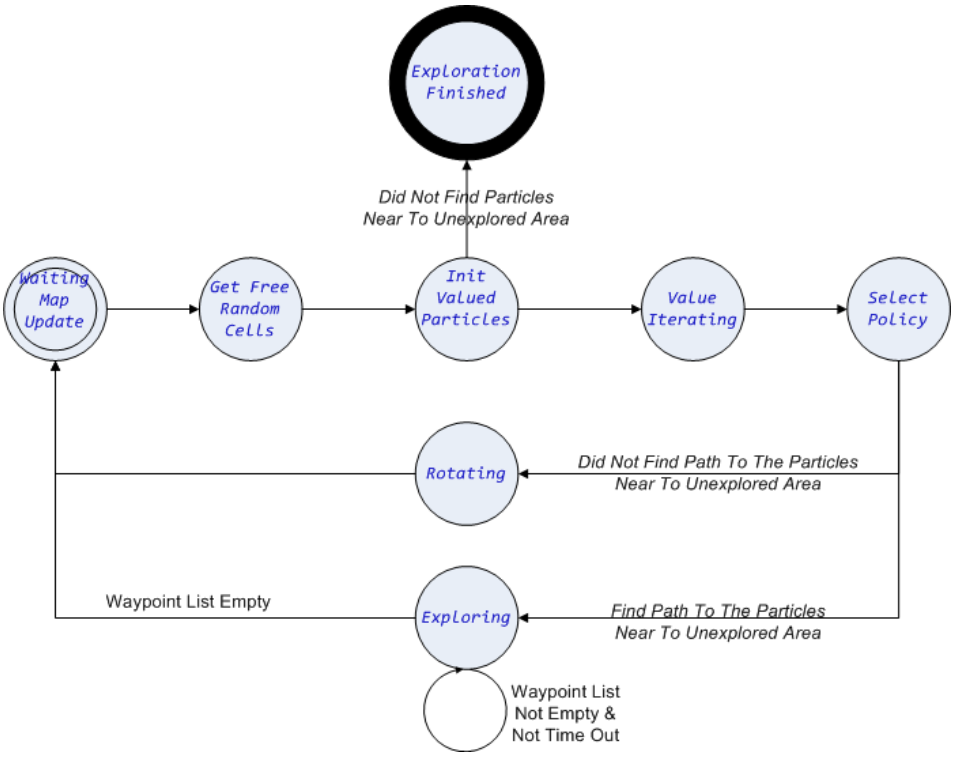



**Figura 50: Máquina de estados[1] de exploração**

Essa máquina de estados funciona em ciclos de planejamentos MDP que levam a regiões ainda não exploradas. Dessa forma, o robô deverá navegar com base no planejamento de cada ciclo até atingir o objetivo de exploração do ciclo. Contudo, a exploração pode revelar regiões antes desconhecidas que, depois de tocadas pelo sensor, se transformam em obstáculos e impeçam o alcance de um objetivo. Nesse caso, a máquina de estados monitora uma variável de *time out* para se reiniciar.

O funcionamento da máquina de estados possui em seu estado inicial uma etapa de atualização do mapa de *grid* (*Waiting Map Update*), que é utilizado no estado seguinte (*Get Free Random Cells*) para selecionar aleatoriamente M regiões livres dentre todas as trafegáveis do mapa de *grid*. Adiante (*Init Valued Particles*), as regiões selecionadas são inicializadas conforme a função de recompensa $r_{\min}$ da Equação 26, executada nos três

[1] A máquina de estados da figura está em Inglês para manter semelhança com o padrão de nomenclatura de código fonte produzido no LCAD.

primeiros passos do algoritmo descrito na Figura 49. Caso não se encontre regiões livres dentre as selecionadas aleatoriamente próximas a regiões inexploradas, a máquina de estados termina. Caso contrário, ela avança (*Value Iterating*) para o algoritmo de otimização *Value Iteration* descrito na Figura 49 (a partir da linha 4). No estado seguinte (*Select Policy*), o algoritmo da Figura 51 é chamado para construir o caminho a partir das ações que levarão a uma região desconhecida. Caso não haja um caminho, passa-se ao estado (*Rotating*) onde são gerados comandos de rotação (assumindo direção diferencial) para expandir o mapa com uma visão 360º. No caso esperado, onde há um caminho do robô até uma região não explorada, o robô entra no estado de exploração (*Exploring*), de onde só sai se alcançar todos os *waypoints* planejados para o caminho ou se atingir uma condição de espera limite (*time out*) para descartar caminhos obstruídos, mesmo temporariamente. Em qualquer uma dessas condições, a exploração retorna ao seu estado inicial (*Waiting Map Update*) para iniciar um novo ciclo com um mapa mais recente. Foi verificado, experimentalmente, que esses ciclos se repetem até se esgotarem quase por completo as regiões desconhecidas do ambiente a ser explorado.

$$1: \quad return \operatorname*{argmax}_{u} \left[ r(x,u) + \sum_{j=1}^{M} \hat{V}(x_j) p(x_j | u, x_i) \right]$$

**Figura 51: *policy_MDP(𝒳, V)***

# 7 EXPLORAÇÃO ROBÓTICA POR BUSCA VISUAL DE PONTOS DE INTERESSE

Nosso sistema de navegação robótica por imagens de pontos de interesse foi implementado por meio de uma máquina de estados que agrega todos os subsistemas descritos nos capítulos anteriores:

I. SLAM: fornece o mapa e a pose atualizados em tempo real;

II. Planejamento MDP para exploração: define a rota de exploração numa estratégia gulosa;

III. Navegação com objetivo definido: conduz o robô em segurança pela rota de exploração e também o leva até os pontos de interesse do mundo calculados no item (v);

IV. Sistema de busca visual: encontra as correspondências dos pontos mais salientes da imagem de referência na imagem recebida da câmera;

V. Controle de vergência: encontra correspondência entre pontos bidimensionais (2D) do par de imagens estéreo, observando a geometria epipolar das câmeras estéreo. A partir dos pares de pontos correspondentes entre a imagem esquerda e direita da câmera estéreo, é calculada a coordenada tridimensional (3D) do ponto de interesse no mundo.

## 7.1 Pontos de interesse

Os pontos de interesse (*features*) em imagens tratados neste trabalho são pontos bidimensionais (2D) identificados em imagens a partir de algoritmos bastante conhecidos em processamento de imagens, como SURF [BAY08] e FAST [ROS06]. A primeira detecta regiões (*blobs*) e a segunda detecta quinas (*corners*). Os pontos de interesse no mundo são

pontos tridimensionais (3D), que neste trabalho são calculados utilizando visão estereoscópica [OLI05].

A partir desses pontos de interesse, é possível definirmos **imagens de pontos de interesse no mundo** representativos de uma determinada cena do caminho que sejam capazes de identificá-lo. Nesse contexto, podemos classificar essas **imagens de pontos de interesse no mundo** como estáticas ou dinâmicas. Por exemplo, para **imagens estáticas de pontos de interesse no mundo**, considere uma sequência de imagens capturadas previamente de um caminho a ser seguido, em que cada imagem possui pontos de interesse do mundo específicos daquela cena. O objetivo é, então, explorar o ambiente à procura dos **pontos de interesse (estáticos) no mundo,** pertencentes à cena capturada previamente. Num contexto mais dinâmico, temos como objetivo encontrar e perseguir um determinado objeto na cena, que a princípio seria único. O objeto alvo presente na cena é que se move no ambiente e determina o **ponto de interesse (dinâmico)**. Por exemplo, no caso de um veículo autônomo que tivesse o objetivo de seguir um carro líder, o carro líder poderia ser escolhido como **ponto de interesse (dinâmico)**. Porém, o estudo de **pontos de interesse (dinâmicos)** não faz parte do escopo deste trabalho, e sim a busca por uma sequência não contígua de imagens que contém **pontos de interesse (estáticos)**.

## 7.2 Máquina de estados

A máquina de estados apresentada na Figura 52 descreve a evolução da exploração robótica em busca das imagens de pontos de interesse (**imagem do caminho**), respeitando a sequência de imagens não contígua fornecida do caminho.

O sistema inicia (INITIALIZATION) a máquina de estado e fica aguardando a pose inicial do robô. Ao recebê-la, a mesma é armazenada como o endereço de origem (HOME) do robô e também como a localização conhecida no mapa mais recente (LAST KNOWN POSE).

O sistema carrega a primeira **imagem do caminho** (GET NEXT REFERENCE IMAGE) e encontra (EXTRACT FEATURES FROM REFERENCE IMAGE) os pontos mais salientes (*features*) nessa imagem usando algoritmos conhecidos de detecção de pontos de interesse

(FAST *features*) em imagem [ROS06]. As *features* são ordenadas de forma decrescente por um critério de relevância próprio do detector e selecionadas de forma a manter um afastamento mínimo entre si, além de descartar aquelas muito próximas da borda da imagem para evitar oclusão de *features* na imagem estéreo. Caso seja encontrado um número mínimo parametrizado (ex. cinco) de *features*, a imagem possui pontos de interesse suficientes para identificar a cena, o que permite que o robô inicie a busca por ela (INIT SEARCH FOR REFERENCE IMAGE). Caso contrário, é informada a falha na detecção de *features* (ERROR EXTRACTING FEATURES) na imagem atual de pontos de interesse e a próxima **imagem do caminho** (GET NEXT REFERENCE IMAGE) é carregada.

Ao iniciar a busca (INIT SEARCH FOR REFERENCE IMAGE), para reconhecimento da **imagem do caminho,** o sistema armazena a pose do robô e o par de imagens estereoscópicas nesse instante. O sistema então verifica se achou (CHECK VISUAL WAYPOINT QUANTITY) uma quantidade mínima parametrizada (ex. três) de correspondências (encontradas pelo sistema de busca visual) entre os pontos de interesse procurados na **imagem do caminho** e aqueles encontrados na imagem direita do par estéreo corrente. Caso contrário, o sistema avançará para o estado (LEARN VISUAL WAYPOINT) de aprendizado da próxima *feature* da **imagem do caminho**, que é passada para a busca visual aprendê-la e tentar encontrá-la na imagem direita do par estéreo. Em seguida, repete-se a verificação de quantidade mínima de correspondência até que esta seja atingida ou o tempo de busca se esgote (ERROR SEARCHING TIMEOUT).

Quando a quantidade mínima de correspondências for atingida, deve-se proceder à verificação geométrica (CHECK VISUAL WAYPOINT GEOMETRY) sobre os pontos correspondentes para assegurar que a cena reconhecida está numa visada próxima da procurada. A verificação da geometria é feita avaliando se os pontos correspondentes respeitam uma relação de ordem vertical e horizontal entre si. Ou seja, dados três pontos $p_1$, $p_2$ e $p_3$ no espaço bidimensional da **imagem do caminho** e seus correspondentes $r_1$, $r_2$ e $r_3$ na imagem direita do par estéreo, $p_1 \prec p_2 \prec p_3 \Rightarrow r_1 \prec r_2 \prec r_3$. Além da verificação geométrica, o sistema verifica também se a confidência média dos pontos correspondentes retornados pela busca visual está acima de um limiar parametrizado (ex. 90%).

Se não reconheceu a cena procurada, o sistema manda o robô iniciar a exploração (GO SEARCH FOR REFERENCE IMAGE) por regiões desconhecidas e reinicia a busca (INIT SEARCH FOR REFERENCE IMAGE) para reconhecimento da **imagem do caminho**. Caso contrário, se reconhecer, o sistema manda o robô parar a exploração e inicia os preparativos para navegação com objetivo definido (GO NAVIGATE TO VISUAL WAYPOINT). Nesse momento, são selecionadas as duas melhores correspondências para cálculo dos seus respectivos pontos tridimensionais pelo sistema de vergência a partir do par de imagens estéreo. Esses pontos são transformados do sistema de coordenadas da câmera equivalente ao apresentado na Figura 25 (Seção 2.6.1) para o sistema de coordenadas do robô e então para o sistema de coordenadas do mapa (2D). Apenas o ponto mais próximo será utilizado como *waypoint* para navegação.

O sistema então define o objetivo (*waypoint*) no mapa e muda para o estado de navegação (NAVIGATING TO VISUAL WAYPOINT). Enquanto o robô se desloca até o objetivo definido, o sistema verifica continuamente:

- Se o robô se aproximou a uma distância mínima parametrizada (ex. 1,5m) do *waypoint* para poder avançar em busca da próxima imagem do caminho (GET NEXT REFERENCE IMAGE) e atualizar sua localização conhecida no mapa mais recentemente;

- Se os pontos de interesse da imagem do caminho continuam visíveis (busca visual) na imagem direita do par estéreo, de forma que seja possível recalcular suas coordenadas usando a vergência e realizar as transformações necessárias para alcançá-lo. Esse rastreamento (*tracking*) é importante durante a navegação para reforçar a certeza sobre o reconhecimento do local para o qual se está navegando.

Quando o tempo de busca se esgota e não foi reconhecida a **imagem do caminho** dentre outras da lista que se deve achar, o sistema manda o robô parar a exploração e retornar rapidamente para a última localização conhecida no mapa (GO BACK LAST KNOWN POSE). O sistema verifica, continuamente, se o robô chegou até a última localização conhecida do mapa (GOING BACK LAST KNOWN POSE) e reinicia o SLAM a partir dessa

posição para permitir que a exploração volte a visitar locais possivelmente obstruídos da vez anterior. Nesse momento, o sistema retoma a busca pela mesma **imagem do caminho** (EXTRACT FEATURES FROM REFERENCE IMAGE).

Ao passar e reconhecer todas as imagens do caminho, o robô retorna ao local de origem (GO BACK HOME) e, chegando lá (GOING BACK HOME), o sistema finaliza (QUIT) com sucesso. Alguns estados previstos para tratamentos em caso de falhas foram omitidos para tornar a explicação mais clara.

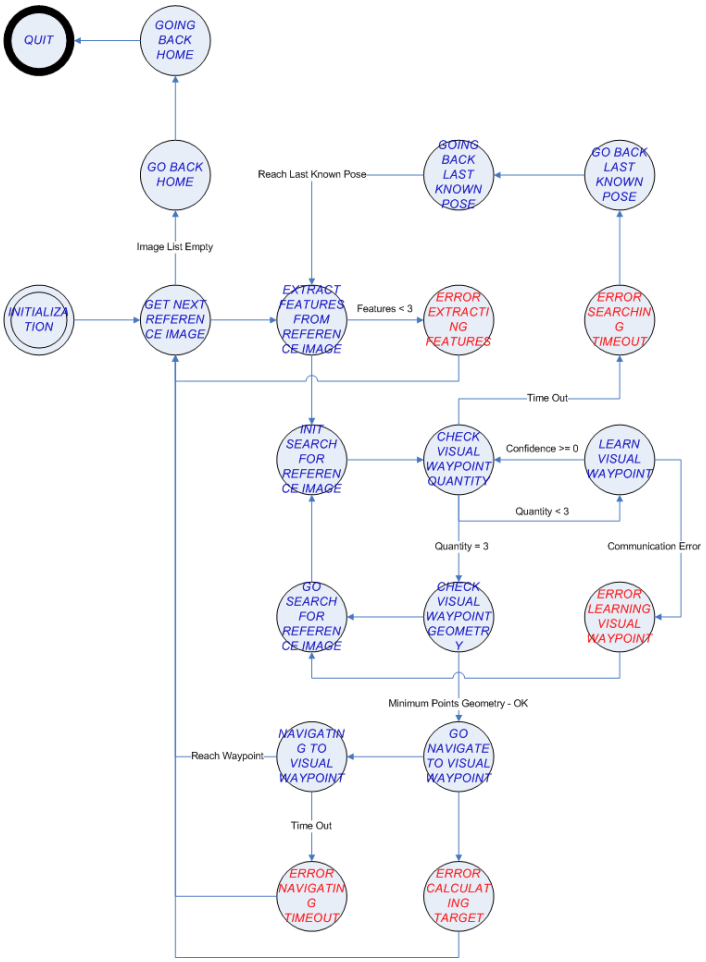



**Figura 52: Máquina de estados[2] da exploração visual**

[2] A máquina de estados da figura está em Inglês para manter semelhança com o padrão de nomenclatura de código fonte produzido no LCAD.

# 8 METODOLOGIA

Neste capítulo será apresentada a metodologia utilizada na implementação da arquitetura proposta neste trabalho de pesquisa. Também serão apresentadas as ferramentas utilizadas e a plataforma robótica empregada neste trabalho.

## 8.1 Framework de inteligência artificial MAE

O *framework* MAE (Máquina Associadora de Eventos) possui código fonte aberto e foi desenvolvido pelo Grupo de Pesquisa em Ciência da Cognição do LCAD-UFES [OLI05]. Com este, pode-se projetar tanto estruturas modulares usando RNSP com neurônios do tipo VG-RAM [KOM02] quanto estruturas com arquitetura em camadas, com a definição de um processamento específico para cada camada, a exemplo de [OLI05].

### 8.1.1 Características do *framework* MAE

Sua utilização para a implementação da arquitetura neural de busca visual, proposta em [NET12], e vergência, proposta neste trabalho, se deveu às facilidades fornecidas, tais como:

- Facilidade de Programação: a MAE permite a descrição de arquiteturas neurais e/ou camadas através de uma linguagem de alto nível, com geração automática do código que as implementam.
- Flexibilidade: na linguagem de descrição de arquiteturas, vários parâmetros podem ser alterados. Esta ferramenta também permite a incorporação de código do usuário.

- Facilidade visual: toda camada descrita pela linguagem utilizada pela MAE pode ter uma representação visual, o que torna possível o acompanhamento da resposta temporal de cada camada.
- Portabilidade: a MAE é implementada utilizando uma linguagem padrão, o C ANSI, possibilitando que seja portada para qualquer sistema que possua um compilador C padrão. Os pacotes e bibliotecas importadas pelo *framework* são pacotes portáteis como Mesa3D, Flex, Bison, entre outros.

### 8.1.2 Visão geral do framework MAE

O sistema MAE é composto, basicamente, por duas partes. A primeira é um programa chamado **netcomp** que recebe como entrada um arquivo com extensão .con contendo a descrição da arquitetura (neural ou de camadas), e gera como saída um arquivo com extensão .c, contendo código C correspondente à tradução do arquivo .con para C. A segunda é a biblioteca de rotinas MAE, que implementa os neurônios, seu treinamento, processamento específico para cada camada, interface com o usuário, etc.

O **netcomp** é um tradutor da linguagem de descrição de arquitetura descrita no arquivo .con para C. O arquivo .c gerado e os arquivos de rotinas do usuário são compilados e lincados com outras bibliotecas necessárias por um *script* chamado **netcompiler**. É este *script* que gera o arquivo executável final, ou aplicação MAE, que implementa a arquitetura descrita no arquivo .con, além de uma interface que permite ao usuário manipular a arquitetura. Na verdade, para gerar uma aplicação MAE, este processo é transparente ao usuário (Figura 53). O **netcompiler** se encarrega de realizar todo o processo automaticamente, gerando como saída uma aplicação MAE. Maiores detalhes podem ser encontrados em [OLI05].

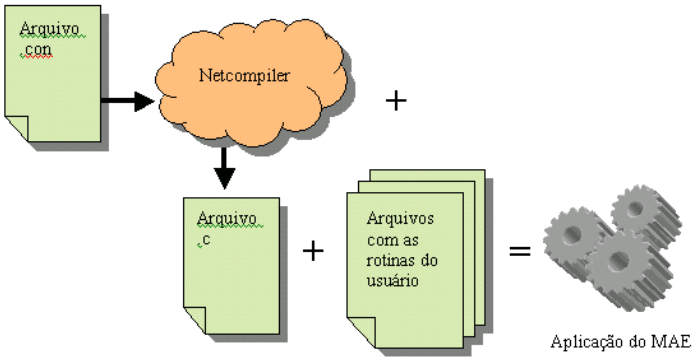



**Figura 53: Esquema de criação de uma aplicação utilizando o framework MAE.**

## 8.2 Framework de robótica CARMEN

Carmen *Robot Navigation Toolkit* ou CARMEN [MON03b] é uma coleção de software aberto GPL (http://carmen.sourceforge.net), desenvolvido na *Carnegie Mellon University* (CMU), para controle de robôs móveis. A plataforma CARMEN permite abstrair grande parte dos detalhes de implementação de um sistema de robótica que incorpora sensores, algoritmos de planejamento, navegação e controle. CARMEN foi projetado sobre uma arquitetura modular, orientada a serviços (SOA - *Service Oriented Architecture*) e fornece nativamente primitivas básicas de navegação para robôs, como: controle de sensores e robôs, persistência de mensagens, detecção e desvio de obstáculos, localização, planejamento de caminho e mapeamento.

### 8.2.1 Características do *framework* CARMEN

A plataforma CARMEN foi adotada nesse trabalho principalmente por suas características de programação se apoiar em três princípios: extensibilidade, robustez e baixa curva de aprendizado [MON03b]. Um programa, ou módulo, desenvolvido utilizando o *framework* CARMEN, deve ser isolado de todos os outros programas do sistema, comunicando-se apenas através de mensagens. Isso gera um baixo acoplamento entre os módulos e minimiza interferências no funcionamento de outros módulos aumentando a robustez do sistema como um todo. Mesmo na eventualidade de uma falha, um módulo não é

capaz de causar uma falha geral em todo o sistema. Assim, o *framework* garante a modularidade, simplicidade dos módulos, tolerância a falhas e a eficácia da aplicação como um todo.

### 8.2.2 Visão geral do framework CARMEN

A comunicação entre módulos no CARMEN é viabilizada por meio de um *middleware* chamado *Inter Process Communication* (IPC) (http://www.cs.cmu.edu/~ipc/), criado em 1994 e mantido até hoje pelo Professor Reid Simmons de *Carnegie Mellon University* (CMU). O IPC fornece uma forma flexível e eficaz de se trocar mensagens entre processos baseada nos paradigmas de comunicação *"Publish-Subscribe"* e "*Request-Reply*" [HOH03], que permite enviar e receber estruturas de dados complexas, incluindo listas e vetores de tamanho variável. Além da capacidade de conexão de alto nível entre processos, a biblioteca do IPC oferece também facilidades para definição (*Register*), serialização (*Marshall*), envio e recebimento de mensagens, abstraindo os detalhes de comunicação via *sockets* sobre o protocolo TCP/IP. Na Figura 54 pode-se observar a comunicação de três módulos via IPC.

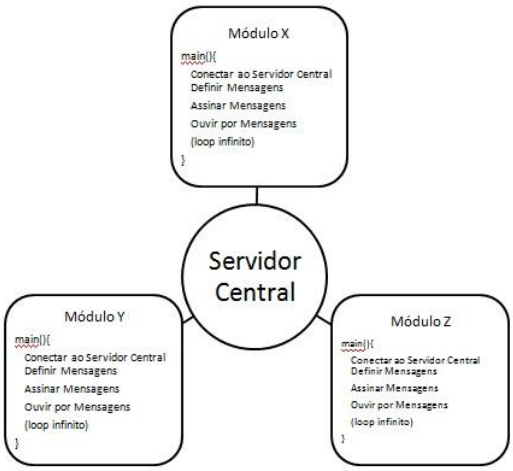



**Figura 54 - Comunicação dos Módulos do Carmen Toolkit utilizando IPC.**

Um sistema desenvolvido no CARMEN consiste em um servidor de aplicativos central independente, conectado a um número de processos específicos (módulos). O servidor central (programa *central* do IPC) é um repositório de informação de todo o sistema responsável pelo

roteamento das mensagens. Preferencialmente, é utilizado no CARMEN o modelo "*Publish-Subscribe*" da Figura 55, onde os módulos indicam o seu interesse em receber certos tipos de mensagens que outros módulos possam publicar.

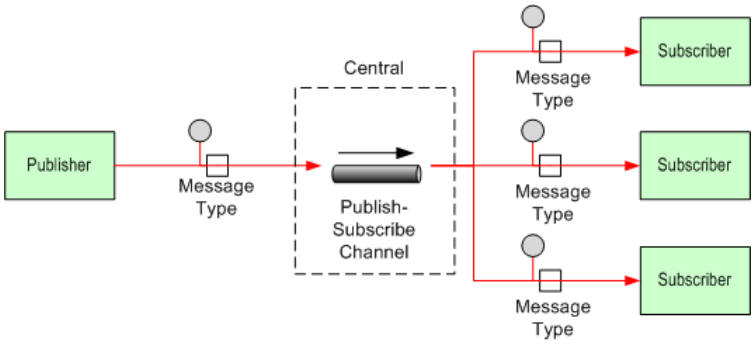



**Figura 55: Modelo de comunicação *Publish-Subscribe* [HOH03]**

Nesse modelo, a recepção de mensagens é assíncrona, ou seja, cada assinante (*subscriber*) possui uma função de *callback*, também conhecida como *handler*, que é invocada para cada instância do tipo de mensagem recebida. Os módulos publicadores (*publishers*) devem se conectar ao processo *central* do IPC, definir suas mensagens, publicá-las, e, eventualmente, ouvir por mensagens que assinaram.

Contudo, há casos em que a comunicação precisa ocorrer em duas vias, como na Figura 56, onde um requisitante (*requestor*) envia uma mensagem de requisição e aguarda por uma mensagem de resposta, e o responsável (*replier*) por atender a mensagem de requisição recebe a mensagem de requisição e retorna uma mensagem de resposta. Nesse modelo de comunicação, o canal de requisição pode ser ponto-a-ponto ou via *publish-subscribe*. A diferença está em enviar a mensagem de requisição para todos os módulos interessados ou somente para um único consumidor, no caso da comunicação ponto-a-ponto. O canal de resposta, por outro lado, normalmente funciona ponto-a-ponto, por que as mensagens de resposta fazem sentido apenas ao requisitante. No IPC o canal de requisição emprega o modelo *publish-subscribe*, que permite a outros módulos receberem a mensagem de requisição, mesmo não sendo responsáveis por respondê-las.

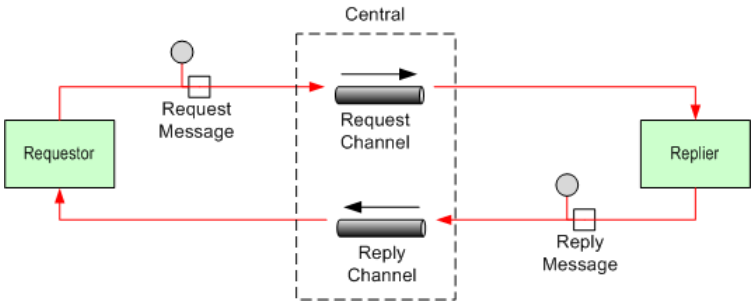



**Figura 56: Modelo de comunicação *Request-Reply* [HOH03]**

## 8.3 Arquitetura do sistema

Na arquitetura proposta neste trabalho, os módulos CARMEN foram desenvolvidos seguindo os princípios e boas práticas de programação com os quais o CARMEN foi projetado [MON03b]. Dentre eles, a separação da visualização e do controle dos dados, conhecido como MVC (*Model-View-Controller*), que podem ser identificados na legenda da Figura 57 por *driver*, GUI, e *filter*, respectivamente.

Módulos denominados *driver* foram classificados no grupo dos componentes que se comunicam diretamente com algum tipo de plataforma de hardware ou sensor, enquanto que os módulos de filtros (*filter*) foram agrupados na classe dos que recebem medidas de sensores e realizam algum tipo de computação sobre elas. Esse último gera novos dados que serão utilizados por outros filtros ou que serão apresentados em uma interface de visualização (GUI) para o usuário final.

Os principais componentes representativos da arquitetura deste trabalho são listados abaixo:

1. **Central:** programa *central* do IPC. Responsável por manter as informações, rotear e fazer o registro do tráfego de mensagens do sistema. Antes de executar qualquer módulo, esse programa deve estar executando.
2. **Parameter Server:** módulo *param_daemon*, nativo do CARMEN. Responsável por armazenar e distribuir os parâmetros lidos de um arquivo de inicialização composto por variáveis de diversos módulos. Os módulos

solicitam via mensagens do tipo *Request-Reply* os valores dos parâmetros que precisam e ainda recebem atualizações dos seus valores sob demanda via *Publish-Subscribe*.

3. **Bumblebee:** módulo *bumblebee_basic*, desenvolvido no LCAD. Responsável por capturar imagens de câmeras *Bumblebee* 2 ou XB3 da fabricante Point Grey. Faz a captura das imagens estéreo (retificadas ou não) através da interface *Firewire* e as publica como mensagens.
4. **Bumblebee GUI:** módulo *bumblebee_basic_view*, desenvolvido no LCAD. Responsável por exibir as imagens esquerda e direita (par estéreo) lado-a-lado na taxa que são publicadas.
5. **Laser:** módulo *laser*, nativo do CARMEN. Responsável pela comunicação e captura de dados de *Laser Range Scans* (LRS), suporta diversos modelos de LRS, entre eles o LMS 200 da fabricante SICK.
6. **Pioneer:** módulo *pioneer*, nativo do CARMEN. Responsável pela comunicação e controle do robô Pioneer 3-DX da fabricante MobileRobots. Aceita comandos de velocidade e fornece dados de odometria.
7. **Robot**: módulo *robot*, nativo do CARMEN. Representa uma abstração do hardware para diferentes robôs, responsável pela fusão de dados de laser com odometria, envio de comandos de movimento para o *pioneer*, além de evitar colisões (*collision avoidance*).
8. **SLAM**: módulo *SLAM*, desenvolvido no LCAD. Trata-se de uma implementação do FastSLAM para mapas de grid que é responsável pelo cálculo e envio em tempo real de poses do robô e mapas contendo informações de obstáculos, regiões livres e desconhecidas.
9. **Explorer**: módulo *explorer*, desenvolvido no LCAD. Capaz de planejar caminhos a partir de mapas de grid e poses que levarão a regiões desconhecidas (não exploradas) realizando o menor número de manobras (comandos de odometria).

10. **Navigator**: módulo *navigator*, nativo do CARMEN. O navegador utiliza um planejador de caminho mais curto (*shortest path*) para encontrar um caminho entre a pose atual do robô até um objetivo definido. Se não existir um caminho direto, o navegador encontrará um caminho para o ponto mais próximo que o robô pode alcançar. O caminho é assumido como sendo um conjunto de pontos no caminho (*waypoints*) conectados por segmentos unidos de reta. Uma vez que um caminho foi calculado, e o navegador esteja no modo autônomo (mensagem *go*), um controlador PD (Proporcional-Derivativo) será acionado para conduzir o robô entre um ponto de passagem (*waypoint*) e o próximo até atingir o objetivo.
11. **Navigator GUI**: módulo *navigator_gui*, nativo do CARMEN. Este módulo, assim como o anterior, foi adaptado para receber mapas de grid enviados em tempo real pelo módulo de SLAM e exibi-los juntamente com a pose do robô em cada instante. Através dessa interface gráfica o usuário pode definir a posição inicial do robô no mapa e também as posições no mapa definidas como objetivos para navegação. Outra funcionalidade importante disponível é a de colocar o robô em modo autônomo ou de pará-lo. Há outras funcionalidades de visualização muito interessantes para investigação do problema de navegação, como mapa de custo, utilidade, projeção de sensores e outras.
12. **Vergence**: módulo *vergence*, desenvolvido no LCAD utilizando o *framework* MAE. Este módulo é capaz de encontrar um ponto correspondente na imagem esquerda a partir de um ponto de interesse (x, y) na imagem direita. O resultado da vergência é a disparidade (no eixo x da imagem) entre o ponto de interesse e o ponto correspondente. Para isso este módulo deve receber uma mensagem contendo o par de imagens estereoscópicas e um ponto de interesse na imagem direita e então retornar uma mensagem com a disparidade encontrada. Essa comunicação é

viabilizada através do modelo de troca mensagens *Request-Reply*, descrito na Figura 57 pelas etapas de treino B1-B2 e teste B3-B4.

13. **Visual Search**: módulo *visual_search*, desenvolvido no LCAD utilizando o *framework* MAE. Este módulo é responsável por realizar buscas visuais por pontos de interesse em uma imagem tendo como entrada outra imagem e um ponto de interesse. Sendo assim, este módulo recebe uma mensagem de requisição contendo duas imagens e um ponto de interesse definido em apenas uma delas. E retorna numa mensagem de resposta a coordenada (x, y) do ponto de interesse na outra imagem e a certeza associada à resposta. Essa comunicação é viabilizada através do modelo de troca mensagens *Request-Reply*, descrito na Figura 57 pelas etapas de treino A1-A2 e teste A3-A4.
14. **Visual Planning**: módulo *visual_planning*, desenvolvido no LCAD. Este é o módulo supervisor responsável pela orquestração dos demais módulos, feita por meio da máquina de estados apresentada na Figura 52, cujo propósito é explorar o ambiente em busca dos locais representados por uma sequência de fotos apenas e depois retornar ao local de partida. As principais trocas de mensagens utilizadas para controlar o robô e receber informações do ambiente através dos sensores estão descritas na Figura 57.

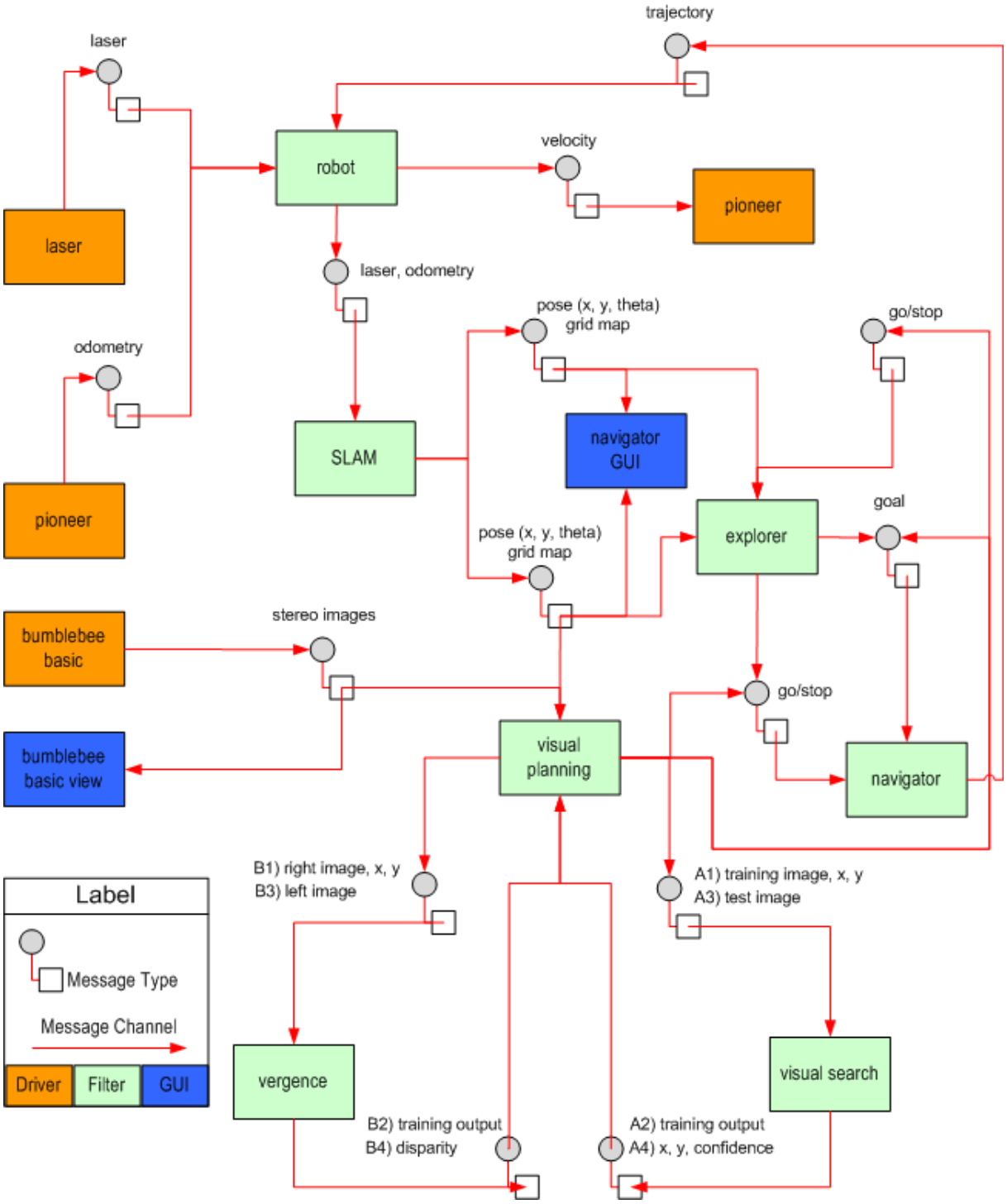



**Figura 57: Arquitetura simplificada do sistema**
**(as comunicações com o *Parameter Server* e o *Central* foram omitidas para simplificação)**

## 8.4 Hardware do sistema

Para que fosse possível realizar os experimentos desse trabalho no ambiente físico, empregamos um robô Pioneer 3-DX, um sensor LRS LMS-200 da SICK e uma câmera estereoscópica Bumblebee XB3 da PointGrey. Além de um notebook DELL Precision M4400 para execução do sistema e interface com os sensores e o robô.

### 8.4.1 Plataforma

A seguir apresentaremos a plataforma utilizada: um robô Pioneer P3-DX e um notebook.

#### 8.4.1.1 Pioneer P3-DX

O robô Pioneer P3-DX, Figura 58, é entregue montado numa estrutura de alumínio, com rodas de 19cm de diâmetro, motores em cada roda com *encoder* de 500 pulsos por volta, 8 sensores frontais ultrasônico (sonar), três baterias 9A hot-swap. O robô emprega modelo de direção diferencial, suporta até 23kg e pode alcançar velocidade de até 1,6m/s.

**Figura 58: Pioneer P3-DX**

O controlador de movimento embutido do robô efetua o controle de velocidade do robô e fornece o estado do robô e informações de controle, incluindo uma estimativa de posição absoluta no mundo (x, y, theta), dados de carga da bateria e dados de varredura do sonar.

#### 8.4.1.2 Notebook Dell Precision M4400

Foi utilizado um microcomputador portátil com processador Intel Core 2 Duo T9800 (2.93GHz, 1066MHz, 6M L2 Cache), memória DDR2 800MHz de 4GB, placa gráfica NVIDIA Quadro FX770M de 512MB, disco rígido SATA com sensor de queda livre de

250GB (7200 RPM), placa de rede sem fio Dell 1397 (802.11 b/g), Firewire 1394a, tela LCD Wide Screen WXGA de 15.4 polegadas.

## 8.4.2 Sensores

Apresentaremos, a seguir, os sensores utilizados neste trabalho de pesquisa.

### 8.4.2.1 SICK LMS-200

O sistema LMS opera através da medição do tempo de vôo de pulsos de luz laser, ou seja, um feixe de luz laser pulsada emitido é refletido se encontra um objeto. A reflexão é registrada pelo receptor do *scanner*. O tempo entre a transmissão e a recepção do impulso é diretamente proporcional à distância entre o *scanner* e o objeto (tempo de voo).

O LMS-200 possui interface de comunicação serial RS 232 com taxa de transmissão de 115000 *baud*, e frequência de operação de 75Hz. O peso é 1,2kg.


**Figura 59: SICK LMS-200**

O LMS-200 suporta campos de visão (*Field of View* – FoV) de 100° e 180° e resolução angular de 0,25°, 0,5° ou 1°, conforme Figura 60.

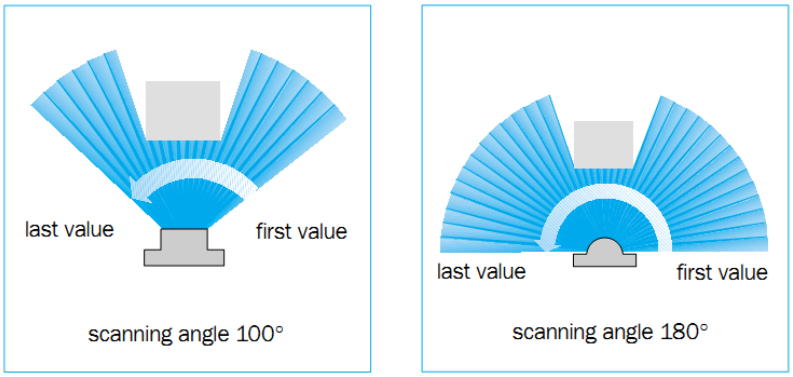



**Figura 60: FOV do SICK LMS-200**

### 8.4.2.2 Bumblebee XB3

A Bumblebee® XB3 é uma câmera estéreo 2 em 1 (*multi-baseline*) projetada para maior flexibilidade e precisão. Suporta taxa de transferência de 800Mb/s usando interface padrão IEEE-1394b, mas trabalha apenas a 16FPS (*Frames Per Second*) . Possui 3 sensores CCD de 1,3 mega-pixel arranjados de forma a oferecer duas opções de *baseline* para processamento estéreo. O *baseline* alargado e a alta resolução (1280x960) proporcionam mais precisão em distâncias mais longas, enquanto o *baseline* estreito melhora a precisão de perto e o alcance mínimo. Peso 505g. 12V.

**Figura 61: Bumblebee XB3**

# 9 EXPERIMENTOS E RESULTADOS

Neste capítulo apresentamos os experimentos realizados para validar a contribuição dos principais componentes da arquitetura do sistema de navegação discutida na Seção 8.3.

## 9.1 Avaliação do sistema de SLAM

Para o robô poder executar qualquer ação num ambiente inicialmente desconhecido, é necessário que ele seja capaz de construir, em tempo real, uma representação fidedigna do ambiente no qual ele está inserido e, simultaneamente, determinar com precisão qual a sua localização nessa representação. A representação escolhida nesse trabalho é a de um mapa de grid formado por células quadriculadas que armazenam informações discretizadas do ambiente projetadas no plano do chão. Essa discretização, definida pelo tamanho de cada célula, determina a precisão com a qual saberemos a localização do robô e dos obstáculos à sua volta. No caso de ambientes fechados, os obstáculos mais comuns são paredes e móveis que podem ser facilmente detectados com um sensor de varredura laser de uma linha apenas instalado numa altura adequada em relação aos obstáculos de interesse. Sendo assim, basta projetarmos os obstáculos sobre o plano do chão para criar uma representação bidimensional (2D) do mundo, semelhante a uma planta baixa de construção civil.

Considerando a representação adotada em nossa implementação do FastSLAM para mapas em formato de *grid*, dividido em células com resolução mínima do mapa de 5cm e dimensões quadradas (da área a ser mapeada) da ordem de centenas de metros, facilmente se esgotaria a memória principal de um computador pessoal disponível atualmente no mercado. Chegamos a duas configurações viáveis na Tabela 4 para a realização dos experimentos de exploração, levando em consideração o balanceamento entre os recursos disponíveis, a precisão desejada e o tamanho da área a ser mapeada.

**Tabela 4: parâmetros de mapeamento**

| *Dimensão* | *Resolução* | *Células* | *Partículas de mapas* |
|---|---|---|---|
| 50m x 50m | 0,05m | 50.000 | 50 |
| 100m x 100m | 0,10m | 100.000 | 50 |

Outro importante ajuste de parâmetros diz respeito aos parâmetros que descrevem o modelo de movimento do robô, que pode ser baseado no modelo de odometria ou de velocidade para o caso do nosso robô com direção diferencial [THR05]. O modelo de odometria possui parâmetros relacionados a rotação e translação, enquanto o modelo de velocidade possui parâmetros relacionados a velocidade angular e linear. Os melhores resultados que alcançamos foram utilizando o modelo de odometria, porque os comandos de odometria são suportados nativamente pelo controle embarcado do robô Pioneer 3-DX. Já no modelo de velocidade, as velocidades angular e linear são estimadas a cada instante pelo filtro *robot* e constituem uma aproximação que torna mais desafiadora a estimativa de parâmetros para o modelo de movimento do robô.

Outro desafio enfrentado por algoritmos de mapeamento é a capacidade de fazer fechamento de *loop*, ou seja, reconstruir as conexões entre caminhos circulares como a planta do 1º. andar do CT-7 da UFES em formato de oito exemplificado na Figura 62. Os mapas mostrados na Figura 58 foram construídos a partir do ambiente simulado do CARMEN usando FastSLAM com *resampling* seletivo. Para um exemplo real de fechamento de *loop*, disponibilizamos no canal do LCAD (http://www.youtube.com/user/lcadufes) no Youtube um vídeo com o nome *Grid-based FastSLAM (loop closure)* mostrando o robô Pioneer 3-DX fazendo SLAM no 2º. Andar do CT-7 da UFES.

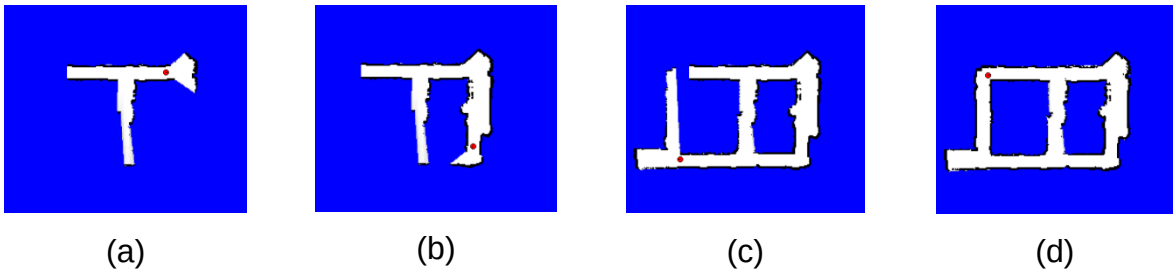



**Figura 62: Fechamento de loop no mapa simulado do CT7 (1º. Andar) usando FastSLAM com *resampling***

O fato de empregarmos *resampling* seletivo não garante sozinho o fechamento de loop, mas contribui fortemente para a preservação de partículas suficientemente variadas (quando $N_{eff}$ < M/2) que possam explicar os movimentos que levam ao fechamento do mapa. No momento do fechamento de loop, por exemplo, há comumente partículas alinhadas com cada um dos seus mapas e outras não. As partículas alinhadas corretamente possuem maior peso que as incorretamente alinhadas, provocando um aumento na variação dos pesos e fazendo com que o número efetivo $N_{eff}$ caia abaixo de um *threshold* de partículas definido (M/2). Quando isso ocorre o *resampling* tende sortear as partículas mais pesadas e, portanto, as que possivelmente fecharão o *loop* [GRI05].

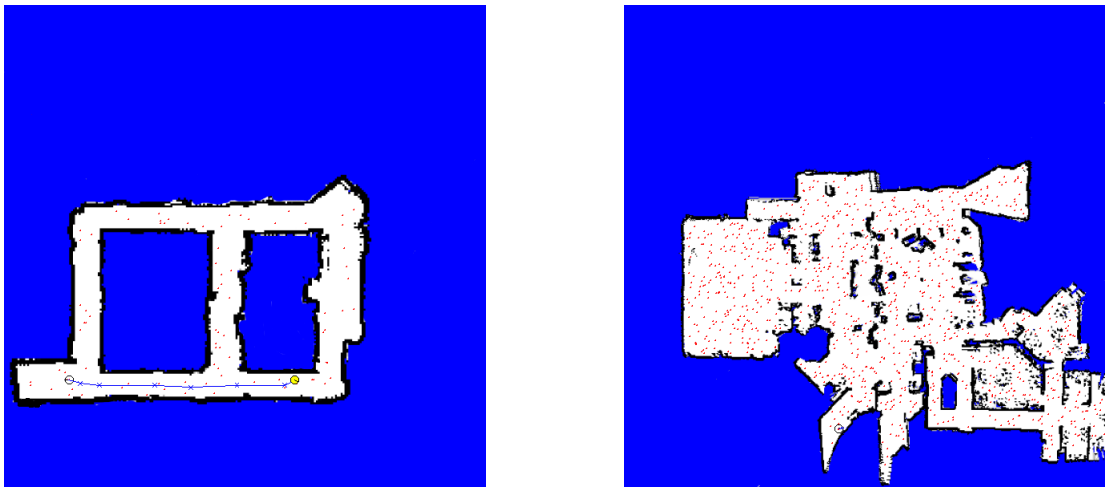
## 9.2 Avaliação do sistema de exploração em *grid*

O sistema de planejamento MDP foi projetado com uma política gulosa de exploração. Portanto, o planejamento deve trabalhar em conjunto com um sistema de mapeamento, no nosso caso SLAM, de forma a expandir as regiões exploradas no mapa de *grid*. Contudo, essa expansão do mapa abre novas células do *grid* a serem consideradas no planejamento dos caminhos possíveis para exploração, o que torna o problema intratável devido à explosão combinatória de políticas a serem geradas. Uma forma de tratar o problema em tempo polinomial é utilizar algoritmos de planejamento aproximativos [THR05], como o que propomos no Capítulo 6.

Nosso algoritmo seleciona aleatoriamente, dentre as células vazias do *grid*, uma quantidade fixa (parametrizada) de posições por onde o robô pode trafegar em busca de

regiões desconhecidas. Sendo assim, o dimensionamento desse parâmetro influencia na abrangência da exploração. Quanto maior a região a ser explorada e mais repleta de obstáculos, maior deverá ser a quantidade de células sorteadas aleatoriamente.

Para mapas semelhantes ao da Figura 63 (a), com poucos obstáculos e dimensões moderadas, 100 partículas são suficientes para explorá-lo completamente. No entanto, mapas maiores contendo muitos obstáculos requerem que o robô realize mais manobras por distâncias maiores para alcançar regiões ainda desconhecidas. Verificamos empiricamente que mapas com essas características necessitam de milhares de partículas para convergir. Por exemplo, um primeiro experimento com o mapa da Figura 63 (b), 500 partículas não foram suficientes para a exploração convergir após 30 min. No entanto, dobrando a quantidade de partículas foi suficiente para garantir uma exploração completa nesse mapa em menos de 30 min.



(a) (b)

**Figura 63: Mapas de *grid* do CT-7 da UFES (a) e longwood de CARMEN (b)**

A seguir é apresentada uma sequência de mapas na Figura 64 (a) até (i) mostrando a evolução da exploração em um mapa complexo de dimensões 50m x 50m com resolução de 10cm, utilizando 1000 partículas. Na Figura 64 (e) e (f) é possível ver um exemplo de planejamento (linha vermelha) complexo de manobras, contornando obstáculos para levar a uma região inexplorada ainda.

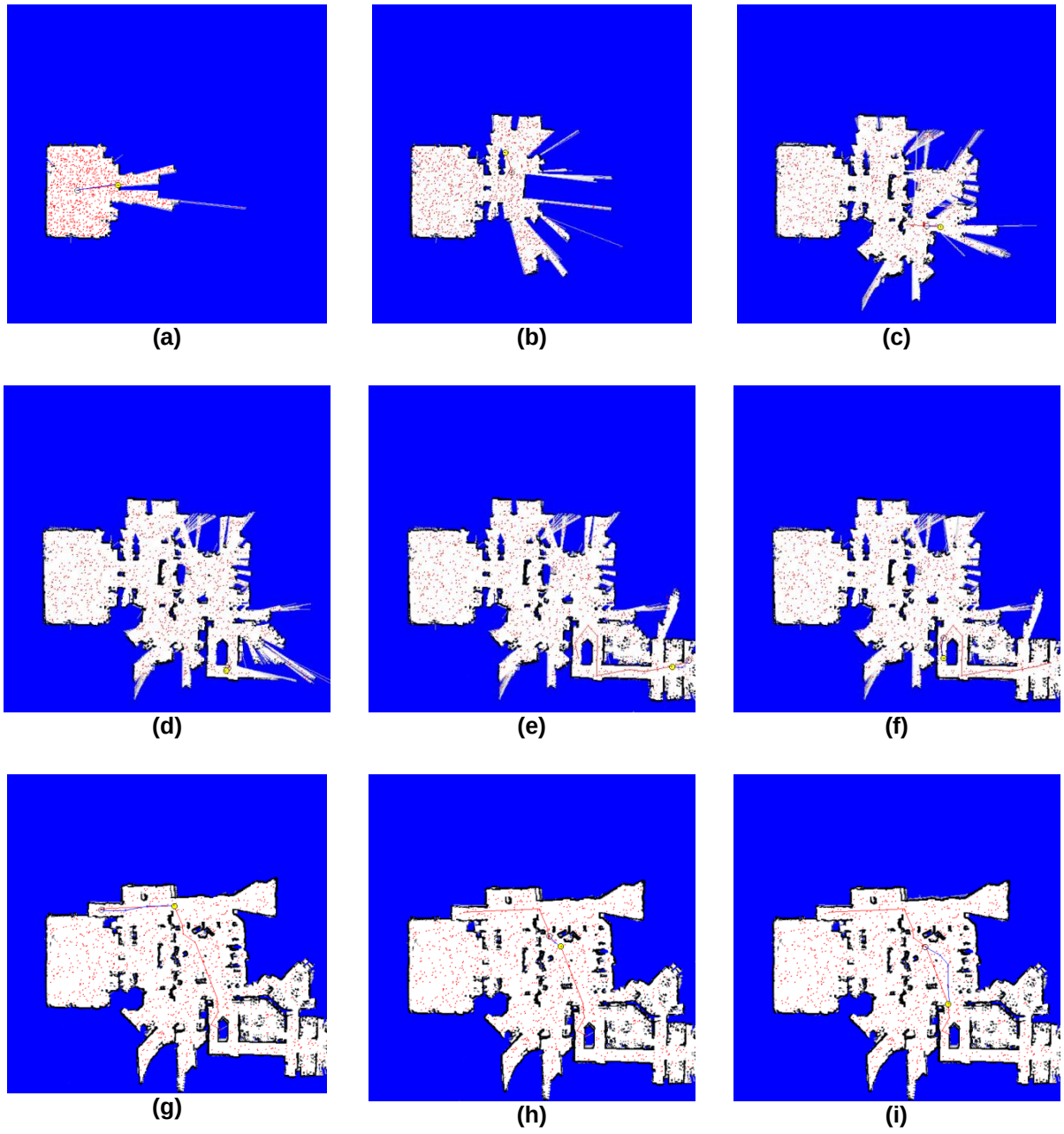



**Figura 64: Sequência de exploração que completa**

Na Figura 64 (g) até (i) é possível ver outro exemplo de planejamento longo e complexo em que pode ocorrer divergência entre o caminho tomado pela navegação (linha azul), conforme descrito no Capítulo 5, e o que foi planejado (linha vermelha), conforme descrito no Capítulo 6. Tal comportamento é esperado e desejável para evitar colisões do robô com os obstáculos, principalmente, por que o planejamento assume ser MDP quando, na realidade, ao fazer SLAM os estados são parcialmente observáveis e mudam constantemente.

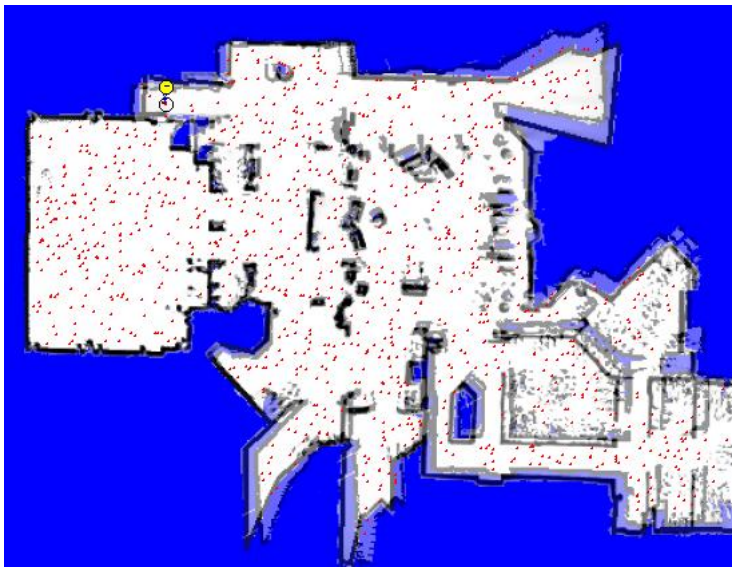


**Figura 65: Mapa original sobreposto à exploração completa**

Ao final da exploração, sobrepomos o mapa explorado no mapa original com o objetivo de visualizar se restou alguma região não explorada. Há um pequeno desalinhamento do mapa provocado pela incerteza na orientação das partículas de pose geradas pelo SLAM, mas ainda assim se pode ver na Figura 65 que todas as regiões desconhecidas foram exploradas.

## 9.3 Avaliação do sistema de vergência

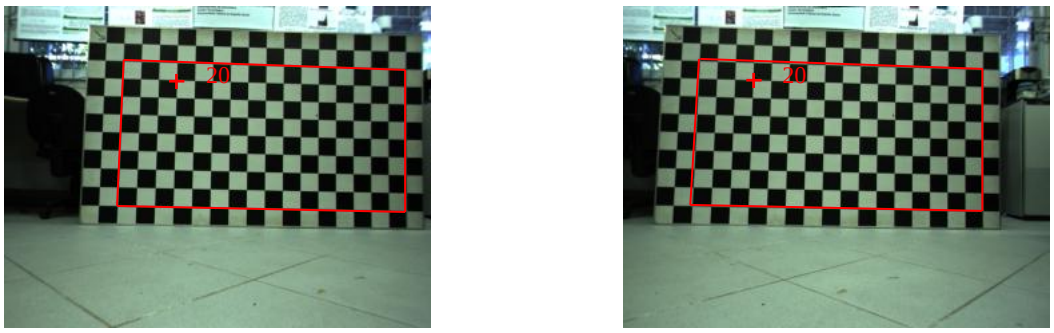
Na Seção 3.3 deste trabalho foi proposta uma nova arquitetura de vergência baseada em redes neurais VG-RAM para medir a disparidade em pixel entre pontos correspondentes nas imagens esquerda e direita. Na arquitetura anterior [KOM02], também baseada em redes neurais VG-RAM, o erro médio era de 3,6 pixel e em nossa nova arquitetura, como mostraremos, o erro médio é inferior a 1 pixel.

A métrica adotada para avaliação da precisão é dado pela raiz da média quadrática RMS (*Root Mean Square*). Ou seja, estamos interessados em medir a diferença na imagem esquerda entre o ponto de vergência calculado e o correto. Para determinarmos o ponto correto utilizamos um padrão reticulado no formato de tabuleiro de xadrez (*chessboard*) conforme Figura 66.

Esse padrão foi escolhido devido a precisão empregada na sua construção e, principalmente, pela facilidade de detecção de padrões de quinas. O método que utilizamos para detectar automaticamente as quinas do tabuleiro de xadrez emprega as funções *cvFindChessboardCorners* e *cvFindCornerSubPix* da biblioteca de visão computacional OpenCV [BRA00].

A função *cvFindChessboardCorners* lista as quinas detectadas linha por linha, da esquerda para a direita e de cima para baixo, a partir da primeira quina detectada. Há, portanto, num tabuleiro quadrado, quatro possibilidades de quinas iniciais que geram ambiguidade na orientação do tabuleiro. Para evitar isso, um tabuleiro de xadrez retangular (que contém M colunas e N linhas com $M \neq N$ e ambos ímpares) é usado em vez de um quadrado, e a quina superior esquerda é branca para ser detectada como a quina inicial.

Existem ao todo 180 quinas que podem ser determinadas com precisão sub-pixel (*cvFindCornerSubPix*) no quadro de calibração de câmeras da Figura 66 mas, devido a questão de simetria explicada logo acima, decidimos utilizar apenas 153 (17 x 9) quinas e descartar as duas primeiras fileiras e colunas. Sendo assim, tomamos as quinas da região selecionada na Figura 66(b) como pontos de interesse para treinamento do sistema de vergência e a Figura 66(a) para teste de vergência.




(a) (b)

**Figura 66: Quadro de calibração de dimensões 1,9m x 1,1m e células quadradas de 0,1m de lado. Imagem esquerda (a) e direita (b) retificadas. A cruz vermelha indica a quina 20 (contagem começando em zero) na imagem direita e o ponto de vergência na imagem esquerda, também representado por um cruz vermelha numerada.**

O gráfico da Figura 67 apresenta os resultados do teste de vergência no cálculo da disparidade (eixo y principal) em pixel para cada uma das quinas (eixo x) do tabuleiro de

xadrez. No eixo y secundário, à direita, é exibida a escala do erro (em preto) em pixel entre a disparidade real (em azul) e a calculada (em vermelho). A disparidade real varia entre 31 e 33 pixel e segundo a curva azul mais suave. Já a disparidade calculada varia um pouco mais, na faixa de 30 a 35 pixels, com curvas mais acentuadas devido à precisão da vergência não ser sub-pixel, como a real.

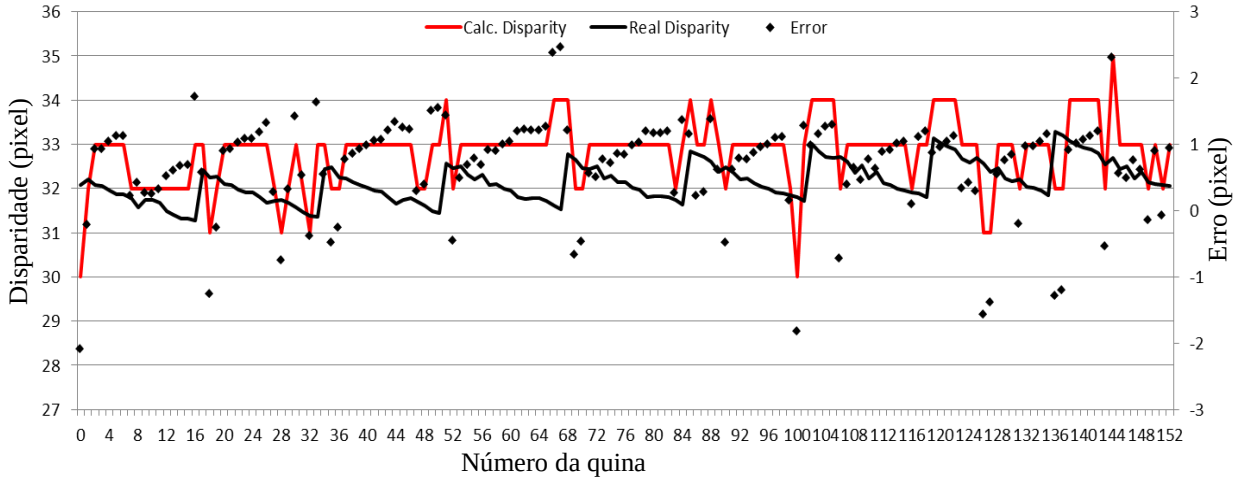



**Figura 67: Gráfico de dispersão do erro do sistema de vergência**

Contudo, o erro médio RMS da disparidade foi de 0,998 pixel em imagens com resolução de 320 x 240 pixels. Esse resultado demonstra uma melhora significativa da nova arquitetura em comparação com a anterior que apresentou erro RMS de 3,6 pixel [KOM02]. O erro de 1 pixel, em média, é suficiente para viabilizar a navegação robótica, em ambientes fechados, com objetivos definidos calculados por meio do sistema de vergência.

## 9.4 Avaliação do sistema de exploração visual

Conforme discutido no Capítulo 7, o sistema de exploração visual coordena a execução dos demais subsistemas componentes da arquitetura descrita na Seção 8.3 por meio de uma máquina de estados. Tal máquina de estados foi projetada para orquestrar a iteração assíncrona entre os módulos (subsistemas) e permitir a evolução do estado global do sistema em busca dos locais no caminho descrito pelas fotos.

Montamos dois experimentos desafiadores de exploração em ambiente fechado para demonstrar a estratégia de busca do sistema e disponibilizamos os vídeos no canal do LCAD (http://www.youtube.com/user/lcadufes) no Youtube sob o nome de *Visual Planning and Navigation powered by VGRAM Neural Networks and Probabilistic Robotics* e *Visual Planning and Navigation (apartment demo)*. Nesses vídeos existem janelas, conforme ilustrado na Figura 68, que representam o funcionamento dos principais subsistemas componentes do sistema de navegação visual.

O subsistema de vergência é exibido no canto superior esquerdo da Figura 68, onde pode ser visualizada uma janela com nome de in_vergence_trained contendo o ponto de atenção (cruz vermelha) na imagem direita da câmera estéreo; uma janela com o nome de in_vergence_current contendo o ponto de vergência (cruz vermelha) na imagem esquerda da câmera estéreo; e uma janela com o nome de out_vergence_v1_activation_map apresentando a saída da camada neural em busca do ponto de vergência.

O subsistema de busca visual é exibido no canto inferior esquerdo da Figura 68, onde pode ser visualizada uma janela com o nome de "in_saccade_trained" contendo os pontos de interesse (cruzes coloridas) encontrados pelo detector de quinas FAST na imagem da cena que se deseja reconhecer; uma janela com o nome de "in_saccade_current" contendo os pontos de interesse correspondentes (cruzes coloridas) encontrados pela busca visual; e uma janela com o nome de "out_saccade_v1_activation_map" apresentando a saída da camada neural em busca do ponto de sacada.

As saídas dos subsistemas de SLAM, exploração e navegação são exibidos no canto inferior direito da Figura 68, onde pode ser visualizada a janela com o nome de "CARMEN Planner" contendo a projeção do robô (círculo vermelho) e do objetivo de navegação (círculo amarelo) no mapa de ocupação 2D.

Acima da janela "CARMEN Planner", no canto superior direito da Figura 68, se encontra a janela de exibição de imagens estéreo da Bumblebee XB3 parcialmente oclusa, onde é possível ver apenas a imagem direita da câmera. Sobre a imagem esquerda da câmera está a janela "Camera View" para exibição de imagens do robô capturadas por uma câmera monocular externa. Por fim, no canto inferior esquerdo da Figura 68 (região retangular com

fundo branco) é impresso na tela o estado do sistema de navegação visual correspondente à máquina de estados descrita na Seção 7.2.

O primeiro experimento foi realizado num ambiente de escritório contendo mobílias padronizadas que provocam ambiguidades na busca visual por pontos de interesse com muita similaridade e repetição de padrões. Isso ocorre com mais frequência ao utilizarmos SURF para detecção de *features*, pois este busca identificar regiões (*blobs*) com certo grau de invariância a transformações de rotação e escala [BAY08].

Visando minimizar a escolha de pontos de interesse ambíguos testamos outro gerador de *features*, especializado em detecção de quinas (*corners*), conhecido como FAST [ROS06] num ambiente de apartamento. Ao tomar quinas como pontos de interesse, frequentemente, serão escolhidas regiões salientes na imagem que possuem um vetor gradiente de intensidade maior que nas demais regiões vizinhas e, consequentemente, se tornam mais discriminativas. Isto favorece o desempenho do sistema de busca visual e nos permite simplificar a validação de reconhecimento da cena. Dada uma quantidade mínima (ex.: três) de pontos de interesse encontrados pela busca visual, basta verificar se eles mantêm a mesma localização relativa entre si como heurística de reconhecimento dos pontos de interesse retratado na foto, como pode ser observado nos pontos em formato de cruz nas cores amarela, verde e azul, que aparecem ampliados (efeito lupa) na Figura 68, extraída do vídeo dos experimentos no apartamento.

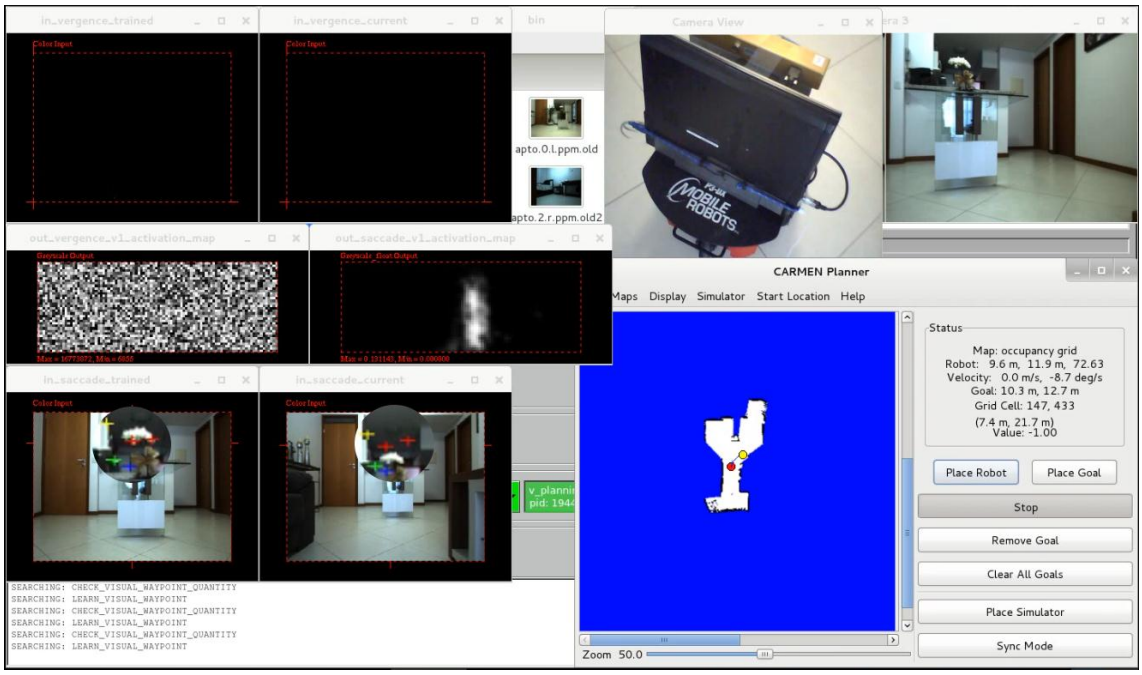



**Figura 68: Reconhecimento de pontos de interesse baseado em sua geometria relativa**

Em seguida ao reconhecimento dos três pontos de interesse são selecionados os dois pontos encontrados pela busca visual com melhor correspondência (que possuem o padrão de ativação mais próximo da barra branca de aprendizado). Esses dois pontos são passados para a vergência encontrar seus correspondentes na imagem esquerda e retornar suas disparidades, conforme ilustrado na Figura 69.

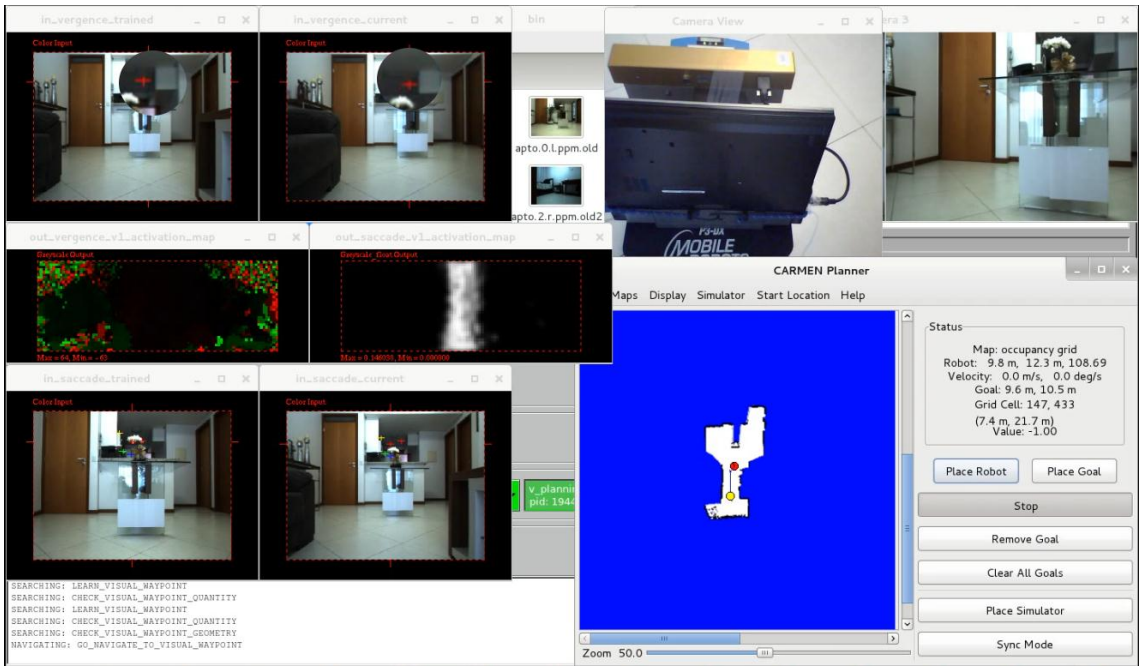



**Figura 69: Vergência de pontos de interesse para navegação**

O sistema utilizará as disparidades retornadas para calcular suas coordenadas no mundo e utilizará o mais próximo como *waypoint* para navegação, como pode ser visto na imagem ampliada dos pontos em formato de cruz na cor vermelha e no círculo amarelo no mapa da Figura 70. Para se aumentar o grau de certeza sobre o ponto de interesse usado para vergência, é feito continuamente um *tracking* do mesmo pela busca visual antes da vergência. O padrão de ativação em formato de barra apresentado na Figura 70 significa um alto grau de certeza para o ponto de interesse rastreado.

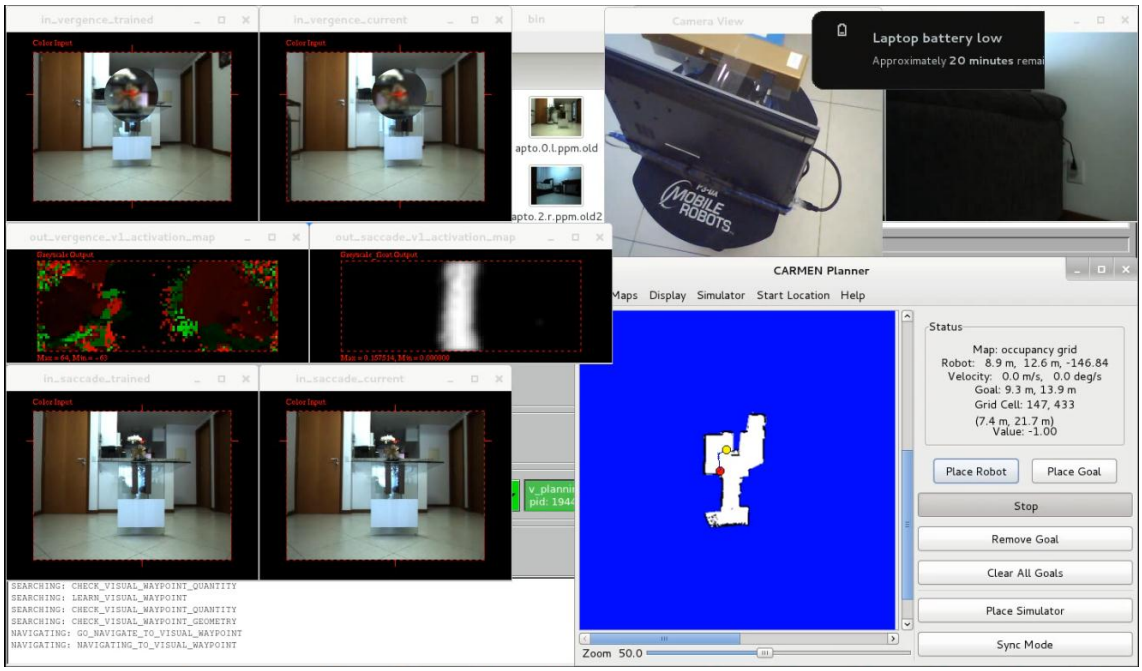



**Figura 70: *Tracking* e vergência do ponto de interesse na navegação**

Ao se aproximar a distância desejada do ponto de interesse, o local é definido como alcançado e um próximo será carregado para busca. As Figura 71 e Figura 72 ilustram as etapas de reconhecimento, vergência, navegação e *tracking* para o próximo ponto de interesse no mundo.

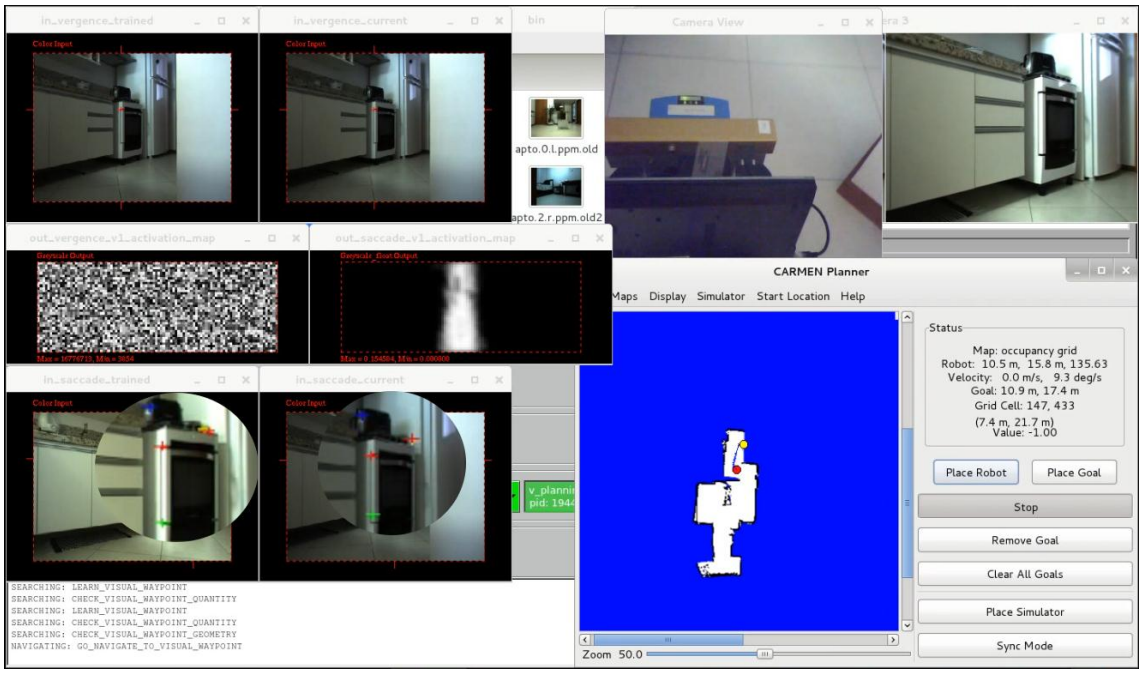



**Figura 71: Reconhecimento do segundo ponto de interesse no mundo**

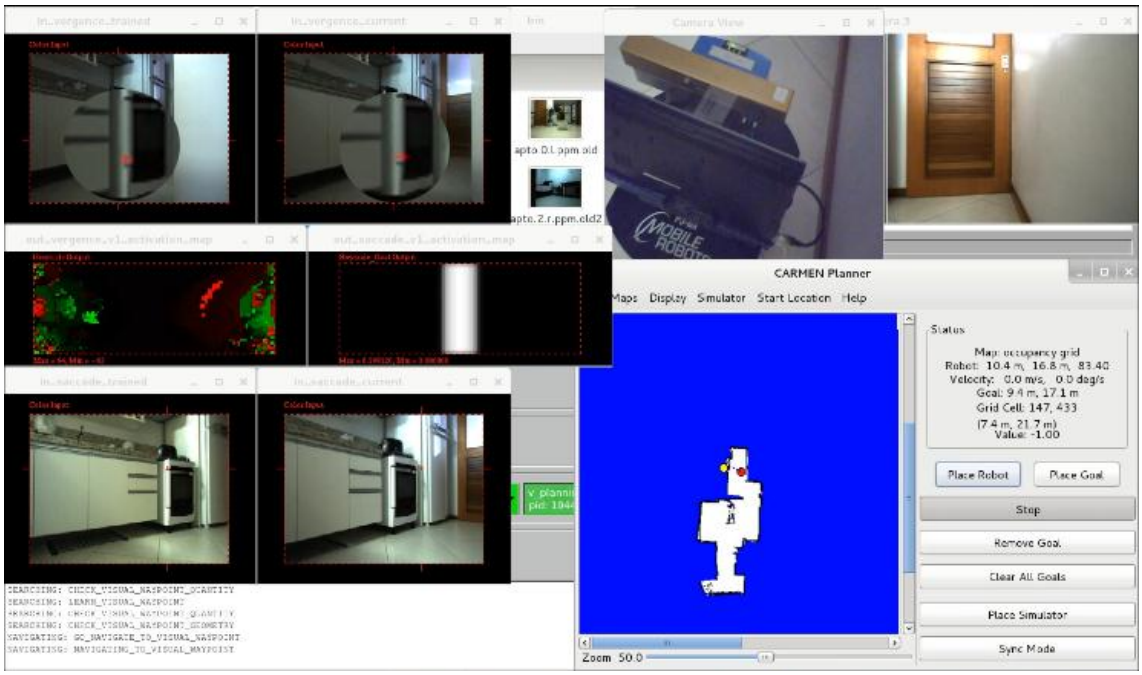



**Figura 72: Vergência e *tracking* do segundo ponto de interesse no mundo**

Após o robô ter se aproximado à uma distância desejada do próximo ponto de interesse e não havendo mais pontos a serem procurados, o sistema define ponto de partida como objetivo de navegação para que o robô retorne até ele e finalize a missão com sucesso, tal qual ilustrado na Figura 73.

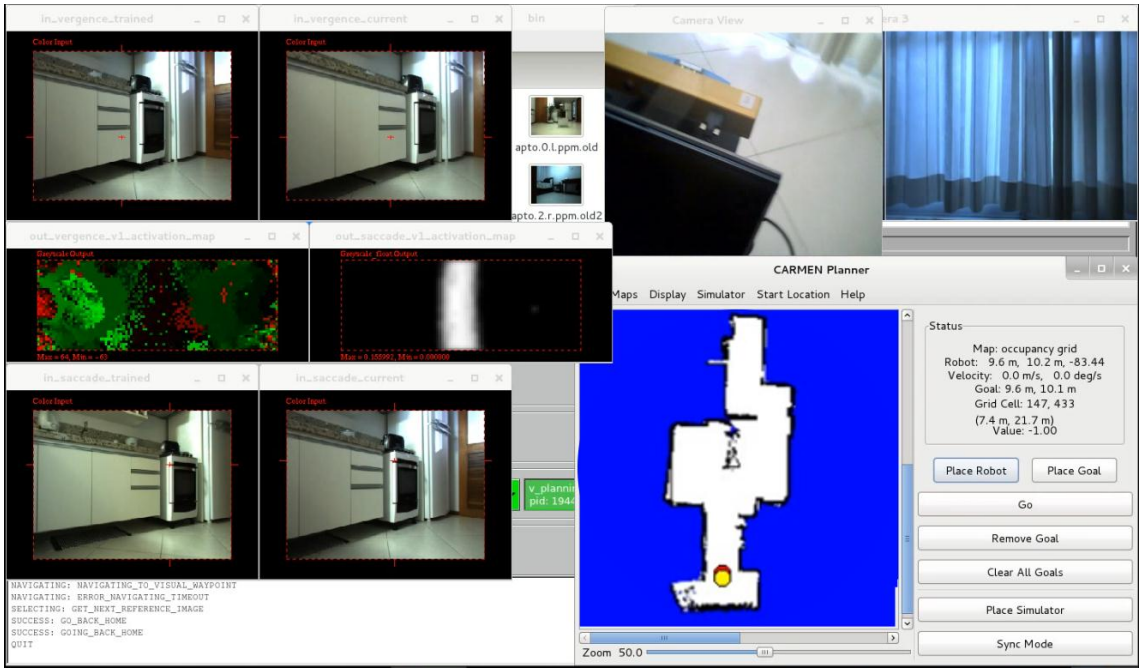



**Figura 73: Busca finalizada com retorno ao ponto de partida**

# 10 DISCUSSÃO

Neste capítulo é apresentada uma análise comparativa deste trabalho de pesquisa com dois outros trabalhos correlatos. Um deles busca a plausibilidade biológica, como o nosso, mas utiliza uma arquitetura cognitiva diferente denominada ACT-R, enquanto o outro segue uma abordagem puramente matemática baseada em robótica probabilística. Em seguida a esta análise comparativa, é apresentada uma análise crítica deste trabalho de pesquisa.

## 10.1 Trabalhos correlatos

Poucos trabalhos na literatura apresentam uma visão integradora de sistema de navegação robótica exploratória por imagens como a apresentada aqui.

No trabalho proposto por Furgale [FUR10] é utilizada uma câmera estereoscópica para mapear o ambiente e localizar o robô utilizando mapas de *landmarks* [THR05]. O robô é conduzido na etapa de aprendizado do caminho de forma que sejam gerados e armazenados submapas do caminho conectados topologicamente. Na etapa seguinte ao aprendizado, basta então ao robô reconstruir os mapas novamente e seguir a rota armazenada na etapa de aprendizado. Nosso sistema de navegação não exige uma etapa *off-line* de aprendizado nem tão pouco necessita de uma sequência contígua de imagens para encontrar o caminho.

O trabalho apresentado por Reitter [REI10] apoia-se em modelos cognitivos de memória e percepção visual baseados na arquitetura ACT-R para testar suas hipóteses de navegação visual, tendo como informação uma visão superior do mapa, os pontos de interesse para navegação e onde o robô está. Para isso, são utilizadas simulações 2D de labirintos e informações *off-line* coletadas num ambiente onde múltiplos robôs foram guiados por seres humanos. Em nosso trabalho, tratamos o problema de navegação robótica em ambientes reais fornecendo apenas fotos dos locais a serem visitados pelo robô, que deverá descobri-los com a ajuda do sistema de exploração, reconhecê-los com a ajuda do sistema de busca visual e navegar de forma autônoma até eles com a ajuda do sistema de vergência, sem colidir com

nenhum obstáculo. Outra importante diferença do nosso trabalho para o trabalho apresentado em [REI10] é que este emprega modelos de percepção e memória baseados em ACT-R e o nosso utiliza modelos de busca visual e vergência baseados em VG-RAM. Infelizmente, não há informações suficientes para comparação de desempenho entre o nosso modelo de percepção visual baseado em VG-RAM e o deles baseado em ACT-R.

## 10.2 Análise crítica deste trabalho de pesquisa

O maior desafio deste trabalho foi integrar algumas soluções do estado da arte em robótica móvel com modelos de cognição visual implementados no LCAD – UFES. A seguir, discutiremos os pontos positivos e negativos a respeito dos subcomponentes empregados e algumas melhorias para aumentar a eficiência global do sistema.

### 10.2.1 Sistema de vergência

O movimento de vergência dos olhos e seu efeito na representação interna ao cérebro para percepção de profundidade têm sido estudados há mais de uma década pelo grupo de ciência da cognição do LCAD – UFES [KOM02, OLI05]. Nesses anos foram alcançados excelentes resultados com diferentes arquiteturas neurais, tendo a nova arquitetura proposta neste trabalho apresentado melhores resultados quanto ao desempenho e à precisão, mantida sua plausibilidade biológica.

Diante de resultados tão promissores, cabe investigar mais a fundo a precisão com diferentes resoluções de imagem e afastamento (*baseline*) entre o par de câmeras estereoscópicas. Proporcionalmente ao aumento do tamanho das imagens, pode ser necessário aumentar o nível de paralelismo no seu processamento. Tal abordagem será importante para a execução de simulações com propósito de analisar a sensibilidade da rede aos parâmetros da arquitetura neural.

### 10.2.2 Sistema de busca visual

O sistema de busca visual desenvolvido por Oliveira Neto [NET12] apresenta excelentes resultados na busca por um ponto em imagens estáticas e dinâmicas. Entretanto, analisando o emprego desse sistema num contexto mais geral de rastreamento é possível identificar algumas melhorias para torná-lo mais robusto e mais rápido.

No sentido de torná-lo mais robusto, é importante reexaminá-lo frente ao sistema *Tracking-Learning-Detection* (TLD), proposto por Kalal [KAL10]. Considerando o arcabouço proposto por Kalal [KAL10], o que falta para tornar o *tracking* pela busca visual mais robusto é uma etapa de detecção que faça o reconhecimento da região ao redor do ponto de sacada (ou ponto de fixação após o movimento sacádico; ver Seção 2.5) para confirmar, ou não, se a busca convergiu para o ponto de interesse treinado. Outra importante melhoria seria a implementação de um modelo de movimento ocular de perseguição suave que tenta predizer a direção do movimento do objeto de interesse. Alternativamente, é interessante estudar formas de retreinar a rede (sobrescrevendo aprendizado antigo ou adicionando aprendizado novo) para um mesmo objeto de interesse em movimento.

Para tornar o sistema de busca visual VG-RAM mais rápido, pretendemos investigar o modelo de Redes Neurais Sem Peso WISARD [LUD99] com busca em tabelas de dispersão (*hash tables*) [NOR12]. Um sistema de busca visual mais rápido permitiria realizar buscas por mais de um ponto de interesse em tempo *quasi*-real, suficiente para o robô assumir velocidades maiores.

### 10.2.3 Sistema de exploração em *grid*

O algoritmo de planejamento MDP para exploração permite reduzir drasticamente o espaço de busca de otimização por caminhos de exploração. Por se tratar de um algoritmo de planejamento que assume os estados completamente observáveis (MDP) e estáticos, seu emprego foi complementado por um algoritmo de SLAM para atualizar os mapas e por um algoritmo de navegação para permitir ao robô navegar desviando de obstáculos e explorando o ambiente em busca de regiões desconhecidas.

### 10.2.4 Sistema de exploração visual

A estratégia da exploração visual é, dada uma sequência de fotos do caminho, utilizar a exploração robótica para navegar pelo ambiente, desviando de obstáculos com ajuda de um sensor de varredura laser e, ao mesmo tempo, identificar os locais representados nas fotos através de câmeras. Há então uma clara divergência de objetivos: o laser varre o ambiente num ângulo aberto à procura de obstáculos e a câmera busca imagens de pontos de interesse num ângulo mais restrito.

Empregamos um sensor laser para mapeamento e detecção de obstáculos e uma câmera estereoscópica para localização de pontos de interesse. O fato de usarmos sensores com propósitos diferentes traz divergências na percepção do ambiente do robô. . Esses dois sensores possuem campos de visão (*Field of View* - FoV) muito diferentes, dificultando o alinhamento de objetivos entre os diferentes subsistemas que os utilizam como fonte de informação a respeito do mundo.

O problema é evidenciado no vídeo http://www.lcad.inf.ufes.br/~avelino/videos/visual_planning_lost_place_recognition.mp4 ao se explorar o mundo com um campo de visão maior do que o sensor que fornece as imagens para reconhecimento dos pontos de interesse. Para minimizar a divergência entre a FoV do laser e a da câmera, reduzimos a FoV do laser de 180º para o mínimo permitido, que é 100º. Ainda assim, resta uma lacuna de 34º em relação a FoV da câmera *Bumblebee* XB 3, que é de 66º. Esse ponto cego existe e tentou-se contorná-lo com o esquecimento da região explorada quando o robô se perde. O vídeo http://www.lcad.inf.ufes.br/~avelino/videos/visual_planning_go_back_last_known_pose.mp4 demonstra o robô retornando ao último ponto de interesse conhecido e reiniciando a exploração a partir deste ponto. Provocar intencionalmente o esquecimento do caminho percorrido pelo robô pode ser útil em duas situações: (1) passar pela cena procurada e não reconhecê-la por alguma oclusão parcial ou total, diferença de ponto de vista entre as cenas ou por tempo insuficiente para realizar a busca visual; (2) em caso do robô se perder e acabar passando pelos pontos de interesse seguintes e descartar inadvertidamente essas áreas de uma exploração futura.

A escolha das imagens de pontos de interesse constitui uma tarefa crítica para o sucesso da missão de busca. Nesse sentido, foi necessário garantir que as fotos não possuem intersecção entre si, de forma a manter a representatividade da cena. No caso de haver intersecção, corre-se o risco de perder um ponto de interesse contido nas duas cenas.

Unificar a percepção do mundo através de um único sensor, preferencialmente, câmeras estereoscópicas, daria maior plausibilidade biológica, além de simplificar o Sistema de Navegação Robótica por Imagens de Pontos de Interesse.

# 11 CONCLUSÃO

Neste capítulo apresentamos um breve sumário deste trabalho de pesquisa e, em seguida, discutimos os principais resultados alcançados e nossas conclusões. Por fim, apontamos direções para trabalhos futuros.

## 11.1 Sumário

Plataformas robóticas têm se tornado uma alternativa economicamente viável e segura para exploração em ambientes inóspitos ou inacessíveis ao ser humano. Contudo, o grau de autonomia de tais plataformas ainda é relativamente baixo, o que exige, frequentemente, a intervenção remota humana para minimizar riscos de acidentes ou mesmo para guiar o robô em tarefas de maior complexidade.

Neste trabalho investigamos modelos matemático-computacionais de cognição visual aplicados ao problema de navegação de veículos autônomos. Para tal, empregamos SLAM e navegação robótica exploratória probabilísticas, em conjunto com modelos de cognição visual dos movimentos oculares de busca visual e vergência, para construir um sistema capaz de navegar um robô autônomo através de um caminho definido por uma sequência de pontos de interesse representados por fotos desses locais numa visão em primeira pessoa. Nossa avaliação deste sistema mostrou que a abordagem empregada é promissora e que, com avanços no desempenho em termos de tempo dos modelos, tal abordagem pode ser empregada em casos reais de interesse.

## 11.2 Conclusões

Nesta dissertação apresentamos contribuições relevantes para problemas de exploração robótica e projeção de pontos de interesse no mundo por meio de sistemas de vergência e busca visual neurais. Além disso, mostramos ser viável o emprego de uma plataforma

robótica para a realização de tarefas que requerem alto nível de capacidade cognitiva de forma autônoma.

Ainda há muito a ser melhorado, mas dispomos agora de plataformas robóticas maduras e de fundamentos no estado da arte de robótica móvel probabilística que nos permitirão avançar em nossos estudos sobre cognição visual em ambientes dinâmicos reais.

## 11.3 Trabalhos futuros

Em trabalhos futuros buscaremos desenvolver um sistema de navegação visual que opere em tempo real utilizando apenas câmeras estereoscópicas que possua plausibilidade biológica. Para isso, será preciso aprimorar o desempenho e a precisão dos modelos de percepção visual empregados nesse trabalho, e desenvolver novos modelos cognitivos para navegação e localização.

# REFERÊNCIAS BIBLIOGRÁFICAS


[ALE66] ALEKSANDER, I. Self-adaptive universal logic circuits. **IEEE Electronic Letters**, v. 2, n. 8, p. 231-232, 1966.

[ALE98] ALEKSANDER, I. RAM-Based Neural Networks From WISARD to MAGNUS: a Family of Weightless Virtual Neural Machines. In: AUSTIN, J. **RAM-Based Neural Networks**. [S.l.]: World Scientific, 1998. p. 18-30.

[BAD08] BADUE, C.; PEDRONI, F.; DE SOUZA, A. F. Multi-Label Text Categorization using VG-RAM Weightless Neural Networks. **Proceedings of the 10th Brazilian Symposium on Neural Networks (SBRN'08)**, Salvador, BA, 2008. 105-110.

[BAY08] BAY, H. et al. SURF: Speeded Up Robust Features. **Computer Vision and Image Understanding (CVIU)**, 2008. 346-359.

[BRA00] BRADSKI, G. The OpenCV Library. **Dr. Dobb's Journal of Software Tools**, 2000.

[BUE05] BUEHLER, M.; IAGNEMMA, K.; SINGH, S. **The 2005 DARPA Grand Challenge:** The Great Robot Race. [S.l.]: Springer, 2005.

[BUE08a] BUEHLER, M.; IAGNEMMA, K.; SINGH, S. Special Issue on the 2007 DARPA Urban Challenge, Part I. **Journal of Field Robotics**, 25, n. 8, 2008. 423-566.

[BUE08b] BUEHLER, M.; IAGNEMMA, K.; SINGH, S. Special Issue on the 2007 DARPA Urban Challenge, Part II. **Journal of Field Robotics**, 25, n. 9, 2008. 567-724.

[DOE10] CHRISTIAN F. DOELLER, C. B. N. B. Evidence for grid cells in a human memory network. **Nature**, n. 463, p. 657-661, 2010.

[DES07] DE SOUZA, A. F. et al. Automated Free Text Classification of Economic Activities using VG-RAM Weightless Neural Networks. **Proceedings of the 7th International Conference on Intelligent Systems Design and Applications**, 2007. 782-787.

[DES08b] DE SOUZA, A. F. et al. Face Recognition with VG-RAM Weightless Neural Networks. **Lecture Notes in Computer Science**, v. 5163, n. 1, p. 951-960, 2008.

[DES08a] DE SOUZA, A. F. et al. Improving VG-RAM WNN Multi-label Text Categorization via Label Correlation. **8th International Conference on Intelligent Systems Design and Applications**, 2008.

[DES09] DE SOUZA, A. F. et al. Automated Multi-label Text Categorization with VG-RAM Weightless Neural Networks. **Neurocomputing**, v. 72, p. 2209-2217, 2009.

[DEA99] DEANGELIS, G. C.; NEWSOME, W. T. Organization of disparity-selective neurons in macaque area MT. **The Journal of Neuronscience**, v. 19, n. 4, p. 1398-1415, 1999.

[DOD03] DODGE, R. Five Types of eye movement in the horizontal meridian plane of the field of regard. **Am. J. Physiol**, v. 8, p. 307-329, 1903.

[DUR06] DURRANT-WHYTE, H.; BAILEY, T. Simultaneous Localisation and Mapping (SLAM): Part I The Essential Algorithms. **IEEE Robotics and Automation Magazine**, v. 13, n. 2, p. 99-110, Junho 2006.

[EBE01] EBENHOLTZ, S. M. **Oculomotor Systems and Perception**. [S.l.]: Cambridge University Press, 2001.

[MOS08] EDVARD I. MOSER, M.-B. M. A metric for space. **Hippocampus**, v. 18, n. 12, p. 1142 -1156, 2008.

[ELF89] ELFES, A. Using Occupancy Grids for Mobile Robot Perception and Navigation. **Computer**, v. 22, n. 6, 1989.

[ERC98] ERCEGOVAC, M. D.; LANG, T.; MORENO, J. H. **Introduction to Digital Systems**. [S.l.]: Wiley, 1998.

[FAR03] FARDIN JR, D.; KOMATI, K. S.; DE SOUZA, A. F. Arquitetura do Sistema Visual Humano: Uma Abordagem Computacional. In: _____ **III Escola Regional de Informática RJ/ES**. Porto Alegre - RS: Sociedade Brasileira de Computação, 2003. p. 95-126.

[FOX97] FOX, D.; BURGARD, W.; THRUN, S. The dynamic window approach to collision avoidance. **IEEE Robotics & Automation Magazine**, v. 4, n. 1, 1997.

[FUR10] FURGALE, P.; BARFOOT, T. Visual Path Following on a Manifold in Unstructured Three-Dimensional Terrain. **2010 IEEE International Conference on Robotics and Automation (ICRA)**, Anchorage, 2010.

[FYH08] FYHN, M. et al. Grid cells in mice. **Hippocampus**, v. 18, n. 12, p. 1230-1238, 2008.

[GEG97] GEGENFURTNER, K. R.; KIPER, D. C.; LEVITT, J. B. Functional Properties of Neurons in Macaque Area V3. **J Neurophysiol**, v. 77, n. 4, p. 1906–1923, Abril 1997.

[GOL99] GOLLEDGE, R. G. (Ed.). **Wayfinding Behavior:** Cognitive Mapping and Other Spatial Processes. 1. ed. [S.l.]: The Johns Hopkins University Press, 1999. 125 p.

[GRE98] GREGORY, R. **Eye and Brain**. 5th Ed. ed. [S.l.]: Oxford University Press, 1998.

[GRE05] GREY TEAM. Team Gray Technical Paper DARPA Grand Challenge 2005, 2005. Disponivel em: <http://www.darpa.mil/grandchallenge05/TechPapers/GreyTeam.pdf>.

[GRI05] GRISETTI, G.; STACHNISS, C.; BURGARD, W. Improving Grid-based SLAM with Rao-Blackwellized Particle Filters by Adaptive Proposals and Selective Resampling. **Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)**, 2005.

[HAF05] HAFTING, T. et al. Microstructure of a Spatial Map in the Entorhinal Cortex. **Nature**, n. 436, p. 801-805, 2005.

[HEY84] HEYDT, R. V. D.; PETHERHANS, E.; BAUMGARTNER, G. Illusory contours and cortical neuron responses. **Science**, p. 224:1260-1262, 1984.

[HOH03] HOHPE, G.; WOOLF, B. **Enterprise Integration Patterns:** Designing, Building, and Deploying Messaging Solutions. 1. ed. [S.l.]: Addison-Wesley Professional, 2003.

[HUB95] HUBEL, D. H. **Eye, Brain and Vision**. [S.l.]: Scientific Amarican Library, 1995.

[HUB62] HUBEL, D. H.; WEISEL, T. N. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. **J. Physiol.**, n. 160, p. 106-154, 1962.

[KAL10] KALAL, Z.; MIKOLAJCZYK, K.; MATAS, J. Tracking-Learning-Detection. **IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE**, v. 6, n. 1, Janeiro 2010.

[KAN00] KANDEL, E. R. . S. J. H. . J. T. M. **Principles of Neural Science**. 4th Ed. ed. [S.l.]: Prentice-Hall International, Inc., 2000.

[KOM02] KOMATI, K. S.; DE SOUZA, A. F. Vergence Control in a Binocular Vision System using Weightless Neural Networks. **Proceedings of the 4th International Symposium on Robotics and Automation**, 2002.

[KON00] KONOLIGE, K. A gradient method for realtime robot control. **In Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)**, 2000. 2436-2441.

[LAU98] LAUMOND, J.-P. (Ed.). **Robot Motion Planning and Control:** Lectures Notes in Control and Information Sciences. 229. ed. [S.l.]: Springer, 1998. 343 p.

[LEE88] LEE, C.; W.H., R.; SPARKS, D. L. Population coding of saccadic eye movements by neurons in the superior colliculus. **Nature**, p. 332: 357–360, 1988.

[LUD99] LUDERMIR, T. B. et al. Weightless neural models: a review of current and past works. **Neural Computing Surveys**, v. 2, 1999. ISSN 41-61.

[MAT12] MATHER, G. The Visual Cortex. Disponivel em: <http://www.lifesci.sussex.ac.uk/home/George_Mather/Linked%20Pages/Physiol/Cortex.html>. Acesso em: 14 Agosto 2012.

[MIT98] MITCHELL, R. J. et al. Comparison of Some Methods for Processing Grey Level Data in Weightless Networks. In: _____ **RAM-Based Neural Networks**. [S.l.]: World Scientific, 1998. p. 61-70.

[MON03b] MONTEMERLO, M.; ROY, N.; THRUN, S. Perspectives on standardization in mobile robot programming: The carnegie mellon navigation (CARMEN) toolkit. **In Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)**, 2003. 2436-2441.

[MON03a] MONTEMERLO, M.; THRUN, S. FastSLAM 2.0: An Improved Particle Filtering Algorithm for Simultaneous Localization and Mapping that Probably Converges, 2003.

[MOV85] MOVSHON, J. A. et al. The analysis of moving visual patterns. In: CHAGAS, C.; GATTASS, R.; GROSS, C. **Pattern Recognition Mechanisms**. New York: Springer, 1985. p. 117-151.

[NET12] NETO, J. D. O. **Um Sistema de Busca Visual Biologicamente Plausível Baseado em Redes Neurais Sem Peso**. 2012. Monografia (Graduação em Engenharia de Computação). Departamento de Informática, Universidade Federal do Espírito Santo, Vitória, 2012.

[NOR12] NOROUZI, M.; PUNJANI, A.; FLEET, D. J. Fast Search in Hamming Space with Multi-Index Hashing. **IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, Providence, 2012.

[KEE71] O'KEEFE, J. . D. J. The hippocampus as a spatial map. Preliminary evidence from unit activity in the freely-moving rat. **Brain Research**, 1971. 171-175.

[OLI05] OLIVEIRA, H. **Uma Modelagem Computacional de Áreas Corticais do Sistema Visual Humano Associadas à Percepção de Profundidade**. 2005. Dissertação (Mestrado em Informática). Departamento de Informática, Universidade Federal do Espírito Santo, Vitória, 2005.

[POR03] PORT, N. L.; WURTZ, R. H. Sequential activity of simultaneously recorded neurons in the superior colliculus during curved saccades. **Journal of Neurophysiology**, 2003. 1887–1903.

[RED05a] RED TEAM. DARPA Grand Challenge 2005 Technical Paper, 2005. Disponivel em: <http://www.darpa.mil/grandchallenge05/TechPapers/RedTeam.pdf>.

[RED05b] RED TEAM TOO. Red Team Too DARPA Grand Challenge 2005 Technical Paper, 2005. Disponivel em: <http://www.darpa.mil/grandchallenge05/TechPapers/RedTeamToo.pdf>.

[REI10] REITTER, D.; LEBIERE, C. A Cognitive model of spatial path-planning. **Computational & Mathematical Organization Theory**, 2010.

[ROS06] ROSTEN, E.; DRUMMOND, T. Machine Learning for High-Speed Corner Detection. **Computer Vision**, v. 3951, n. 2006, p. 430-443, 2006.

[STA05] STANFORD RACING TEAM. Stanford Racing Team's Entry in the 2005 DARPA Grand Challenge, 2005. Disponivel em: <http://www.darpa.mil/grandchallenge05/TechPapers/Stanford.pdf>.

[STA07] STANFORD RACING TEAM. Stanford's Robotic Vehicle 'Junior': Interim Report, 2007. Disponivel em: <http://www.darpa.mil/grandchallenge/TechPapers/Stanford.pdf>.

[TER05] TEAM TERRAMAX. Team TerraMax Darpa Grand Challenge 2005, 2005. Disponivel em: <http://www.darpa.mil/grandchallenge05/TechPapers/TeamTerraMax.pdf>.

[THR05] THRUN, S. . B. W. . F. D. **Probabilistic Robotics**. [S.l.]: MIT Press, 2005.

[THR06] THRUN, S. E. A. Stanley: The robot that won the DARPA Grand Challenge. **Journal of Robotic Systems**, 2006. 661 – 692.

[TOO82] TOOTELL, R. B. et al. Deoxyglucose analysis of retinotopic organization in primate striate córtex. **Science**, n. 218, p. 902-904, 1982.

[WAL05] WALTON, M. M. G.; SPARKS, D. L.; GANDHI, N. J. Simulations of Saccade Curvature by Models That Place Superior Colliculus Upstream From the Local Feedback Loop. **Journal of Neurophysiology**, 2005. 2354-2358.

[WEI10] WEINSTEIN, A. J.; MOORE, K. L. Pose estimation of Ackerman steering vehicles for outdoors autonomous navigation. **IEEE International Conference on Industrial Technology (ICIT)**, 2010. 579-584.

[ZEK73] ZEKI, S. M. Colour coding of the rhesus monkey prestriate cortex. **Brain Research**, 1973. 422-427.