Gilberto Antonio Marcon dos Santos
Victor Terra Ferrão

Proposta de Plataforma de Pesquisa em Robótica Móvel Autônoma

Goiânia, Brasil

2016

Gilberto Antonio Marcon dos Santos
Victor Terra Ferrão

Proposta de Plataforma de Pesquisa em Robótica Móvel Autônoma

Trabalho apresentado como requisito parcial para a conclusão do projeto final de curso 2 de engenharia de computação da Universidade Federal de Goiás.

Universidade Federal de Goiás – UFG

Escola de Engenharia Elétrica, Mecânica e de Computação – EMC

Engenharia de Computação

Orientador: Profº Dr. Cássio Dener Noronha Vinhal

Goiânia, Brasil

2016

Gilberto Antonio Marcon dos Santos
Victor Terra Ferrão

Proposta de Plataforma de Pesquisa em Robótica Móvel Autônoma

Trabalho apresentado como requisito parcial para a conclusão do projeto final de curso 2 de engenharia de computação da Universidade Federal de Goiás.

Goiânia, Brasil, 4 de março de 2016:

Profº Dr. Cássio Dener Noronha Vinhal
Orientador

Profº Dr. Carlos Galvão Pinheiro Júnior
Membro 1

Profº Dr. Marco Antônio Assfalk de Oliveira
Membro 2

Profº Dr. Gélson da Cruz Júnior
Membro suplente

Goiânia, Brasil
2016

Resumo

Este documento apresenta a proposta de uma plataforma de pesquisa em robótica autônoma móvel cujo objetivo é servir como base para estudo e desenvolvimento de métodos relacionados. Tal plataforma visa proporcionar um patamar base para desenvolvimento, implementação e teste de sistemas de sistemas, sistemas de controle, sistemas de navegação, sistemas de mapeamento e sistemas de sensoriamento. Em adição, a plataforma a ser proposta visa também servir como ferramenta para o desenvolvimento e teste de algoritmos de cooperação multi-agentes. É apresentado um detalhado levantamento das principais tecnologias e métodos necessários para a construção da plataforma proposta, além de uma seleção de tecnologias e métodos alternativos, que podem ser usados no lugar dos métodos propostos, tendo diferentes custos e benefícios. Por fim, é apresentada uma implementação da plataforma e uma avaliação dos resultados obtidos.

Palavras-chave: Robótica móvel autônoma. Sistemas de controle. Reconhecimento de padrões. Processamento de imagens.

Abstract

This document presents a research platform for autonomous robotics research. The goal of this platform is to support studies and development of related methods. This platform aims to be the basis for developing, implementing and testing systems of systems, control systems, navigation systems, mapping systems, and sensing systems. Additionally, this proposed platform aims to serve as a tool for developing and testing multi-agent cooperation algorithms. A detailed survey of the main technologies and methods necessary to build this proposed platform is presented, in addition to alternative technologies and methods, that may be used to substitute the proposed methods, having different cost-benefit trade-offs. Finally, an implementation of the platform and an evaluation of the results are presented.

Keywords: Mobile autonomous robotics. Control systems. Pattern recognition. Image processing.

Lista de ilustrações

Lista de abreviaturas e siglas

ADALINE	Nerônio adaptativo linear , do inglês: *adaptive linear neuron*.
ASIC	Circuito integrado específico de aplicação, do inglês: *application-specific integrated circuit*.
DCM	Gerenciador digital de *clock*, do inglês: *digital clock manager*.
DIP	Pacote dual em linha, do inglês: *dual in-line package*.
CLB	Bloco lógico configurável, do inglês: *configurable logic block*.
DSP	Processamento digital de sinais, do inglês: *digital signal processing*.
EBP	Retro-propagação de erro, do inglês: *error backpropagation*.
FPGA	Arranjo de portas programáveis por campo, do inglês: *field-programmable gate array*.
GPU	Unidade de processamento gráfico, do inglês: *graphic processing unit*.
HL	Camada oculta, do inglês: *hidden layer*.
IMU	Unidade de medida inercial, do inglês: *inertial measurement unit*.
IOB	Bloco de entrada e saída, do inglês: *input/output block*.
LAN	Rede de área local, do inglês: *local area network*.
LED	Diodo emissor de luz, do inglês: *light emitting diode*.
LUT	Tabela de referência, do inglês: *look-up table*.
MCU	Microcontrolador, do inglês: *microcontroller unit*.
MLP	Perceptron de múltiplas camadas, do inglês: *multi-layer perceptron*.
MPU	Microprocessador, do inglês: *microprocessing unit*.
OL	Camada de saída, do inglês: *output layer*.
PAN	Rede de área pessoal, do inglês: *personal area network*.
PWM	Modulação por largura de pulso, do inglês: *pulse-width modulation*.
PID	Proporcional-diferencial-integral, do inglês: *proportional-integral-derivative*.

RBF	Função baseada em raio, do inglês: *radial basis funciton*.
RNA	Redes neurais artificiais.
SBM	Correspondência estéreo de blocos, do inglês, Stereo Block Matching.
SGBM	Correspondência estéreo semi-global de blocos, do inglês, Semi-Global Stereo Block Matching.
SoC	Sistema em chip, do inglês: *system on a chip*.
SPI	Interface periférica serial, do inglês: *serial peripheral interface*.
UART	Transmissor/receptor assíncrono universal, do inglês: *universal asynchronous receiver/transmitter*.
USB	Barramento serial universal, do inglês: *universal serial bus*.
VAF	Fator de aumento de velocidade, do inglês: *velocity augmentation factor*.
VGA	Arranjo gráfico de vídeo, do inglês: *video graphics array*.

Sumário

Introdução

Um número significativo de plataformas de pesquisa em robótica móvel foi proposto recentemente. São soluções que tipicamente oferecem hardware e software de baixo nível, provendo serviços à camada de aplicações. Em geral, permitem ao desenvolvedor de aplicações abstrair os detalhes de hardware e focar seu trabalho no desenvolvimento de aplicações de sensoriamento, controle e tomada de decisões de mais alto nível. Tais plataformas têm o objetivo de suportar o desenvolvimento de métodos relacionados a robótica móvel, tais como métodos de navegação, mapeamento, cooperação, entre outros.

Plataformas focadas no suporte a enxames (do inglês, *swarm*) são a tendência atual. A ideia é fortemente inspirada nas soluções apresentadas pela natureza. Enxames de abelhas, cardumes de peixes e exércitos de formigas se unem para a realização de tarefas extraordinárias. Muitas vezes a distinção entre coletivo e indivíduo se torna tênue e o grupo se comporta como uma só entidade. A aplicação da ideia de usar *swarms* de agentes para o cumprimento de um objetivo tem inúmeros exemplos de sucesso no meio natural, e seu uso tem sido explorado também em aplicações de robótica móvel.

Algumas plataformas focam no baixo custo de hardware, como as apresentadas por (COUCEIRO et al., 2011) e (ROCCO; GALA; ULIVI, 2013). Visam, assim, explorar e viabilizar a multitude de indivíduos através da redução do custo individual. Outras plataformas buscam disponibilizar recursos mais avançados para cada robô, limitando, assim, o número total de indivíduos devido à elevação de seu custo, como a plataforma apresentada por (ROCKEL, 2011).

Este projeto visa apresentar uma plataforma de tamanho intermediário. Isso permite o desenvolvimento de algoritmos de cooperação e coordenação de múltiplos robôs, porém possibilita uma diversidade de aplicações e desenvolvimento *solo*, favorecendo também o estudo de diversos métodos e tecnologias componentes do robô. Em plataformas excessivamente minimalistas, o desenvolvimento de sensores, controladores, e mapeamento fica muitas vezes limitado e restrito. Assim, propõe-se uma plataforma que disponha de recursos satisfatórios a cada indivíduo, porém sem tornar inviável a escalabilidade como *swarm*.

Projetos de pesquisa e desenvolvimento em robótica móvel e robótica autônoma agregam diversas áreas de engenharia e de computação. De característica multidisciplinar, estes projetos tipicamente requerem a integração de múltiplos subsistemas, tal como sistemas mecânicos, sistemas de controle, sistemas de visão computacional, sistemas de tomada de decisão, entre outros. Dadas as dificuldades de desenvolver projetos que cruzam diversas áreas de estudo, modularizarão e encapsulamento são fatores recorrentes no

desenvolvimento de plataformas de robótica móvel e autônoma.

Modularizarão consiste em desenvolver o sistema como um conjunto de módulos fechados, de forma que o funcionamento interno de um módulo não depende do funcionamento interno dos demais módulos. Encapsulamento consiste em ocultar a lógica interna de cada módulo, de forma que apenas sua interface de entrada e saída fique visível para os outros módulos. Juntos, encapsulamento e modularizarão permitem que se trabalhe componentes do todo de forma isolada. Isso permite distribuir o desenvolvimento de um mesmo projeto ao longo do espaço e do tempo. Diferentes departamentos e centros de pesquisa podem desenvolver cada módulo a seu tempo, uma vez que se abstrai o funcionamento interno dos módulos alheios. Adicionalmente, um módulo pode ser melhorado ou trocado sem que seja necessária qualquer alteração nos demais módulos.

Uma abordagem modularizada permite então o desenvolvimento de pesquisas de áreas especializadas sobre cada módulo do sistema. Uma arquitetura modular pode ser a base de variados projetos de desenvolvimento. Enquanto um grupo explora métodos concorrentes para controle de locomoção, outro grupo explora métodos de visão computacional sobre a mesma plataforma.

Este projeto propõe uma plataforma cujo objetivo é ser uma base a partir da qual projetos de diferentes áreas podem ser desenvolvidos. Projetada com os princípios de modularidade e encapsulamento, esta plataforma permite que se abstraia o funcionamento dos demais componentes e se foque no desenvolvimento de partes isoladas. Ainda que aqui se proponham detalhes de funcionamento de todos os módulos que constituem a plataforma, estes podem ser facilmente trocados e melhorados.

Um levantamento bibliográfico é apresentado na Parte I, descrevendo brevemente as principais tecnologias disponíveis para a construção da plataforma proposta (Capítulo 1), além das teorias que fundamentam sua construção e os métodos propostos (Capítulo 2 e Capítulo 3). A Parte II descreve a arquitetura proposta do ponto de vista de sistemas e os módulos componentes. O hardware e a arquitetura de controle de baixo nível, que implementa a camada de locomoção, é apresentado no Capítulo 4. Seu meio de sensoriamento primário, a visão computacional por meio de reconstrução estéreo, é apresentado no Capítulo 5. Por fim, testes sobre o modelo implementado são apresentados para validação da arquitetura proposta (Capítulo 6).

Parte I

Fundamentação Bibliográfica

1 Tecnologias de suporte à plataforma proposta

Este capítulo resume as variadas tecnologias disponíveis para a construção da plataforma de pesquisa proposta. Adicionalmente, são apresentadas tecnologias concorrentes ou equivales, que podem ser utilizadas alternativamente. Cada ferramenta apresenta diferentes benefícios, desvantagens e custos. A escolha de determinada tecnologia e a comparação dos resultados obtidos a partir de diferentes escolhas são dignos de investigação. São analisadas tecnologias de processamento embarcado (seção 1.1), comunicação sem fio (seção 1.2), sistemas atuadores (seção 1.3), sistemas sensores (seção 1.4) e kits estruturais (seção 1.5).

1.1 Plataformas de processamento embarcado

Nesta seção é apresentado um breve levantamento elencando os principais tipos de ambientes de desenvolvimento embarcado disponíveis no mercado. São destacadas vantagens e desvantagens de diferentes arquiteturas e é feita uma análise focada em sua aplicação à implementação de agentes robóticos móveis. Placas com microcontroladores (MCU, do inglês: *microcontroller unit*) (subseção 1.1.1), microprocessadores (MPU, do inglês: *microprocessing unit*) (subseção 1.1.2), unidades de processamento gráfico (GPU, do inglês: *graphic processing unit*) (subseção 1.1.3), arranjos de portas programáveis por campo (FPGA, do inglês: *field-programmable gate array*) (subseção 1.1.4) e plataformas híbridas (subseção 1.1.5) têm se mostrado cada vez mais acessíveis ao consumidor final, e portanto têm se apresentado como componentes viáveis para implementação de agentes robóticos de baixo custo.

1.1.1 Placas de desenvolvimento com MCU

A recente popularização de plataformas embarcadas de desenvolvimento com micro controladores para prototipação rápida e de baixo custo foi impulsionada e financiada pelo surgimento de um novo mercado de desenvolvedores leigos ou semileigos. A fabricação em massa e atuação de uma vasta comunidade de desenvolvedores fomentou o desenvolvimento de amplas bases de conhecimento aplicado sobre questões práticas e de implementação. Adicionalmente, esse movimento favoreceu a criação e o aprimoramento de bibliotecas especializadas que implementam rotinas de software para estas plataformas, cobrindo desde interfaceamento de comunicação serial até processamento de sinais. O fácil acesso, baixo custo de hardware e de aprendizado são os principais pontos positivos para o uso de plataformas de micro controladores no desenvolvimentos de projetos de robótica móvel.

Em contrapartida, tais plataformas muitas vezes restringem o poder de desenvolvimento e a flexibilidade do projeto. Isso decorre do fato de se estar preso a rotinas pré-definidas. Outras vezes, o desconhecimento das reais limitações e da qualidade das bibliotecas disponíveis pode se um fator negativo para o projeto. Em muitos casos não há garantia de que as rotinas implementam as funcionalidades a que se propõem, ou simplesmente o fazem de maneira ineficiente. Por fim, para alguns projetos, as limitações de memória e de processamento das plataformas podem ser um fator crucial, sendo necessário recorrer a outros tipos de plataforma.

Podemos citar uma ampla variedade de soluções dessa categoria oferecidas no mercado atual. Por um lado temos as plataformas abertas, como Arduino e seus variantes: Freeduino, Seeeduino e outros, que contam com ampla base de usuários (ARDUINO, 2016). Por outro lado temos as plataformas fechadas de prototipação, que são geralmente oferecidas pelos próprios fabricantes dos MCUs, tais como Freescale (FREESCALE, 2016) e PIC (MIKROELEKTRONIKA, 2016). A Figura 1 apresenta um dispositivo de desenvolvimento baseado em MCU.

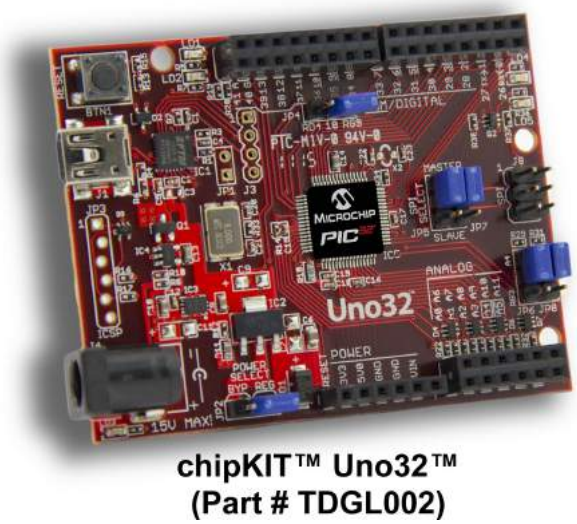


Figura 1 – Plataforma de desenvolvimento ChipKIT baseada no MCU PIC32 (MICROCHIP, 2016a).

1.1.2 Placas de desenvolvimento com MPU

Placas de desenvolvimento dotadas de microprocessadores geralmente oferecem memória e processamento capazes da execução de um sistema operacional (SO) típico de computadores de mesa. Comumente seguem a abordagem de computação com reduzido conjunto de instruções (RISC, do inglês: *reduced instruction set computing*), em sua maior parte com MPUs ARM. Essa plataforma se beneficia da disponibilidade e maturidade de SOs abertos voltados para processadores embarcados. Atualmente dispõe-se de distribuições estáveis de linux e outros SOs baseados em unix, tais como Debian (DEBIAN, 2016)

e Ubuntu (ELINUX, 2016). Isso permite o uso de todo o arcabouço de bibliotecas e *frameworks* existentes para arquiteturas x86 nesses dispositivos. Assim, todo o *know-how* e as ferramentas legadas e maduras das plataformas não-embarcadas se tornam disponíveis com pouca ou nenhuma adaptação. Bibliotecas para processamento de imagens, visão computacional, sistemas gerenciadores de bancos de dados, além do suporte nativo de muitos SOs a interfaces de rede e comunicação fazem parte desse elenco.

Estas plataformas também facilitam o desenvolvimento e a prototipação com relação à portabilidade de software. Sistemas desenvolvidos para uma plataforma podem ser facilmente portados para outras plataformas. Pode-se também desenvolver e executar o software em computadores de mesa e então portar para o dispositivo embarcado. Além da superioridade de processamento em relação às plataformas com MCU, plataformas com MPUs permitem o uso de dispositivos desenvolvidos para sistemas x86, tal como modems, dispositivos de interface humana (mouse, teclado, etc), *webcams*, adaptadores *wireless*, entre outros, com *drivers* maduros e estáveis.

Por outro lado, em alguns casos, o custo, o tamanho e o consumo de energia podem ser um fator crítico, favorecendo plataformas com MCUs. O mercado atual oferece uma ampla variedade de plataformas com MPUs, tais como Texas Instruments BeagleBone (BEAGLEBONE, 2016), Intel Edson (INTEL, 2016c), Raspberry Pi (RASPBERRYPI, 2016), entre outras. A Figura 2 apresenta um dispositivo de embarcado baseado em MPU Intel Atom.

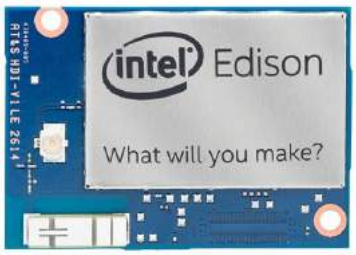


Figura 2 – Plataforma Intel Edison baseada no MPU Intel Atom (INTEL, 2016b).

1.1.3 Sistemas com GPU

GPUs têm possibilitado a exploração de computação massivamente paralela a nível de software. A proliferação de placas gráficas e os pesados investimentos observados no setor nas últimas décadas aceleraram o desenvolvimento da tecnologia. Várias aplicações em robótica podem se beneficiar com o uso de processamento paralelo, tais como processamento de visão computacional, filtragem de sinais, planejamento de rotas, mapeamento e simulações em tempo real.

Sistemas com GPU são tradicionalmente centralizados em um MPU e têm a GPU como coprocessador. Isso ocorre porque muitas rotinas, por sua natureza serial, não se

beneficiam da paralelização oferecida pelas GPUs. Assim, sistemas de software podem ser facilmente portados de plataformas com MPUs para plataformas MPU-GPU. Estes fatores tornam atrativas as plataformas com GPU, por possibilitar o aproveitamento de sistemas e ferramentas legadas, e ainda se beneficiar da computação paralela.

Como desvantagens desse tipo de arquitetura, podemos citar o custo, o tamanho das placas, que são geralmente maiores que as placas que portam apenas uma MPU, e o consumo de energia, que pode ser um fator limitante em determinados projetos. Encontra-se atualmente grande variedade de fabricantes de plataformas embarcadas com GPU, com destaque para o kit de desenvolvimento NVIDIA Jetson TK1, que é oferecido pelo próprio fabricante do GPU (NVIDIA, 2016b), e o Gizmo 2, que traz um GPU AMD (GIZMOSPHERE, 2016). A plataforma Jetson TK1 (Figura 3) é capaz de executar um SO Ubuntu completo. Possui o SoC Tegra TK1, que contém a GPU Kepler GK20a e um MPU com quatro núcleos ARM e um núcleo de baixo consumo de energia. A GPU Kepler GK20a tem capacidade de processamento nominal de 326 Giga Flops e os núcleos ARM trabalham a 2.32 GHz por padrão (NVIDIA, 2016a).

As principais características de hardware da plataforma Jetson TK1 são:

- 2GB DRAM DDR3L 933MHz
- 16GB de armazenamento interno eMMC4.51
- mini-PCIe
- 1 USB 3.0
- 1 micro-USB 2.0
- HDMI
- RS232
- Entradas de áudio
- Ethernet 10/100/1000
- SATA
- 7 GPIOs

1.1.4 Placas de desenvolvimento com FPGA

FPGAs são dispositivos semicondutores que permitem a implementação de circuitos digitais arbitrários com grande flexibilidade. Sua arquitetura garante um comportamento muito próximo daquele obtido em implementações em circuitos integrados específicos de

Figura 3 – Plataforma MPU-GPU Nvidia Jetson-TK1 (NVIDIA, 2016a).

aplicação (ASIC, do inglês: *application-specific integrated circuit*), porém com os benefícios de prototipação rápida, reduzido ciclo de desenvolvimento e dramática redução de custos para a produção em pequenas quantidades. A Figura 4 apresenta um comparativo dos ciclos de desenvolvimento de baseados em FPGA e ASIC.

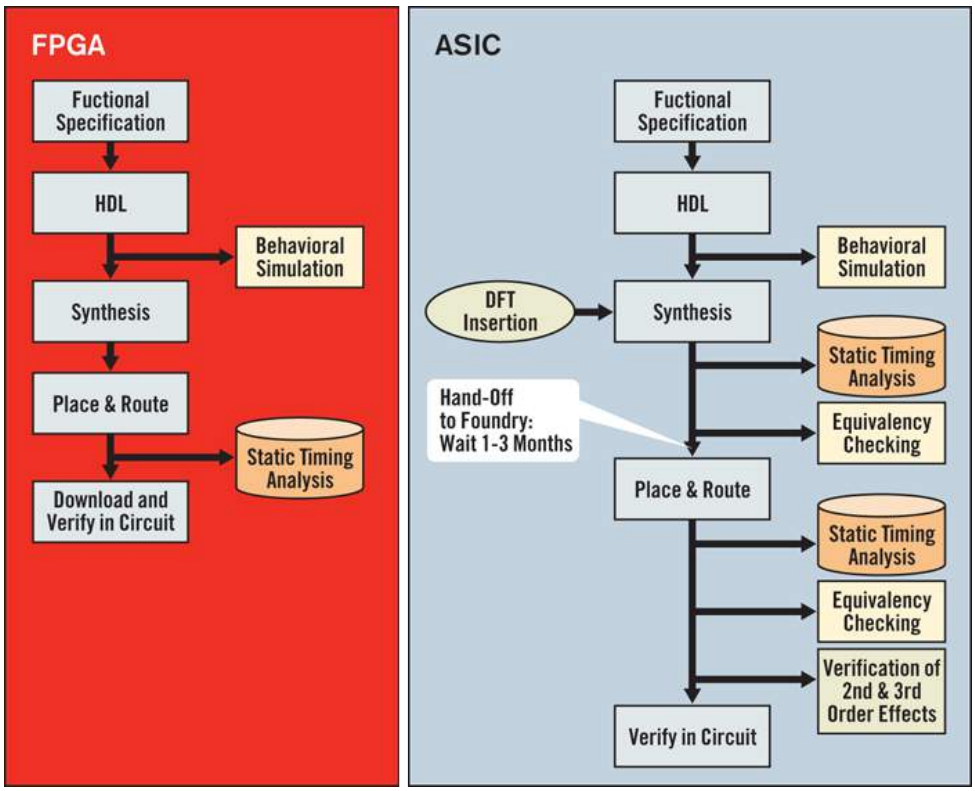


Figura 4 – Comparativo dos ciclos de desenvolvimento FPGA e ASIC (XILINX, 2015a).

Os dispositivos dispõem primariamente de blocos lógicos configuráveis (CLB, do inglês, *configurable logic blocks*), bancos de registradores, blocos de memória de acesso aleatório (BRAM, do inglês: *block random-access memory*), blocos de entrada e saída configuráveis (IOB, do inglês, *input/output block*), unidades aritméticas e blocos de processamento digital de sinais (DSP, do inglês, *digital signal processing*).

A programação da FPGA ocorre, então, a nível de hardware na forma de redirecionamento de conexões entre estes elementos, além da configuração dos CLB, capazes de implementar tabelas lógicas arbitrárias, emulando lógica combinacional. Adicionalmente, dispositivos FPGA também são dotados de gerenciadores de *clock* digital (DCM, do inglês, *digital clock manager*) que permitem a configuração de um ou vários sinais de *clock* em frequências arbitrárias para a sincronização dos sistemas digitais implementados. A Figura 5 apresenta a estrutura básica de uma FPGA.

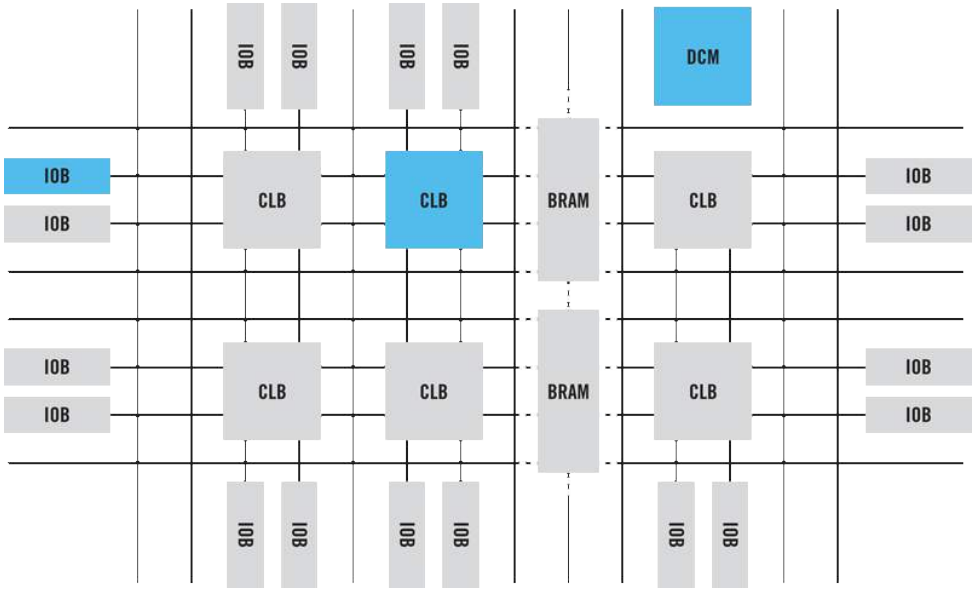


Figura 5 – Estrutura interna de um dispositivo FPGA (XILINX, 2015b).

Os blocos CLB são a estrutura fundamental em uma FPGA. Diferentes dispositivos contém blocos CLB em diferentes quantidades. Tais blocos contém matrizes de comutação configuráveis, circuitos multiplexadores, blocos RAM e *flip-flops*. A Figura 6 apresenta a estrutura básica de um CLB.

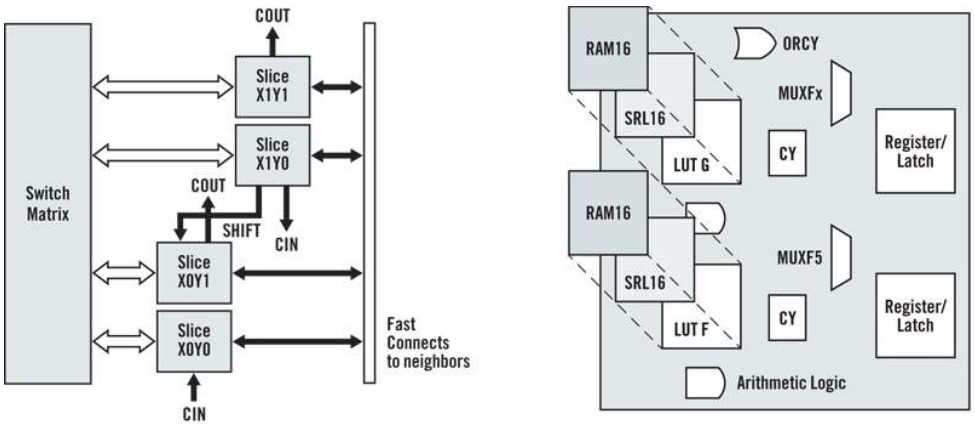


Figura 6 – Estrutura interna de um CLB (XILINX, 2015b).

Associada a uma ampla gama de entradas e saídas igualmente redirecionáveis, a

FPGA permite a realização de sistemas de elevado paralelismo ou até mesmo a implementação de múltiplos sistemas no mesmo dispositivo trabalhando de maneira verdadeiramente paralela – diferentemente do pseudo-paralelismo oferecido pela intercalação de processos em sistemas de software.

Esse paradigma, assim, permite uma abordagem completamente independente de software e de arquiteturas computacionais rígidas. Algoritmos são transformados em máquinas de estado, garantido tempos de execução amarrados aos ciclos de *clock*. Com isso, FPGAs abrem uma nova gama de possibilidades para a prototipação rápida em sistemas embarcados, permitindo a agregação de diversos processos onde a temporização e o paralelismo são essenciais, tais como a geração de sinais, implementação de protocolos de comunicação seriais e paralelos, implementação de filtros digitais em tempo real, entre outros.

Fatores limitantes nesses sistemas são as quantidades de blocos aritméticos, lógicos e de memória disponíveis. Muitas aplicações, tais como processamento de imagens e vídeo, exigem grandes quantidades de memória e muitas vezes requerem o uso de unidades de memória externas ao *chip* FPGA. Isso frequentemente se apresenta como uma barreira à viabilidade da implementação porque os gargalos da comunicação serial entre a FPGA e os dispositivos de memória limitam seus ganhos de desempenho em paralelismo. Adicionalmente, tais dispositivos apresentam preços mais elevados que sistemas com MPUs. Assim, FPGAs são ferramentas excelentes para certas aplicações, porém inadequadas para outras.

Atualmente encontra-se grande diversidade de kits de desenvolvimento e placas com dispositivos FPGA, de diferentes dimensões e formatos. Dispõe-se desde placas pequenas, no formato de pacote dual em linha (DIP, do inglês: *dual in-line package*), como a Xilinx Cmod S6 (DIGILENT, 2016a), placas de tamanho médio, como a Xilinx Nexys 4 (DIGILENT, 2016b), até placas grandes, como a Altera Arria V GX DK (ALTERA, 2016). Tais placas de desenvolvimento oferecem uma rica variedade de dispositivos acessórios integrados, tais como conversores analógico/digital e digital/analógico, sensores de temperatura, pressão, transdutores de áudio, além de conectores de diversos padrões, tais como barramento serial universal (USB, do inglês, *universal serial bus*), Ethernet, arranjo gráfico de vídeo (VGA, do inglês, *video graphics array*), cartões de memória, entre outros. A Figura 7 apresenta como a Xilinx Cmod S6.

1.1.5 Placas de desenvolvimento híbridas

Têm se tornado difundidas também plataformas de desenvolvimento híbridas. Podemos dividir as plataformas híbridas e duas classes, segundo o nível de integração. No primeiro temos placas com múltiplos dispositivos discretos, sendo eles MPUs, GPUs, entre outros. Como exemplo dessa classe podemos citar a DUO-VPC-LV2, uma placa embarcada com dois MPUs ARM e um GPU (LABORATORIES, 2016a).

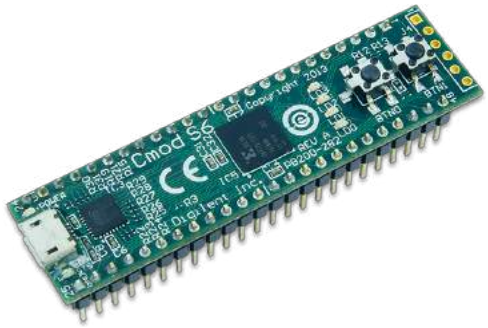


Figura 7 – Cmod S6: dispositivo DIP baseado em um FPGA Xilinx (DIGILENT, 2016a).

No segundo caso, temos placas que trazem múltiplos dispositivos integrados em um único semicondutor. Como exemplo, podemos citar os kits Digilent ZYBO Zynq (DIGILENT, 2016f) e Digilent ZedBoard Zynq (DIGILENT, 2016e). Ambos portam um sistema em chip (SoC, do inglês: *system on a chip*) Xilinx Zynq que integra um MPU ARM a um dispositivo FPGA.

Sistemas híbridos comumente portam uma ampla variedade de portas de comunicação, sensores e dispositivos de interface humana, possibilitando fácil integração com outros sistemas. Por outro lado, as placas disponíveis são geralmente mais caras, maiores e mais robustas, o que pode ser um fator negativo para certos projetos. A Figura 8 apresenta como o Digilent ZYBO Zynq.

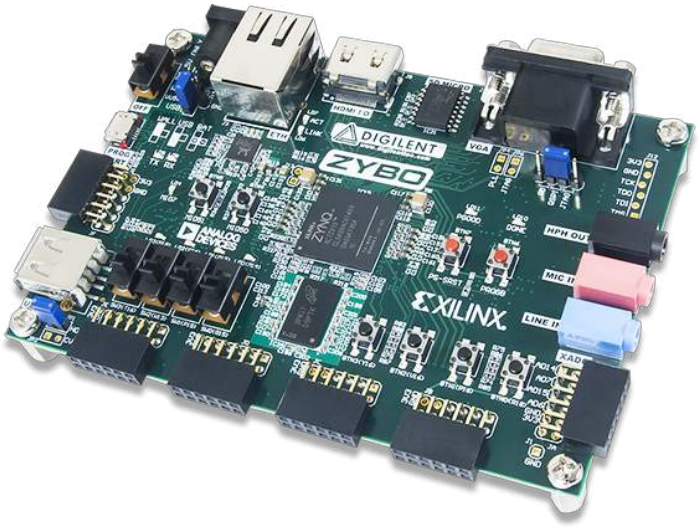


Figura 8 – Plataforma de desenvolvimento híbrida MPU-FPGA Digilent ZYBO Zynq (DIGILENT, 2016f).

1.2 Soluções de comunicação sem fio embarcadas

Comunicação sem fio tem dois papéis centrais em robótica móvel. O primeiro é na comunicação entre o robô e uma estação-base, em momento de execução, para telemetria e telecomando. A telemetria permite receber dados sobre o robô e o ambiente à sua volta, tal como leitura de sensores, estados de operação, fatores de desempenho. O telecomando permite o envio de instruções para o robô, para alternar entre modos de operação, para definição dinâmica de parâmetros e para avaliação e teste de desempenho. O segundo é o de permitir a intercomunicação entre robôs para a realização de tarefas cooperativamente.

Nesta sessão apresentamos cinco diferentes tecnologias de comunicação sem fio que se apresentam como opções viáveis para diferentes necessidades de projeto em robótica móvel autônoma. Duas definições se fazem necessárias para esta análise:

Comunicação half-duplex permite o envio e recebimento de dados em ambos os sentidos, não simultaneamente. Envio e recebimento ocorrem em tempos diferentes.

Comunicação full-duplex permite o envio e recebimento de dados em ambos os sentidos, simultaneamente. Envio e recebimento ocorrem ao mesmo tempo.

1.2.1 Sistemas de comunicação de mão única com modulação por amplitude

Modulação por amplitude é a opção mais rudimentar e de baixo custo para comunicação digital sem fio. Seu protocolo permite a implementação de sistemas de recepção e transmissão com apenas componentes eletrônicos discretos simples, como capacitores e transistores. Seguindo esse paradigma, muitos sistemas de transmissão a custos acessíveis são capazes de garantir um enlace de comunicação funcional. Estes sistemas, porém, apresentam instabilidade e susceptibilidade a ruído. Adicionalmente, são tipicamente voltados para comunicação de mão única, não permitindo sequer canais half-duplex. Enquanto algumas aplicações de telemetria de baixo custo podem encontrar nestes sistemas uma solução acessível, projetos que exigem comunicação de mão dupla exigem outros tipos de sistema.

1.2.2 Sistemas half-duplex

Aplicações que exigem comunicação de mão dupla podem se beneficiar de sistemas half-duplex. Tais sistemas apresentam comumente um nível mais elevado de complexidade que os sistemas de mão única, oferecendo ao usuário modos de operação programáveis. Adicionalmente, soluções dessa categoria oferecem certos níveis de sofisticação, tais como envio de mensagens por pacotes, reenvio e confirmação. A complexidade extra e a interface de configuração podem ser um fator negativo, a depender das exigências de projeto. A

exemplo dessa categoria, temos o Nordic Semiconductor nRF24L01+, de grande disponibilidade nas prateleiras de fornecedores especializados e baixo custo (NORDIC, 2016). Opera na faixa de 2.4 GHz, permitindo ao usuário a parametrização de largura de banda, canal, ganho e tratamento de pacotes através de interface periférica serial (SPI, do inglês: *serial peripheral interface*). A Figura 9 apresenta um dispositivo de comunicação half-duplex baseado no transceptor nRF24L0.

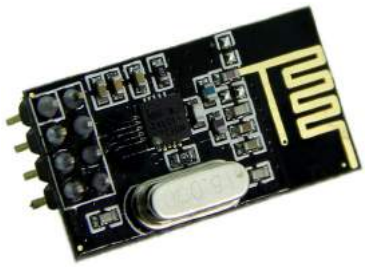


Figura 9 – Dispositivo de comunicação half-duplex baseado no transceptor nRF24L01 (NORDIC, 2016).

1.2.3 Sistemas full-duplex

Sistemas full-duplex permitem a implementação de canais de mão dupla, habilitando telecomando e telemetria simultaneamente. Em adição, alguns protocolos permitem a implementação de redes *mesh*, onde cada nó da rede se comunica diretamente com os nós adjacentes, não havendo necessariamente um nó mestre ou sequer diferenciação entre nós.

Redes Bluetooth permitem comunicação ponto a ponto full-duplex e são voltadas para curtas distâncias, implementando redes de área pessoal (PAN, do inglês: *personal area network*). Módulos embarcados que implementam o protocolo permitem o estabelecimento de enlaces de comunicação serial transparentes, incorporando o gerenciamento de conexão e a troca de pacotes. O módulo HC-06 é um exemplo de solução de baixo custo que implementa uma conexão serial full-duplex transparente (MIKROKOPTER, 2016). Realizado o pareamento entre dois dispositivos, o usuário tem à sua disposição uma interface de transmissor/receptor assíncrona universal (UART, do inglês: *universal asynchronous receiver/transmitter*).

Redes Wifi permitem conexões full-duplex com bandas e alcances mais amplos que redes Bluetooth e são consideradas redes de área local (LAN, do inglês: *local area network*) (IEEE, 2016a). Com wifi pode-se mais facilmente integrar dispositivos embarcados, como robôs móveis, a redes estruturais e à internet. A intercomunicação entre diversos dispositivos é também facilitada através da adoção do protocolo IP. Estas vantagens, porém, têm o custo de hardware mais elevado, geralmente associado também a um consumo mais elevado de energia. Há grande variedade de soluções

para wifi embarcado, a exemplo dos módulos Microchip RN171XV (MICROCHIP, 2016b), Digi XBee® Wi-Fi (DIGI, 2016c), Digilent PmodWiFi (DIGILENT, 2016d) e WiFi Intel 7260 HMW (INTEL, 2016a).

Redes Zigbee também são caracterizadas como PAN, porém voltadas para a implementação de redes *mesh* heterogêneas (IEEE, 2016b). As redes foram concebidas originalmente para intercomunicação entre dispositivos domésticos, tais como televisores, sistemas de aquecimento e condicionamento de ar e controles-remotos. Devido às suas origens, o protocolo é caracterizado por diferenciação entre nós da rede, formando redes heterogêneas hierárquicas. O principal nó da rede é o nó coordenador (do inglês, *Coordinator*), que estabelece a rede e registra informações essenciais para seu funcionamento. Outros nós centrais, chamados de nós roteadores (do inglês, *Router*), repassam mensagens entre si com a finalidade de entrega-las a dispositivos finais (do inglês, *End Device*). *End Devices* permanecem em modo de espera até que sejam acordados pelos roteadores para troca de mensagens. Dessa forma, é possível otimizar a o consumo de energia destes dispositivos finais. No ambiente doméstico, estes dispositivos têm o consumo de energia como requerimento crítico, a exemplo de controles remotos e outros dispositivos de interface humana. Assim, a carga energética é transferida dos nós-folha para os nós centrais, que tipicamente fazem parte da infraestrutura do ambiente e estão conectados à rede elétrica. Ainda que voltado inicialmente para o ambiente doméstico, este protocolo encontrou diversas aplicações científicas e industriais, para redes de sensores e redes para controle e automação. Assim como os módulos wifi, também encontra-se grande variedade de módulos embarcados Zigbee, tais como Digilent PmodRF2 (DIGILENT, 2016c) e Digi XBee® ZigBee (DIGI, 2016d).

Redes DigiMesh são definidas por um protocolo proprietário para redes *mesh* homogêneas da Digi International Inc. (DIGI, 2016a). Oferecido como alternativa ao Zigbee, o protocolo DigiMesh permite implementar redes menos centralizadas, onde todos os nós desempenham papéis iguais, recebendo e roteando mensagens. É especialmente interessante para a implementação de *swarms* de agentes robóticos por caracterizar uma arquitetura de rede *mesh* mais simples e mais robusta. Por não concentrar a responsabilidade de roteamento em nós centrais, estas redes apresentam maior tolerância a falhas e maior flexibilidade que as redes ZigBee. Como exemplo de módulo DigiMesh de baixo custo temos o Digi XBee® 802.15.4 (DIGI, 2016b).

A Figura 10 apresenta um comparativo entre as redes *mesh* ZigBee e as redes *mesh* DigiMesh.

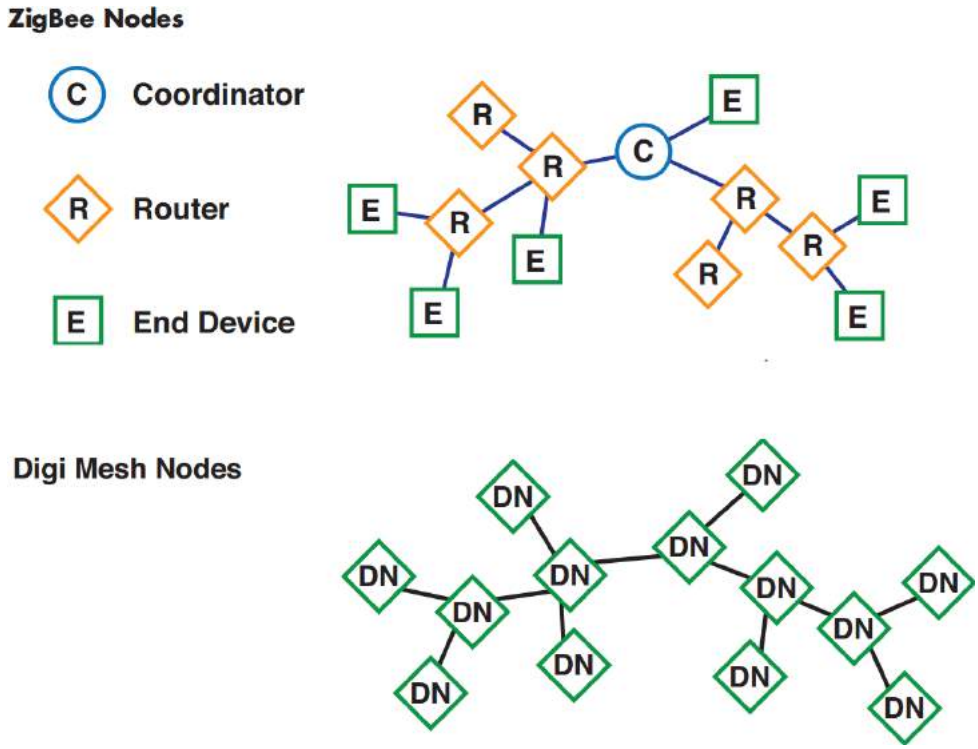


Figura 10 – Comparativo entre as redes *mesh* heterogêneas ZigBee e as redes *mesh* homogêneas DigiMesh (DIGI, 2016a).

1.3 Sistemas Atuadores

Esta seção apresenta algumas definições relevantes referentes aos principais componentes disponíveis para a implementação de interfaces de atuação para robótica móvel de baixo custo.

1.3.1 Motores elétricos de corrente contínua

Atuadores que exercem papel central em robótica móvel são os motores elétricos de corrente contínua. Podemos dividi-los em quatro classes gerais: motores com bucha, motores sem bucha, motores de passo e servo-motores.

Motores com bucha são caracterizados por bobinas acopladas ao ao rotor, exigindo buchas para conexão e comutação dos terminais da bobina às conexões elétricas externas. Motores com bucha apresentam o mais baixo custo, comparados às outras categorias, porém são, em geral, menos eficientes, mais ruidosos e têm vida útil menor devido ao atrito gerado pelas buchas.

Motores sem bucha são caracterizados por bobinas acopladas ao ao estator e ímãs permanentes fixados ao rotor. Essa configuração é mais eficiente, silenciosa e longeva, por não exigir conexão elétrica entre partes móveis. Por outro lado, requerem maior complexidade de controle, uma vez que é exigido um preciso gerenciamento síncrono

das fases em cada bobina. Tais motores são geralmente mais caros porque vêm acompanhados dos controladores eletrônicos. São comumente usados em robótica aérea, onde eficiência energética é mais crítica que em robótica terrestre.

Motores de passo se assemelham aos motores sem bucha por apresentarem bobinas acopladas ao estator e ímãs permanentes acoplados ao rotor. Seu funcionamento, porém, é mais simples, de forma que as fases necessárias para seu funcionamento são ondas quadradas. Motores de passo permitem grande precisão angular e podem facilmente controlados com sistemas de controle de laço aberto. São comumente adotados em robótica de manipuladores de baixo custo. Seu rendimento é geralmente inferior comparado às demais categorias e seu tamanho também é proporcionalmente elevado.

Servo-motores são sistemas constituídos de motores com bucha acoplados a sistemas de controle de posição com retroação embarcados, geralmente baseados em codificadores resistivos rotacionais. Permitem fácil operação, uma vez que já dispõem de um ciclo fechado de controle e são usados comumente em manipuladores e atuadores de baixo custo.

1.4 Sensores

Esta seção apresenta uma breve descrição dos principais sensores acessíveis no mercado atual para consumidores finais que são relevantes para robótica móvel.

1.4.1 Codificadores rotacionais

Codificadores rotacionais, ou codificadores de eixo, permitem a medida de posição e velocidade angular em eixos e rodas. Podemos dividir esta categoria em três classes:

Codificadores resistivos são constituídos por um divisor de tensão com um resistor e um potenciômetro rotacional. O potenciômetro é acoplado ao eixo e a voltagem de saída do divisor de tensão indica a posição angular do mesmo. São dispositivos de baixo custo, muito comuns em servo-motores. Seu principal ponto negativo é a falta de precisão, em relação às outras tecnologias de codificadores, e o desgaste devido ao contato.

Codificadores ópticos permitem a leitura de posição digitalmente. Seu esquema básico é de um disco vazado acoplado ao eixo. De cada lado do disco dispõe-se um dispositivo emissor de luz – geralmente um diodo emissor de luz (LED, do inglês: *light emitting diode*), e um fototransistor. A alternância entre pontos vazados e não vazados no disco gera uma onda quadrada no fototransistor, que indica a rotação. A contagem

dos períodos da onda permite que se avalie o deslocamento do eixo. Sistemas de duas fases permitem determinar também o sentido da rotação. São tipicamente mais robustos ao desgaste de longo prazo que os codificadores resistivos.

Codificadores magnéticos Seguindo um princípio semelhante ao dos codificadores ópticos, os codificadores magnéticos permitem mensurar o deslocamento angular pela formação de uma onda quadrada oriunda da indução magnética de uma coroa ferromagnética acoplada ao eixo do motor. O sistema também é robusto ao desgaste de longo prazo, porém susceptível à interferência de campos magnéticos intensos.

1.4.2 Dispositivos de medida inercial

Dispositivos de medida inercial, tais como acelerômetros e giroscópios, utilizam a inércia para aproximar orientação, deslocamento ou rotação. A combinação de giroscópios e acelerômetros em um único dispositivo resulta na chamada unidade de medida inercial (IMU, do inglês: *inertial measurement unit*), voltada para a mensuração de orientação e de velocidades angulares. Tais dispositivos inerciais são comuns em aviação e exploração espacial, tendo presença unânime em aviação comercial e veículos espaciais atuais. Sua miniaturização e implementação em *chips* de silício, porém, é recente, sendo possibilitada pela tecnologia de sistemas microeletromecânicos (MEMS, do inglês: *microelectromechanical systems*) (HSU, 2008). A Figura 11 apresenta um dispositivo integrado IMU para estimação de orientação triaxial. O dispositivo combina dados de um acelerômetros, um giroscópio e um magnetômetro utilizando um filtro de Kalman estendido (CHROBOTICS, 2016).

Figura 11 – IMU para estimação de orientação triaxial (CHROBOTICS, 2016).

1.4.3 Sensores de distância

Sensores de distância podem ser divididos quanto à sua natureza de operação: óptica ou sonar, e quanto à sua dimensão de cobertura: uni, bi ou tridimensional. Em se

tratando de sensores ópticos, podemos também dividi-los entre sensores ativos ou passivos.

Sensores ópticos utilizam luz para aproximar distâncias a superfícies vizinhas. Sensores passivos são detalhados na subseção 1.4.4. Sensores ativos emitem e captam luz (geralmente *lazer*), medindo o tempo de ida e volta. Arranjos de vários sistemas sensores ópticos unidimensionais e sistemas de varredura permitem a mensuração de múltiplas distâncias simultaneamente, gerando percepção bidimensional ou tridimensional das superfícies à sua volta. Dispositivos atuais permitem adquirir dados com precisão de milímetros e alcance de centenas de metros, porém a custos elevados. O consumo de energia, a complexidade mecânica, a existência de partes móveis e o tamanho ainda é um fator proibitivo para o uso sistemas ativos multidimensionais em robótica móvel. A exemplo de sensor bidimensionais e tridimensionais por varredura podemos citar PulsedLight LIDAR-Lite (PULSEDLIGHT, 2016) e Velodyne HDL-64E (VELODYNE, 2016), respectivamente.

Sensores sonares emitem e captam som, geralmente em frequências inaudíveis, registrando o tempo de ida e volta para aproximar distância. Por limitações da tecnologia, sensores sonares comuns permitem apenas medidas unidimensionais. Ampla variedade de dispositivos são encontrados para aplicações industriais, a exemplo da linha MaxBotix, que cobre diferentes faixas de preços e limitações (MAXBOTIX, 2016).

1.4.4 Sensores ópticos passivos e visão computacional

Visão computacional trata de agregar diversos métodos para adquirir e extrair informação a partir de dados visuais. Este campo cruza diversas áreas da computação, incluindo processamento de imagens, inteligência computacional, reconhecimento de padrões e otimização. Sua aplicação à robótica móvel permite a obtenção de informações sobre as superfícies do ambiente, posicionamento e características de objetos adjacentes e estimação de posição, orientação e velocidade do robô em relação a um referencial externo. Sua aplicação ao processamento de imagens obtidas por câmeras permite a implementação de sistemas de sensoriamento de distância passivos. Tais sistemas permitem a mensuração tridimensional de distâncias sem a necessidade de emissão de *lasers*. É assim, uma opção que não envolve partes móveis, simples e de baixo custo em relação aos sistemas ópticos ativos.

DUO M (Figura 12) é um sistema integrado para visão computacional estéreo dotado de um par de câmeras monocromáticas idênticas comercializado pela Code Laboratories Inc. O dispositivo tem como áreas de aplicação visão computacional, interação homem-máquina, automotivo, militar, médico, navegação e campos relacionados. O sistema opera como periférico para computadores pessoais e plataformas embarcadas capazes de executar SOs Linux, Windows e MacOS. Se comunica com o computador *host* através de interface

USB. As câmeras possuem diferentes níveis de resolução, taxa de quadros por segundo (do inglês, Frames Per Second – FPS) configuráveis, e o tamanho (52x25x13 mm) e o peso (6,5 gramas) do conjunto são favoráveis para a construção de robôs móveis (LABORATORIES, 2016b).

Figura 12 – Sistema embarcado de visão estéreo DUO M (LABORATORIES, 2016b).

1.5 Kits estruturais para robótica

Kits de peças plásticas, ligas de alumínio ou de aço para a construção de estruturas para robôs são atualmente encontradas no mercado a preços acessíveis. Em diversos projetos, o uso de peças pré-fabricadas pode acelerar a prototipação, reduzir ciclos de implementação e cortar custos.

Cabe mencionar os kits oferecidos pela VEX Robotics, que têm escopo educacional, porém se mostram muito versáveis para a implementação de plataformas robóticas para pesquisa (VEX, 2016). A Figura 13 apresenta o conteúdo oferecido por kit estrutural VEX Robotics. Por permitir que as peças sejam facilmente cortadas, perfuradas e dobradas e garantir robustez estrutural, este kit foi adotado para a implementação da arquitetura proposta.

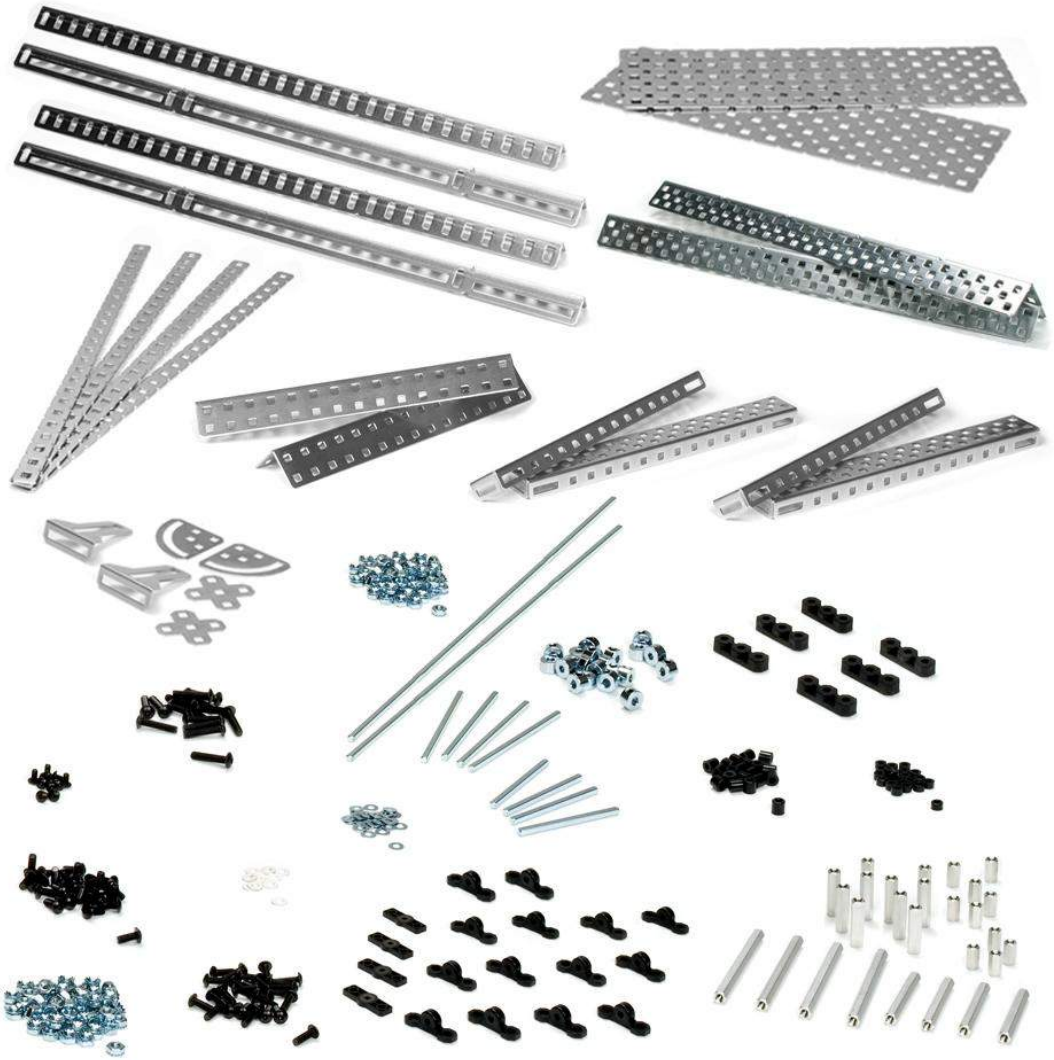


Figura 13 – Kit de metais estruturais e ferragens para robótica educacional VEX (VEX, 2016).

2 Redes neurais artificiais

2.1 Biomimética

Antes de introduzir os temas de redes neurais artificiais (RNA), algoritmos evolucionários, ou lógica *fuzzy*, é importante destacar o conceito que define a origem de ambas teorias, a biomimética. Biomimética é o uso e a adaptação de processos e mecanismos encontrados na natureza para a construção de sistemas e obras humanas. Muitos processos biológicos podem ser vistos como o resultado de um longo processo de otimização. O estudo de tais processos é comprovadamente uma fonte extensiva de inspiração para a solução de variados problemas de engenharia.

Várias áreas da computação e da engenharia se baseiam em bio-inspiração. O desenho de estruturas aero e hidrodinâmicas, novos materiais e estruturas são aplicações típicas. Áreas de mais alto nível de abstração, tais como a engenharia de sistemas de controle e de sistemas de classificação e de tomada de decisão também têm se beneficiado. Dente estas, podemos citar o desenvolvimento de teorias tais como as relacionadas ao aprendizado de máquina ou à computação evolucionária (PASSINO, 2005).

2.2 Introdução a RNA

Redes neurais artificiais são estruturas computacionais inspiradas em redes neuronais biológicas. São um representante clássico das estruturas computacionais para aprendizado de máquina. Adequadas e comumente utilizadas para aprendizado supervisionado, estas estruturas também permitem aprendizado por reforço e outras técnicas indiretas de aprendizado. O aprendizado supervisionado, assim como o aprendizado humano, se dá por meio de exemplos. O sistema é alimentado com diversos exemplos e o aprendizado é alcançado quando este se torna capaz de gerar saídas válidas para entradas ainda não apresentadas (generalização).

De maneira geral, o aprendizado em RNA ocorre da seguinte maneira: um conjunto de entradas e saídas correspondentes válidas é apresentado à RNA. Este conjunto, também chamado de conjunto de treinamento, é a base para o aprendizado e constitui o conjunto de exemplos válidos para o treinamento. Através de um algoritmo de aprendizagem, a rede vai se adaptar segundo os exemplos. Após a aprendizagem, espera-se que as relações inerentes entre as entradas e as saídas apresentadas tenham sido captadas pela rede. Por meio do processo de aprendizagem, a RNA vai de fato aproximar estas relações.

Em outras palavras, uma RNA é um aproximador de funções. Suponha que há um

conjunto de entradas $X = \{x_1, x_2, \ldots, x_p\}, x \in \mathbb{R}^I$ e um conjunto de saídas correspondentes $D = \{d_1, d_2, \ldots, d_p\}, d \in \mathbb{R}^K$. Suponha também que há uma relação inerente entre X e D representada por $f(x) : \mathbb{R}^I \to \mathbb{R}^K$. Através do processo de aprendizagem, a rede neural vai aproximar a função $f(x)$. Assim, pode ser usada para prever valores de saída a partir de entradas arbitrárias. RNA são comumente usadas como aproximadores globais de funções de múltiplas entradas e múltiplas saídas (KECMAN, 2001).

2.3 Origens: neurônios artificiais lineares

A teoria de aprendizado supervisionado de máquina por meio de estruturas inspiradas em redes neuronais tem suas origens no início da década de 1960. Foi nesse período que surgiram, a partir de campos distintos, dois modelos bio-inspirados capazes de aproximar funções lineares quaisquer.

2.3.1 ADALINE

O primeiro, publicado em 1960, foi batizado de neurônio adaptativo linear (ADALINE, do inglês: *adaptive linear neuron*) (WIDROW; HOFF et al., 1960). Oriundo do campo de processamento de sinais, sua função inicial era de proporcionar um mecanismo para obtenção de filtros lineares para cancelamento de ruído. A rede, constituída de uma camada de nós de processamento cuja função de ativação é proporcional, permite modelar qualquer função (transformação) linear de $\mathbb{R}^N \to \mathbb{R}^M$. A Figura 14 apresenta o modelo original de uma ADALINE de N entradas e M saídas.

Note que o elemento $N + 1$ não é de fato uma entrada, mas sim um valor constante unitário. Este elemento é chamado de fator de viés, compensação ou limiar e permite modelar hiperplanos que não necessariamente passam pela origem do espaço em questão. Como exemplo, no espaço $\mathbb{R}^3$, um plano é a transformação $f(x) : \mathbb{R}^2 \to \mathbb{R}^1$ definido por

$$z = w_1 x + w_2 y + w_3 \tag{2.1}$$

Assim, o elemento w_3 é o peso correspondente à entrada constante, 1, enquanto w_1 e w_2 correspondem às entradas factuais, x e y. Sem a entrada constante e seu peso, w_3, a capacidade de representação estaria restrita a planos que passam pela origem do espaço. É esse fator constante que permite que a transformação efetuada por $W = \{w_1, w_2, w_3\}$ represente qualquer plano em $\mathbb{R}^3$.

Uma rede ADALINE também pode ser representada matricialmente:

$$Wx = o \tag{2.2}$$

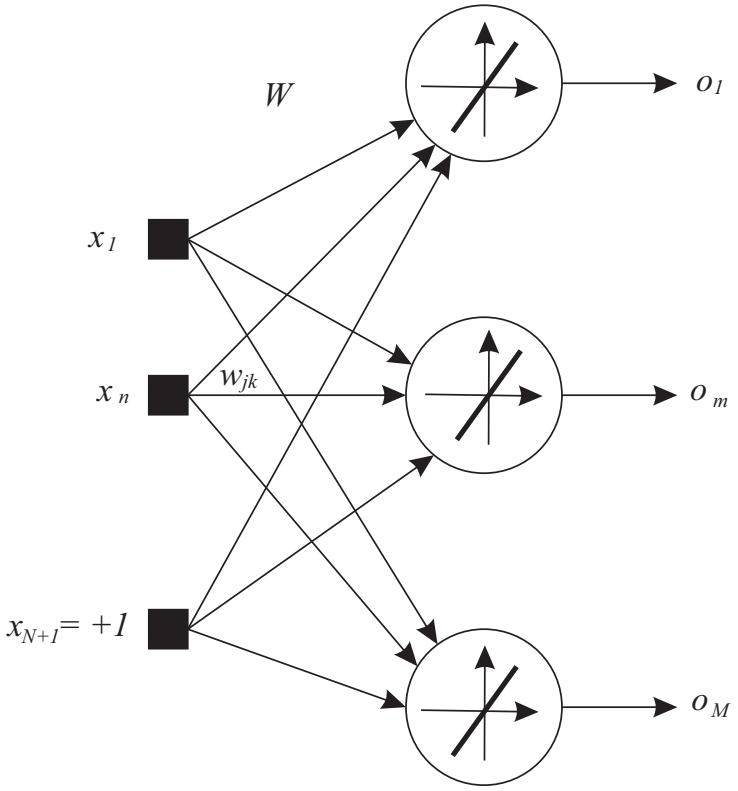


Figura 14 – Rede ADALINE.

Observe que a relação representada pela Equação 2.2 é a representação geral para uma transformação linear. De fato, é do que se trata uma ADALINE. Ao solucionar sistemas lineares, porém, dispõe-se comumente dos elementos W e o e procura-se determinar x. No caso do treinamento em ADALINE, dispõe-se de um conjunto de $X = \{x_1, x_2, \ldots, x_p\}, x \in \mathbb{R}^N$ e um conjunto $D = \{d_1, d_2, \ldots, d_p\}, d \in \mathbb{R}^M$ e deseja-se encontrar o conjunto de pesos ideais W^*, que melhor aproxima a relação linear entre as entradas X e as saídas D, resolvendo

$$WX = D \tag{2.3}$$

Como apresentado por Kecman (2001), W^* é obtido por

$$W^* = (XX^T)^{-1}XD = X^+D \tag{2.4}$$

Onde X^+ é a pseudo-inversa de X. Este método permite a obtenção imediata dos pesos ótimos, porém é um processo monolítico. Uma abordagem mais gradual associada ao uso de redes ADALINE é o chamado método de descida gradiente, um processo de otimização comum, não apenas para problemas lineares. Aplicado à ADALINE, há garantia de convergência para W^*, dado que a otimização ocorre em uma superfície convexa.

Este método iterativo consiste então em ajustar continuamente os pesos segundo o erro δ – a diferença entre as saídas esperadas, d, e as saídas fornecidas pelo sistema, o. Por esse motivo, o método também é chamado de regra delta. O passo de ajuste é definido por um escalar arbitrário η, também chamado de taxa de aprendizado. Dessa forma, os pesos são obtidos iterativamente:

$$W_{i+1} = W_i + \eta(d_i - o_i)x_i \tag{2.5}$$

$$W_{i+1} = W_i + \eta\delta x_i \tag{2.6}$$

2.3.2 Perceptron

O segundo modelo, chamado de perceptron, foi publicado por Rosenblatt (1962). Voltado para a percepção visual, foi desenvolvido para tratar do reconhecimento de padrões e classificação. Perceptrons são capazes de classificar padrões quaisquer, desde que estes sejam linearmente separáveis. A questão da separabilidade linear é representada na Figura 15. Enquanto uma rede baseada em perceptrons seria capaz de classificar os elementos dos grupos de (a) com total acurácia, o mesmo não pode ser esperado de (b).

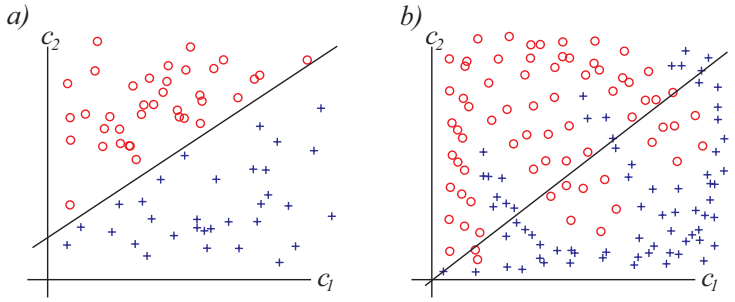


Figura 15 – Separabilidade. Os conjuntos em (a) são linearmente separáveis, enquanto os conjuntos em (b) não são.

O que ocorre, de fato, é que o processo de aprendizagem aplicado a uma rede linear de perceptrons é uma minimização dos erros, assim, espera-se que o sistema vai encontrar uma linha que melhor divide os dois grupos de modo a minimizar os erros de classificação. O fator que diferencia redes de perceptrons, como definidas originalmente por Rosenblatt e redes de ADALINE é o uso de funções de ativação *signum*. As funções *signum* se assemelham a uma função degrau, porém, são funções ímpares, i.e., $f(x) = -f(-x)$. São chamadas de *signum* porque para se gerar a saída leva-se em consideração apenas o sinal do valor de entrada:

$$signum(x) = \begin{cases} +1, \text{ se } x \geq 0 \\ -1, \text{ se } x < 0 \end{cases} \tag{2.7}$$

De maneira semelhante à ADALINE, as saídas de um perceptron também podem ser representadas matricialmente:

$$o = signum(Wx) \tag{2.8}$$

Em sua publicação original, Rosenblatt mostrou que a convergência de aprendizado também ocorre para redes de perceptrons e definiu um teorema de convergência. Em resumo, o teorema afirma que, dado um conjunto de treinamento com entradas e saídas cujo padrão é linearmente separável, o processo de aprendizado vai necessariamente classificar os dados em tempo finito. Coincidentemente, este referido processo de aprendizado é exatamente o mesmo associado à ADALINE e definido pela Equação 2.5. A Figura 16 apresenta um modelo de rede clássica de perceptrons.

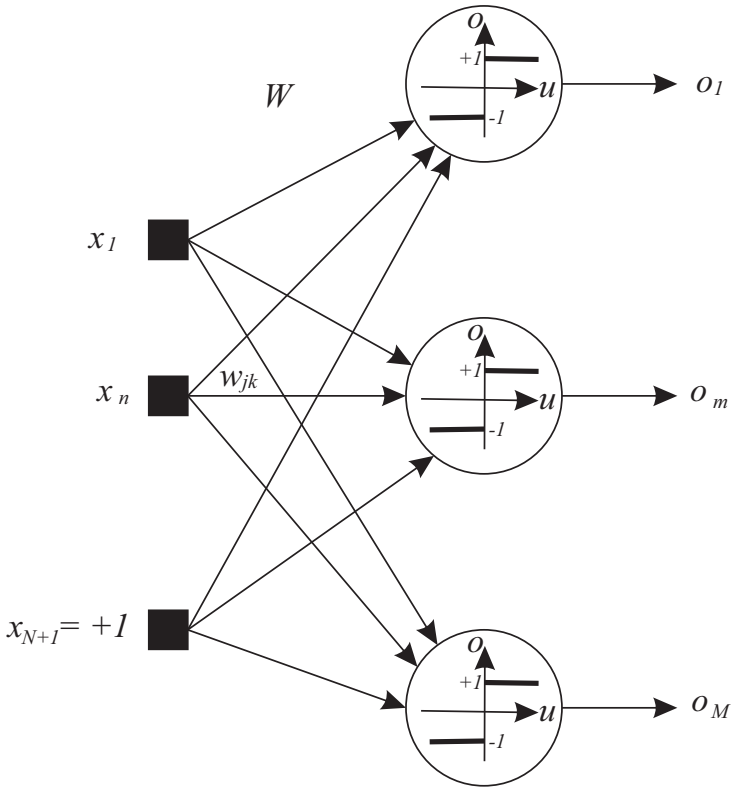


Figura 16 – Rede clássica de perceptrons.

2.4 Perceptrons de múltiplas camadas

Enquanto as redes de ADALINE conseguem aproximar qualquer função linear e as redes clássicas de perceptrons conseguem classificar qualquer conjunto linearmente separável, ambas se limitam ao domínio linear. Por esse motivo, a área de aprendizagem supervisionada ficou relativamente estagnada durante as duas décadas que seguiram após sua origem.

O potencial de uso de redes de múltiplas camadas com funções de ativação não lineares para representação de funções quaisquer e classificação de grupos quaisquer, independente de linearidade, já era conhecido. Entretanto, não se dispunha de um algoritmo para efetuar o treinamento de maneira generalizada em trais estruturas. Foi apenas com a publicação do método de retro-propagação de erro (EBP, do inglês, *error backpropagation*) por Rumelhart, em seu artigo revolucionário de 1985, que o ramo voltou a se dinamizar e toda uma nova geração de investigação foi desencadeada (RUMELHART; HINTON; WILLIAMS, 1985).

O método de Rumelhart, também chamado de regra delta generalizada, posteriormente expandido em seu livro de 1988, permite o aprendizado em redes de perceptrons com múltiplas camadas e funções de ativação não lineares, desde que estas sejam diferenciáveis (RUMELHART; HINTON; WILLIAMS, 1988). Apesar de ter causado um grande avanço para a área de aprendizado de máquina na década de 1980, o método EBP já era há muito utilizado em outras áreas, como em controle ótimo, e já havia sido investigado por diversos outros pesquisadores (KECMAN, 2001). Este método é uma versão do método de descida gradiente apresentado pela Equação 2.5 e será detalhado nas seções a seguir.

2.4.1 Funções de ativação

Para que uma rede seja um aproximador global, é necessário, que ao menos uma camada da mesma seja constituída por neurônios cuja função de ativação seja não linear. Diversas funções de ativação foram estudadas desde a publicação do EBP. A classe mais comum é a das funções sigmoidais, caracterizadas por um formato que se assemelha à letra S e cuja derivada é positiva para todos os valores reais: $\frac{d}{dx}f(x) > 0, \forall x \in \mathbb{R}$. Como exemplo de função sigmoidal podemos citar a função logística unipolar:

$$\frac{1}{1+e^{-u}} \tag{2.9}$$

Função logística bipolar:

$$\frac{2}{1+e^{-u}} - 1 \tag{2.10}$$

A tangente hiperbólica:

$$tanh(u) = \frac{e^{u} - e^{-u}}{e^{u} + e^{-u}} \tag{2.11}$$

A função de erro gaussiana:

$$erf(u) = \frac{2}{\sqrt{\pi}} \int_{0}^{u} e^{-t^2 dt} \tag{2.12}$$

Entre outras. Redes neurais de múltiplas camadas e funções de ativação sigmoidais são chamadas de perceptron de múltiplas camadas (MLP, do inglês, *multi-layer perceptron*). Um esquemático básico de MLP é apresentado na Figura 17 e um exemplo de função sigmoidal é apresentado no Figura 18 (a).

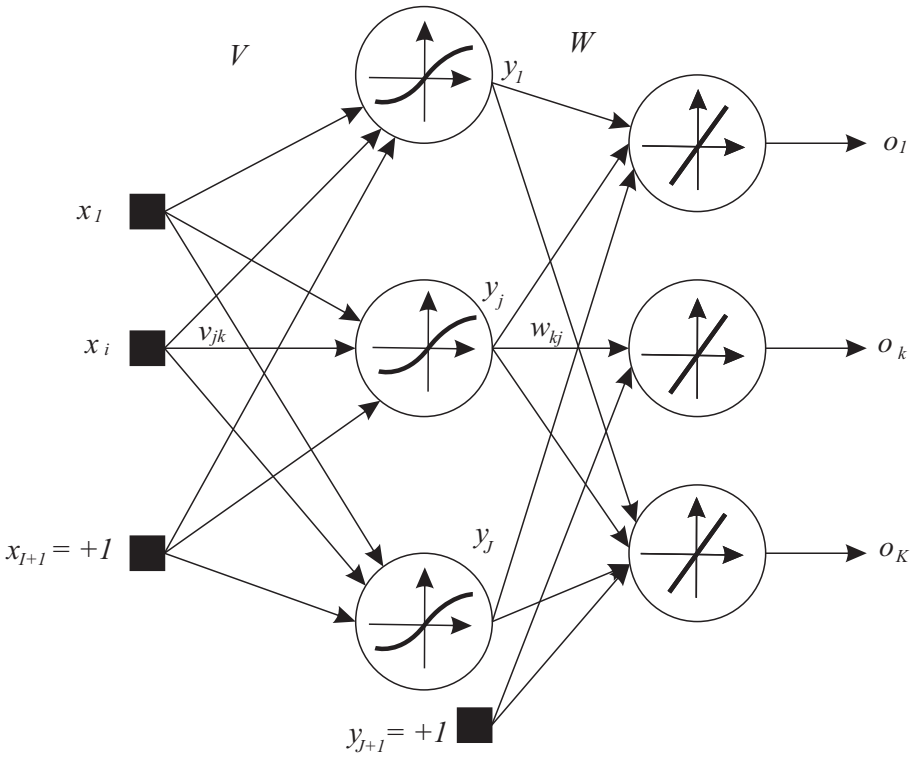


Figura 17 – MLP.

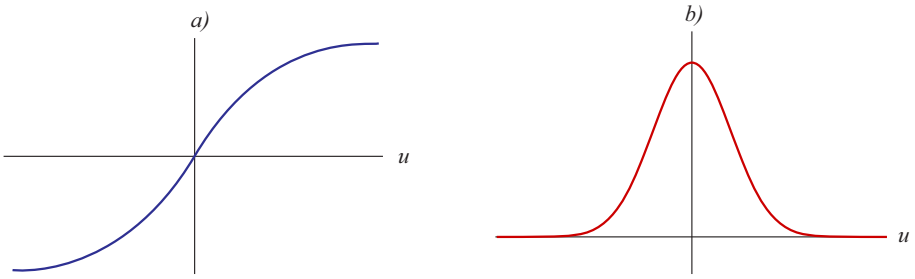


Figura 18 – Funções de ativação mais comuns e relevantes na área de RNA. (a) Função de ativação sigmoidal. (b) Função de ativação gaussiana.

Outra classe de funções de ativação muito importante é a classe das funções baseadas em raio (RBF, do inglês, *radial basis funciton*). Um exemplo de RBF é a função gaussiana:

$$f(u, \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{2.13}$$

Como exemplificado pela gaussiana da Figura 18 (b), as funções RBF são caracterizadas por curvas contínuas, suaves e simétricas. Redes neurais baseadas em funções RBF

também são referenciadas pelo termo RBF. Assim como redes MLP, redes RBF também podem ser treinadas pelo método EBP. Um fator que diferencia as RBF é que estas têm melhor capacidade de aproximação que as MLP, como mostrado por Poggio e Girosi (1990). Adicionalmente, redes RBF se destacam por sua fundamentação teórica, segundo Kecman (2001), baseada na chamada teoria da regularização, que trata da aproximação de funções de múltiplas variáveis a partir de dados esparsos.

2.4.2 Topologia de rede

Em se tratando de redes ADALINE ou perceptron com funções de ativação lineares, cada camada da rede pode ser representada por uma transformação linear. Neste contexto, uma rede ADALINE de múltiplas camadas é chamada de MADALINE. Assim, uma rede MADALINE com três camadas lineares, A, B e C, com entradas x e saídas o pode ser representada como

$$ABCx = o \tag{2.14}$$

Por essa notação, fica evidente que pode-se adotar uma matriz W capaz de representar qualquer transformação linear efetuada por ABC:

$$W = ABC \tag{2.15}$$

$$Wx = o \tag{2.16}$$

Portanto, o uso de múltiplas camadas de perceptrons cuja função de ativação é linear é uma redundância. Em se tratando de camadas não lineares, o assunto é mais controverso. Muita discussão no tópico ocorreu no final dos anos 1980 / início dos anos 1990, tanto de pesquisas que evidenciavam vantagens do uso de múltiplas camadas não lineares, quanto do oposto (CYBENKO, 1989), (HORNIK; STINCHCOMBE; WHITE, 1989), (KUURKOVA, 1992), (HUSH; HORNE, 1993), (FUNAHASHI, 1989), (CHESTER, 1990). Em suma, é matematicamente comprovado que tanto redes com apenas uma camada não linear quanto redes com múltiplas camadas não lineares funcionam como aproximadores universais. Assim, é comum o uso de redes com apenas uma camada não linear para a solução de problemas de engenharia, pelo princípio de se encontrar a solução mais simples e rápida para o problema em questão.

A rede MLP mostrada na Figura 17 é um exemplo típico para aplicações de aproximação. Tem-se a camada interna, também chamada de camada oculta (HL, do inglês, *hidden layer*), composta por neurônios cuja função de ativação é não-linear e a camada externa, também chamada de camada de saída (OL, do inglês, *output layer*)

composta por neurônios com funções de ativação linear. Para fins de classificação é mais comum o uso de redes MLP nas quais as funções da OL são funções degrau ou *signum*.

2.4.3 Quantidade de neurônios na camada interna

A questão referente ao número ideal de neurônios na HL, J, é um dilema importante na teoria de RNA, também chamado de dilema da variância-deslocamento (do inglês, *bias-variance dilemma*). Trata-se de um balanço entre obter algo mais próximo de uma aproximação ou algo mais próximo de uma interpolação.

É um fenômeno análogo ao que ocorre em interpolação polinomial. Quando a ordem do polinômio aproximador iguala a quantidade de pontos, obtém-se uma interpolação de fato, ou seja: a curva obtida passa necessariamente por cada um dos pontos tratados. Ao diminuir a ordem do polinômio, a curva resultante passa a ser uma curva de aproximação apenas, que não passa necessariamente pelos pontos.

O mesmo ocorre em redes neurais com funções de ativação não lineares. Em geral, ao elevar o número de elementos na HL, a curva resultante vai se aproximar de uma interpolação. Ao reduzir o número de elementos, a curva torna-se uma aproximação cada vez mais suave – e cada vez menos sujeita à influência de cada ponto individualmente. Em outras palavras, a redução faz com que a curva aproximadora passe a ignorar mais as características individuais de cada amostra. Isso a torna menos sujeita a ruídos – é como um filtro passa-baixas que elimina variações de alta frequência, gerando curvas mais suaves. Parte do sinal, contudo, também será ignorado juntamente com o ruído.

Quando se eleva excessivamente a ordem da HL, há uma tendência de se modelar o ruído pela aproximação. Em outras palavras, a curva está aproximando componentes espúrios dos dados. Esse fenômeno acarreta na elevação da variância da solução: a cada vez que dados são coletados e aproximados, uma nova curva é obtida porque o ruído está sendo agregado à curva modelada. Quanto mais ruído modelado, mais variação é esperada entre uma iteração e outra. Quando a ordem da HL é excessivamente baixa, por outro lado, a variância é também tipicamente baixa, de forma que a mesma curva é obtida sempre a cada nova coleta de dados. Entretanto, há um viés (do inglês, *bias*), um deslocamento na curva modelada, de forma que esta difere da função inerente que relaciona os dados. Busca-se então um balanço entre variância e bias ao determinar a quantidade de neurônios da HL, como apresentado na Figura 19. Este balanço depende inerentemente dos dados modelados, a quantidade de entradas, de saídas, a largura de banda do sinal e relação sinal-ruído.

2.4.4 A taxa de aprendizagem e inicialização dos pesos

A taxa de aprendizado η, também chamada de largura de passo, é um escalar de grande importância na descida gradiente. Valores demasiado elevados para η podem levar à instabilidade do algoritmo, impedindo que o mesmo venha a convergir. Por outro lado, valores muito pequenos de η podem tornar o tempo de convergência excessivamente longo, ao ponto de fazer com que o algoritmo pareça não chegar a lugar algum após longo tempo de execução. Para diferentes aplicações, η pode diferir em várias ordens de magnitude, assim, uma regra básica ao encontrar problemas de convergência com o EBP é variar os valores de η.

Uma prática comum na inicialização do EBP é definir pesos aleatórios para W e V no intervalo $(-1, 1)$. Dependendo da aplicação, porém, outros intervalos podem favorecer uma convergência mais rápida.

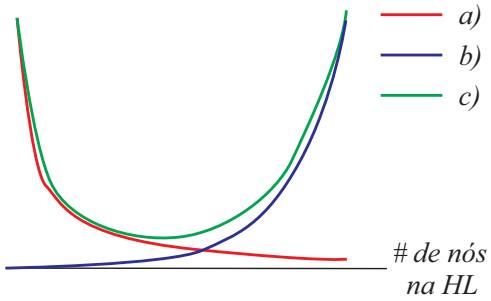


Figura 19 – Representação gráfica do dilema da variância-deslocamento. (a) Viés ou bias. (b) Variância. (c) Soma das influências negativas de bias e variância. Busca-se encontrar um balanço entre os dois problemas. Adaptado de Kecman (2001).

2.4.5 O algoritmo Backpropagation

O princípio fundamental do EBP é de se determinar os erros nas saídas da OL e, a partir deste erro, determinar os erros referentes às saídas das camadas internas, propagando o erro. Seguindo o princípio da Equação 2.5, os erros são usados para iterar os pesos de cada camada. No caso das redes MLP, como apresentado na Figura 17, temos que o delta da OL é

$$\delta_o = d - o \tag{2.17}$$

E que o delta da HL é

$$\delta_h = (1 - y)y\delta_o W \tag{2.18}$$

Por fim, os pesos devem ser iterados segundo:

$$W_{i+1} = W_i + \eta\delta_o y \tag{2.19}$$

$$V_{i+1} = V_i + \eta\delta_h x \tag{2.20}$$

O EBP trata a função de erro médio quadrático como uma função objetiva a ser minimizada, da mesma forma que o método de otimização gradiente, caminhando no sentido oposto ao gradiente da função objetiva. Assim, uma maneira de avaliar o progresso da aprendizagem é observar a função de erro quadrado definida por

$$E = \frac{1}{2}\delta_o^2 \tag{2.21}$$

No caso em que o número de saídas da OL é maior que um, $K > 1$, faz-se a soma dos erros quadrados de cada saída. E pode ser utilizado como condição de parada, estabelecendo um valor de erro almejado E_a, porém, muitas vezes prefere-se estabelecer um limite de iterações e avaliar os resultados empiricamente. Cada par $\{x_p, d_p\}$ dos P pares dos conjuntos X e D são apresentados ao MLP, efetuando P iterações. A cada vez que todo o conjunto de treinamento é iterado, diz-se que se passou uma época (do inglês, *epoch*) de aprendizado. Comumente usam-se várias épocas, repetindo-se o ciclo várias vezes. Quanto mais épocas, melhor o aprendizado, porém mais longo o tempo de treinamento. Assim, procura-se experimentalmente encontrar o melhor balanço entre número de épocas e tempo de treinamento. O algoritmo 1 apresenta o processo EBP.

2.5 Aplicações de RNA

Dada sua característica de atuar aproximador universal, as RNA encontram aplicações em diversos ramos científicos, da indústria, e da saúde. Em geral, pode-se dividir as aplicações de RNA em diversas classes:

Aplicações de regressão incluem casos em que a RNA é utilizada para modelar e prever padrões em séries temporais, funções de aptidão em otimização ou qualquer outra aplicação que tenha regressão como elemento central.

Aplicações de classificação que se caracterizam por reconhecimento de padrões e sequencias.

Processamento de dados que envolvem filtragem, agrupamento e compressão.

Robótica, onde RNA encontram as mais variadas aplicações, desde controle dinâmico, reconhecimento de padrões, até interface entre próteses e sistemas orgânicos.

Dados:
Matriz de entradas $X_{P,I}$,
Matriz de saídas $D_{P,K}$,
número de nós HL J,
taxa de aprendizado η, e
erro almejado E_a.
Resultado:
Matriz de pesos HL $V_{J,I+1}$, e
matriz de pesos OL $W_{K,J+1}$

Inicialização dos pesos W e V.
repeat
  $u = VX$;
  $Y = sigmoid(u)$;
  $O = WY$;
  $\delta_o = D - O$;
  $\delta_h = (1 - y)y\delta_o W$;
  $W_{i+1} = W_i + \eta\delta_o Y$;
  $V_{i+1} = V_i + \eta\delta_h X$;
  $E = \frac{1}{2}\delta_o^2$;
until $E > E_a$;

Algoritmo 1: Algoritmo EBP.

Sistemas de controle, onde RNA são aplicadas em controles adaptativos, cognitivos e inteligentes.

Dentro das categorias citadas pode-se mencionar controle de processos, veículos, gerenciamento de recursos e tomada de decisões, reconhecimento alvos, reconhecimento de padrões em processamento de imagens, visão computacional, processamento de áudio e sinais diversos, reconhecimento de voz, texto, diagnósticos médicos, previsões e sistemas de negociação financeira, mineração de dados e filtragem de email.

2.6 Desenvolvimentos recentes de RNA e computação paralela

A partir do momento em que o algoritmo EBP foi publicado, grande esforço foi realizado no sentido de explorar o paralelismo característico das redes neurais. Diversas implementações em hardware foram estudadas e desenvolvidas. Os chamados neuro-computadores, surgidos nesse movimento, são uma proposta de sistemas de hardware voltado para a implementação paralela de redes neurais. Muito foi desenvolvido usando ASIC, porém os custos e a rigidez do ciclo de desenvolvimento de ASIC invariavelmente tornava os chips obsoletos antes mesmo de chegarem ao mercado – o desempenho de implementações de software em processadores atuais se mostrava um rival insuperável dos neuro-computadores em termos de desempenho. Em termos de flexibilidade, as implemen-

tações em hardware dificilmente apresentam vantagens sobre as de software (OMONDI; RAJAPAKSE, 2006).

Com o avanço e a popularização de sistemas FPGA, novas abordagens de RNA em hardware passaram a ser estudadas nos anos 1990 e mais intensamente após 2000. A facilidade com que se pode reprogramar um sistema FPGA permite testar diferentes arquiteturas em um mesmo dispositivo a um ritmo muito semelhante ao de desenvolvimento de software. Adicionalmente, dispositivos FPGA recentes têm apresentado arquiteturas híbridas que contém centenas de nós aritméticos de ASIC distribuídos pela matriz. Com tais dispositivos pode-se utilizar a FPGA como uma rede programável e reconfigurável, explorando a flexibilidade de projeto de FPGA e ainda com desempenho comparável ao obtido em implementações ASIC (OMONDI; RAJAPAKSE, 2006).

Em geral, são três as principais dificuldades encontradas em implementações de redes neurais em FPGA: limitações de precisão numérica, representação de funções de ativação e limitação do tamanho do hardware. A questão da precisão numérica está associada à limitação do tamanho do hardware. Trabalhar com elevado número de bits para a representação numérica exige uma grande expansão no mesmo, que no contexto de RNA, é uma expansão de ordem polinomial. Ao elevar o número de bits de precisão acarreta-se em uma elevação no número de unidades aritméticas de cada nó da rede e também na complexidade do roteamento entre nós (ZHU; SUTTON, 2003).

Para muitas aplicações, adicionalmente, o número de nós da rede inviabiliza uma implementação totalmente paralelizada, exigindo que uma mesma unidade aritmética seja usada para realizar o processamento referente a vários perceptrons. Essa circunstância exige um escalonamento e multiplexação por divisão de tempo, de tal forma que requer uma fila para ordenar o uso de cada unidade, elevando o *overhead* de controle. Muitas implementações estudadas buscam, assim, encontrar um balanço entre paralelismo de hardware e divisão de tempo. Poucas aplicações práticas apresentam redes pequenas o suficiente para possibilitar implementações completamente paralelas. Além da limitação de processamento, outra limitação comum no hardware atual é a limitação de memória. Enquanto bancos de memória são abundantes e baratos atualmente, o acesso aos mesmo é serial, o que inviabiliza, muitas vezes, o acesso simultâneo de centenas de unidades aritméticas. Sistemas FPGA atuais também dispõem de pequenos blocos de memória RAM distribuídos pela matriz. Assim, como as unidades aritméticas, porém, sua quantidade é demasiado limitada para muitas aplicações (OMONDI; RAJAPAKSE, 2006).

A questão da representação das funções de ativação também é de grande relevância, uma vez que a convergência do EBP depende intrinsecamente das mesmas. Problemas relacionados à precisão da representação e do *overhead* gerado têm sido tratados em pesquisas recentes. Uma abordagem comum é o uso de tabelas de referência (LUT, do inglês *look-up table*) para a representação das funções, tanto alocadas em registradores *flip-*

flop quanto em memória RAM. Gomperts, Ukil e Zurfluh (2010) propõe uma paradigma de implementação paramétrico que visa explorar os recursos híbridos oferecidos em sistemas FPGA contemporâneos. Para tal, sugere uma representação de funções de ativação em LUT diferenciada, que permite uma acurácia superior às representações em LUT simples através de interpolação dinâmica sobre os elementos da tabela. Enquanto esta proposta permite uma economia dramática no uso e acesso à memória, causa um grande *overhead* de hardware. É digno de nota que sua implementação se restringe à representação numérica a ponto fixo, uma vez que o uso de ponto flutuante consumiria uma parcela maior de unidades aritméticas por nó.

Um caminho que tem sido explorado de forma intensa atualmente é o oferecido pelos sistemas de GPU. Beneficiada pelos investimentos massivos do dinâmico mercado de jogos eletrônicos, a tecnologia proporciona uma plataforma de pesquisa e desenvolvimento consolidada de processamento massivamente paralelo via software. A existência de interfaces de programação refinadas e estáveis e uma comunidade bem estabelecida favorece projetos na área. O uso de tais sistemas de computação paralela programáveis tem pavimentado o caminho para ramos de pesquisa em redes neurais de alta densidade e elevado número de unidades neuronais. Aplicações voltadas ao processamento de imagens, visão computacional, classificação e reconhecimento de padrões são as áreas mais beneficiadas (CIRESAN et al., 2011). Muito tem se proposto com relação às chamadas redes neurais convolucionais, que são inspiradas em estruturas neurais biológicas cuja função é o reconhecimento de padrões (KRIZHEVSKY; SUTSKEVER; HINTON, 2012).

3 Visão Computacional Estéreo

3.1 Estereoscopia

A visão humana permite percepção de distância e profundidade. Tal percepção é possível devido à característica binocular de nossa visão. As imagens captadas, uma de cada olho, possuem diferenças que são causadas pela distância relativa entre os olhos, sendo cada imagem obtida de uma perspectiva ligeiramente diferente. O par de imagens é processado e fundido pelo cérebro em apenas uma imagem, que contém informações de profundidade, como ilustrado pela Figura 20. A estereoscopia é uma técnica que imita esse efeito. Baseada nas diferenças entre um par de imagens obtidas de posições ligeiramente distintas, obtém informações de profundidade de um espaço 3D.

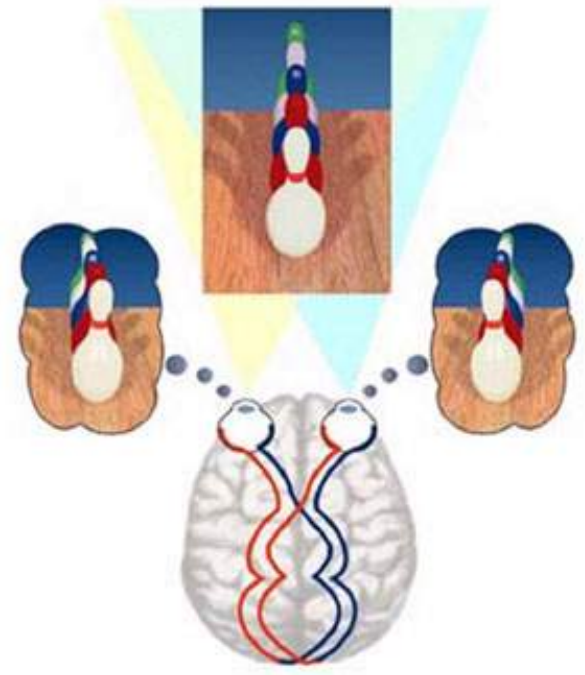


Figura 20 – Visao Binocular (MUNRO; GERDELAN, 2009)

A estereoscopia é obtida computacionalmente combinando métodos de processamento de imagens e de reconhecimento de padrões e permite calcular posição, tamanho e velocidade de objetos no espaço tridimensional a partir de um par de imagens bidimensionais. Conhecendo as características intrínsecas de cada câmera (distância focal, formato do sensor e distorções da lente) é possível estimar com precisão as posições e as formas dos objetos no espaço.

O par estéreo consiste de duas imagens de uma mesma cena obtidas de posições

distintas. Assim, um ponto específico, por exemplo, o topo de um pinheiro, deverá aparecer em coordenadas distintas no referencial de cada imagem. A distância absoluta entre cada ponto, ao sobrepor as imagens, é chamada de disparidade binocular do ponto, como ilustrado pela Figura 21.

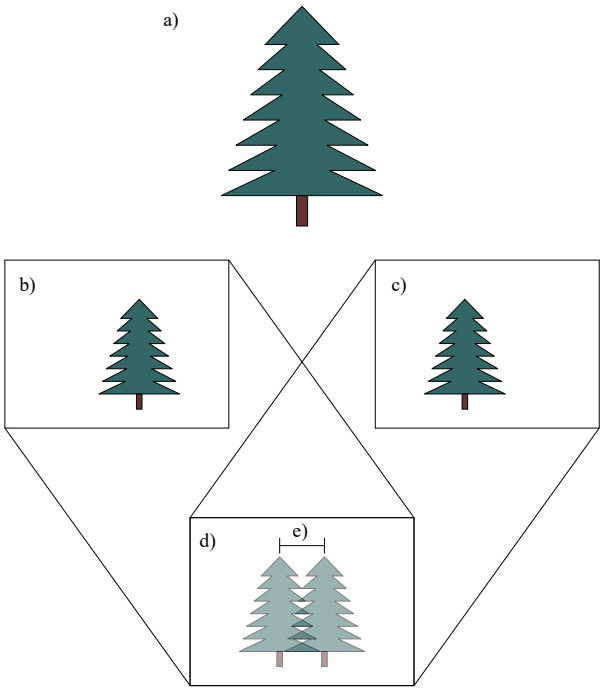


Figura 21 – Disparidade binocular. (a) Objeto tridimensional real. (b) Imagem bidimensional esquerda. (c) Imagem bidimensional direita. (d) Sobreposição das imagens esquerda e direita. (e) Disparidade binocular no ponto correspondente ao topo do pinheiro.

A partir das características da câmera, é possível inferir a profundidade de um dado ponto, dada sua disparidade. Para encontrar a disparidade entre dois pontos correspondentes, é necessário antes identificar tal correspondência, isto é, é necessário determinar quais pontos da imagem esquerda correspondem a quais pontos da imagem direita. Este processo de encontrar pontos correspondentes em pares estéreo é facilitado pela retificação das imagens, que as coloca no mesmo plano de projeção. O produto final da reconstrução estéreo é a nuvem de pontos, que consiste da lista dos pontos cuja posição no espaço 3D foi recuperada. As seções a seguir introduzem os conceitos fundamentais e detalham cada passo do procedimento para recuperação dos pontos tridimensionais.

3.2 Retificação

No caso geral, a busca de pontos correspondentes entre duas imagens de um par estéreo é uma busca bidimensional de elevado custo computacional. Quando se pode admitir alinhamento perfeito das câmeras, de forma a garantir a coplanaridade entre os

planos de formação de imagem, a busca se torna unidimensional porque correspondências se restringem a pontos situados em linhas equivalentes. Em outras palavras, quando se tem câmeras idênticas perfeitamente alinhadas, um ponto que aparece na linha l da imagem esquerda aparecerá na linha l da imagem direita, tornando o escopo de busca por correspondências unidimensional. Como o alinhamento mecânico perfeito é dificilmente obtido e mantido, é comum a adoção do alinhamento virtual, via software.

Este alinhamento via software, chamado de retificação, se utiliza das características intrínsecas e extrínsecas do par estéreo. As características intrínsecas são comprimento focal f, formato do sensor (m_x e m_y são fatores de escala horizontal e vertical e γ representa uma distorção cisalhante) e ponto principal (u_0, v_0) de cada câmera. As características intrínsecas são representadas matricialmente pela matriz K:

$$K = \begin{bmatrix} f \cdot m_x & \gamma & u_0 \\ 0 & f \cdot m_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

As características extrínsecas derivam das diferenças de posição T e orientação R entre as câmeras, formando a matriz de coeficientes extrínsecos: $\begin{bmatrix} R & T \end{bmatrix}$. O vetor T representa a posição da origem do sistema de coordenadas global expressa em coordenadas do sistema de coordenadas centralizado na câmera. R representa uma matriz de rotação, tal que $C = -R^T T$, sendo C a posição do centro da câmera expressa em coordenadas globais (FUSIELLO; TRUCCO; VERRI, 2000). Com este modelo de câmera, a transformação projetiva que leva pontos do espaço tridimensional para o espaço bidimensional (do mundo real para a imagem) é dada, em coordenadas homogêneas por:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \begin{bmatrix} R & T \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{3.2}$$

O processo de retificação tem interpretação geométrica demonstrável por meio da geometria epipolar, como ilustrado pela Figura 22. Duas câmeras distintas, cujos pontos focais são representados por c_1 e c_2, obtém imagens de um ponto p do espaço tridimensional. A imagem de p aparece no ponto p_1 da imagem 1 e no ponto p_2 da imagem 2. O ponto p_1 é determinado pela intersecção entre o plano da imagem 1 e a reta que liga p a c_1. p_2 é determinado de maneira análoga.

A projeção do ponto c_2 sobre a imagem 1, determinada pela intersecção entre o plano da imagem 1 e a reta que liga c_2 a c_1, é chamada de epipolo da imagem 1. O epipolo da imagem 2 é determinado analogamente. A linha e_1, contida no plano da imagem 1, liga o epipolo da imagem 1 ao ponto p_1 e é chamada de linha epipolar. Há infinitas linhas epipolares e todas elas se encontram no epipolo. O processo de retificação leva os epipolos

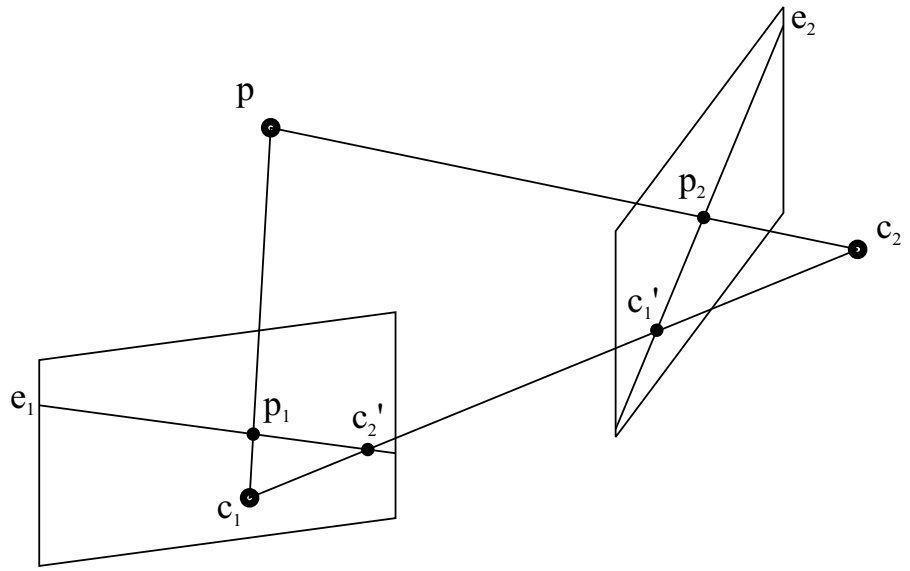


Figura 22 – Elementos fundamentais da geometria epipolar

das duas imagens para o infinito e faz com que todas as linhas epipolares sejam horizontais e paralelas. Como consequência, um ponto arbitrário p no espaço tridimensional, que tem sua imagem no pixel p_1 da imagem 1, terá sua imagem no pixel p_2 da imagem 2, sendo que p_1 e p_2 se encontram em linhas equivalentes. Em outras palavras, por exemplo, se p_1 está na linha 137 da imagem 1, p_2 estará na linha 137 da imagem 2. Entretanto, devido ao ruído e à oclusão, esta asserção pode não ser exatamente válida, mesmo em pares estéreo de câmeras perfeitamente calibradas. Para imagens reais, a asserção é apenas aproximadamente válida, oferecendo uma boa estimativa de onde buscar por pontos correspondentes. Por isso, os métodos mais utilizados buscam também em áreas vizinhas à linha epipolar.

3.3 Obtenção do mapa de disparidades

Após a retificação, o próximo passo do processo de reconstrução tridimensional é determinar os pontos correspondentes entre as duas imagens. Este processo permite a criação do mapa de disparidades binoculares, que permite a estimação das profundidades. Esta seção apresenta os dois métodos mais comuns para determinação de correspondências e geração do mapa de disparidades.

3.3.1 Stereo Block Matching

A correspondência estéreo de blocos (SBM, do inglês, Stereo Block Matching) é um algoritmo simples que utiliza da diferença dos níveis de intensidade entre duas imagens para obter a informação de distância. Neste método as imagens são percorridas por meio de uma janela N x N pixels, onde N é um número ímpar. Para cada janela de uma imagem, a outra imagem é percorrida por uma janela de mesmo tamanho sobre determinada linha

epipolar. Os valores de intensidade dos pixels então são comparados por meio da soma das diferenças absolutas ou soma das diferenças quadradas entre as janelas. A menor diferença absoluta é escolhida, determinando o ponto da correspondência. A Figura 23 apresenta graficamente o processo de varredura das janelas sobre a linha epipolar.

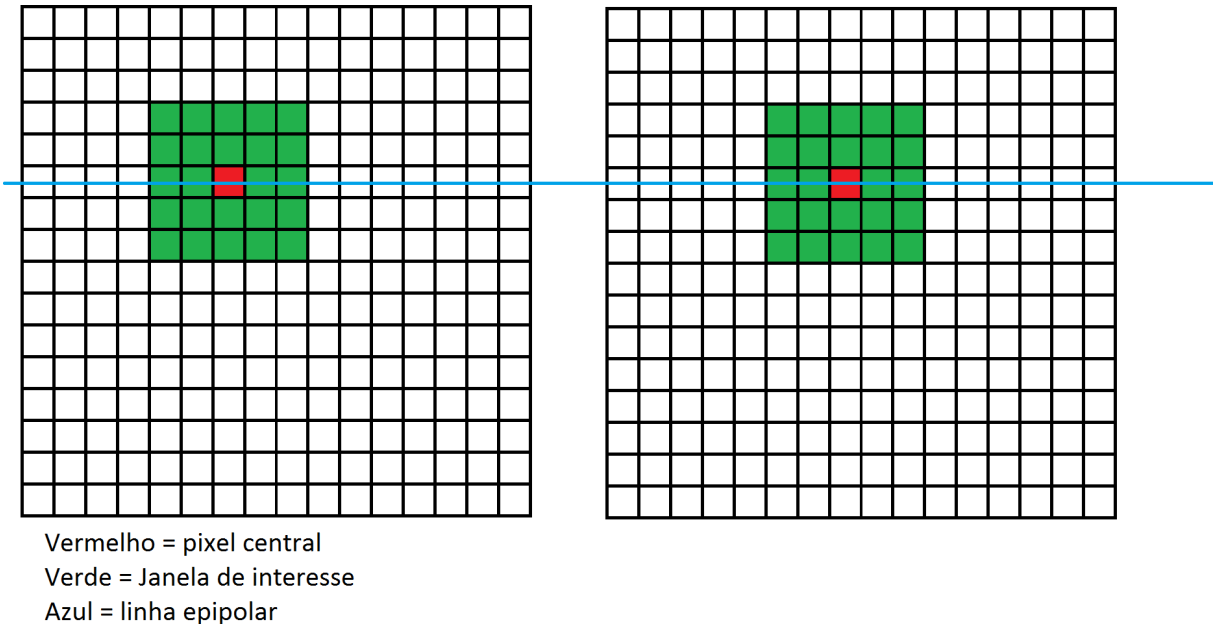


Figura 23 – Exemplo gráfico do algoritmo SBM.

3.3.2 Semi-Global Stereo Matching

O método da correspondência estéreo semi-global de blocos (SGBM, do inglês, Semi-Global Stereo Block Matching) é um processo mais complexo que utiliza de heurísticas para gerar reconstruções mais densas e precisas. Em vez de utilizar apenas a linha epipolar, como no SBM, SGBM faz o cálculo da função custo do pixel por meio de vários caminhos, como mostrado na Figura 24, para encontrar a melhor correspondência. A função custo para um pixel p e disparidade D_p é dada pela Equação 3.3, onde $h_{I_1}(p)$ é a entropia (HIRSCHMÜLLER, 2005) do pixel p da imagem esquerda, $H_{I_2}(D_p)$ é a entropia da imagem direita do pixel correspondente, e H_{I_1,I_2} é a entropia das duas imagens sobrepostas do pixel correspondente.

$$-C(p, D_p) = h_{I_1}(p) + H_{I_2}(D_p) - H_{I_1,I_2}(p, D_p) \tag{3.3}$$

Para cada caminho, é calculado o custo para chegar a um pixel com uma certa disparidade. Para cada pixel e para cada disparidade, os custos são somados sobre todos os caminhos. Depois disso, para cada pixel, a disparidade com o menor custo é escolhida.

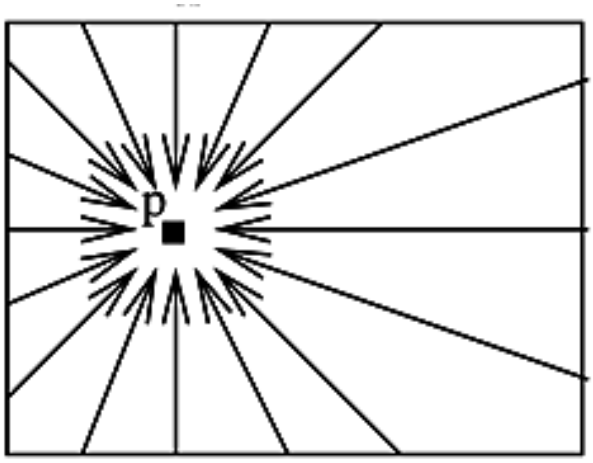


Figura 24 – Caminhos utilizados pelo SGBM.(HIRSCHMÜLLER, 2008)

3.4 Obtenção da nuvem de pontos a partir do mapa de disparidades

Uma nuvem de pontos é um conjunto de pontos em um sistema de coordenadas obtido a partir de reconstrução estéreo ou outras formas de sensoriamento tridimensional. Cada ponto geralmente é definido em um sistema de coordenadas cartesianas X, Y e Z, e pode possuir informações adicionais, tais como cor ou intensidade luminosa. As nuvens de pontos geralmente são obtidas por meio de aparelhos chamados LIDAR (do inglês, Light Detection and Ranging), mas podem ser geradas por meio de mapa de disparidade de visão estéreo, conversão de malha de polígonos, modelos CAD (Computer-aided design) entre outros. A Figura 25 apresenta um exemplo de nuvem de pontos obtida via LIDAR.

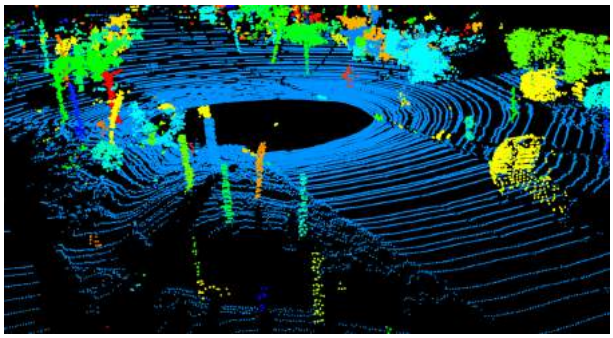


Figura 25 – Exemplo de nuvem de pontos (DOUILLARD et al., 2011).

Para construir a nuvem de pontos a partir de um mapa disparidades, precisamos determinar a correspondência pixel-ponto, ou seja, cada pixel irá gerar um ponto na nuvem de pontos com a sua coordenada específica. Para conseguir essa correspondência, é utilizada a matriz de transformação de perspectiva, ou matriz Q. Esta matriz é obtida a partir dos parâmetros intrínsecos e extrínsecos da câmera. Esta matriz representa a transformação projetiva inversa, que leva pontos do espaço bidimensional para o espaço

tridimensional.

$$Q = \begin{bmatrix} 1 & 0 & 0 & -u_0 \\ 0 & 1 & 0 & -v_0 \\ 0 & 0 & 0 & f \\ 0 & 0 & \frac{-1}{T_x} & \frac{u_0 - u'_0}{T_x} \end{bmatrix} \tag{3.4}$$

As coordenadas do ponto principal da câmera esquerda são u_0 e v_0. A coordenada x do ponto principal da câmera direita é dada por u'_0, f é a distância focal da lente e T_x a distância absoluta entre as duas câmeras. A obtenção do ponto $P(x, y, z)$ se dá utilizando a seguinte equação

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = Q \cdot \begin{bmatrix} u \\ v \\ disparidade(u, v) \\ 1 \end{bmatrix} \tag{3.5}$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \frac{x'}{w'} \\ \frac{y'}{w'} \\ \frac{z'}{w'} \end{bmatrix} \tag{3.6}$$

Inicialmente, foi necessária uma própria implementação para conseguir gerar a nuvem de pontos, mas após uma atualização de driver na DUO M, a mesma já entrega a nuvem de pontos através da API do driver. Após a geração da nuvem de pontos, os pontos são visualizados por meio do OpenGL.

3.5 OpenGL

OpenGL é uma biblioteca multi-plataforma de renderização e desenvolvimento de ambientes 2D e 3D. Desenvolvida pela Silicon Graphics, essa biblioteca é muito utilizada em CAD, realidade virtual, visualização científica, visualização de informação, simulação de vôo e jogos.

O OpenGL foi desenvolvido com a linguagem C/C++ em mente, mas passou a ser utilizado em diversas outras linguagens de programação como Ada, Fortran e Java.(OPENGL, 2016)

3.6 OpenCV

OpenCV (Open Source Computer Vision Library) é uma biblioteca de Visão Computacional, distribuída sob a licença BSD (Berkeley Software Distribution), que

suporta os sistemas operacionais Windows, Linux, Android e Mac OS. Possui interface com C++, C, MATLAB, Java e Python. Devido à grande variedade de algoritmos otimizados, desde básicos a avançados, o OpenCV é muito utilizado em meio acadêmico e industrial.

Entre as diversas aplicações do OpenCV, algumas são:

- Edição de imagens
- Visão Estéreo
- Robótica Móvel
- Reconstrução 3D
- Gravação e edição de vídeos
- Rastreamento de movimentos

Parte II

Plataforma e abordagem propostas

Este trabalho tem o objetivo de propor uma plataforma de robótica móvel e autônoma. O propósito da plataforma proposta é constituir uma base para estudo e desenvolvimento de métodos e abordagens em robótica móvel e áreas relacionadas. Entre os mais importantes requerimentos de tal plataforma, pode-se citar:

Mobilidade holonômica, permitindo que movimentos de deslocamento sejam independentes de movimentos de rotação sobre superfícies planas. Tal mobilidade elevada permite maior variedade de casos de estudo, sobretudo quando a plataforma é aplicada ao desenvolvimento de algoritmos de cooperação multi-agente. Ainda se pode impor limitações virtuais de movimentação para simular o comportamento de robôs com mobilidade mais limitada. Impostas via software, estas limitações podem ser alteradas segundo as necessidades específicas de cada uso.

Comunicação sem fio permite que o robô troque informações com outros robôs e com elementos da infraestrutura. Este requerimento é especialmente importante para o estudo de sistemas multi-agente, que exigem comunicação entre robôs. Entretanto, este requerimento também se faz importante para outros tipos de estudo, tais como desenvolvimentos em mapeamento, navegação e controle. Isso ocorre devido à necessidade de coletar informações em tempo de execução para análise de rendimento, entre outros diagnósticos.

Sensoriamento espacial para navegação e mapeamento. A plataforma deve conter sistemas que permitam que o robô identifique obstáculos e outros objetos à sua volta. Este requerimento é essencial para navegação e é constituinte fundamental de sistemas autônomos. O sensoriamento espacial é o meio primário de interação do robô com o meio, seja por sensores ativos ou passivos.

Arquitetura modular. Visto como um sistema de sistemas, o robô é constituído de módulos. Um módulo pode ser substituído sem que seja necessário alterar os demais módulos. Isso eleva a robustez do modelo, facilitando a detecção de erros de implementação do projeto. É também a modularizarão que permite que componentes do robô sejam desenvolvidos de maneira individual. Pode-se assim, por exemplo, desenvolver um sistema de visão computacional abstraindo o funcionamento do sistema de locomoção e vice-versa.

Construído a partir de componentes 'de prateleira'. Permite redução de custos e tempo de montagem e troca de componentes defeituosos.

Estes requerimentos levam a um modelo de sistema de sistemas como apresentado na Figura 26. Por esta perspectiva, o robô proposto pode ser visto como o conjunto de cinco módulos complementares:

Locomoção permite o deslocamento e a rotação do robô. Este módulo contempla motores, sensores de rotação e os controladores responsáveis para a realização destas tarefas.

Visão Estéreo permite extrair informações visuais sobre a geometria do ambiente no qual o robô se insere. Este módulo contempla os sistemas de câmeras estéreo e os módulos de processamento das imagens para estimação da geometria e interpretação dos dados visuais.

Posicionamento Global permite identificar a posição e a orientação do robô em relação a um referencial externo. Consiste de sistema de posicionamento global (do inglês, global positioning system – GPS) ou outro sistema análogo adequado para o propósito em questão. Para robótica sub-aquática ou mesmo robótica para ambientes fechados, sistemas alternativos ao GPS são comumente usados, devido às dificuldades de comunicação com satélites.

Comunicação permite que o robô troque informações com outros robôs e com elementos da infraestrutura.

Tomada de Decisão constitui a camada de mais alto nível do robô. É responsável por processar todas as informações fornecidas pelas camadas inferiores, tomar decisões de alto nível e enviar comandos para as camadas abaixo em resposta.

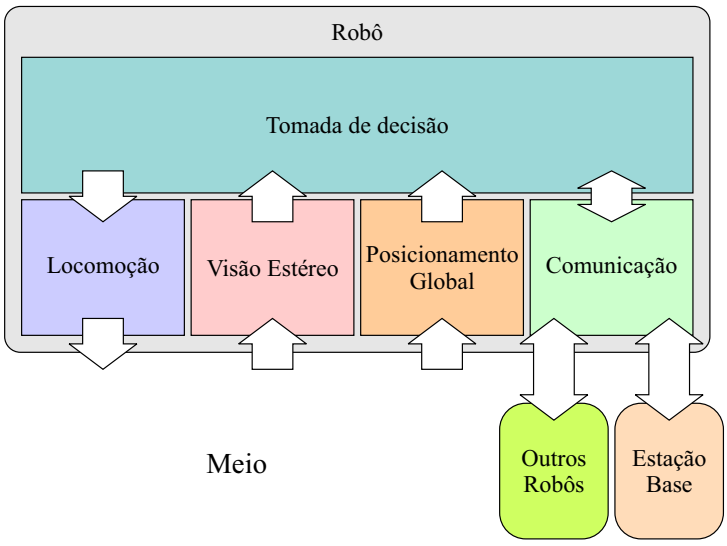


Figura 26 – Arquitetura de sistemas da plataforma proposta.

Tendo em vista os requerimentos apresentados, o estado atual das tecnologias de suporte à construção de robótica móvel de baixo custo (apresentadas no Capítulo 1) e se limitando ao uso de componentes 'de prateleira', se propõe a arquitetura de implementação apresentada pela Figura 27. Esta arquitetura incorpora os requerimentos de modularidade, e apresenta a divisão dos módulos propostos como distribuídos pelos dispositivos de suporte.

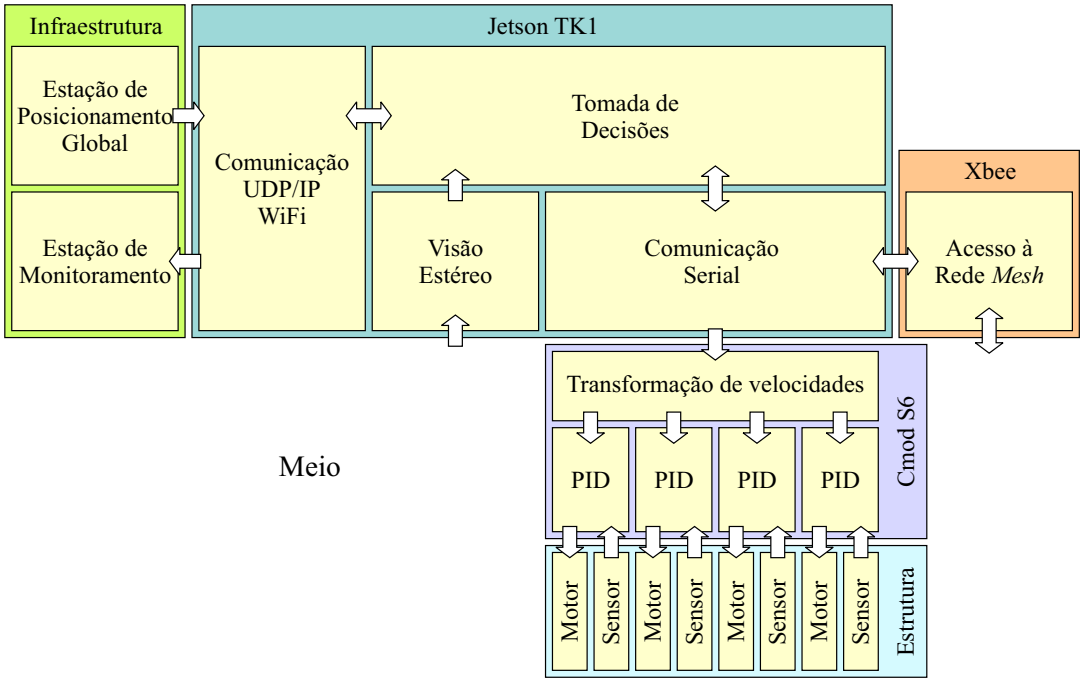


Figura 27 – Arquitetura de hardware da plataforma proposta.

O sistema de locomoção consiste da união de elementos físicos acoplados à estrutura do robô – motores e sensores associados – e de sistemas virtuais na implementados em hardware na forma de configuração do dispositivo de FPGA Cmod S6. Adotou-se este dispositivo por permitir implementar sistemas de processamento de sinal digital com taxa de amostragem fixa em período de *clock* além de sua natureza paralela. O paralelismo é especialmente adequado a este fim, pois permite a implementação de múltiplos controladores independentes, não estando sujeitos a escalonamento de tempo para uso de processamento. O Cmod S6 é especificamente adequado para nossa proposta por seu baixo custo, tamanho reduzido e compatibilidade com placas perfuradas largamente disponíveis no mercado de componentes eletrônicos.

Os sistemas de visão computacional, de tomada de decisão e de comunicação e de posicionamento foram implementados na plataforma híbrida MPU-GPU Jetson TK1. Devido à sua complexidade, tais sistemas requerem flexibilidade a nível de *software* para implementação em baixo custo, especialmente o sistema de visão computacional. Outro fator que favoreceu a adoção desta plataforma microprocessada para a implementação destes módulos é a disponibilidade de *drivers* e bibliotecas otimizadas para fins de comunicação, visualização e visão computacional.

Na arquitetura proposta, sistemas que constituem a infraestrutura – elementos externos que fornecem informação de posicionamento e recebem informações de telemetria – se comunicam via protocolo UDP/IP sobre enlace WiFi. Por esse meio, é facilitado o desenvolvimento de aplicações para estes fins, dada a maturidade das interfaces de

programação de aplicações (do inglês, application programming interface – API) disponíveis em ambientes linux.

A comunicação com módulos de rede *mesh*, assim como a comunicação para comando do módulo de locomoção é proposta através de comunicação serial UART, dado que o uso do protocolo UDP/IP com módulos embarcados atualmente disponíveis adicionaria complexidade desnecessária ao sistema. Através de linhas seriais a camada de tomada de decisões envia a uma taxa de amostragem fixa as velocidades de referência para que a camada de locomoção as execute. Os capítulos a seguir detalham a implementação de cada módulo, do ponto de vista de *hardware* e de *software*. O Apêndice A apresenta o guia de construção do robô para implementação do sistema proposto.

4 Implementação da arquitetura proposta

Este capítulo apresenta os detalhes de implementação de cada módulo da arquitetura proposta. A subseção 4.2.1 apresenta o sistema de locomoção; a subseção 4.2.2 apresenta o sistema de comunicação; a subseção 4.2.3 apresenta o sistema de posicionamento global. O sistema de visão computacional embarcado é apresentado, do ponto de vista de hardware, na subseção 4.2.4 e do ponto de vista de software na . Também é descrito o modelo de corpo rígido do agente robótico, responsável por delinear as características gerais do módulo de locomoção (seção 4.1).

4.1 Modelo Cinemático

Rodas omnidirecionais permitem a construção de sistema de direção holonômicos. A cinemática que sustenta tais sistemas é derivada da superposição das forças exercidas por cada roda individualmente sobre o corpo rígido. No caso ideal, as rodas não vão sofrer qualquer força de atrito contra movimentos laterais, i.e., perpendiculares à sua direção de rotação. Esta arquitetura permite que a velocidade do robô ultrapasse a velocidade individual de cada roda, em proporção quantificada pelo fator de aumento de velocidade (VAF, do inglês: *velocity augmentation factor*) (ASHMORE; BARNES, 2002). Diferentes números de rodas permitem diferentes VAFs. Considerando que o veículo se desloca sobre uma superfície plana, suas velocidades resultantes podem ser descritas por

$$\vec{v} = \begin{bmatrix} v_x \\ v_y \\ R\omega \end{bmatrix} \tag{4.1}$$

Onde v_x e v_y são suas velocidades na direção dos eixos x e y, respectivamente, e $R\omega$ é a velocidade rotacional, o produto entre o raio do robô, R, e sua velocidade angular resultante, ω, como apresentado pela Figura 28.

Considerando que o veículo consiste de um corpo rígido, pode-se traduzir $\vec{v}$ nas velocidades individuais de cada roda usando uma relação linear para qualquer número de rodas em qualquer configuração (ROJAS; FÖRSTER, 2006). No caso onde temos quatro rodas equidistantes do centro de massa do robô e alinhadas ao longo dos dois eixos de simetria ortogonais, a relação é simplificada para

$$\vec{u} = T \cdot \vec{v} \tag{4.2}$$

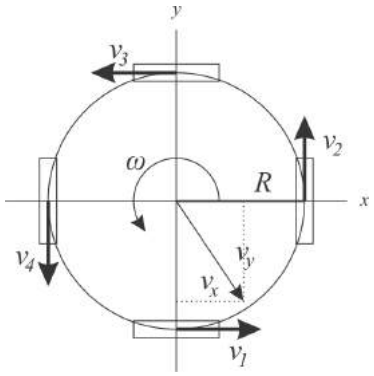


Figura 28 – Modelo cinemático para um sistema de direção holonômico de quatro rodas.

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ -1 & 0 & 1 \\ 0 & -1 & 1 \end{bmatrix} \cdot \begin{bmatrix} v_x \\ v_y \\ R\omega \end{bmatrix} \tag{4.3}$$

Onde $\vec{u}$ é o vetor que contém a velocidade escalar de cada roda e T é uma matriz de transformação linear quatro por três. A matriz pseudo-inversa de T, T^+, faz a transformação oposta, resultando nas velocidades do corpo como uma função da velocidade de cada roda individualmente, como em

$$\vec{v} = T^+ \cdot \vec{u} \tag{4.4}$$

$$\begin{bmatrix} v_x \\ v_y \\ R\omega \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & 0 & -\frac{1}{2} & 0 \\ 0 & \frac{1}{2} & 0 & -\frac{1}{2} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{bmatrix} \cdot \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} \tag{4.5}$$

Partindo da Equação 4.2 e da Equação 4.4, tomando I como a matriz identidade, segue que

$$\left[I - TT^+\right] \vec{u} = 0 \tag{4.6}$$

$$\begin{bmatrix} 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 \end{bmatrix} \vec{u} = 0 \tag{4.7}$$

Dada a Equação 4.7, e devido à dependência linear entre as velocidades, no caso ideal, i.e., quando todas as velocidades das rodas estão mutuamente consistentes e nenhuma opõe as demais, temos necessariamente que

$$v_1 - v_2 + v_3 - v_4 = 0 \tag{4.8}$$

Na prática, porém, esta soma não será sempre zero, indicando que algumas das velocidades das rotas estão em oposição. A patinação das rodas permite essa oposição. Quanto maior a magnitude da soma, mais intensa será a inconsistência entre as rodas. Assim, v_{inc} é proposto como um indicador de inconsistência de velocidades das rodas:

$$v_{inc} = |v_1 - v_2 + v_3 - v_4| \tag{4.9}$$

4.2 Projeto de hardware

O robô proposto segue o modelo holonômico de quatro rodas e foi implementado com peças pré-fabricadas dos kits VEX Robotics, descrito na seção 1.5. A adoção dos kits agiliza a implementação e flexibiliza o modelo para modificações sob-demanda, tais como a adição de novos níveis estruturais ou extensões laterais. Adotou-se peças em aço pelo seu baixo custo, porém o uso de peças de alumínio pode ser interessante para reduzir o peso total do robô. Rodas holonômicas de 60 mm foram adotadas, cada uma acoplada diretamente ao eixo de cada motor.

4.2.1 Sistema de locomoção

Motores com bucha, descritos na subseção 1.3.1, foram adotados por oferecer um bom balanço entre tamanho, potência e custo. Motores do mesmo modelo são adotados para cada uma quatro rodas, como mostrado no projeto da Figura 29. Estão acoplados a uma caixa de redução 34,014:1 que garante tração suficiente para que o robô possa portar cargas de vários quilogramas, ainda com velocidade máxima nominal de 1,2 metros por segundo. Cada motor também vem com um codificador magnético de duas fases acoplado ao eixo, que permite a leitura da direção de rotação, posição e derivação da velocidade do eixo. A subseção 1.4.1 descreve este tipo de codificador. Para impelir cada motor foi designada uma placa de regulação de corrente que recebe comandos digitais de direção e de intensidade em modulação por largura de pulso (PWM, do inglês: *pulse-width modulation*). Por fim, uma bateria de polímero de íon de lítio DuraTrax Onyx de 5 Ampére-hora foi adotada por sua elevada capacidade de carga e descarga e pela robustez de seu invólucro plástico. O conjunto é apresentado por diferentes vistas nos esquemáticos da Figura 29. Para o controle dos motores é proposto o uso de quatro controladores

proporcional-diferencial-integral (PID, do inglês: *proportional-integral-derivative*) atuando em paralelo. Neste projeto, a implementação dos controladores é realizada em uma placa de desenvolvimento FPGA para garantir temporização fidedigna, sincronia e paralelismo real. FPGAs são descritas na subseção 1.1.4.

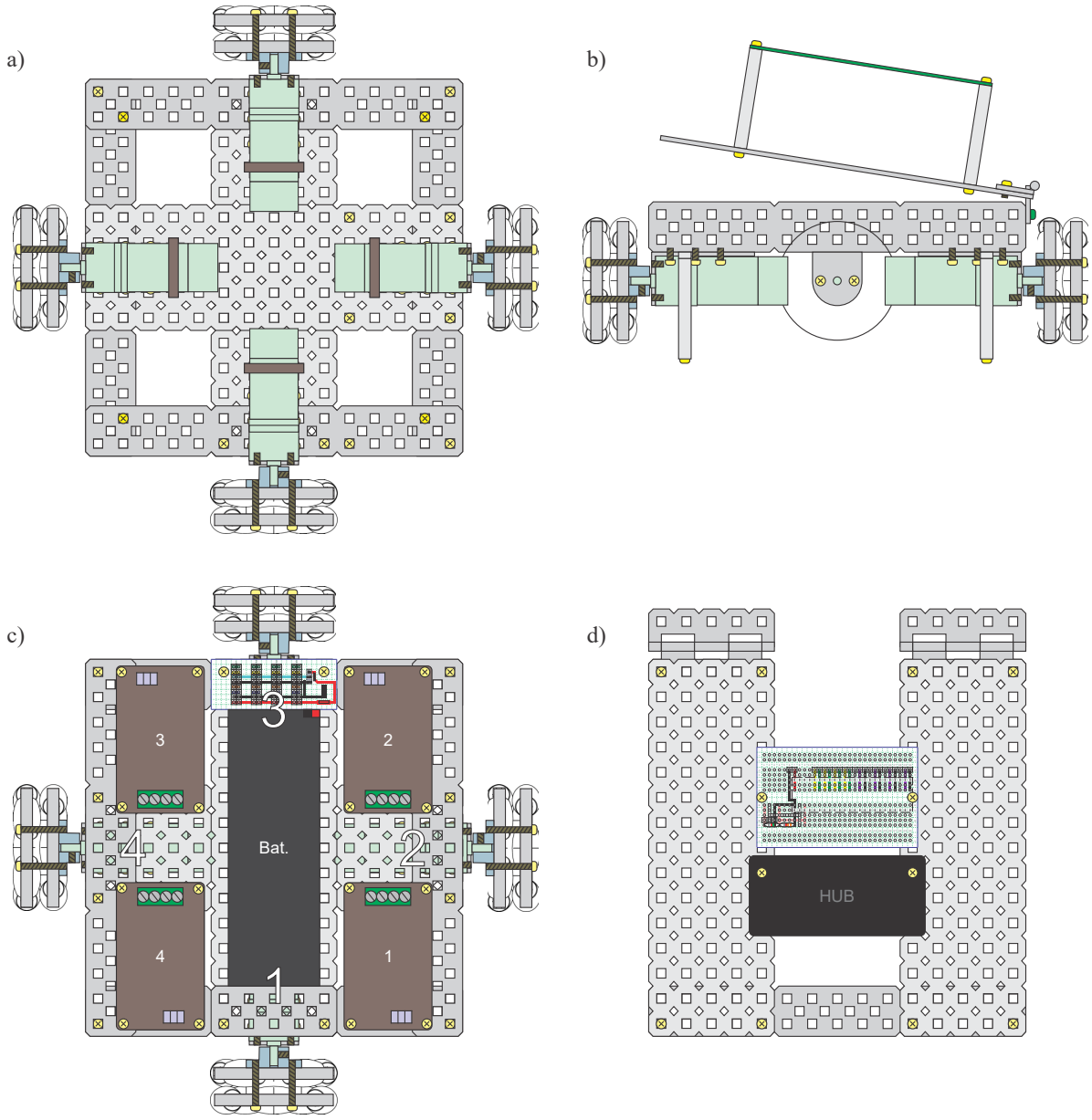


Figura 29 – Estrutura do robô proposto. (a) Visão inferior, apresentando os quatro motores e suas rodas. (b) Visão lateral, apresentando abertura por dobradiça para acesso interno. (c) Visão interior com numeração dos motores/rodas e respectivas placas de regulação de corrente, bateria e placa interna de circuito. (d) Visão da tampa superior, apresentando o posicionamento da placa externa de circuito e do hub USB.

Para a implementação dos controladores, um dispositivo Cmod S6 foi utilizado. Este dispositivo recebe as velocidades de referência referentes a cada roda via comunicação

serial. Decodifica a velocidade lida por cada sensor angular, alimentando os controladores PID que geram sinais de controle em modulação PWM para cada motor. A Figura 30 apresenta o diagrama que descreve estas relações entre motores, sensores e o controlador.

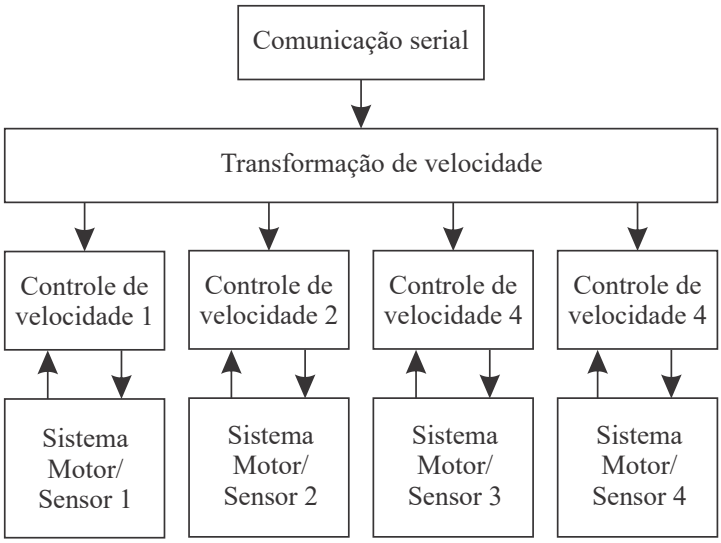


Figura 30 – Arquitetura do módulo de locomoção.

As conexões elétricas entre os dispositivos mencionados foram realizadas através da implementação de duas placas de circuito. A *Placa interna* distribui a energia proveniente da bateria para a alimentação dos dispositivos embarcados e faz a conexão entre os terminais provenientes dos motores – sinais de PWM e leitura dos codificadores rotacionais – e seus respectivos destinos. Esta placa também acomoda um regulador de tensão que alimenta os sensores rotacionais dos motores. A *Placa externa* acomoda o dispositivo Cmod S6 e faz a conexão de seus terminais aos respectivos destinos. A Figura 31 apresenta os diagramas de conexões elétricas referente às duas placas, assim como a codificação de corres de fios proposta.

Para facilitar a definição dos pesos dos controladores, uma Digilent Nexys 4 foi utilizada. Esta plataforma é retro-compatível com a Cmod S6, e os códigos desenvolvidos podem ser mutuamente portados com um mínimo de adaptações. O desenvolvimento é favorecido na Nexys 4 porque a plataforma oferece maior quantidade de componentes programáveis, além de uma rica variedade de conectores incorporados, tais como Ethernet, VGA e USB. Esta plataforma de desenvolvimento é baseada na FPGA Xilinx Artix 7, que dispõe de 63400 tabelas-verdade programáveis, 126800 registradores flip-flop 240 módulos aritméticos e 4,860 Kbits em blocos de memória RAM integrados. Os pesos do controlador foram definidos empiricamente com a ajuda de um adaptador de vídeo também implementado na FPGA. Este adaptador permite a visualização dos sinais de controle e dos sinais dos codificadores em um monitor de vídeo genérico, sem a necessidade de qualquer dispositivo intermediário.

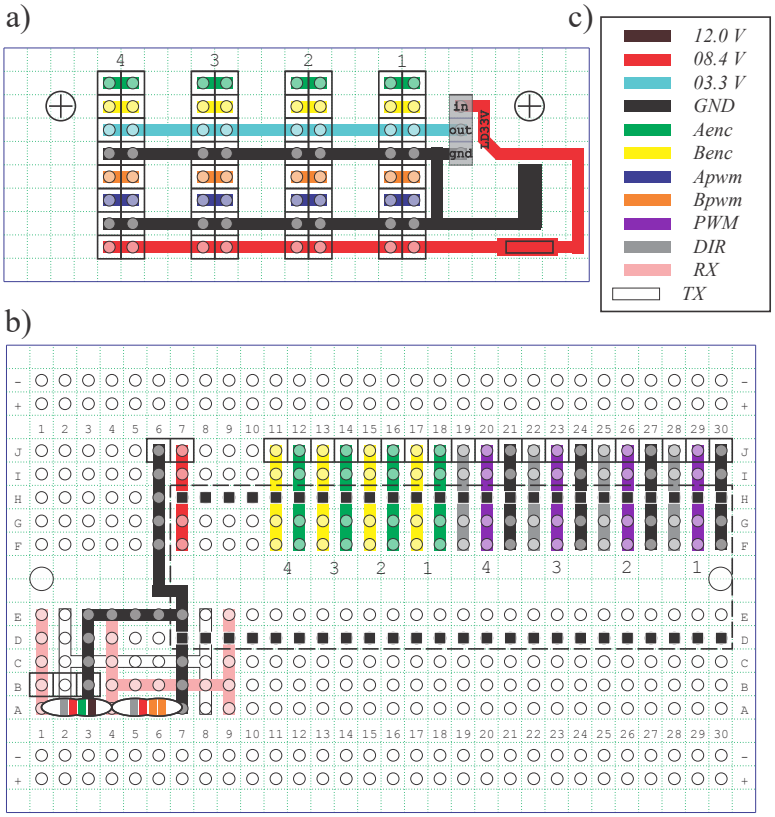


Figura 31 – Diagrama elétrico das placas de circuito. (a) Placa de circuito interna e (b) placa de circuito externa. (c) Código de cores para conexão dos componentes elétricos.

O sistema de controle implementado consiste em um arranjo de quatro controladores PID independentes. Implementados em verilog, seu funcionamento é verdadeiramente paralelo, e sua temporização é regularizada pelos pulsos de clock da FPGA. A Figura 32 apresenta a arquitetura digital do controlador implementado. O integrador é dotado de um limitador de saturação cujo limite é definido como um parâmetro externo. Quando é detectada uma saturação, o sistema congela o valor do acumulador.

A Figura 33 apresenta o sistema no qual os controladores PID foram desenvolvidos. Nesta figura apresenta-se apenas um dos quatro controladores por economia de espaço. Este sistema permite ao usuário a seleção de parâmetros de controle dinamicamente, em tempo de execução. Esta seleção se dá por meio das chaves e botões encontrados na placa de desenvolvimento FPGA. Os três pesos dos controladores (coeficientes proporcional, diferencial e integral), assim como o limite do integrador e as velocidades de referência podem ser definidos durante a execução do sistema.

Para a definição dos pesos, o controlador de vídeo gera sinais VGA para um monitor de vídeo convencional. No vídeo são apresentados os sinais advindos dos quatro sensores, as quatro referências de velocidade e os quatro sinais de controle (esforço de controle) em tempo real. Um controlador de mostrador de sete segmentos também faz parte do sistema

e permite ao usuário visualizar diversos parâmetros numericamente através do mostrador contido na placa de desenvolvimento. Os sinais de controle que saem do controlador são modulados em dois sinais distintos: intensidade (modulado em PWM) e direção de rotação para o motor. Tais sinais são então encaminhados para a placa de controle de potência, que trata de acionar os motores devidamente.

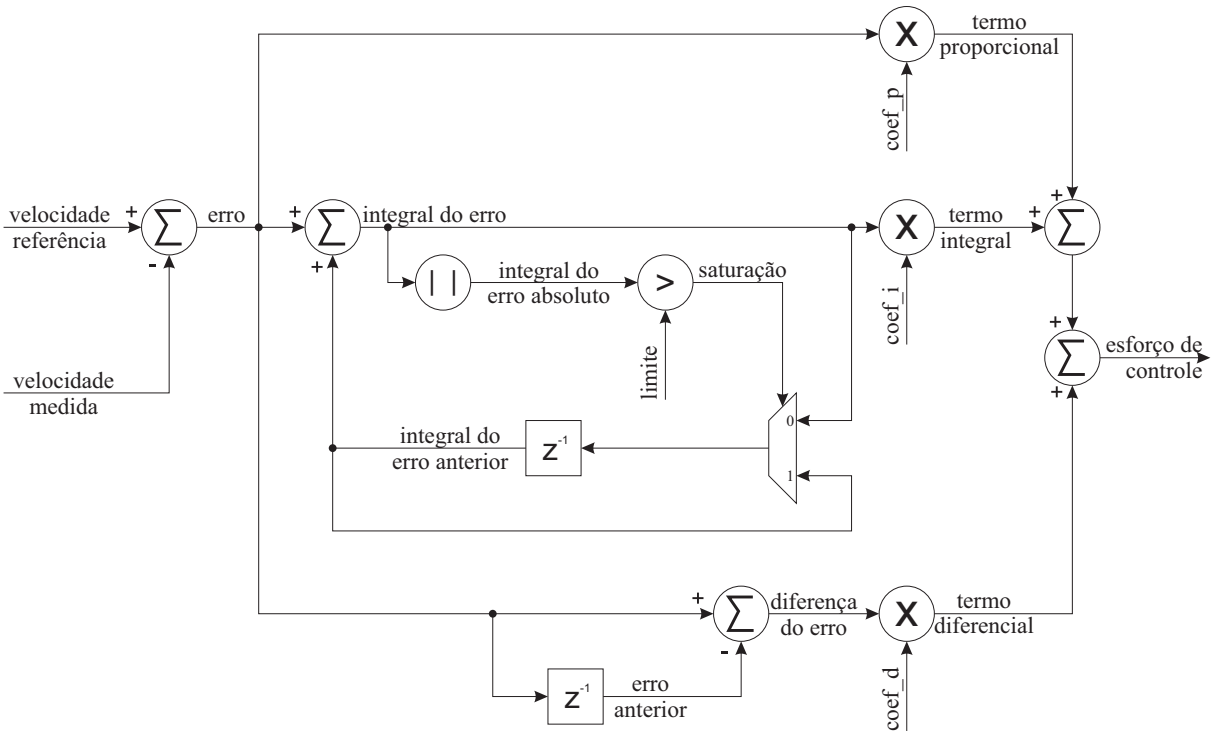


Figura 32 – Arquitetura do controlador PID digital.

4.2.2 Sistemas de comunicação

Durante o desenvolvimento da plataforma foram testadas diferentes soluções de comunicação sem fio. Nos testes utilizaram-se os módulos para estabelecer um enlace sem fio entre o robô e um computador de mesa, com o objetivo de possibilitar a coleta remota de dados do robô em tempo de execução e permitir o controle remoto do mesmo. Para esta comunicação ponto-a-ponto entre o robô e a infraestrutura, foram testados: (i) rádios de modulação AM genéricos, de origem chinesa, que se apresentaram pouco robustos e sensíveis a ruído; (ii) rádios Nordic Semiconductor nRF24L01+, que foram capazes de garantir conexão transparente ponto-a-ponto, ainda que half-duplex; (iii) módulos HC-06, que implementam uma ponte transparente UART full duplex com resposta satisfatória; (iv) módulos WiFi Intel 7260 HMW, que disponibilizam largura de banda suficiente para transmissão de imagens sem compactação em tempo real; e (v) módulos XBee® 802.15.4, que apresentaram o menor tempo de resposta dos módulos avaliados. Os módulos XBee®

802.15.4 e Xbee ZigBee foram testados também para a implementação da rede *mesh* DigiMesh entre robôs.

Como resultado dos testes, concluiu-se que que os módulos (iii) e (v) são os mais adequados em termos de confiabilidade para conexão ponto-a-ponto para telecomando e transferência de dados com baixa taxa de bits. Os módulos Intel 7260 HMW são mais adequados para telemetria envolvendo transmissão de imagens por sua maior largura de banda. Os módulos XBee® 802.15.4 e ZigBee são os mais adequados para a comunicação entre robôs. Dados os resultados apresentados, propõe-se adotar o módulo WiFi Intel 7260 HMW para telemetria e coleta de dados com elevada largura de banda e os módulos XBee® 802.15.4 para implementação de rede *mesh* homogênea.

4.2.3 Sistema de posicionamento global

Para suprir a demanda de posicionamento em relação a um referencial global, é típico o uso de GPS em ambientes externos. Limitações de orçamento, de espaço dedicado e limitações climáticas, entretanto, tornam atrativo o desenvolvimento de plataformas para uso em ambientes fechados, onde o sistema de GPS tradicional não pode ser utilizado. Dadas as restrições de desenvolvimento da plataforma proposta, a mesma foi otimizada para operação em ambientes fechados. Por tanto, o uso de GPS deixa de ser uma opção.

Em face ao exposto, propõe-se um sistema de suporte ao posicionamento global que não depende da recepção de sinais GPS. Baseado em processamento de imagens, o sistema proposto registra com acurácia a posição e a orientação do robô em cada momento de sua trajetória, dentro de um espaço delimitado. Uma câmera de vídeo acoplada ao topo do ambiente, apontada para baixo gera imagens que são utilizadas para tanto. Um software de processamento de imagens e visão computacional foi implementado para detectar a posição do robô com base em análise de tonalidades. Dois marcadores em cores uniformes no topo do mesmo, como mostrado na Figura 34, permitem o processamento.

O processo de localização do robô e determinação de sua orientação em relação ao referencial da câmera se dá por: (a) Determinar a posição do marcador amarelo e do marcador verde. (b) Determinar a posição do robô ao obter o ponto médio entre os dois marcadores: $p_{robô} = \frac{1}{2} \cdot (p_{verde} + p_{amarelo})$. (c) Determinar a orientação do robô a partir de operações sobre os vetores de posição dos marcadores: $o_{robô} = (p_{verde} - p_{amarelo})^{\perp}$. A Figura 35 detalha graficamente a obtenção de posição e orientação do robô através de operações vetoriais.

A determinação da posição dos marcadores coloridos se dá através de limiarização seletiva por tonalidade. Primariamente, a codificação da imagem é convertida do espaço de cores RGB (do inglês *red-green-blue* – vermelho, verde e amarelo) para o espaço HSL (do inglês *hue-saturation-lightness* – tonalidade, saturação e luminosidade). No espaço

HSL, são selecionados apenas os pixels cujos valores de tonalidade se enquadram em uma faixa definida pelo usuário. Esta faixa é determinada manualmente através de controles deslizantes da interface gráfica do software desenvolvido. Este processo de calibração deve ocorrer no ambiente de testes utilizando a câmera em sua posição final e os marcadores do robô como referência. O usuário também define limites para as dimensões de saturação e luminosidade. Uma vez calibrado, o sistema é capaz de calcular o ponto médio entre todos os pixels que se enquadram aos critérios do usuário. Este ponto médio é adotado como sendo a posição estimada para o marcador colorido. A Figura 36 apresenta uma captura da tela do sistema de localização em execução. Os pixels referentes aos marcadores encontrados pelo software estão destacados em cores verde e amarelo vibrantes. As linhas azuis indicam a posição e a orientação do robô detectados pelo sistema.

O software, executado em um computador de mesa, se encarrega de enviar periodicamente os dados de posicionamento e orientação para o robô. A Figura 37 apresenta as relações entre o robô e os sistemas da estrutura de suporte, que oferecem serviços de posicionamento global e monitoramento.

4.2.4 Sistema de visão computacional embarcado

Para permitir ao robô a obtenção de dados visuais sobre as superfícies que o cercam, propõe-se o uso de um sistema de visão computacional baseado em câmeras estéreo. Este método de sensoriamento, muito comum e acurado na natureza entre animais terrestres, permite a obtenção extensiva sobre posições e características dos objetos e superfícies ao redor do indivíduo. O uso de visão estéreo permite a obtenção de informações sobre tamanho, proporções e formato de múltiplos objetos simultaneamente, além de fornecer boa aproximação da superfície sobre a qual está situado o indivíduo. Esta noção de superfície facilita a navegação e desvio de obstáculos. Tal abordagem apresenta-se como um grande desafio computacional por se tratar de uma ciência recente com aplicação ainda limitada. As limitações de processamento embarcado atuais incrementam o desafio, porém acredita-se que o hardware atual proporcione desempenho satisfatório para os propósitos apresentados.

Para implementar um sistema de visão computacional é proposto o uso de câmeras DUO M, apresentadas na subseção 1.4.4. A estrutura proposta para incorporar este dispositivo ao projeto é apresentada pela Figura 38. O processamento das imagens provenientes destes sistemas requer poder computacional elevado e grande disponibilidade de memória. Adicionalmente, o suporte dos fornecedores do dispositivo se limitam a poucas arquiteturas. Com vista destas restrições, propõe-se adotar a plataforma Nvidia Jetson-TK1 para gerenciar o processamento da visão computacional. O posicionamento da mesma é visto na Figura 29 (b), no topo da tampa do robô. O sistema de visão proposto é apresentado em maiores detalhes no Capítulo 5, com ênfase no software.

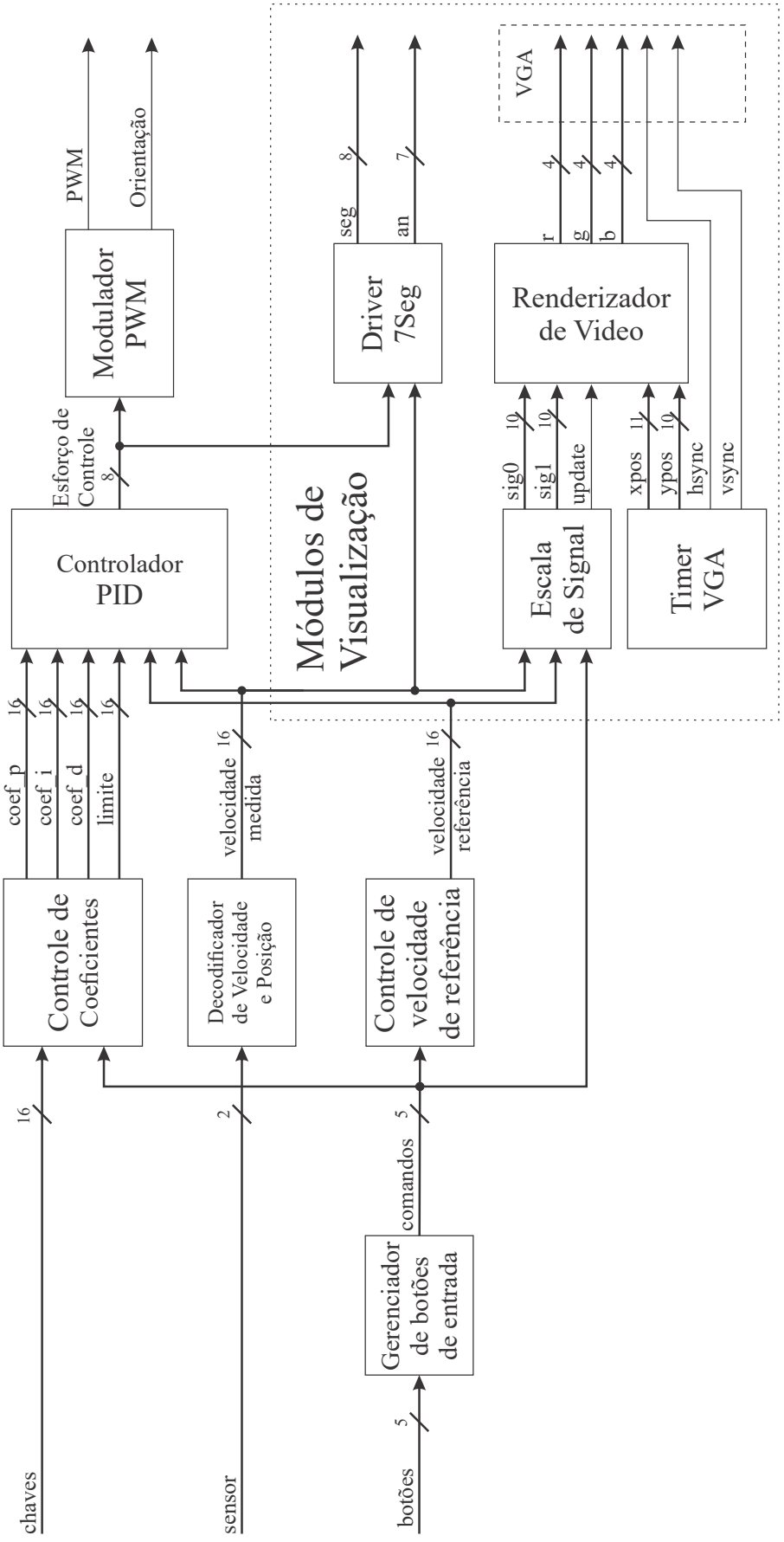


Figura 33 – Sistema de calibração do controlador PID.

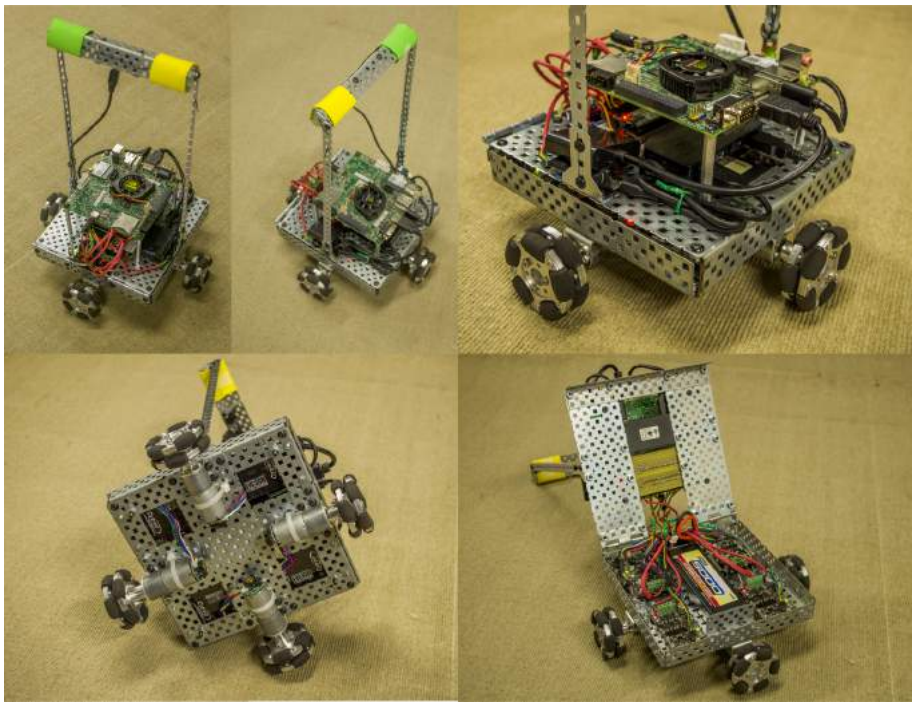


Figura 34 – Vistas perspectivas do robô completo, com marcadores em cores uniformes nas extremidades superiores do suporte ao dispositivo de visão computacional.

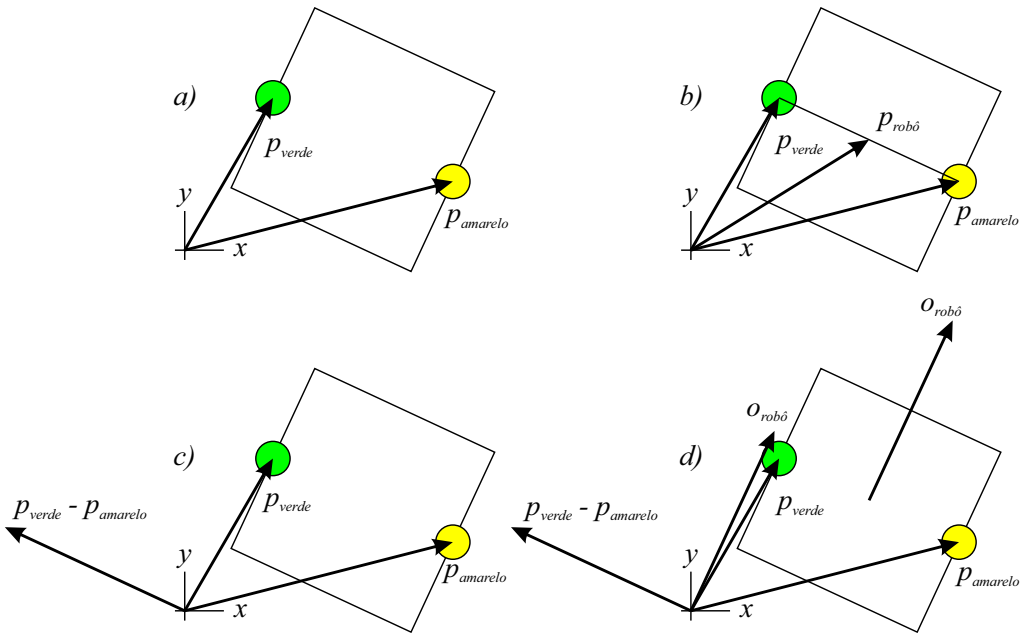


Figura 35 – Operações vetoriais para obtenção de posição e orientação do robô em um referencial externo. (a) Vetores de posição dos marcadores verde p_{verde} e amarelo $p_{amarelo}$. (b) Obtenção de $p_{robô}$ a partir da média de p_{verde} e $p_{amarelo}$). (c) Diferença vetorial de p_{verde} e $p_{amarelo}$). (d) O vetor ortogonal a $p_{verde} - p_{amarelo}$ representa a orientação do robô.

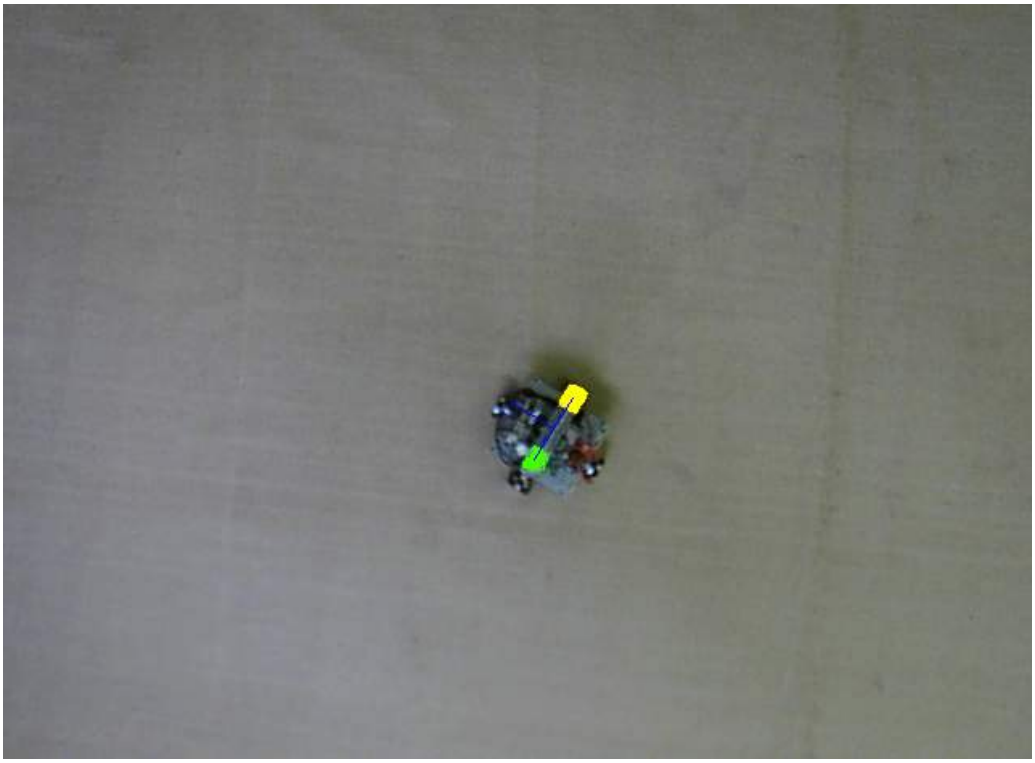


Figura 36 – Captura da tela do sistema de localização em execução.

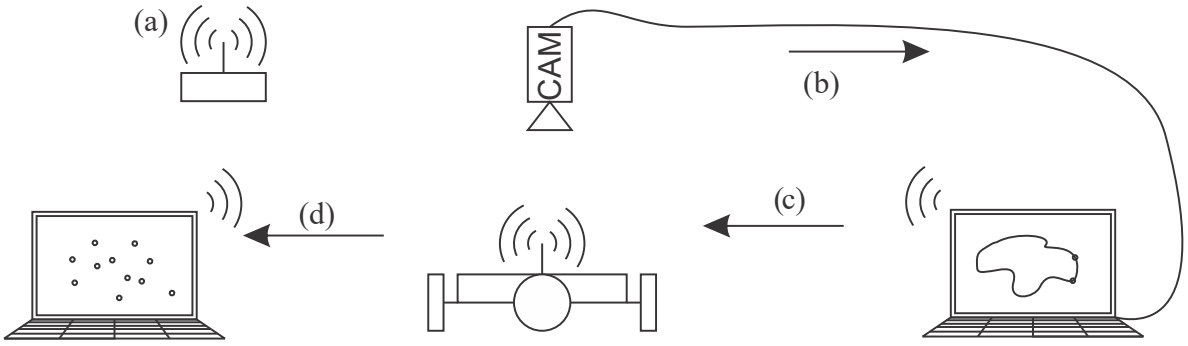


Figura 37 – Esquemático da estrutura de suporte. (a) Ponto de acesso para a rede WiFi. (b) Imagens da vista superior do ambiente são enviadas para a estação de suporte de posicionamento. (c) A estação de posicionamento processa as imagens para extrair informações de posicionamento e orientação e as envia periodicamente ao robô. (d) O robô envia periodicamente dados de telemetria à estação de monitoramento.

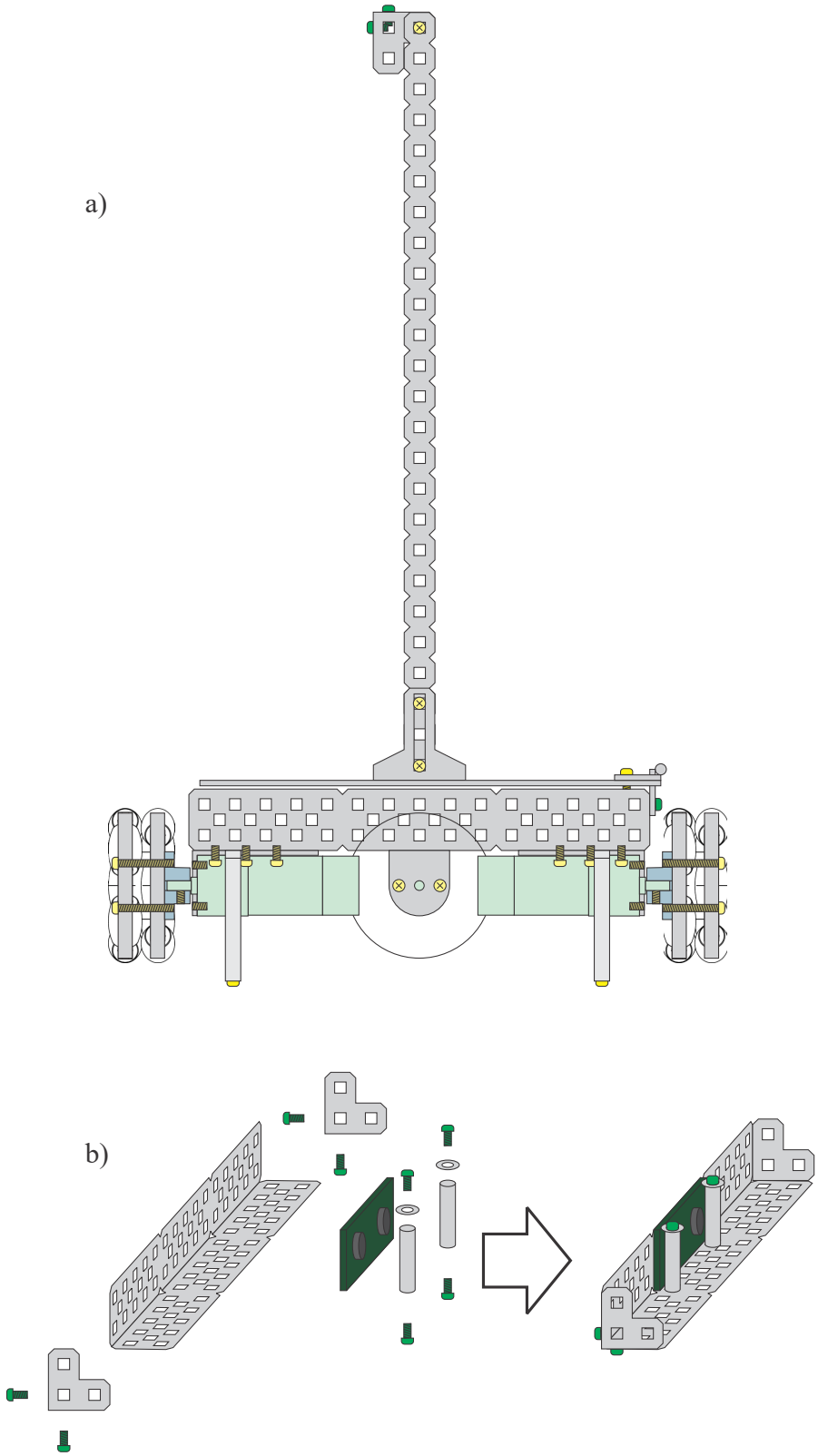


Figura 38 – Estrutura de suporte ao hardware de visão estéreo. (a) Visão lateral apresentando a estrutura de suporte ao hardware de visão estéreo. (b) Detalhe da montagem para o sistema de visão estéreo.

5 Sistema de visão computacional proposto

O Capítulo 3 resumiu como a visão computacional pode fornecer dados tridimensionais detalhados sobre a geometria de um ambiente a partir de pares binoculares de imagens bidimensionais. Tais dados tridimensionais são traduzidos em uma nuvem de pontos, que é ponto de partida para processos de interpretação geométrica. Tais processos permitem extrair informações de alto nível da nuvem, tal como detectar posição e quantidade de obstáculos, determinar áreas de risco e áreas livres para o deslocamento do robô, entre outras informações. Como parte do módulo de visão computacional propõe-se uma abordagem de segmentação e detecção de obstáculos que utiliza métodos de computação geométrica, estimação robusta e redes neurais artificiais. O produto final deste método é uma lista de objetos presentes em cena. A lista contém características geométricas dos objetos detectados, tal como posição, dimensões e pontos pertinentes. Este capítulo descreve todos os passos desta abordagem, incluindo passos preliminares necessários para calibração da câmera utilizada e controle de ganho/exposição.

5.1 Operação da DUO M

As lentes das câmeras DUO M possuem amplo ângulo de visão de (170º) e geram imagens com grande distorção esférica, resultando no efeito conhecido como "olho de peixe". A DUO disponibiliza um pacote encapsulado para a retificação e obtenção de nuvem de pontos, o DUO Dashboard. Para fazer a retificação, o padrão da Figura 39 foi impresso, e após várias tentativas de calibração utilizando a ferramenta disponibilizada, os parâmetros de retificação desejados foram obtidos.

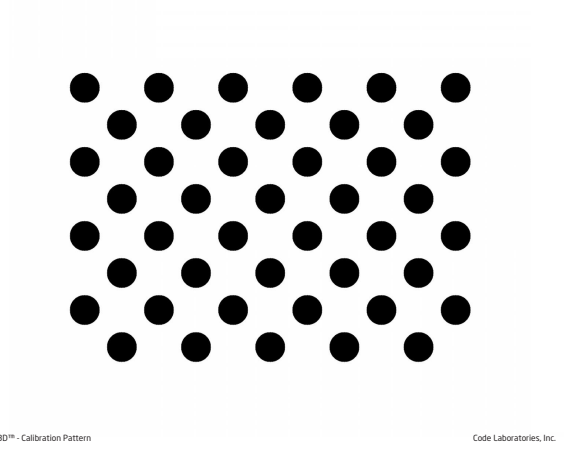


Figura 39 – Padrão de Calibração (DUO3D, 2016)

5.2 Controle de ganho da câmera

Os algoritmos de obtenção de disparidade são sensíveis a mudança de luminosidade em uma cena. Imagens muito escuras podem apresentar pouca informação e poucas variações entre os níveis de intensidade. Já as imagens muito claras podem criar áreas completamente brancas em que também não é possível extrair as informações de disparidade.

Para garantir que as variações de intensidade luminosa não afetem muito a obtenção dos dados, foi implementado um controlador de exposição e ganho ao fazer a captura da imagem. Durante cada captura de quadro, uma quantidade fixa de pixels aleatórios é coletada, e então uma média dos níveis de intensidade é calculada. A média é fornecida como entrada em um controlador proporcional integral, e esse controlador determina os valores de exopsição e de ganho.

O controlador funciona mudando apenas um valor que vai de 0 a 200, que foi chamado de nível de luminosidade. Quando este valor está entre 0 e 100, apenas o Nível de Exposição, que é proporcional ao tempo de exposição da câmera é modificado, e o ganho é colocado como zero. A partir de 100, o nível de ganho da câmera é modificado, como mostrado pela Figura 40. Os pesos do controlador foram obtidos empiricamente.

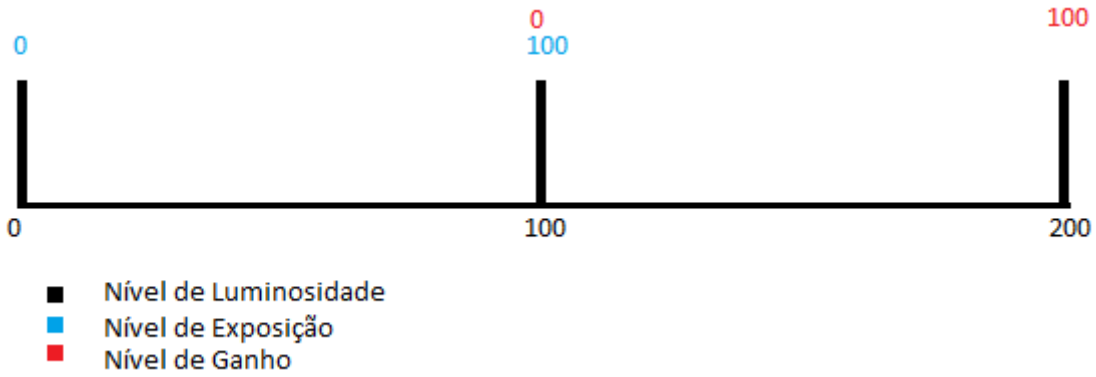


Figura 40 – Niveis de Luminosidade

5.3 Segmentação da Nuvem de Pontos

Uma vez obtida uma nuvem de pontos do sistema de reconstrução estéreo, é necessário extrair dela informações significativas para a navegação. Este processo de extração de informações geométricas a partir de nuvens de pontos é chamado de segmentação. O termo deriva do processo de segmentar a nuvem, identificando objetos distintos na mesma.

Para a plataforma proposta, propõe-se um método de segmentação otimizado para robótica terrestre, que utiliza de suposições aplicáveis à navegação em solo. A abordagem aqui apresentada se trata de uma expansão e aprimoramento sobre o algoritmo apresentado

e analisado pelos autores em publicações recentes (MARCON et al., 2015; MARCON et al., 2016). A segmentação da nuvem de pontos se dá em duas etapas. Na primeira etapa, é detectado o plano principal do piso sobre o qual está o robô. Na segunda etapa, baseado no que foi detectado na primeira, são descartados todos os pontos que fazem parte do piso e é criado um mapa de onde estão os objetos.

5.3.1 Detecção do Plano principal

A detecção do plano principal se dá por meio de do algoritmo RANSAC. Inicialmente são selecionados vários conjuntos de três pontos aleatórios da nuvem de pontos, para cada um desses conjuntos, um plano que passa por esses três pontos é criado como plano candidato. A partir de vários conjuntos de três pontos aleatórios da nuvem de pontos, são criados vários planos que passam por esses três pontos.

Após a geração dos planos candidatos, o melhor plano é selecionado. O cálculo do melhor plano se dá pelo menor valor da função custo. Esta é dada pela soma das distâncias de cada ponto da nuvem ao plano candidato.

$$C(P) = \sum_{i=0}^{n} d(p_i, P) \tag{5.1}$$

Onde $d(p, P)$ é definida pela Equação 5.2, equação que define a distancia de um ponto a um plano.

$$d(p, P) = \frac{|ax_p + by_p + cz_p + d|}{\sqrt{a^2 + b^2 + c^2}} \tag{5.2}$$

Caso a distância seja maior do que desvio padrão esperado do ponto ao plano, $d(p, P)$ é limitado para o valor do desvio padrão. Isso é feito para que a distância de pontos muito longe do plano não influenciem mais o custo do que pontos que não estão tão longe, já que queremos saber apenas se o ponto está dentre os limites do plano ou não.

Utilizando a Equação 5.1, o plano selecionado será o que possui uma maior quantidade de pontos que fazem parte do plano.

5.3.2 Rotação do Plano Principal

Após a detecção, o sistema de coordenadas é rotacionado, para que o plano principal encontrado fique paralelo ao plano XY. Para isso, o plano encontrado pelo RANSAC será rotacionado θ em torno de $\vec{u}$, onde $\vec{u}$ é o vetor unitário de $\vec{u}^*$.

$$\vec{u} = \frac{\vec{u}^*}{|\vec{u}^*|} \tag{5.3}$$

$\vec{u}^*$ é o produto vetorial entre $\vec{v}_n$,e o vetor unitário $\vec{k}$.

$$\vec{u}^* = \vec{v}_n \times \vec{k} \tag{5.4}$$

$\vec{v}_n$ é o vetor normal ao plano que será rotacionado

$$\vec{v}_n \;=\; [\; a \;\; b \;\; c \;]^T \tag{5.5}$$

θ é o ângulo entre $\vec{k}$ e $\vec{v}_n$.

Assumindo que $\vec{v}_n$ é unitário, $cos(\theta) = c$ e $sin(\theta) = \sqrt{a^2 + b^2}$. A Fig. 41 mostra a perspectiva geométrica explicada acima.

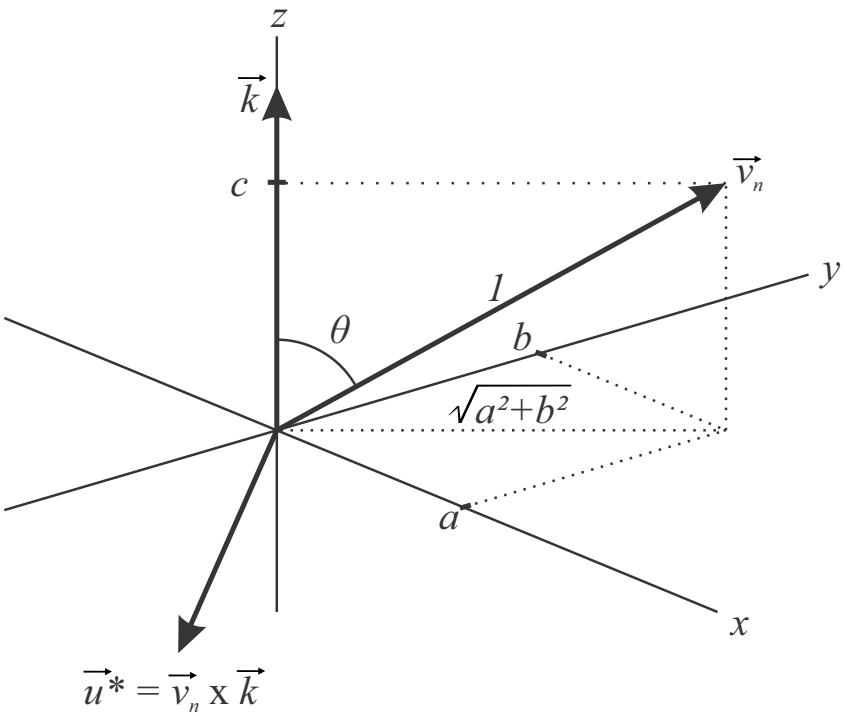


Figura 41 – Espaço de Rotação.

Sob essas suposições, temos

$$\vec{u}^* = \begin{bmatrix} b \\ -a \\ 0 \end{bmatrix} \tag{5.6}$$

e

$$\vec{u} = \begin{bmatrix} \frac{b}{\sqrt{a^2+b^2}} \\ \\ \frac{-a}{\sqrt{a^2+b^2}} \\ \\ 0 \end{bmatrix} \tag{5.7}$$

Da fórmula de rotação de Rodrigues (MURRAY; SASTRY; ZEXIANG, 1994), dado o eixo de rotação definido por $\vec{u}$ e o ângulo de rotação definido por θ, a matriz de rotação é

$$\begin{bmatrix} cos\theta + u_x^2(1-cos\theta) & u_x u_y(1-cos\theta) & u_y sin\theta \\ \\ u_x u_y(1-cos\theta) & cos\theta + u_y^2(1-cos\theta) & -u_x sin\theta \\ \\ -u_y sin\theta & u_x sin\theta & cos\theta \end{bmatrix} \tag{5.8}$$

$$\begin{bmatrix} \frac{c+b^2(1-c)}{a^2+b^2} & \frac{-ab(1-c)}{a^2+b^2} & -a \\ \\ \frac{-ab(1-c)}{a^2+b^2} & \frac{c+a^2(1-c)}{a^2+b^2} & -b \\ \\ a & b & c \end{bmatrix} \tag{5.9}$$

Que podemos simplificar e obter a matriz de rotação H

$$H = \begin{bmatrix} \frac{a^2c+b^2}{a^2+b^2} & \frac{ab(z-1)}{a^2+b^2} & -a \\ \\ \frac{ab(z-1)}{a^2+b^2} & \frac{b^2z+a^2}{a^2+b^2} & -b \\ \\ a & b & c \end{bmatrix} \tag{5.10}$$

Aplicando H para todos os pontos da nuvem, eles são rotacionados e produz uma nova nuvem com o plano principal aproximadamente paralelo ao eixo XY.

Após completar o próximo passo do algoritmo, a nuvem segmentada deve ser rotacionada de volta para sua posição original. Isso pode ser feito aplicando a matriz de transformação inversa H^{-1}, que é igual a H^T.

$$H^{-1} = \begin{bmatrix} \frac{a^2c+b^2}{a^2+b^2} & \frac{ab(z-1)}{a^2+b^2} & a \\ \\ \frac{ab(z-1)}{a^2+b^2} & \frac{b^2z+a^2}{a^2+b^2} & b \\ \\ -a & -b & c \end{bmatrix} \tag{5.11}$$

5.3.3 Segmentação

Para separar uma superfície plana de uma complexa, separamos os pontos em blocos, e levamos em conta a diferença de altura dos pontos que estão em cada bloco.

É de se esperar que terrenos que são praticamente planos que fazem parte do plano principal, mesmo com o ruído da captura, possuam pontos com pouca diferença de altura. Em vez de separar os pontos por blocos previamente definidos, como em um *grid*, é proposta uma árvore quaternária (Figura 42) e realizar uma subdivisão adaptativa, assim é possível focar em áreas que possuem maior complexidade.

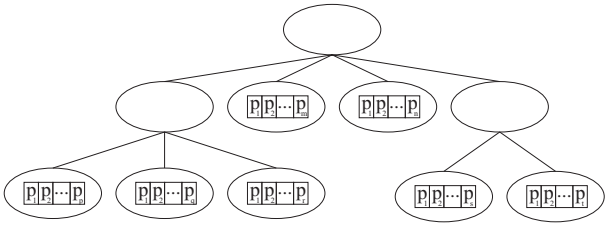


Figura 42 – Árvore quaternária

Cada nó da árvore contém uma lista L da nuvem de pontos, e uma forma retangular R que contêm todos eles. O processo de subdivisão consiste em percorrer a árvore recursivamente.

Inicialmente, o nó raiz é inicializado com todos os elementos da nuvem de pontos e colocados na lista L. Para cada nó, é calculada a diferença das menores e maiores distâncias dos pontos até o plano principal previamente calculado. Essa diferença é obtida utilizando a Equação 5.12, onde L_z representa os valores do eixo Z dos pontos da lista L.

$$v_r = |max(L_z) - min(L_z)| \tag{5.12}$$

Como o plano principal foi rotacionado para ficar paralelo ao eixo XY, as distâncias podem ser calculadas apenas considerando a coordenada Z de cada ponto.

Para ocorrer a descida do nó da árvore, os lados do retângulo do bloco devem ser maiores do que um tamanho mínimo R_t, para evitar segmentação excessiva, e $v_r < V_r$, onde V_r é o limiar mínimo da diferença das maiores e menores distâncias dos pontos do bloco ao plano principal. O retângulo R do bloco, é então dividido em quatro retângulos menores, cada um com um quarto da área de R e com a mesma proporção. A lista L de pontos é então subdividida em quatro sub-listas, cada uma contendo os pontos de cada sub-retângulo. Para cada sub-retângulo que contenha pontos, um novo nó é criado e percorrido recursivamente. Durante a recursão, a lista que foi subdividida é apagada para evitar redundância, assim apenas os nós folhas da árvore a lista de nós, e cada nó possui apenas os pontos do seu respectivo ao seu bloco. O algoritmo 2 descreve o processo que é demonstrado na Figura 43.

Data: Lista de Pontos L e Retângulo R
Result: Nuvem de pontos subdividida
$v_r = |max(L_z) - min(L_z)$;
if $v_r > V_r$ *e* $R_t >$ *tamanho mínimo* **then**
  Divide R igualmente em R_1, R_2, R_3 e R_4;
  Separa os pontos de L em L_1, L_2, L_3 e L_4;
  if $size(L_1) > 1$ **then**
    instancia e percorre (L_1, R_1)
  end
  if $size(L_2) > 1$ **then**
    instancia e percorre (L_2, R_2)
  end
  if $size(L_3) > 1$ **then**
    instancia e percorre (L_3, R_3)
  end
  if $size(L_4) > 1$ **then**
    instancia e percorre (L_4, R_4)
  end
end

Algoritmo 2: Segmentação da árvore quaternária

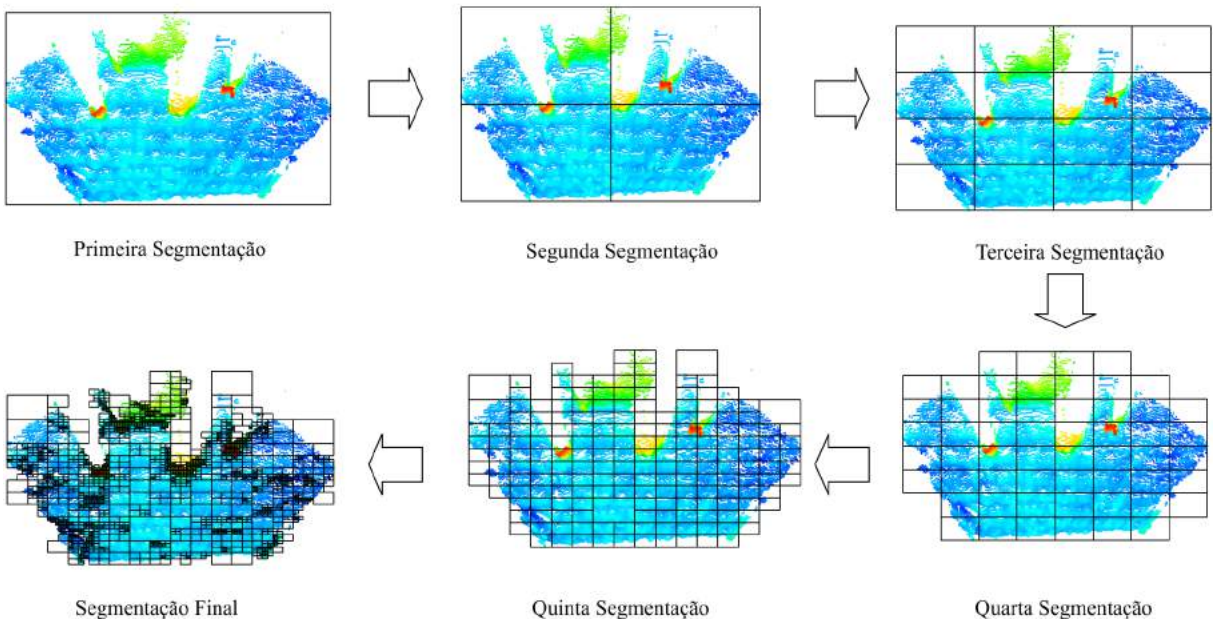


Figura 43 – Processo de Segmentação

5.4 Classificação da Nuvem de Pontos

Após a subdivisão da nuvem de pontos em vários blocos, uma rede neural artificial multi-camada (RNA) é utilizada para classificar os nós entre terreno e obstáculo. Três características de cada bloco é inserida na rede neural, essas são: desvio padrão da altura σ_z, perímetro do retângulo p, e a média da distância dos pontos até o plano principal d_f.

O conjunto de treinamento foi obtido a partir de oito nuvens de pontos distintas e foram manualmente marcados os pontos da nuvem como terreno e não terreno. Após a subdivisão da nuvem, cada nuvem contém centenas de amostras de treinamento, sendo cada bloco uma amostra. A Figura 44 apresenta uma representação do conjunto de treinamento no espaço de características.

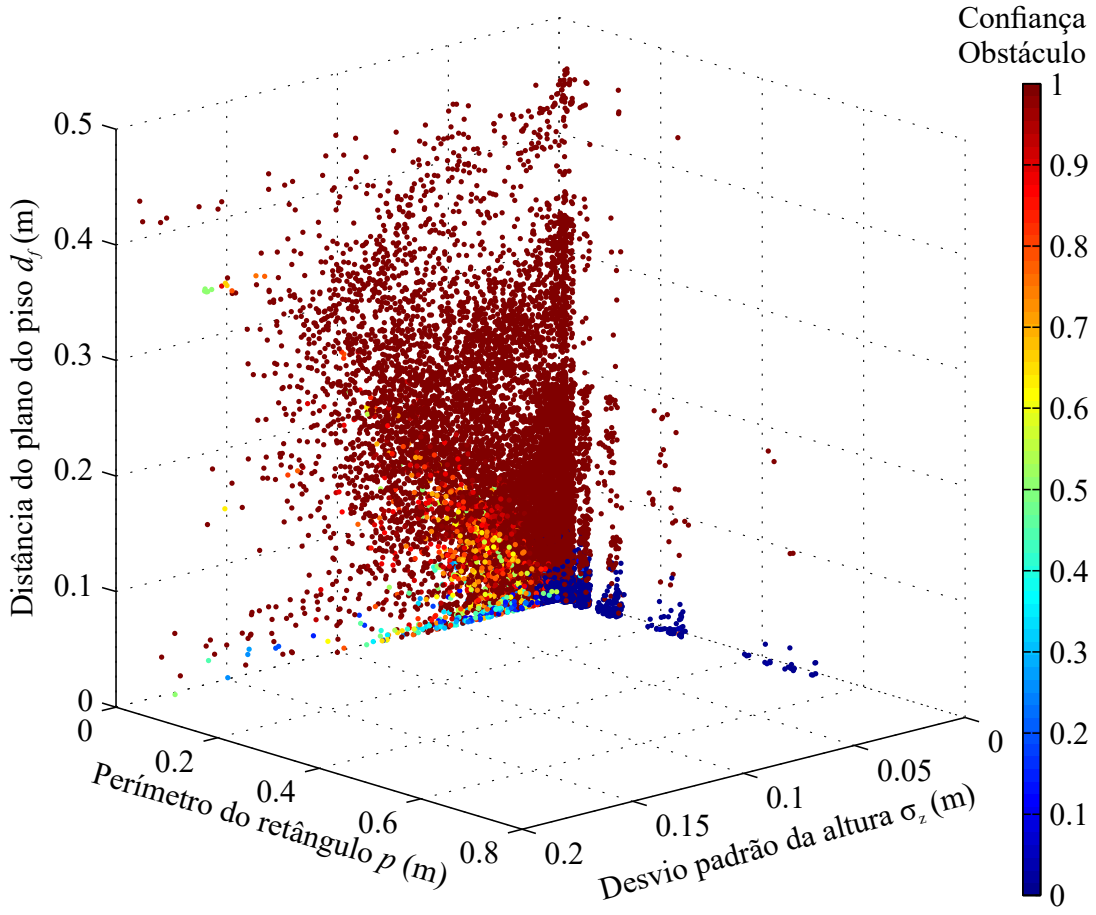


Figura 44 – Conjunto de Treinamento

A proporção da quantidade de pontos que são e não são terrenos de cada bloco é registrada como o nível de certeza que o respectivo bloco é ou não um terreno. A Figura 45 mostra o conjunto de treinamento da RNA, em que os níveis de certeza de certo ponto se é ou não terreno são mostrados com cores que vão de azul até vermelho respectivamente. Após execução da RNA, os pontos passam a estar classificados como terreno e não terreno. O Apêndice B apresenta um artigo recente submetido pelos autores que descreve em maiores detalhes o processo de detecção de obstáculos em nuvens de pontos.

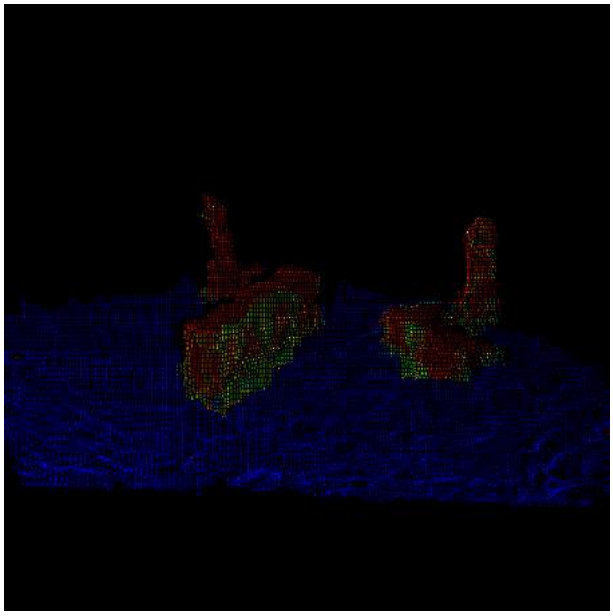


Figura 45 – Conjunto de Treinamento

5.5 Criação do mapa de objetos

Após a classificação dos pontos em terreno e não terreno, a última etapa do algoritmo trata de agrupar os pontos em objetos discretos. Através desse processo é gerado uma lista dos diferentes objetos do ambiente.

Inicialmente uma matriz K de dimensões W por H é criada. W é calculado como $\frac{N_{lx}}{N_{mw}}$, sendo N_{lx} o tamanho total da nuvem no eixo X e N_{mw} a largura do menor bloco criado na subdivisão. H é calculado análogo a W, mas com os valores do eixo Y e altura do menor bloco. Essa matriz será uma projeção ortogonal dos blocos de obstáculos criados pela árvore quaternária, com cada célula da matriz correspondente a área do menor bloco A_m, ou seja, se um bloco possui uma área de $4 \times A_m$, este irá ocupar quatro células da matriz.

Após a matriz K ser criada, a árvore quaternária é percorrida. Para cada nó folha da árvore, a área correspondente a este nó na matriz K é preenchida. Para cada célula da matriz K, é guardados o nó correspondente a esta célula. Após a árvore ser completamente percorrida, temos a matriz preenchida como na Figura 46 a). Na visualização da matriz cada cor representa um conjunto diferente de blocos marcados como não terreno.

Como é possível na Figura 46 a), existem pequenas granulações na matriz K devido às limitações do método de segmentação e classificação dos pontos. Para retirar essas granulações é aplicado um filtro de mediana b), assim deixando a matriz de projeção de obstáculos mais suave. O filtro de mediana resolve os problemas de granulação, mas alguns problemas de detecção de descontinuidade, ou seja, um mesmo objeto representado como

sendo mais do que um. Para isso é aplicado o método de dilatação após a aplicação do filtro c).

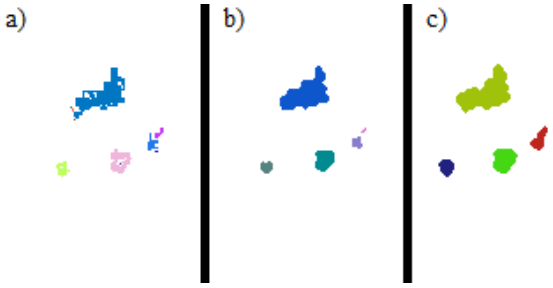


Figura 46 – Representação gráfica de um matriz K exemplo. (a) Sem filtro de mediana e sem dilatação. (b) Com filtro de mediana e sem dilatação. (c) Com filtro de mediana e com dilatação.

Após o filtro ser aplicado e a dilatação da imagem ser feita, é aplicado um método de crescimento de região em K os diferentes agrupamentos de objetos que K possui, e então esses agrupamentos são guardados em uma Lista L_a. L_a é percorrida e caso a área do agrupamento seja maior que um limiar mínimo A_t, o invólucro convexo e a centroide dos pontos do agrupamento é calculado. Uma Lista de objetos L_o guarda os objetos que foram encontrados, com seus respectivos pontos, e localização espacial baseada na centróide calculada. A Figura 47 mostra o resultado final.

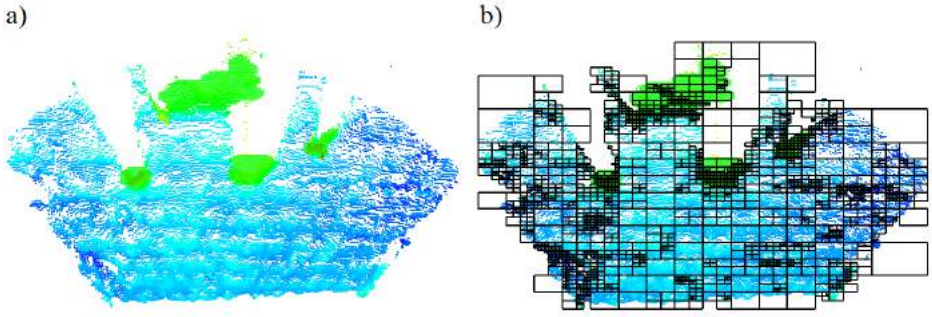


Figura 47 – Sobreposição dos objetos da matriz K sobre a visão ortogonal de uma nuvem de pontos. Cores representam altura e azul indica pontos mais baixos. Objetos da matriz K em verde. a) Nuvem e objetos de K. b) Nuvem, objetos de K e segmentação por árvore quaternária.

6 Validação e teste do modelo implementado

Do ponto de vista de controle, o robô proposto, em ambientes controlados, representa uma planta dinamicamente estável. A estabilidade, aliada ao sistema de controle por retroação, permite que um sistema de tomada de decisão simples possa comandar a locomoção do robô para a realização de tarefas específicas. Este sub-sistema, representado na Figura 26, tem implementação proposta a nível de software, como mostra a Figura 27. A definição dos objetivos deste sistema decisório é arbitrária e pode ser determinada de acordo com o uso da plataforma. No estudo de sistemas de mapeamento, por exemplo, este módulo de decisão pode ser configurado para varrer uma área seguindo um algoritmo simples de cobertura de espaço útil. Por outro lado, em um estudo que envolve coleta de itens, um módulo de decisão de maior complexidade pode ser implementado. A definição deste módulo, portanto, fica a critério dos usuários da plataforma.

Para demonstrar o funcionamento da plataforma e a integração de seus módulos, foram implementados dois sistemas de tomada de decisão semelhantes. O primeiro tem como objetivo a execução de uma trajetória pré-definida, que permite avaliar o comportamento do sistema de locomoção para a realização de trajetos específicos. O segundo é uma expansão do primeiro, tendo como objetivo a execução de uma trajetória pré-definida, porém seguindo os alvos que encontrar ao longo do caminho. Ambos os sistemas permitem também comprovar o funcionamento do sistema de comunicação, uma vez que a localização no referencial externo depende do recebimento de dados de posição originários da base de posicionamento e os dados de telemetria são constantemente transmitidos para a base de monitoramento. Este capítulo descreve estes sistemas de tomada de decisão e apresenta os resultados obtidos em laboratório.

6.1 Sistema decisório para execução de uma trajetória pré-definida (Sistema Decisório 1)

O sistema carrega na memória um caminho pré-estabelecido e direciona o robô a um ponto-chave, p_{dir}, do caminho. O direcionamento ocorre enviando uma velocidade de referência à camada de locomoção, cujo módulo é uma constante arbitrária e cuja direção, o_{ref}, aponta para p_{dir}. Tal direção é obtida através de $o_{ref} = \frac{p_{dir} - p_{robô}}{|p_{dir} - p_{robô}|}$. A relação entre estes pontos é apresentada geometricamente pela Figura 48.

À medida que o robô se aproxima desta posição objetivo, p_{dir}, a mesma se distancia, mantendo uma distância limiar constante e seguindo a trajetória pré-definida. Do ponto de vista de implementação, o caminho é uma lista de vetores. Durante a execução da

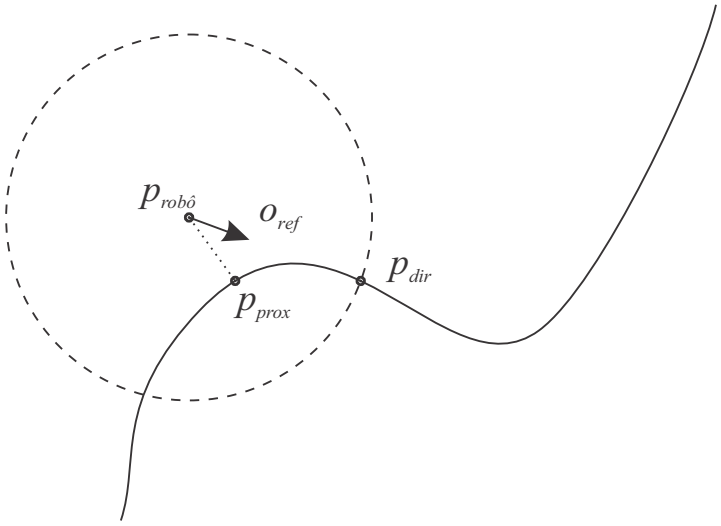


Figura 48 – Direção da velocidade de referência para que o robô siga uma trajetória pré-definida. p_{prox} indica o ponto do caminho mais próximo ao robô.

trajetória, um ponteiro salta para o próximo item da lista cada vez que a distância entre o vetor apontado e o robô seja menor que o limiar.

A Figura 49 apresenta as trajetórias resultantes da aplicação de variados valores de p_{dir}. Observa-se que valores pequenos de p_{dir} geram *overshoot*, enquanto valores grandes geram *undershoot*. Isso ocorre porque p_{dir} funciona como um fator de tolerância ao determinar se o robô chegou ou não a uma posição-objetivo do caminho. Valores pequenos aumentam as chances de o robô passar fora do ponto objetivo e ter que voltar a ele antes de prosseguir para os próximos pontos-objetivo do caminho. Isso causa *overshoot* e gera certa instabilidade. Valores grandes fazem com que o robô 'considere' que já alcançou seu ponto objetivo, estando ainda muito distante do mesmo, prosseguindo para o próximo ponto da lista. Isso faz aumentar a incidência de curvas 'cortadas', o que chamamos aqui de *undershoot*.

6.2 Sistema decisório para derrubada de obstáculos (Sistema Decisório 2)

O segundo sistema decisório permite avaliar o funcionamento integrado dos sistemas de locomoção e de visão estéreo. Este sistema é composto por dois sub-sistemas decisórios:

a: Sistema para execução de uma trajetória pré-definida; e

b: Sistema para seguir/derrubar obstáculos detectados pela visão estéreo.

O critério de escolha entre os dois sistemas é:

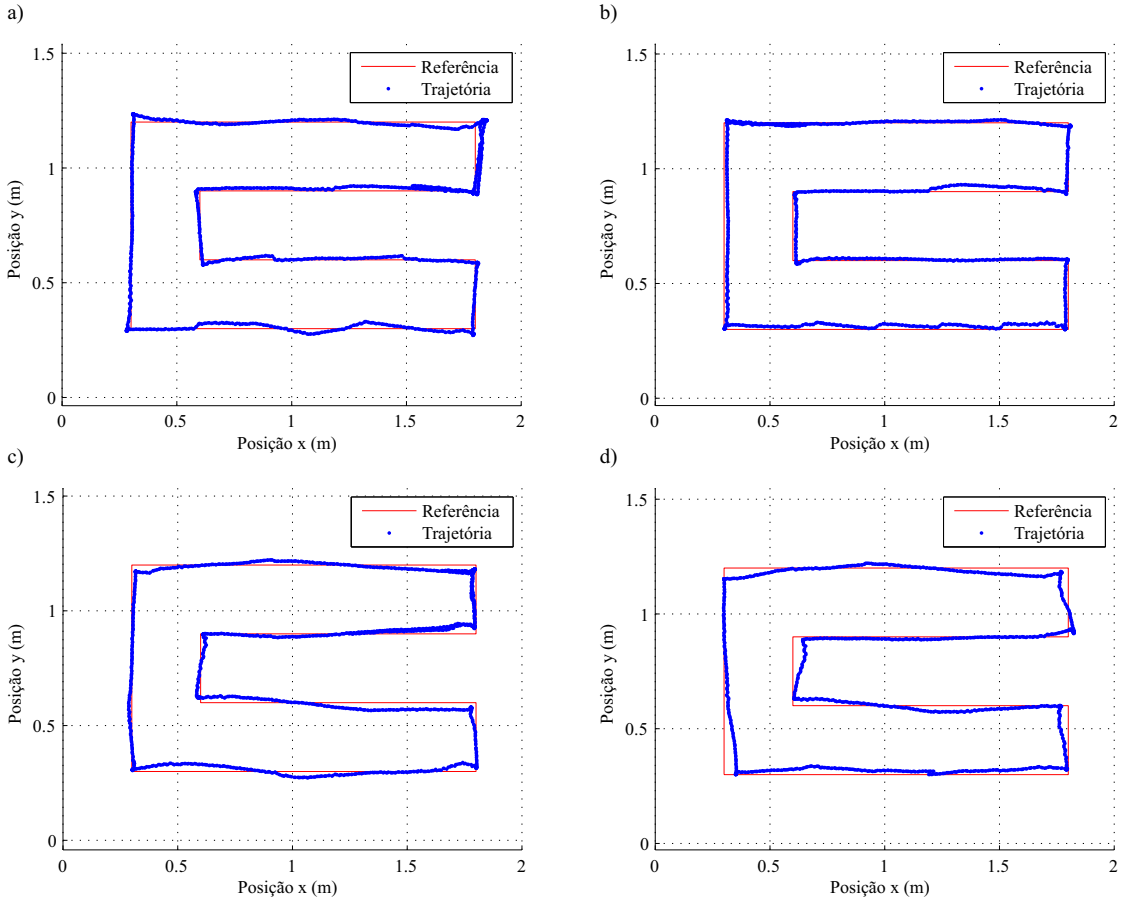


Figura 49 – Trajetória executada no sentido horário com sistema decisório (a) para variados valores de p_{dir}. (a) $p_{dir} = 3cm$. (b) $p_{dir} = 5cm$. (c) $p_{dir} = 7cm$. (d) $p_{dir} = 9cm$.

i: Sistema (a) se nenhum obstáculo detectado no momento; e

ii: Sistema (b) se algum obstáculo detectado no momento.

A execução alterna entre os sub-sistemas (a) e (b) dinamicamente segundo este critério. Como resultado, espera-se que o robô percorra sua trajetória pré-definida, porém mudando de direção dinamicamente para derrubar objetos dispostos em posições arbitrárias. Assim, espera-se demonstrar de maneira clara e simples a integração dos sub-sistemas para realização de uma tarefa específica, que se traduz em percorrer uma área e responder dinamicamente a estímulos eventuais. O comportamento demonstrado por este segundo sistema decisório é uma versão simplificada e generalizada de casos mais complexos e específicos, tais como a coleta de resíduos, resgate de vítimas, busca de passageiros, etc. O obstáculo utilizado para esta experimentação é apresentado pela Figura 50. Este elemento foi escolhido por apresentar tamanho e peso compatíveis com as dimensões do robô.

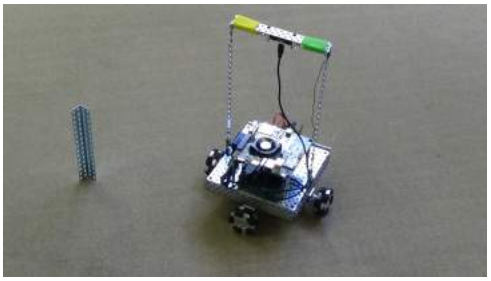


Figura 50 – Obstáculo utilizado para experimentação do sistema decisório (2), à esquerda do robô.

Durante a execução, objetos foram colocados em posições arbitrárias, próximos ao caminho do robô. Observou-se que em todos os casos em que os objetos se encontravam dentro de seu campo de visão, o robô se aproximou e derrubou com sucesso os objetivos. A Figura 51 apresenta um trecho da trajetória resultante da execução deste sistema decisório. Apenas um trecho é apresentado nesta figura para evitar a sobreposição das trajetórias, de forma a deixar mais clara a sequência de posições. Observe como o robô detectou o obstáculo a aproximadamente 80 centímetros de distância e se aproximou, derrubando-o com sucesso. Após a derrubada, o robô voltou à realização da trajetória, partindo para o próximo ponto do caminho.

6.3 Métricas de desempenho

Para avaliar a efetividade e o desempenho da execução de trajetórias e sugerir um ponto de partida para desenvolvimentos posteriores, propõem-se métricas de qualidade a seguir. A primeira métrica, v_{inc}, deriva do modelo cinemático do robô (Equação 4.9) e consiste de somas e subtrações entre as velocidades de cada roda. Esta métrica representa quão fiéis estas velocidades são à restrição linear que idealmente há entre elas. Quanto maior o módulo do valor calculado, mais conflitantes são as forças exercidas pelas rodas.

A segunda métrica, v_{erro}, consiste da diferença absoluta entre a velocidade medida pelo sistema de localização externo, v_{cam}, (subseção 4.2.3) e a velocidade medida pelo controlador de locomoção v_{loc}. Esta métrica reflete a discrepância ente as duas velocidades medidas, indicando a patinação das rodas sobre o piso. A Figura 52 apresenta dados coletados durante testes de locomoção referentes à patinação. Os dados apresentados foram coletados enquanto o robô seguia uma trajetória pré-definida a uma velocidade de referência constante de 0.600 m/s.

Note que aos dois segundos de execução, enquanto v_{loc} segue seu regime permanente, v_{cam} registrou uma desaceleração súbita. Esta desaceleração resultou em um pulso na métrica v_{erro}. Ente os segundos dez e 14 houve outra ocorrência do fenômeno, que neste caso se estendeu por mais tempo. Em ambas as ocasiões o robô patinou devido a irregularidades

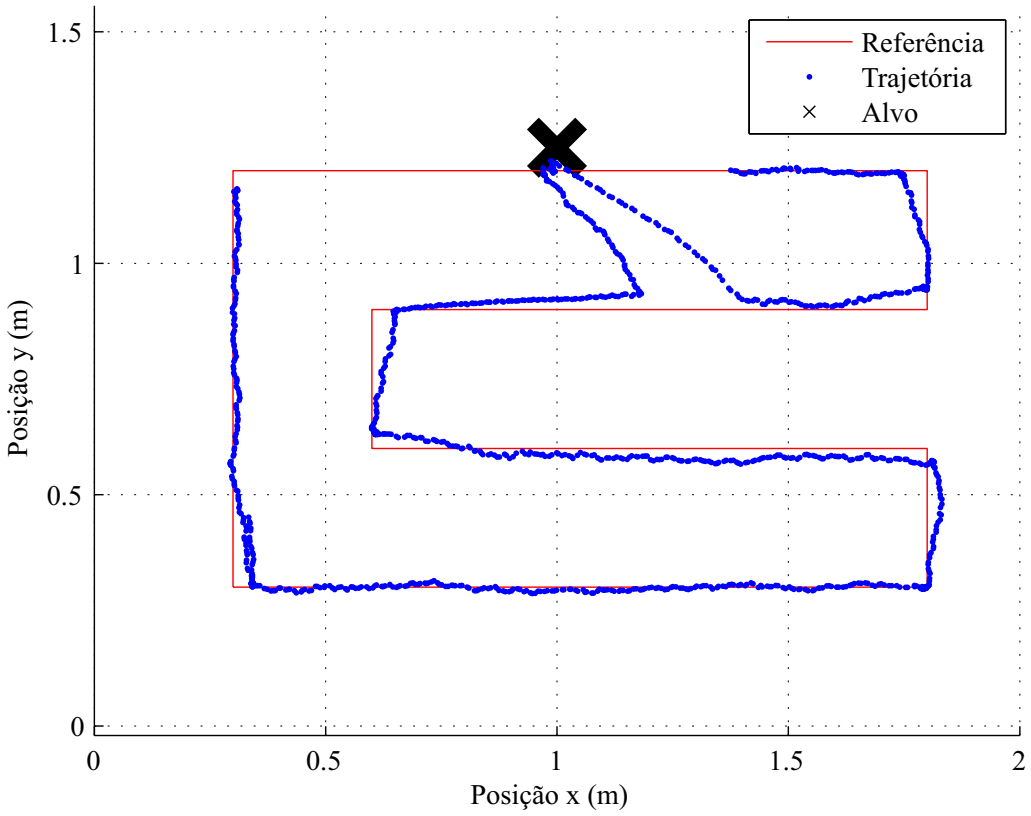


Figura 51 – Trajetória no sentido horário com sistema decisório (2).

no piso. Em caso de patinação das rodas, os sensores de rotação são incapazes de detectar alterações na velocidade. A câmera, entretanto, registra mais fielmente a velocidade do robô e assim a métrica v_{erro} se apresenta como um indicador de patinação.

Para quantificar a intensidade das curvaturas dos caminhos, propõe-se o uso de

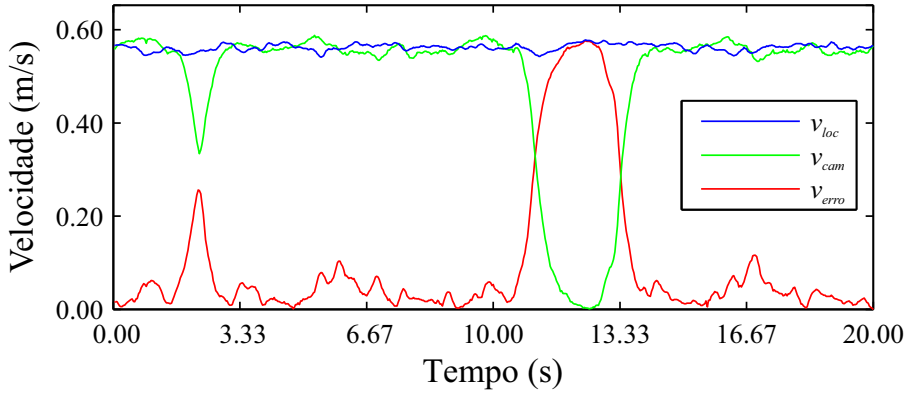


Figura 52 – Dados de v_{cam}, v_{loc} e v_{erro} coletados durante a execução de uma trajetória pré-definida.

duas posições-chave sobre o caminho, p_0 e p_1. Suas posições são parametrizadas por dois diferentes limiares de distância à frente do robô, ao longo da curva, como mostrado geometricamente pela Figura 53. O fator de curvatura f_{curv} é a projeção da diferença unitária entre p_0 e p_1, $\frac{p_0-p_1}{|p_0-p_1|}$, sobre o vetor normal à trajetória no ponto p_0:

$$f_{curv} = p_N \cdot \frac{p_0 - p_1}{|p_0 - p_1|} \tag{6.1}$$

A Figura 54 apresenta os valores calculados de f_{curv} ao longo de duas trajetórias distintas executadas pelo robô. Note que os valores mais altos de f_{curv} antecedem as curvas mais acentuadas.

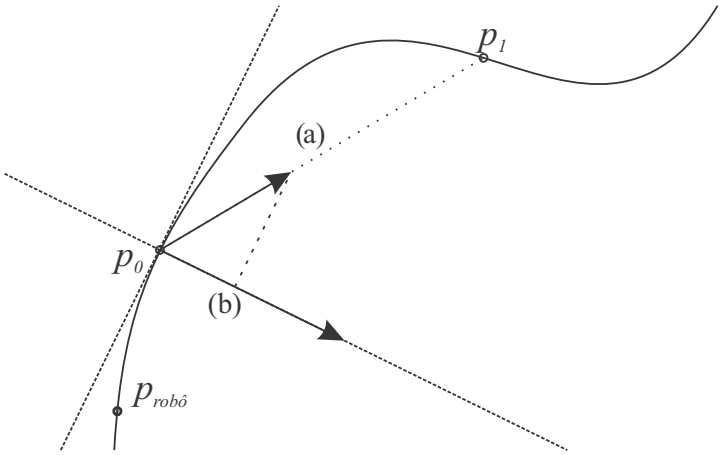


Figura 53 – Definição geométrica do fator de curvatura, f_{curv}. (a) Diferença unitária entre p_1 e p_0. (b) Projeção que define f_{curv}.

O erro de percurso, $e_{percurso}$ é definido como a distância linear absoluta entre $p_{robô}$ e p_{prox}, o ponto do caminho mais próximo ao robô. Este fator quantifica o quão fora o robô está do caminho a ser seguido. Para contabilizar o erro de percurso acumulado ao longo da trajetória, cujo caminho é definido por k posições consecutivas, se propõe o uso do erro médio quadrado sobre $e_{percurso}$:

$$E_{MSE} = \frac{1}{k}\sum_{i=1}^{k} |e_{percurso_i}|^2 \tag{6.2}$$

A Tabela 1 lista todas as métricas descritas. A Figura 55 mostra como as métricas se comportam em um trecho de trajetória executado em laboratório. Note como os picos da curva f_{curv}, calculada a partir da trajetória pré-definida, antecede os picos das curvas de v_{inc} e v_{erro}. Isso mostra como o fator de curvatura do trajeto podem ser utilizados para prever derrapamentos e erros de velocidade.

Os experimentos demonstraram qualitativamente e quantitativamente que o sistema decisório foi capaz de executar as trajetória pré-definidas com sucesso. O objetivo maior dos testes realizados foi demonstrar a integração do sistema e a capacidade de seus módulos

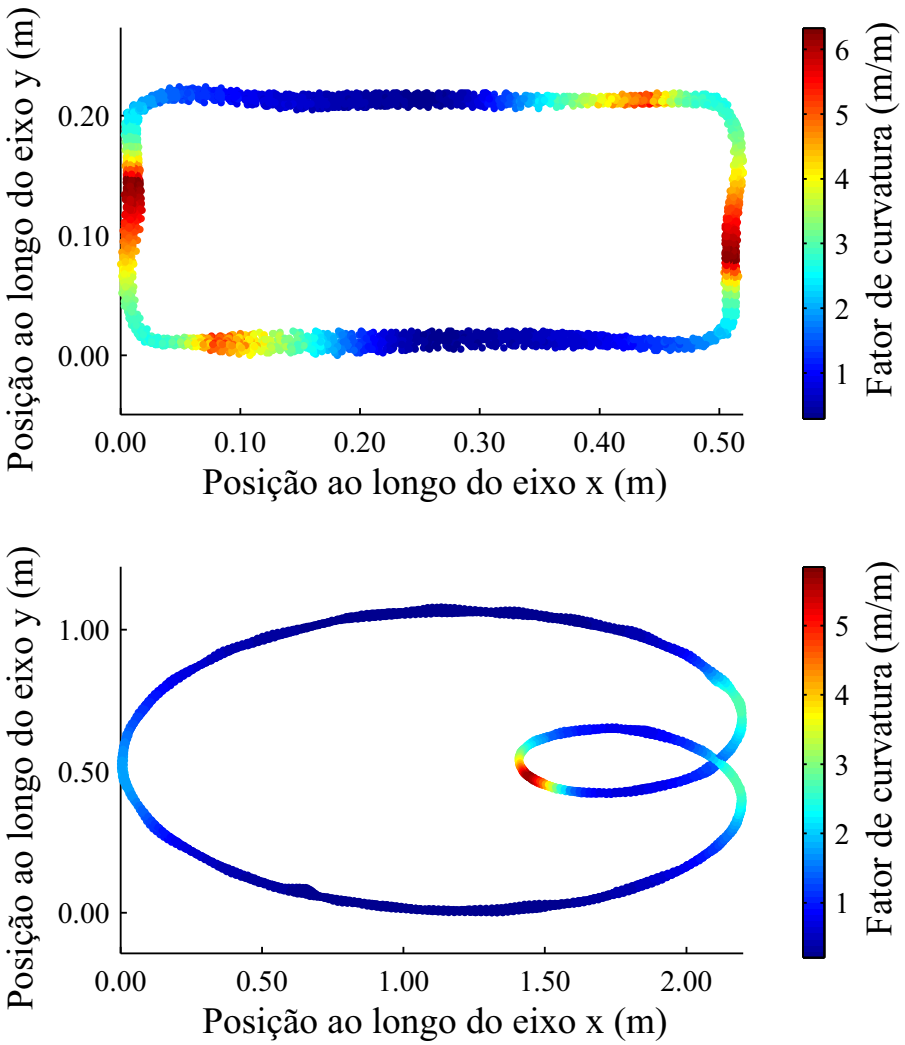


Figura 54 – Valores de f_{curv} calculados à medida que o robô executa uma trajetória retangular e uma trajetória em cardióide no sentido horário.

componentes de cooperar para completar uma tarefa específica. A simplicidade para o usuário final de definir via software um sistema decisório prova que o sistema está apto a cumprir o propósito para o qual foi proposto, de constituir uma plataforma modular e modificável para desenvolvimento e pesquisa em robótica móvel.

Tabela 1 – Métricas de desempenho de trajetória

Métrica	Símbolo	Fórmula	Unidade
Inconsistência de velocidade	v_{inc}	$\lvert v_1 - v_2 + v_3 - v_4 \rvert$	m/s
Erro de velocidade	v_{erro}	$\lvert v_{cam} - v_{loc} \rvert$	m/s
Fator de curvatura	f_{curv}	$p_N \cdot \frac{p_0 - p_1}{\lvert p_0 - p_1 \rvert}$	m/m
Erro de percurso	$e_{percurso}$	$\lvert p_{robô} - p_{prox} \rvert$	m
Erro MSE	E_{MSE}	$\frac{1}{k} \sum_{i=1}^{k} \lvert e_{percurso} \rvert^2$	m^2

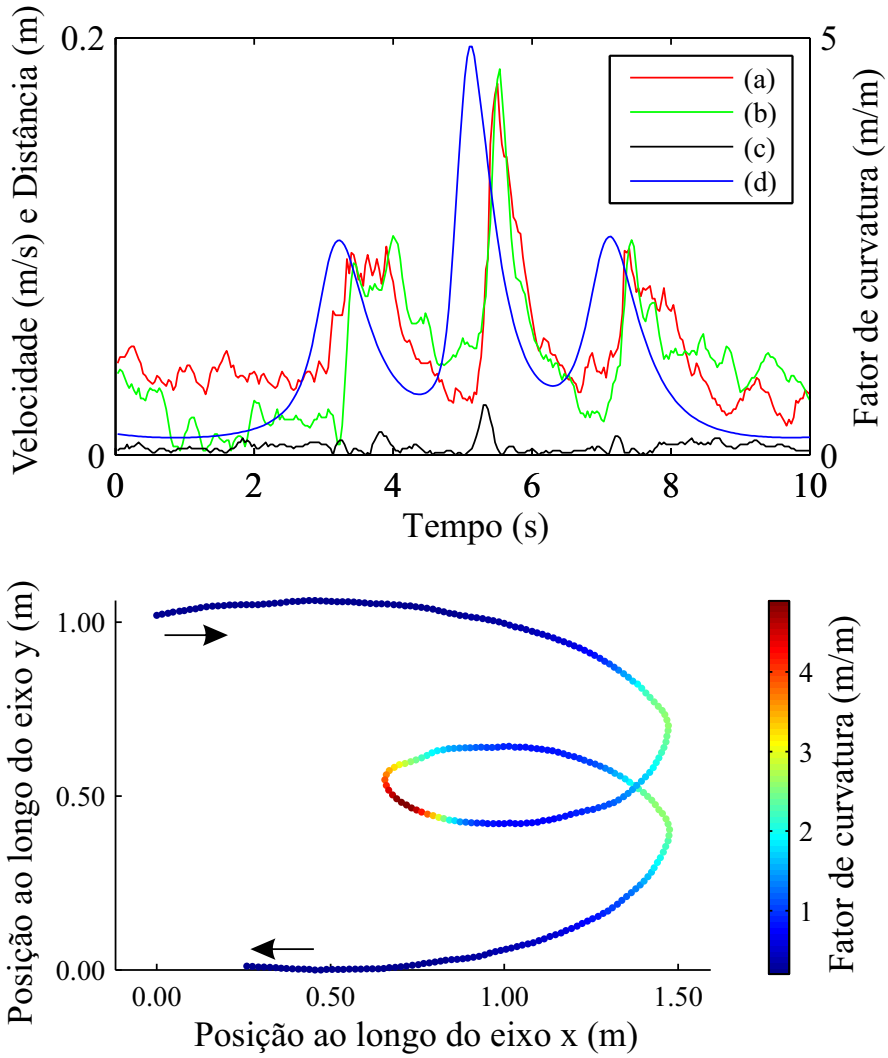


Figura 55 – Trecho de uma trajetória e os fatores de qualidade calculados ao longo de sua execução. (a) v_{inc}. (b) v_{erro}. (c) $e_{percurso}$. (d) f_{curv}.

7 Conclusão

Este trabalho detalhou os métodos e as técnicas adotadas na proposta e implementação de uma plataforma de pesquisa em robótica móvel. Os sub-sistemas propostos foram apresentados, do ponto de vista de hardware e de software. A efetividade e a integração dos módulos da implementação realizada foram avaliados através de testes que simulam casos de aplicação, tendo a plataforma sua efetividade comprovada.

A arquitetura proposta, entretanto, tem como objetivo funcionar como uma base para desenvolvimento a aprimoramento contínuo. Sua constituição modular tem o propósito de permitir que seus componentes sejam aprimorados de forma independente e que novos módulos sejam acrescentados. Com isso, a implementação aqui apresentada objetiva apenas demonstrar o funcionamento básico do modelo modular proposto, e constitui o passo inicial em um processo iterativo de desenvolvimento contínuo. Futuros desenvolvimentos podem seguir caminhos diversos, tanto no campo da robótica e da integração de sistemas, como em áreas específicas, como na avaliação de métodos alternativos para controle, tomada de decisões, visão, etc.

Referências

ALTERA. *Arria V GX FPGA Development Kit*. 2016. Portal Altera. Disponível em: <https://www.altera.com/products/boards_and_kits/dev-kits/altera/kit-arria-v-gx.html>. Acesso em: 23 jan. 2016. Citado na página 21.

ARDUINO. *Getting Started with Arduino*. 2016. Portal Arduino. Disponível em: <http://www.arduino.cc/en/Guide/HomePage>. Acesso em: 23 jan. 2016. Citado na página 16.

ASHMORE, M.; BARNES, N. Omni-drive robot motion on curved paths: The fastest path between two points is not a straight-line. In: MCKAY, B.; SLANEY, J. (Ed.). *AI 2002: Advances in Artificial Intelligence*. Springer Berlin Heidelberg, 2002, (Lecture Notes in Computer Science, v. 2557). p. 225–236. ISBN 978-3-540-00197-3. Disponível em: <http://dx.doi.org/10.1007/3-540-36187-1_20>. Citado na página 59.

BEAGLEBONE. *What is BeagleBone Black?* 2016. Portal BeagleBone. Disponível em: <http://beagleboard.org/BLACK>. Acesso em: 23 jan. 2016. Citado na página 17.

CHESTER, D. L. Why two hidden layers are better than one. In: *Proceedings of the international joint conference on neural networks*. [S.l.: s.n.], 1990. v. 1, p. 265–268. Citado na página 39.

CHROBOTICS. *UM7-LT Orientation Sensor*. 2016. Portal CHRobotics. Disponível em: <https://www.chrobotics.com/shop/um7-lt-orientation-sensor>. Acesso em: 23 jan. 2016. Citado na página 28.

CIRESAN, D. C. et al. Flexible, high performance convolutional neural networks for image classification. In: *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*. [S.l.: s.n.], 2011. v. 22, n. 1, p. 1237. Citado na página 45.

COUCEIRO, M. S. et al. A low-cost educational platform for swarm robotics. *International Journal of Robots, Education and Art*, 2011. Citado na página 12.

CYBENKO, G. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, Springer, v. 2, n. 4, p. 303–314, 1989. Citado na página 39.

DEBIAN. *Debian on arm*. 2016. Portal Debian. Disponível em: <http://www.debian.org/ports/arm/>. Acesso em: 23 jan. 2016. Citado na página 16.

DIGI. *Wireless Mesh Networking: ZigBee® vs. DigiMesh™*. 2016. Portal Digi International Inc. Disponível em: <http://www.digi.com/pdf/wp_zigbeevsdigimesh.pdf>. Acesso em: 23 jan. 2016. Citado 2 vezes nas páginas 25 e 26.

DIGI. *XBee® 802.15.4*. 2016. Portal Digi International Inc. Disponível em: <http://www.digi.com/products/wireless-wired-embedded-solutions/zigbee-rf-modules/point-multipoint-rfmodules/xbee-series1-module>. Acesso em: 23 jan. 2016. Citado na página 25.

DIGI. *XBee® Wi-Fi*. 2016. Portal Digi International Inc. Disponível em: <http://www.digi.com/products/wireless-wired-embedded-solutions/zigbee-rf-modules/point-multipoint-rfmodules/xbee-wi-fi>. Acesso em: 23 jan. 2016. Citado na página 25.

DIGI. *XBee® ZigBee*. 2016. Portal Digi International Inc. Disponível em: <http://www.digi.com/products/xbee-rf-solutions/modules/xbee-zigbee>. Acesso em: 23 jan. 2016. Citado na página 25.

DIGILENT. *Cmod S6*. 2016. Portal Digilent. Disponível em: <https://www.digilentinc.com/Products/Detail.cfm?NavPath=2,400,1227&Prod=CMOD-S6>. Acesso em: 23 jan. 2016. Citado 2 vezes nas páginas 21 e 22.

DIGILENT. *Nexys^{TM}4 DDR Artix-7 FPGA Board*. 2016. Portal Digilent. Disponível em: <https://www.digilentinc.com/Products/Detail.cfm?NavPath=2,400,1338&Prod=NEXYS4DDR>. Acesso em: 23 jan. 2016. Citado na página 21.

DIGILENT. *PmodRF2 - IEEE 802.15 RF Transceiver*. 2016. Portal Digilent. Disponível em: <https://www.digilentinc.com/Products/Detail.cfm?NavPath=2,401,927&Prod=PMOD-RF2>. Acesso em: 23 jan. 2016. Citado na página 25.

DIGILENT. *PmodWiFi - 802.11b WiFi Interface*. 2016. Portal Digilent. Disponível em: <https://www.digilentinc.com/Products/Detail.cfm?NavPath=2,401,884&Prod=PMOD-WIFI>. Acesso em: 23 jan. 2016. Citado na página 25.

DIGILENT. *ZedBoard Zynq^{TM}-7000 Development Board*. 2016. Portal Digilent. Disponível em: <https://www.digilentinc.com/Products/Detail.cfm?NavPath=2,400,1028&Prod=ZEDBOARD>. Acesso em: 23 jan. 2016. Citado na página 22.

DIGILENT. *ZYBO Zynq^{TM}-7000 Development Board*. 2016. Portal Digilent. Disponível em: <https://www.digilentinc.com/Products/Detail.cfm?NavPath=2,400,1198&Prod=ZYBO>. Acesso em: 23 jan. 2016. Citado na página 22.

DOUILLARD, B. et al. On the segmentation of 3d lidar point clouds. In: IEEE. *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. [S.l.], 2011. p. 2798–2805. Citado na página 51.

DUO3D. *Duo Calibration Circles*. 2016. Duo Calibration Circles. Disponível em: <http://duo3d.com/docs/content/files/DUO_CALIBRATION_CIRCLES.pdf>. Acesso em: 23 jan. 2016. Citado na página 72.

ELINUX. *Ubuntu on BeagleBone Black*. 2016. Portal Elinux. Disponível em: <http://elinux.org/Beagleboard:Ubuntu_On_BeagleBone_Black>. Acesso em: 23 jan. 2016. Citado na página 17.

FREESCALE. *Freescale Freedom Development Boards*. 2016. Portal Freescale. Disponível em: <http://www.freescale.com/webapp/sps/site/overview.jsp?code=FREDEVPLA>. Acesso em: 23 jan. 2016. Citado na página 16.

FUNAHASHI, K.-I. On the approximate realization of continuous mappings by neural networks. *Neural networks*, Elsevier, v. 2, n. 3, p. 183–192, 1989. Citado na página 39.

FUSIELLO, A.; TRUCCO, E.; VERRI, A. A compact algorithm for rectification of stereo pairs. *Machine Vision and Applications*, Springer, v. 12, n. 1, p. 16–22, 2000. Citado na página 48.

GIZMOSPHERE. *Gizmo 2*. 2016. Portal GizmoSphere. Disponível em: <http://www.gizmosphere.org/products/gizmo-2/>. Acesso em: 23 jan. 2016. Citado na página 18.

GOMPERTS, A.; UKIL, A.; ZURFLUH, F. Implementation of neural network on parameterized fpga. In: *AAAI Spring Symposium: Embedded Reasoning*. [S.l.: s.n.], 2010. Citado na página 45.

HIRSCHMÜLLER, H. Accurate and efficient stereo processing by semi-global matching and mutual information. In: IEEE. *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. [S.l.], 2005. v. 2, p. 807–814. Citado na página 50.

HIRSCHMÜLLER, H. Stereo processing by semiglobal matching and mutual information. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, IEEE, v. 30, n. 2, p. 328–341, 2008. Citado na página 51.

HORNIK, K.; STINCHCOMBE, M.; WHITE, H. Multilayer feedforward networks are universal approximators. *Neural networks*, Elsevier, v. 2, n. 5, p. 359–366, 1989. Citado na página 39.

HSU, T.-R. *MEMS & microsystems: design, manufacture, and nanoscale engineering*. [S.l.]: John Wiley & Sons, 2008. Citado na página 28.

HUSH, D. R.; HORNE, B. G. Progress in supervised neural networks. *Signal Processing Magazine, IEEE*, IEEE, v. 10, n. 1, p. 8–39, 1993. Citado na página 39.

IEEE. *IEEE 802.11TM Wireless Local Area Networks*. 2016. The Working Group for WLAN Standards. Disponível em: <http://www.ieee802.org/11/>. Acesso em: 23 jan. 2016. Citado na página 24.

IEEE. *IEEE 802.15 Working Group for WPAN*. 2016. The IEEE 802.15 Working Group. Disponível em: <http://www.ieee802.org/15/>. Acesso em: 23 jan. 2016. Citado na página 25.

INTEL. *Intel® Dual Band Wireless-AC 7260 Plus Bluetooth*. 2016. Portal Intel. Disponível em: <http://www.intel.com/content/www/us/en/wireless-products/dual-band-wireless-ac-7260-bluetooth.html>. Acesso em: 24 jan. 2016. Citado na página 25.

INTEL. *Intel® Edison Compute Module, Boards, and Kits*. 2016. Portal Intel. Disponível em: <http://www.intel.com/content/www/us/en/do-it-yourself/edison.html>. Acesso em: 23 jan. 2016. Citado na página 17.

INTEL. *Intel® Edison—One Tiny Platform, Endless Possibility*. 2016. Portal Intel. Disponível em: <http://www.intel.com/content/www/us/en/do-it-yourself/edison.html>. Acesso em: 23 jan. 2016. Citado na página 17.

KECMAN, V. *Learning and Soft Computing: Support Vector Machines, Neural Networks, and Fuzzy Logic Models*. Bradford Book, 2001. (A Bradford book). ISBN 9780262112550. Disponível em: <https://books.google.com.br/books?id=W5SAhUqBVYoC>. Citado 5 vezes nas páginas 33, 34, 37, 39 e 41.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2012. p. 1097–1105. Citado na página 45.

KUURKOVA, V. Kolmogorov's theorem and multilayer neural networks. *Neural networks*, Elsevier, v. 5, n. 3, p. 501–506, 1992. Citado na página 39.

LABORATORIES, I. D. C. *DUO DDK - Product Breif*. 2016. Portal Duo3d. Disponível em: <https://duo3d.com/public/pdf/duo-ddk-lv2.pdf>. Acesso em: 23 jan. 2016. Citado na página 21.

LABORATORIES, I. D. C. *DUO M - Product Breif*. 2016. Portal Duo3d. Disponível em: <https://duo3d.com/product/duo-mini-lv1>. Acesso em: 23 jan. 2016. Citado na página 30.

MARCON, G. A. et al. An adaptive algorithm for embedded real-time point cloud ground segmentation. In: *Soft Computing and Pattern Recognition (SoCPaR), 2015 7th International Conference on*. [S.l.: s.n.], 2015. p. 76–83. ISBN 978-1-4673-9360-7/15. Citado na página 74.

MARCON, G. A. et al. Performance analysis for a novel adaptive algorithm for real-time point cloud ground segmentation. *The International Journal of Hybrid Intelligent Systems (IJHIS)*, IOS Press, v. 12:3, 2016. ISSN 1448-5869. Citado na página 74.

MAXBOTIX. *Ultrasonic Sensors for OEMs, Engineers, Distributors and Educators*. 2016. Portal MaxBotix Inc. Disponível em: <http://www.maxbotix.com/>. Acesso em: 23 jan. 2016. Citado na página 29.

MICROCHIP. *Microchip and Digilent Launch First Arduino^{TM} Compatible 32-bit Microcontroller Development Platform*. 2016. Portal Microchip Technology. Disponível em: <http://www.microchip.com/pagehandler/en-us/chipkit-development-platform.html>. Acesso em: 23 jan. 2016. Citado na página 16.

MICROCHIP. *Microchip RN171XV*. 2016. Portal Microchip. Disponível em: <http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en560635>. Acesso em: 23 jan. 2016. Citado na página 25.

MIKROELEKTRONIKA. *PIC Development Boards*. 2016. Portal MikroElektronika Embedded Solutions. Disponível em: <http://www.mikroe.com/pic/development-boards/>. Acesso em: 23 jan. 2016. Citado na página 16.

MIKROKOPTER. *HC-06 Bluetooth Module*. 2016. Portal MikroKopter. Disponível em: <http://wiki.mikrokopter.de/en/HC-06>. Acesso em: 23 jan. 2016. Citado na página 24.

MUNRO, P.; GERDELAN, A. P. Stereo vision computer depth perception. *Country United States City University Park Country Code US Post code*, Citeseer, v. 16802, 2009. Citado na página 46.

MURRAY, R. M.; SASTRY, S. S.; ZEXIANG, L. *A Mathematical Introduction to Robotic Manipulation*. 1st. ed. Boca Raton, FL, USA: CRC Press, Inc., 1994. ISBN 0849379814. Citado na página 76.

NORDIC. *2.4GHz RF System-on-Chip (SoC) with Flash and USB, nRF24LU1+*. 2016. Portal Nordic Simiconductor. Disponível em: <http://www.nordicsemi.com/eng/Products/2.4GHz-RF/nRF24L01>. Acesso em: 23 jan. 2016. Citado na página 24.

NVIDIA. *Jetson TK1*. 2016. Portal eLinux.org. Disponível em: <http://elinux.org/Jetson_TK1>. Acesso em: 20 jan. 2016. Citado 2 vezes nas páginas 18 e 19.

NVIDIA. *The World's First Embedded Supercomputer*. 2016. Portal Nvidia. Disponível em: <http://www.nvidia.com/object/jetson-tk1-embedded-dev-kit.html>. Acesso em: 23 jan. 2016. Citado na página 18.

OMONDI, A. R.; RAJAPAKSE, J. C. *FPGA implementations of neural networks*. [S.l.]: Springer, 2006. Citado na página 44.

OPENGL. *OpenGL - The Industry's Foundation for High Performance Graphics*. 2016. Portal OpenGL. Disponível em: <http://www.opengl.org/>. Acesso em: 23 jan. 2016. Citado na página 52.

PASSINO, K. M. *Biomimicry for optimization, control, and automation*. [S.l.]: Springer Science & Business Media, 2005. Citado na página 32.

POGGIO, T.; GIROSI, F. Networks for approximation and learning. *Proceedings of the IEEE*, IEEE, v. 78, n. 9, p. 1481–1497, 1990. Citado na página 39.

PULSEDLIGHT. *LIDAR-Lite Laser Module*. 2016. Portal PulsedLight. Disponível em: <http://pulsedlight3d.com/products/lidar-lite>. Acesso em: 23 jan. 2016. Citado na página 29.

RASPBERRYPI. *The making of pi*. 2016. Portal Raspberry pi Foundation. Disponível em: <https://www.raspberrypi.org/about/>. Acesso em: 23 jan. 2016. Citado na página 17.

ROCCO, M. D.; GALA, F. L.; ULIVI, G. Testing multirobot algorithms: Saetta: A small and cheap mobile unit. *Robotics Automation Magazine, IEEE*, v. 20, n. 2, p. 52–62, June 2013. ISSN 1070-9932. Citado na página 12.

ROCKEL, S. *A Multi-Robot Platform for Mobile Robots with Multi-Agent Technology*. Tese (Doutorado) — University of Hamburg, 2011. Citado na página 12.

ROJAS, R.; FÖRSTER, A. G. Holonomic control of a robot with an omnidirectional drive. *KI-Künstliche Intelligenz*, v. 20, n. 2, p. 12–17, 2006. Citado na página 59.

ROSENBLATT, F. Principles of neurodynamics. Spartan Book, 1962. Citado na página 35.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. *Learning internal representations by error propagation*. [S.l.], 1985. Citado na página 37.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. *Cognitive modeling*, v. 5, 1988. Citado na página 37.

VELODYNE. *HDL-64E LiDAR sensor*. 2016. Portal Velodyne. Disponível em: <http://velodynelidar.com/lidar/hdlproducts/hdl64e.aspx>. Acesso em: 23 jan. 2016. Citado na página 29.

VEX. *Aluminum Kits*. 2016. VEX Robotics. Disponível em: <http://www.vexrobotics.com/aluminum-kits.html>. Acesso em: 23 jan. 2016. Citado 2 vezes nas páginas 30 e 31.

WIDROW, B.; HOFF, M. E. et al. Adaptive switching circuits. Defense Technical Information Center, 1960. Citado na página 33.

XILINX. *FPGA vs. ASIC*. 2015. Portal Xilinx. Disponível em: <http://www.xilinx.com/fpga/asic.htm>. Acesso em: 14 jun. 2015. Citado na página 19.

XILINX. *What is a FPGA?* 2015. Portal Xilinx. Disponível em: <http://www.xilinx.com/fpga>. Acesso em: 14 jun. 2015. Citado na página 20.

ZHU, J.; SUTTON, P. Fpga implementations of neural networks–a survey of a decade of progress. In: *Field Programmable Logic and Application*. [S.l.]: Springer, 2003. p. 1062–1066. Citado na página 44.

Anexos

ANEXO A – Guia de construção do robô LEIA 1.

LEIA – Laboratório para Educação e Inovação em Automação
Escola de Engenharia elétrica, Mecânica e de Computação
Universidade Federal de Goiás

GUIA DE CONSTUÇÂO DO ROBÔ LEIA 1

Gilberto Antonio Marcon dos Santos

Goiânia,

Janeiro de 2016

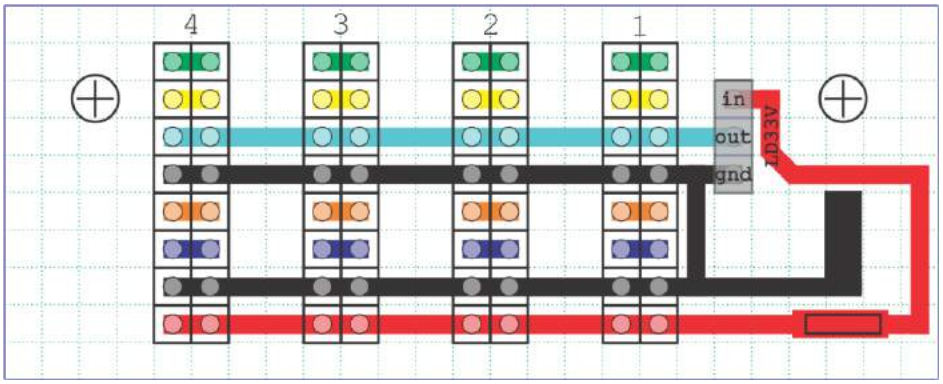
1. Lista de componentes utilizados na construção do robô. Elementos genéricos, encontráveis em lojas de eletrônica locais e lojas de ferragens foram omitidos da lista. Preços em US$.

Cod	Nome	Loja	C. Loja	Preço	Qtd	Pkt	Preço Total
AA	Hinge	Vex	275-1272	$ 9.99	2	2	$ 9.99
AB	Plate 15x5	Vex	275-2023	$ 4.99	4	2	$ 9.98
AC	Angle 2x2x35	Vex	275-1143	$ 17.99	3	4	$ 13.49
AD	Gusset Pack	Vex	276-1110	$ 7.49	2	2	$ 7.49
AE	Bar 1x25	Vex	275-1141	$ 12.99	2	8	$ 3.25
AF	Standoff 1.00”	Vex	275-1016	$ 3.95	2	10	$ 0.79
AG	Standoff 2.00”	Vex	275-1018	$ 7.95	8	10	$ 6.36
BA	Screw 8-32 x 0.250”	Vex	275-1002	$ 7.49	36	100	$ 2.70
BB	Screw 8-32 x 0.375”	Vex	275-1003	$ 7.49	10	100	$ 0.75
BC	Screw 8-32 x 0.500”	Vex	275-1004	$ 7.49	28	100	$ 2.10
CA	Nut 8-32 Nylock	Vex	275-1027	$ 3.99	28	100	$ 1.12
CB	Nut 8-32 Keps	Vex	275-1026	$ 2.99	34	100	$ 1.02
DA	Bracket	Pololu	2676	$ 7.45	4	2	$ 14.90
DB	Hub	Robotshop	RB-Nex-78	$ 10.30	4	1	$ 41.20
DC	Wheel	Robotshop	RB-Nex-75	$ 15.45	4	1	$ 61.80
DD	Gearmotor	Pololu	2273	$ 36.95	4	1	$ 147.80
EA	DUO	Duo	-	$ 595.00	1	1	$ 595.00
EB	Cytron	Robotshop	RB-Cyt-132	$ 13.82	4	1	$ 55.28
EC	12 V Reg	Pololu	2568	$ 13.95	1	1	$ 13.95
ED	TK1	Nvidia	-	$ 192.00	1	1	$ 192.00
EE	Bateria	RbtMktplc	0-DTXC1864	$ 44.99	1	1	$ 44.99
EF	7260HMW	Amazon	-	$ 9.99	1	1	$ 9.99
EG	Antena	Amazon	-	$ 5.98	2	2	$ 5.98
EH	Perma-Proto Half	Adafruit	571	$ 12.50	1	3	$ 4.17
EI	FTDI Serial-USB	SparkFun	9716	$ 14.95	1	1	$ 14.95
EJ	3.3 V Reg	Pololu	526	$ 1.95	1	1	$ 1.95
EK	Bat. Plug	Robotshop	RB-Cyt-51	$ 1.32	1	1	$ 1.32
EL	Cmod S6	Digilent	-	$ 69.00	1	1	$ 69.00
EM	HDMI head	Amazon	-	$ 14.98	1	1	$ 14.98
FA	Cabo F-F 6 in Marrom	Pololu	-	$ 2.49	1	10	$ 0.25
FB	Cabo F-F 6 in Vermelho	Pololu	-	$ 2.49	6	10	$ 1.49
FC	Cabo F-F 6 in Preto	Pololu	-	$ 2.49	10	10	$ 2.49
FD	Cabo F-F 6 in Verde	Pololu	-	$ 2.49	4	10	$ 1.00
FE	Cabo F-F 6 in Amarelo	Pololu	-	$ 2.49	4	10	$ 1.00
FF	Cabo F-F 6 in Alaranjado	Pololu	-	$ 2.49	4	10	$ 1.00
FG	Cabo F-F 6 in Azul	Pololu	-	$ 2.49	4	10	$ 1.00
FH	Cabo F-F 6 in Roxo	Pololu	-	$ 2.49	2	10	$ 0.50
FI	Cabo F-F 6 in Cinza	Pololu	-	$ 2.49	2	10	$ 0.50
FJ	Cabo F-F 6 in 6 in Rosa	Pololu	-	$ 2.49	1	10	$ 0.25
FK	Cabo F-F 6 in Branco	Pololu	-	$ 2.49	1	10	$ 0.25
FL	Cabo F-F 12 in Preto	Pololu	-	$ 4.49	2	10	$ 0.90
FM	Cabo F-F 12 in Roxo	Pololu	-	$ 4.49	2	10	$ 0.90
FN	Cabo F-F 12 in Cinza	Pololu	-	$ 4.49	2	10	$ 0.90
GA	Crimp 1x2-Pin	Pololu	1901	$ 0.69	13	25	$ 0.36
GB	Crimp 1x3-Pin	Pololu	1902	$ 0.79	8	25	$ 0.25
GC	Crimp 1x4-Pin	Pololu	1903	$ 0.59	4	10	$ 0.24
HA	Headers	SparkFun	116	$ 1.50	93	40	$ 3.49
				VEX:			$ 59.03
				Total			$ 1,365.02

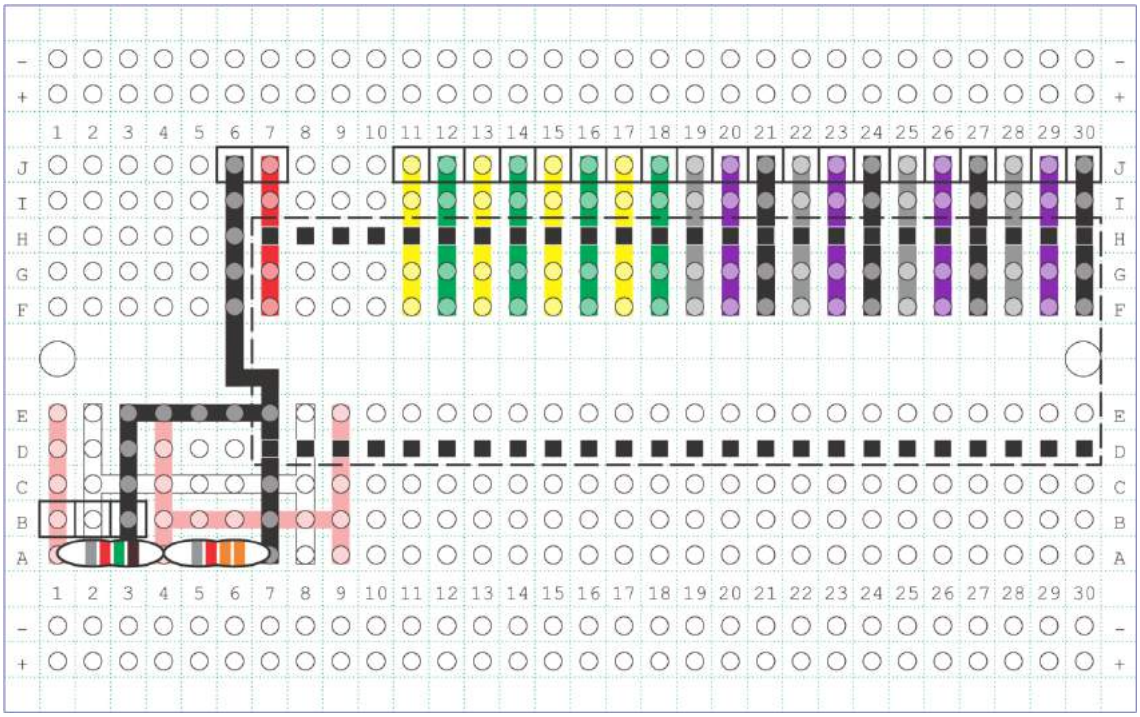
2. Instalação ISE
   2.1. Instale o software ISE. Se receber mensagens de erro ao executar o software, prossiga:
   2.2. Abra o diretório **C:\Xilinx\14.6\ISE_DS\ISE\lib\nt64** e busque por **"libPortability"**. O resultado da busca deve retornar o seguinte arquivo: **libPortability.dll**
   2.3. Renomeie **"libPortability.dll"** para **"libPortability.dll.orig"**. Caso algo errado ocorra, este backup pode ser usado para reverter o original.
   2.4. Copie e cole **"libPortabilityNOSH.dll"** no mesmo diretório. Nomeie a cópia como **"libPortability.dll"**.
   2.5. Os passos anteriores são geralmente suficientes para resolver os problemas. Porém, se a instalação ainda apresentar problemas, siga os passos em ArquivosAuxiliares/ISEInstall.txt na íntegra.
3. Configuração da placa de FPGA Cmod S6. Estes passos programam a memória FLASH do dispositivo, de forma que o controlador para os motores ficará gravado no mesmo após a remoção da fonte de energia.
   3.1. O projeto a ser sintetizado está acessível através do repositório git: **git clone https://leialabufg@bitbucket.org/leialabufg/cmods6_motionlayer.git**
   3.2. Sintetize o projeto ISE e certifique-se de que não ocorreram erros.
   3.3. Abra o software iMPACT, parte da distribuição ISE.
   3.4. Clique em **'Create PROM File ...'** no painel esquerdo **'iMPACT Flows'.**
   3.5. Mantena todas as opções as mesmas e clique na **seta verde**.
   3.6. Clique em '**Auto Select PROM**' e clique na **segunda seta verde**.
   3.7. Antes de clicar em **OK**, certifique-se de que:
      A) O campo **'Output File Location'** contém o diretório do projeto.
      B) **'File Format'** está como **'MCS'**.
   3.8. Clique em **OK** em **'Add Device - Start adding device file to ...'**.
   3.9. Selecione o arquivo **bit stream** do diretório do projeto.
   3.10. Clique em **No** em **'Add Device - Would you like to add ...'**.
   3.11. Clique em **OK** em **'Add Device - You have completed the ...'**.
   3.12. Clique em **'Generate File...'** no painel esquerdo **'iMPACT Processes'**.
   3.13. Clique em **'Boundary Scan'** no painel esquerdo **'iMPACT Flows'**.
   3.14. Clique com botão direito na area principal e selecione **'Add Xilinx Device'**.
   3.15. Selecione o arquivo **bit stream**.
   3.16. Dê duplo clique na etiqueta **SPI/BPI** sobre o ícone **FPGA**.
   3.17. Selecione o arquivo **.mcs**.
   3.18. Selecione **'S25FL128S'** e clique em **OK**.
   3.19. Clique em **Yes** em **'iMPACT Warning - The ...'**.
   3.20. Selecione o ícone **FLASH** sobre o ícone **FPGA**.
   3.21. Clique em **'program'** e em seguida clique em **OK**.
   3.22. O processo vai levar algum tempo. Ao fim, a programação foi finalizada.

4. Placas

4.1. Utilize **uma placa perfurada genérica** para construir a placa de potência como indicado. Insira e solde **64 headers machos (HA)** nas posições indicadas pelas formas retangulares. Insira e solde **um regulador de tensão LD33V (EJ)** como indicado e insira **um plug Deans macho (EK)**. Faça as conexões como nas cores indicadas. Perfure nas duas posições de parafuso utilizando broca de 4 mm.

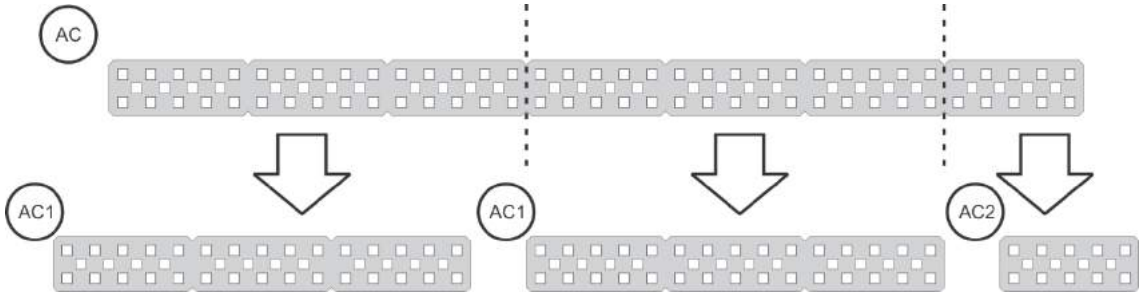


4.2. Utilize **uma perma-proto half (EH)** para construir a placa de potência como indicado. Aumente o tamanho dos furos de parafuso usando broca de 4 mm de forma que um parafuso VEX padrão (BA, BB, BC) possa passar. Insira e solde **25 headers (HA)** nas posições indicadas. Insira e solde **dois resistores (1.5 KΩ e 3.3 KΩ)** e **dois soquetes de 24 pinos** como indicado. Faça as conexões como nas cores indicadas.

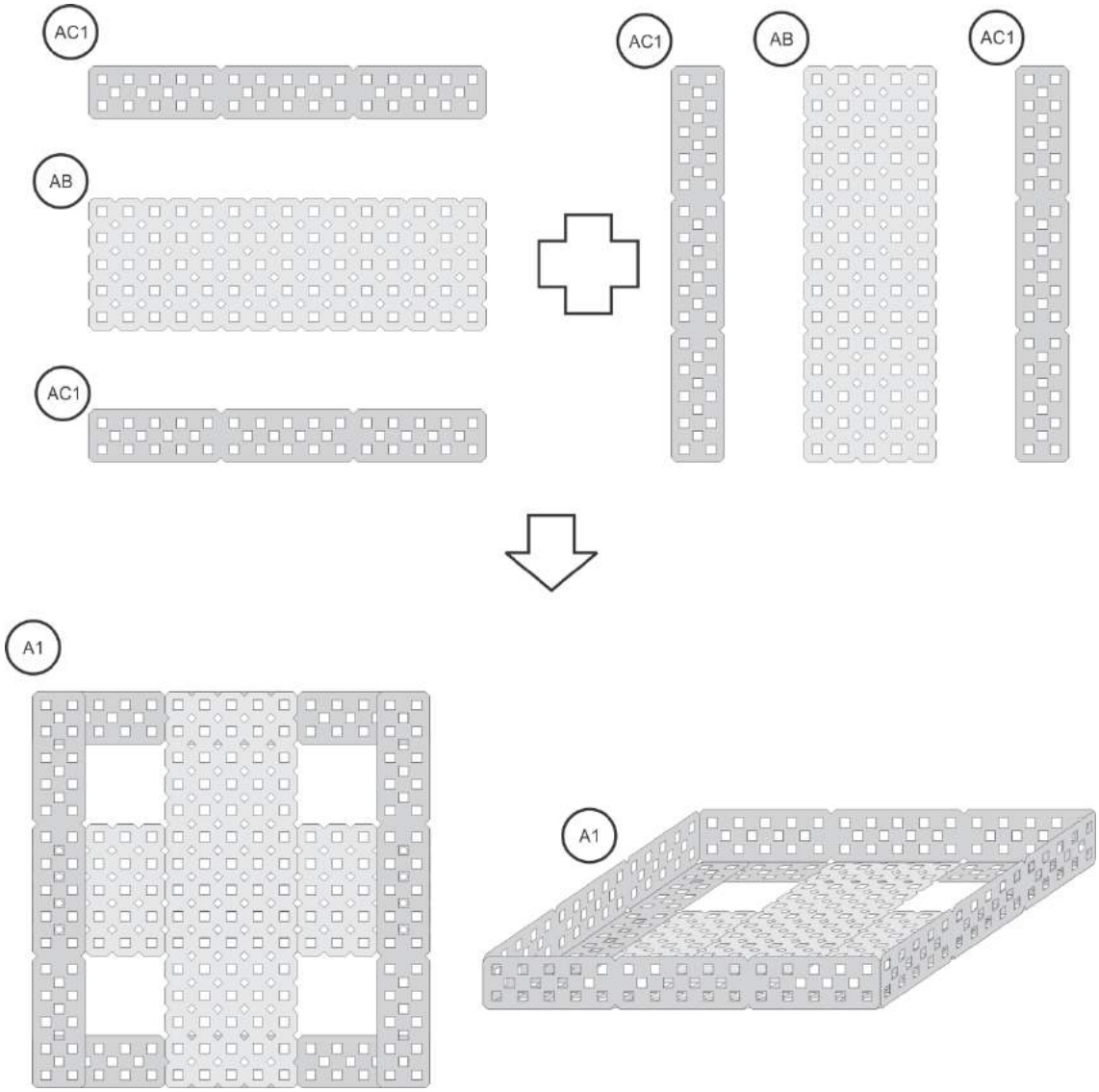


5. Chassi

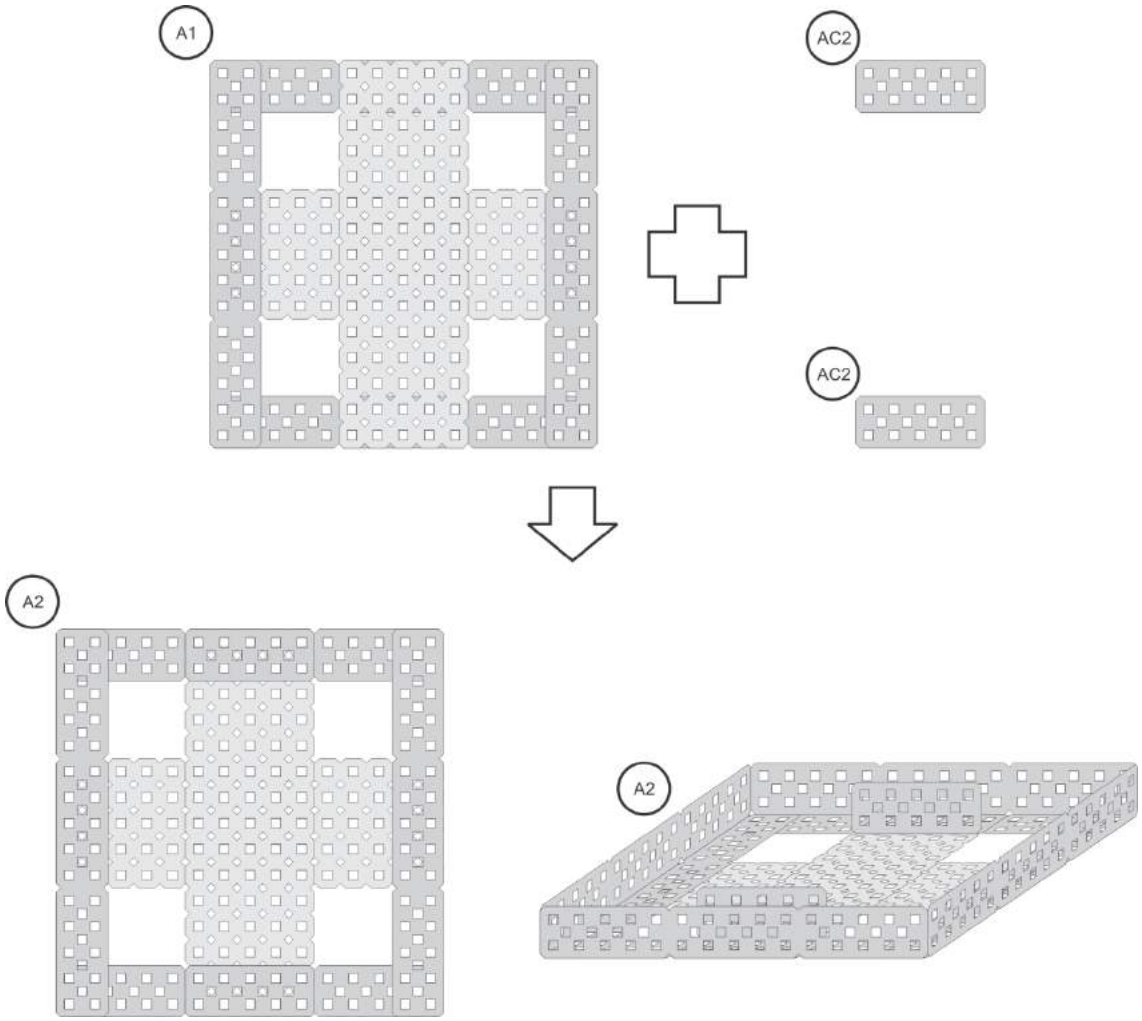
5.1. Corte **três barras (AC)** como ilustrado, obtendo seis barras AC1 e três AC2.



5.2. Monte a parte A1 a partir de quatro peças AC1 e **duas chapas (AB),** como indicado. Posicione as peças verticais sobre as peças horizontais. As abas verticais das peças AC1 devem ficar no exterior, tal que a peça A1 se assemelhe a uma caixa.



5.3. Monte a parte A2 como indicado. Insira duas peças AC2 no interior da caixa A1. As abas verticais das peças AC2 devem ficar no lado mais interior, de forma a comprimir a bateria.



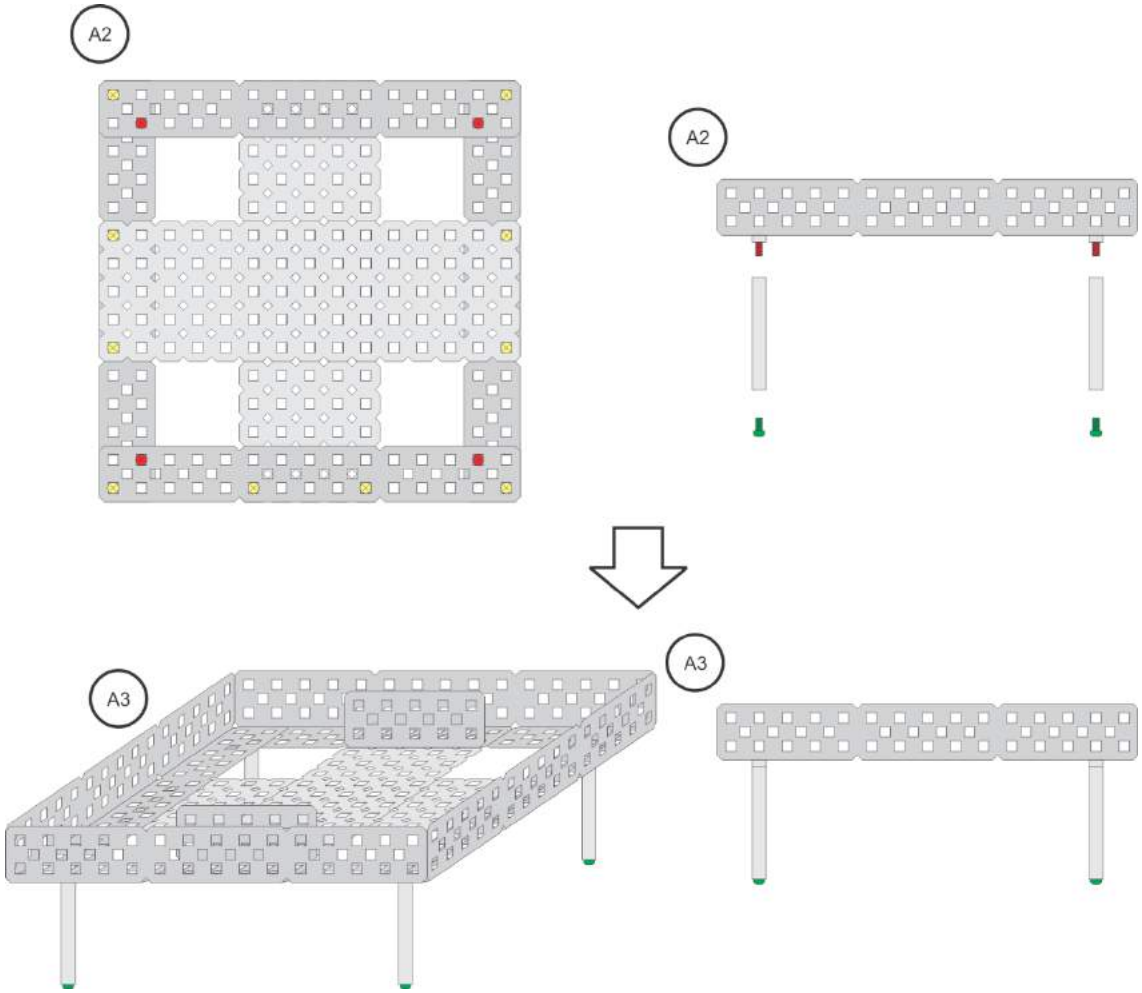
5.4. Monte a parte A3 como indicado.

A) Parafuse utilizando **10 parafusos (BB)** e **10 porcas (CA)** nos locais indicados em amarelo. As cabeças destes parafusos devem ficar do lado externo (inferior) da caixa A2.

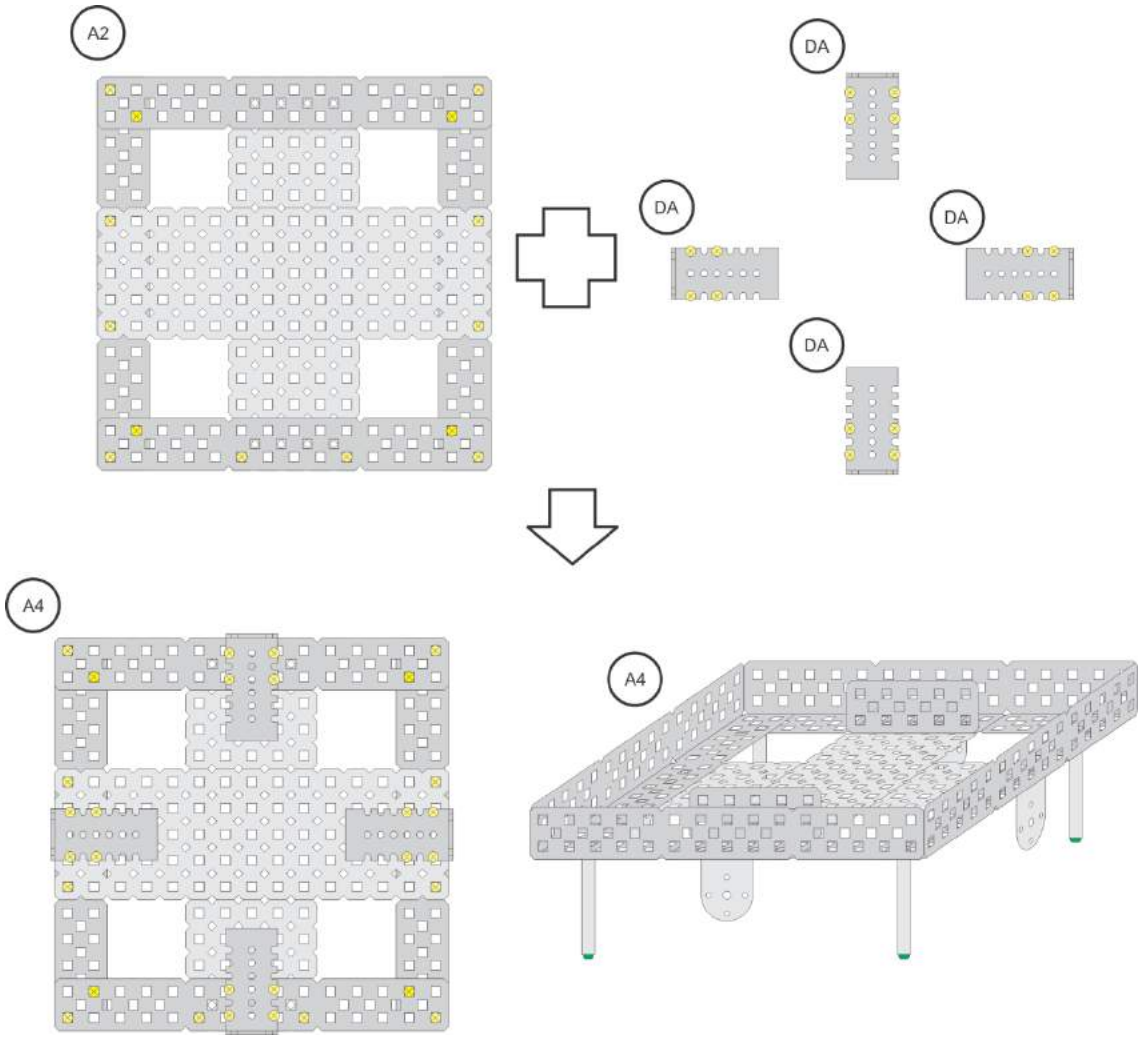
B) Parafuse utilizando **quatro parafusos (BC)** e **quatro porcas (CB)** nos locais indicados em vermelho. As cabeças destes parafusos devem ficar do lado interno da caixa.

C) Parafuse nas pontas destes quatro parafusos **quatro standoffs (AG)**.

D) Parafuse **quatro parafusos (BC)** nas pontas dos standoffs, como indicado na figura.



5.5. Monte a parte A4 como indicado. Parafuse **4 brackets (DA)** sobre a peça A3 como mostrado, de forma que fiquem na parte inferior da caixa e que as abas verticais dos mesmos fiquem nas extremidades exteriores. Utilize **16 parafusos (BC)** e **16 porcas (CA)**, mantendo a cabeça do mesmo no lado exterior da caixa.



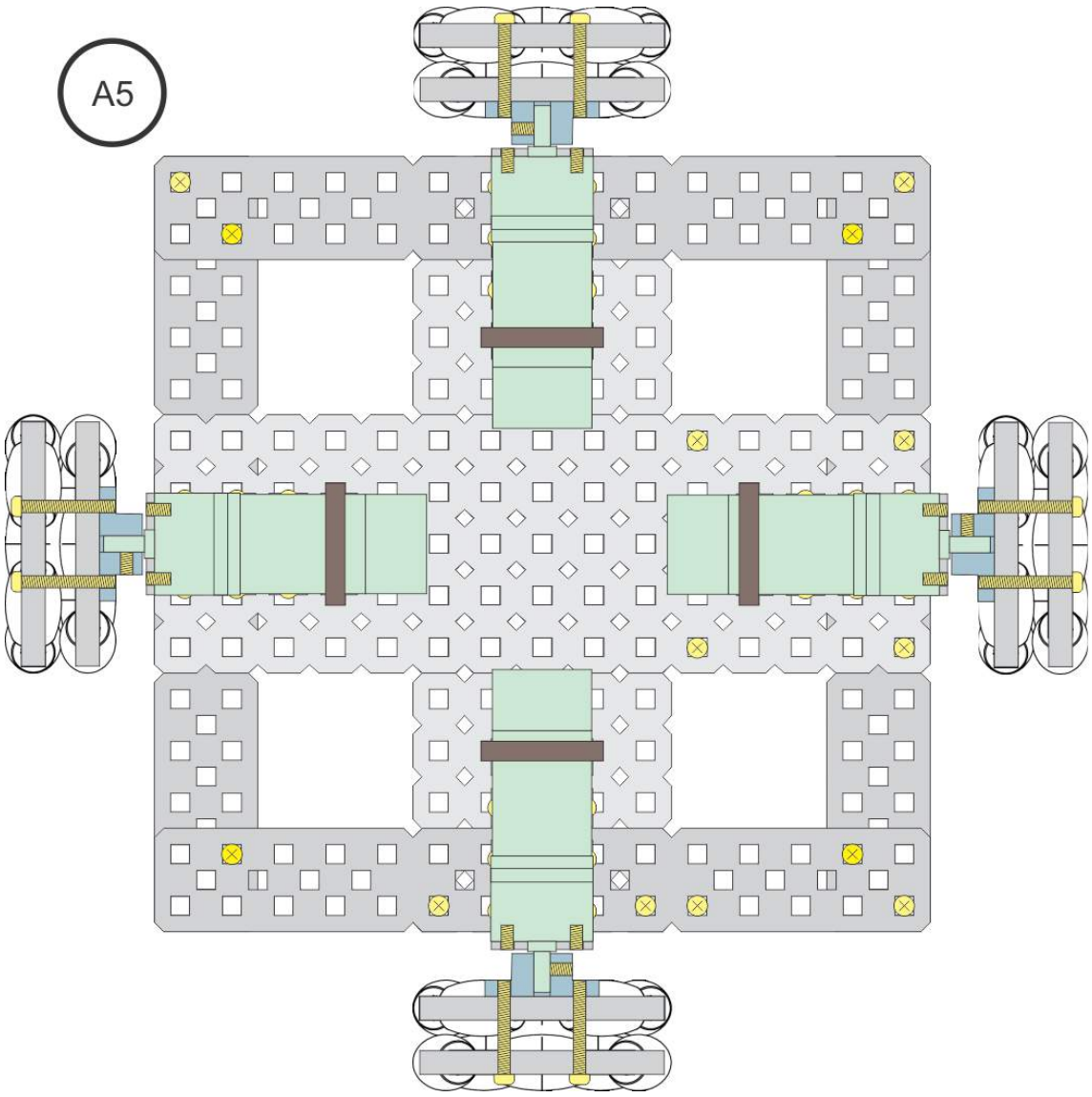
5.6. Monte a parte A5 como indicado.

A) Fixe **quatro hubs (DB)** a **quatro rodas (DC)**.

B) Fixe **quatro motores (DD)** aos quatro brackets da parte A4.

C) Fixe os quatro hubs aos quatro motores.

D) Reforce o apoio dos motores sobre os brackets usando **quatro braçadeiras** em torno de cada motor.

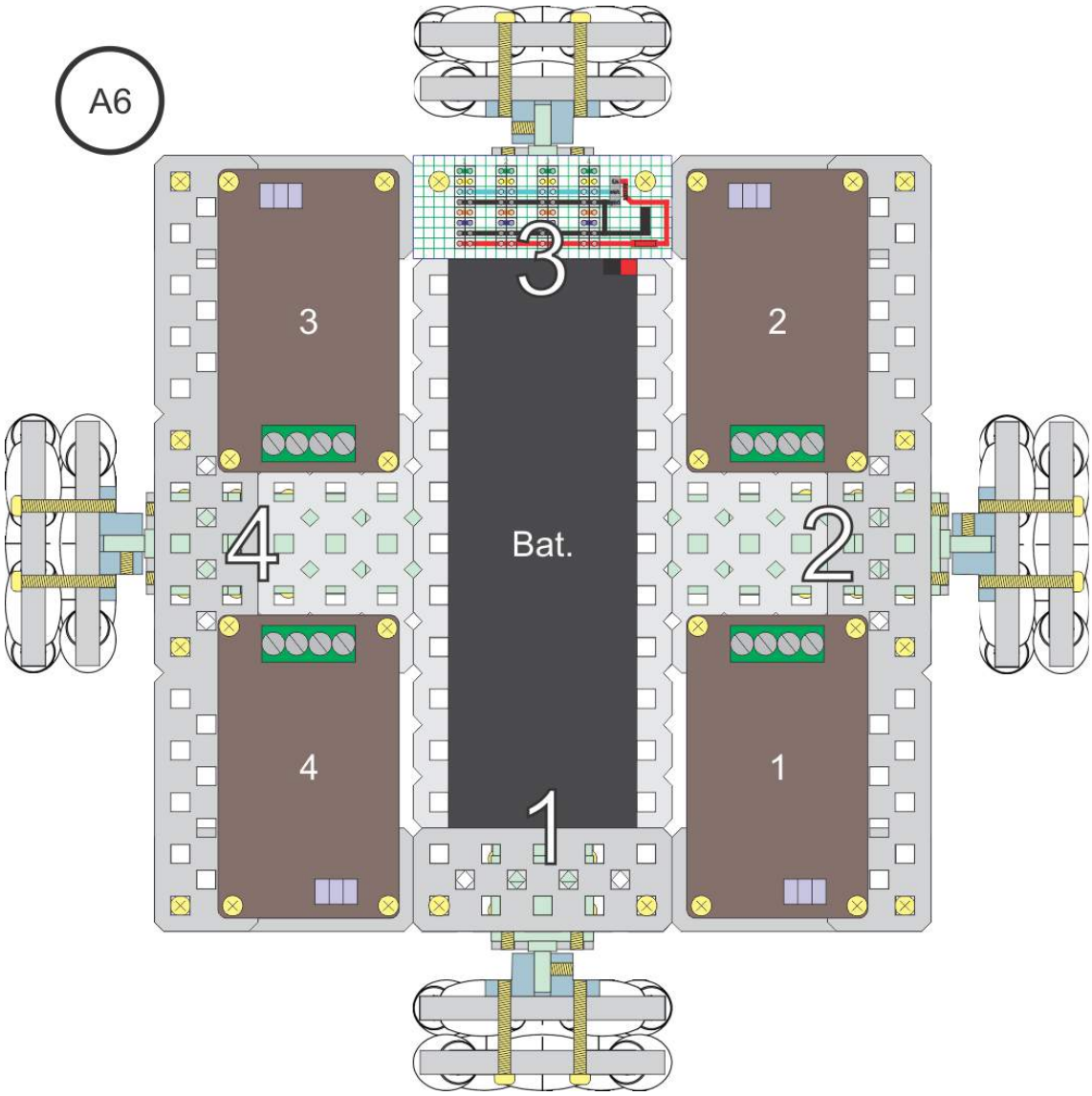


5.7. Monte a parte A6 como indicado.

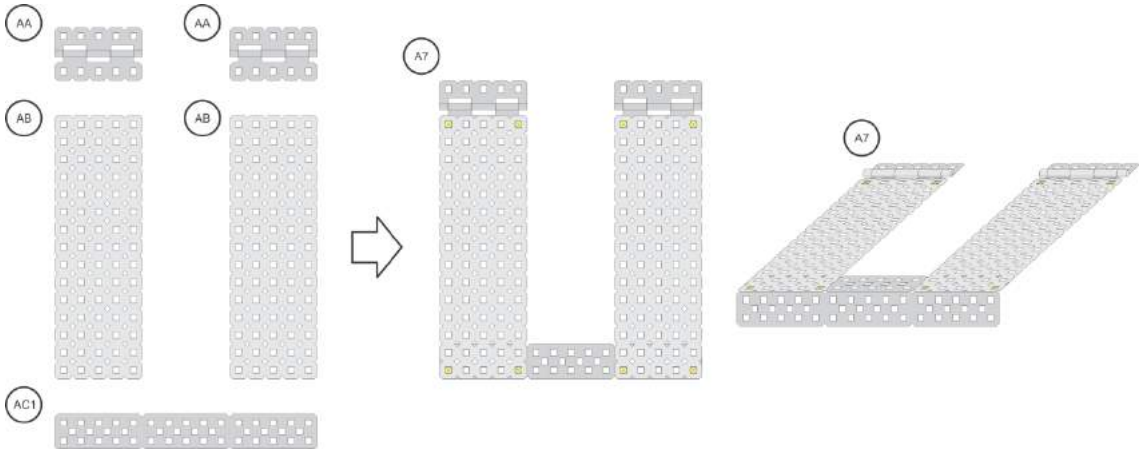
A) Posicione **quatro placas controladoras de corrente (EB)** como indicado. Perfure a base onde necessário (dois furos por placa, nas posições mais interiores à caixa). Utilize **16 espaçadores de plástico** para isolamento entre as placas e a estrutura. Utilize **16 parafusos**, **16 porcas** e **16 arruelas** genéricas.

B) Parafuse a placa de potência na posição indicada. Para tal, utilize **dois parafusos (BC) e duas porcas (CA)** colocando-os com a cabeça do lado da placa. Coloque um recorte de espuma entre a placa e o chassi, para isolamento. O recorte deve ter o mesmo formato que a placa. Utilize **dois separadores plásticos** entre a espuma e o chassi.

C) Posicione **uma bateria (EE)** como ilustrado.



5.8. Monte a parte A7 como indicado. Una **duas dobradiças (AA)**, **duas chapas (AB)** e uma barra AC1 como indicado (vista superior), mantendo as chapas AB sobre as demais peças. A aba vertical de AC1 deve aportar para baixo, no lado exterior da peça. O lado saliente da dobradiça deve ficar no lado superior. Utilize **oito parafusos (BA)** e **oito porcas (CB)** nos locais indicados, mantendo as cabeças dos parafusos do lado superior.

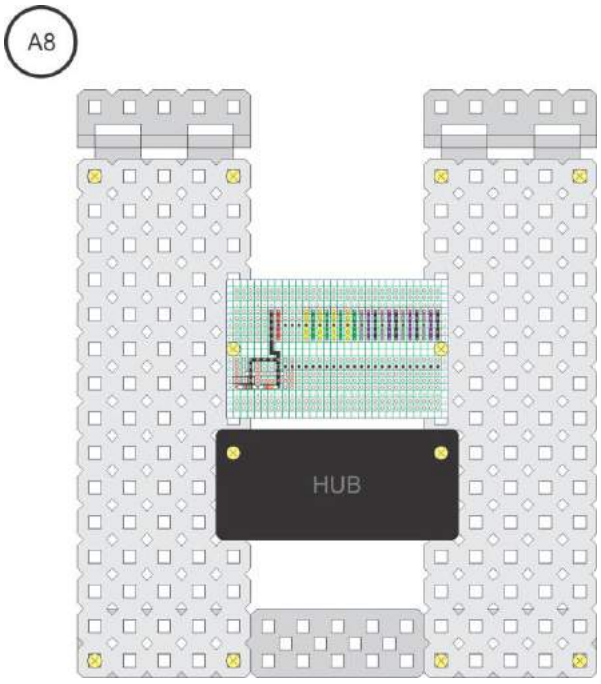


5.9. Monte a parte A8 como indicado.

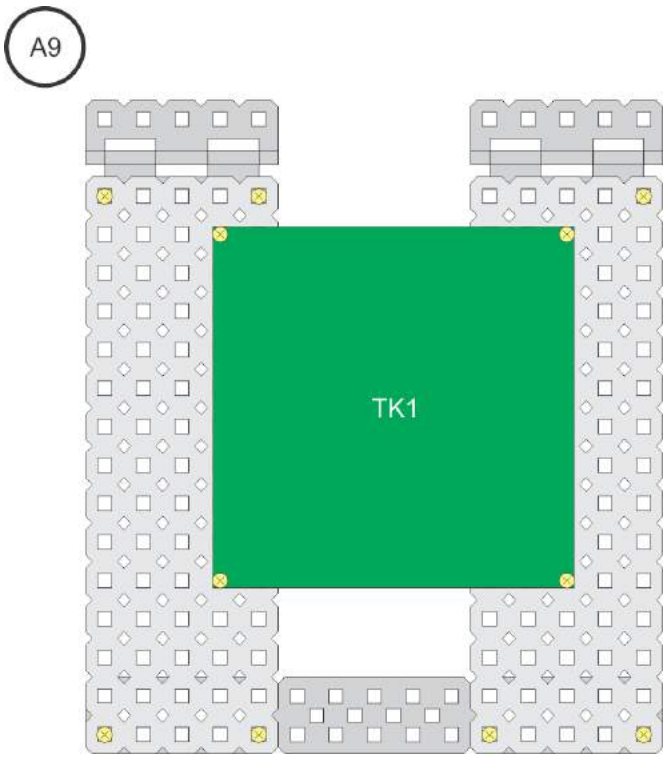
A) Parafuse a placa de controle no local indicado, sobre a parte A7. Utilize **dois parafusos (BC)** e **duas porcas (CB).** Use **dois espaçadores de plástico** ente a placa e a superfície metálica para isolamento.

B) Parafuse **um Hub (DB)** como indicado, abrindo sua caixa e parafusando por dento de sua base de plástico. Utilize **dois parafusos (BA)** e **duas porcas (CB)**.

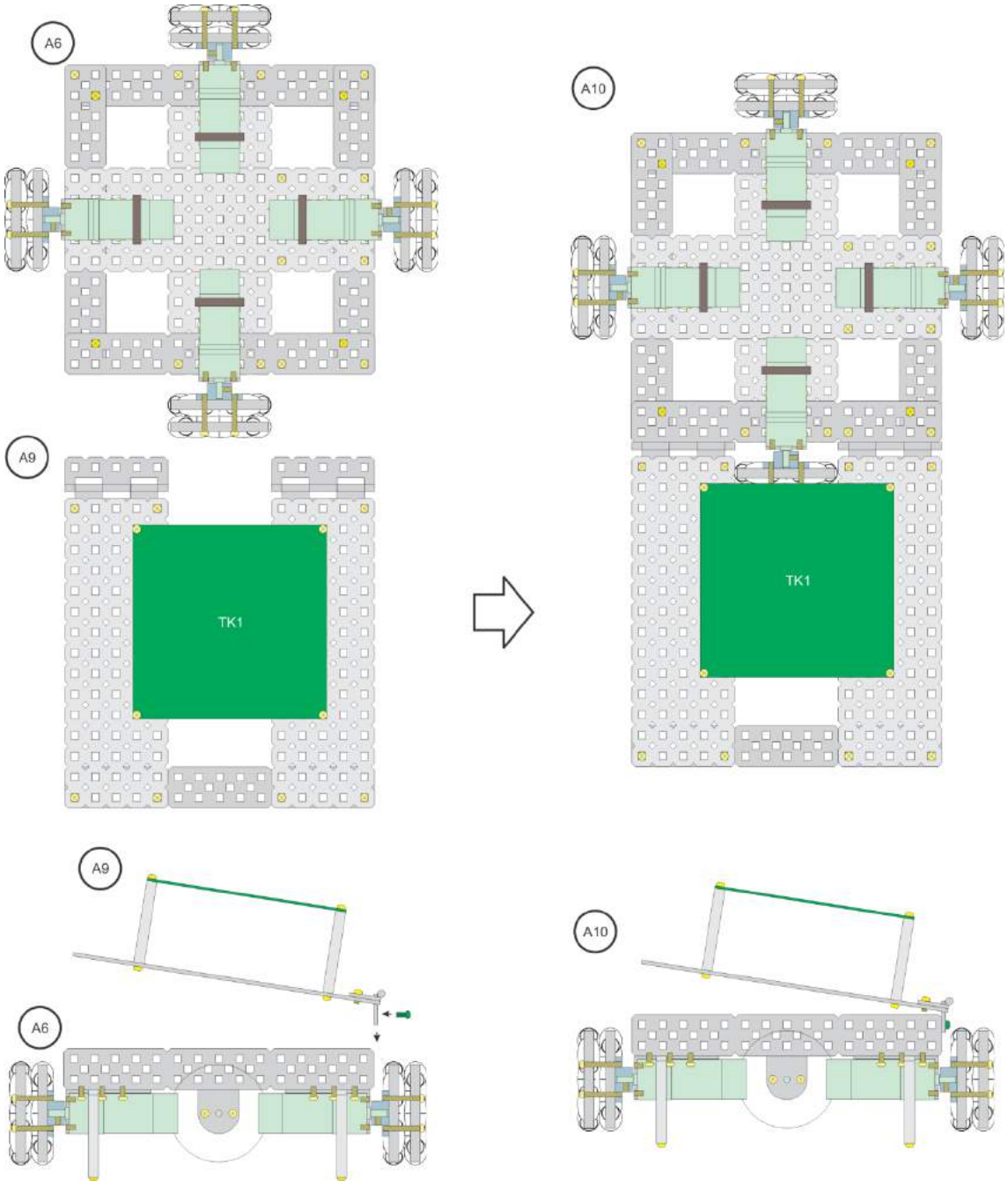
C) Insira **uma Cmod S6 (EL)** no socket da placa de controle. Cuidado no manuseio deste dispositivo é muito importante. O dispositivo é sensível a descargas eletrostáticas e muitos componentes não podem ser tocados.



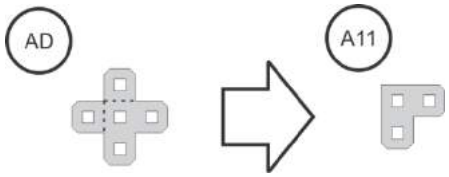
5.10. Monte a parte A9 como indicado. Parafuse a placa **TK1 (ED)** utilizando **oito parafusos (BA)** e **quatro standoffs (AG)**, como ilustrado. Plugue o Hub à TK1.



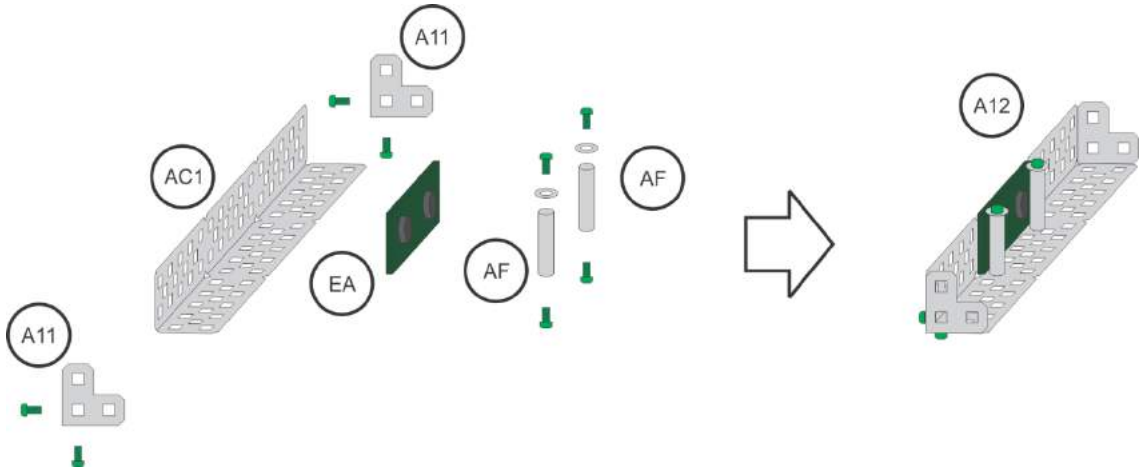
5.11. Monte a parte A10 como indicado. Una e parafuse as partes A6 e A9, de forma a fechar a caixa, obtendo a parte A10. Utilize **quatro parafusos (BA)** e **quatro porcas (CB)**. Parafuse nas extremidades das dobradiças.



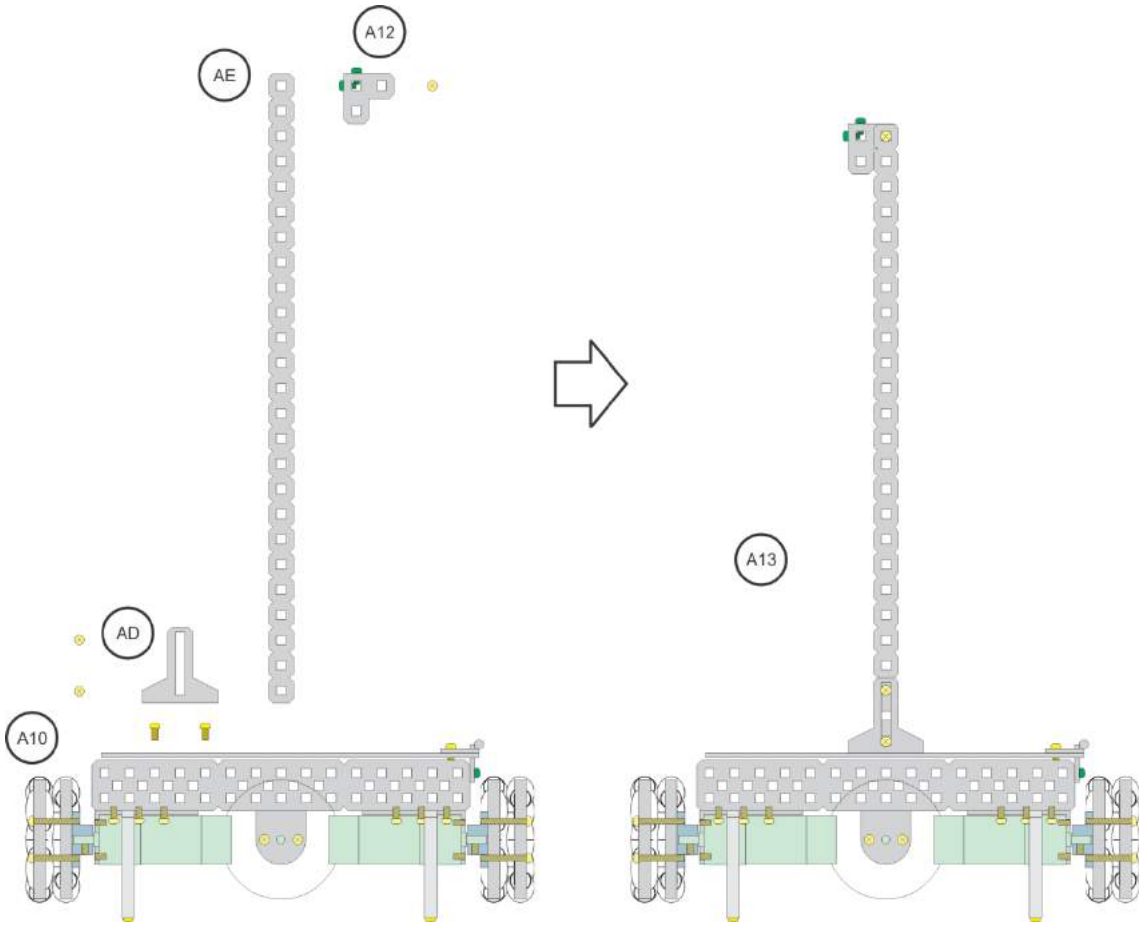
5.12. Obtenha duas partes A11, como indicado, dobrando **duas cruzetas (AD)**.



5.13. Monte a parte A12 como indicado. Parafuse uma peça AC1, duas peças A11 e **dois standoffs (AF)**, fixando **uma câmera (EA),** como mostrado.



5.14. Monte a parte A13 como indicado. Parafuse **duas barras (AE)** e **dois ângulos (AD)** unindo as peças A6 e A7. Utilize **14 parafusos (BA)** e **14 porcas (CB)**.



5.15. Conexões finais.

A) Faça o cabeamento seguindo o padrão de cores e a numeração dos motores.

B) Conecte **um regulador (EC)** à placa de potência nos pinos de 8.4 V. Sua saída será conectada à TK1, portanto o plug de uma das pontas do cabo de saída deve ser do tipo Barrel.

C) Conecte a câmera ao hub utilizando um cabo USB curto.

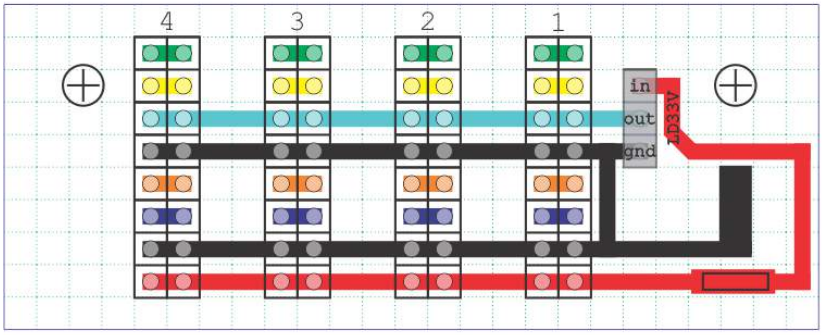
D) Conecte um **adaptador FTDI Serial-USB (EI)** aos pinos RX/TX da placa de controle.

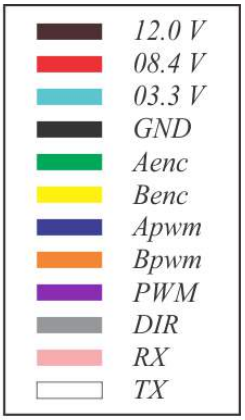
E) Conecte o FTDI ao Hub utilizando outro cabo USB curto.

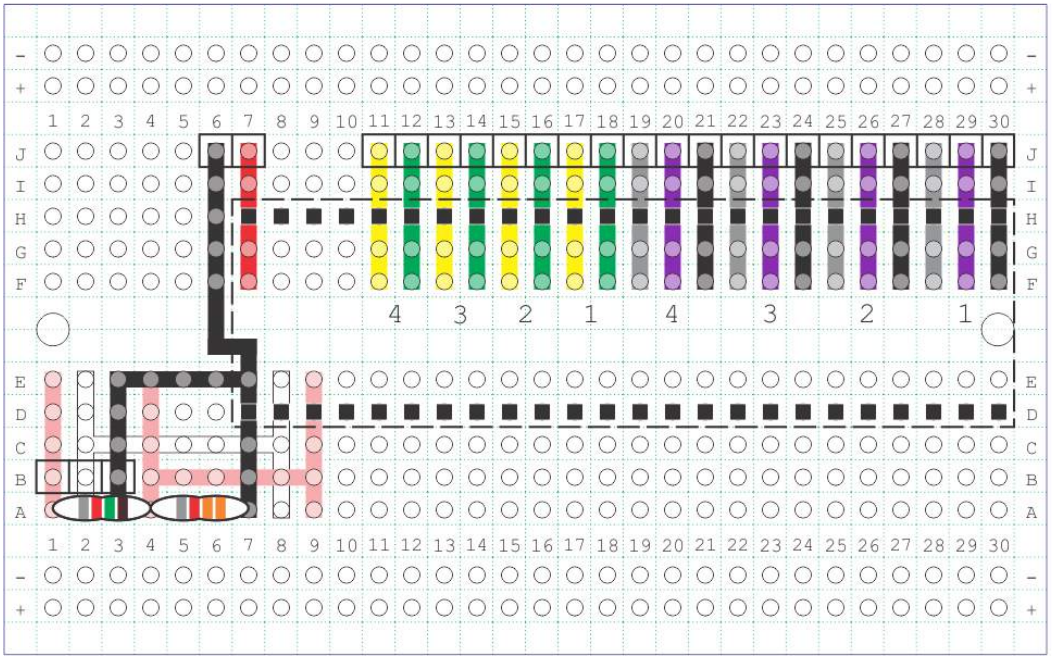
F) Plugue **um adaptador WiFi (EF)** à TK1 e parafuse.

G) Plugue **duas antenas (EG)** ao adaptador WiFi e passe-as pela haste da câmera.

H) Insira **um dummy HDMI (EM)** no plug HDMI da TK1 para emular um monitor de vídeo.







6. Configuração da TK1: Instalando Ubuntuc compatível com a DUO3D na TK1.
   6.1. Baixe os drivers da TK1:
   http://developer.download.nvidia.com/embedded/L4T/r21_Release_v3.0/Tegra124_Linux_R21.3.0_armhf.tbz2
   6.2. Baixe o sistema de arquivos da TK1:
   http://developer.download.nvidia.com/embedded/L4T/r21_Release_v3.0/Tegra_Linux_Sample-Root-Filesystem_R21.3.0_armhf.tbz2
   Nota: Caso qualquer um desses dois links estejam quebrados, eles podem ser baixados em **https://developer.nvidia.com/linux-tegra-r213**
   6.3. Descompacte os arquivos e monte o rootfs:
   sudo tar xpf Tegra124_Linux_R21.3.0_armhf.tbz2
   cd Linux_for_Tegra/rootfs/
   sudo tar xpf ../../Tegra_Linux_Sample-Root-Filesystem_R21.3.0_armhf.tbz2
   cd ../
   sudo ./apply_binaries.sh
   6.4. Coloque o rootfs dentro da eMMC interna do sistema:
   A) Coloque o sistema em **"reset recovery mode"** segurando **botão de RECOVERY** e apertando o **botão de RESET** uma vez na placa.
   B) Veja se o Linux está conectado com o dispositivo através do cabo USB usando **lsusb**. A placa deve estar listada entre um dos dispositivos USB conectados.
   C) Inicie o processo, que deve levar aproximadamente 10 minutos, com
   sudo ./flash.sh jetson-tk1 mmcblk0p1
   Após completar o processo, o dispositivo deve reiniciar automaticamente e pedir usuário e senha:
   usuário: ubuntu
   senha: ubuntu
   6.5. Instale os seguintes pre-requisitos:
   sudo apt-add-repository universe
   sudo apt-get update
   sudo apt-get install build-essential gdb libncurses5-dev
   sudo apt-get install cmake qt5-default qtcreator libopencv-dev
   6.6. Antes de utilizar a câmera DUO, se deve construir e carregar o kernel apropriado:
   cd ~/Documents
   mkdir Kernel
   cd Kernel
   Baixe do código-fonte do kernel (Versão R21.3):
   wget
   http://developer.download.nvidia.com/embedded/L4T/r21_Release_v3.0/sources/kernel_src.tbz2
   Extraia o código-fonte do kernel:
   tar -xvf kernel_src.tbz2
   cd kernel
   Copie a configuração atual:
   zcat /proc/config.gz > .config
   Certifique-se de que o arquivo config está ok (não deve retornar erros se as versões

dos kernels combinarem)

make menuconfig

Va para: **"General setup -> Local version"** e coloque o nome da versão : **-gc017b03**

Para habilitar adaptadores wireless, va para**: "Device Drivers -> Network device support -> Wireless LAN -> Realtek 8187 and 8187B USB support"** e habilite os drivers para o dispositivo.

sudo apt-get install linux-firmware

Salve e saia.

6.7. Edite o arquivo **xhci-mem.c**

gedit drivers/usb/host/xhci-mem.c

Comente as seguintes linhas do código (linhas 1439 e 1440):

1439: if (usb_endpoint_xfer_bulk(&ep->desc))

1440: max_packet = 512;

Salve o arquivo.

6.8. Compile o kernel (leva em torno de 5 minutos):

make -j4

make modules

Instalar os módulos

sudo make modules_install

Copie a nova zImage do kernel para o diretório /boot

sudo cp arch/arm/boot/zImage /boot/zImage

6.9. Altere o config file para habilitar dispositivos FTDI. Copie o script **CompileFTDI** para a pasta atual - **~/Documents/Kernel/kernel**:

sudo chmod +x CompileFTDI

sudo ./CompileFTDI

6.10. Reinicie e verifique a versão do kernel:

uname –r

6.11. Baixe o driver da DUO3D para ARM:

https://duo3d.com/download_file/DUO3D-ARM-ALL-v1.0.35.226

6.12. Construa o módulo kernel linux da DUO:

cd ~/Downloads/DUO3D-ARM-ALL-v1.0.35.226/DUODriver

O comando **make** irá utilizar os novos headers kernel que foram construídos (~/Documents/Kernel/kernel):

make

6.13. Habilite permissões para os scripts de driver

sudo chmod +x InstallDriver LoadDriver UnloadDriver

6.14. Instale o driver:

sudo ./InstallDriver

6.15. Carregue o driver DUO Kernel Module:

sudo ./LoadDriver

6.16. Para descarregar o driver:

sudo ./UnloadDriver

6.17. Configuração de software

Após finalizados os processos de configuração de kernel instalação de drivers, deve-se obter o software de visão computacional / tomada de decisão, que roda na TK1.

Para tal, instale e o software git e clone o diretório do projeto na TK1:

sudo apt-get install git

git clone https://leialabufg@bitbucket.org/leialabufg/decision-layer.git

Atenção: além dos comandos acima, é necessário também usar os comandos de configuração de usuário no git, após sua instalação.

Após clonar o projeto, copile usando **make** e execute o binário **main**.

Este programa envia imagens da câmera e de seus processos através de pacotes UDP via WiFi. Para receber e visualizar estas imagens, instale em um PC regular o seguinte projeto:

git clone https://leialabufg@bitbucket.org/leialabufg/robotmonitor.git

Para treinar a MLP do sistema de visão, capture nuvens de ponto usando o software **"DuoDashboard"**, que acompanha o driver da câmera, e processe-la usando o marcador de obstáculos:

git clone https://leialabufg@bitbucket.org/leialabufg/plytagger.git

Após marcar os obstáculos nas nuvens de treinamento, realize o treinamento usando o projeto **decision-layer.** O processo de treinamento pode ser realizado no próprio robô um em um PC e tem como resultado o arquivo MLP_WEIGHTS.

7. Procedimentos básicos de teste e detecção de erros.
   7.1. Teste e detecção de erros pode exigir a remoção de componentes para testes individuais. Entretanto, há três checkpoints de teste na arquitetura do robô que agilizam a detecção e isolamento de erros ou dispositivos faltosos. Estes pontos de checagem se encontram em camadas crescentes da hierarquia de comando. São eles:
   A) Teste de comando sobre as placas controladoras de corrente.
   B) Teste de comando sobre a placa de controle de velocidade (Cmod).
   C) Software de teste de comando para a TK1.
   Os três pontos de checagem permitem enviar comandos diretamente sobre os elementos abaixo da hierarquia, eliminando a possibilidade de erros acima da hierarquia. Os tópicos abaixo detalham cada ponto de checagem.
   7.2. **Teste de comando sobre as placas controladoras de corrente**. As placas Cytron possuem dois botões de comando "A" e "B" para comandar envio de corrente diretamente sobre o hardware da placa. O uso destes botões permite detectar placas defeituosas, curto-circuito e outros problemas de conexão na placa de potência, além da detecção de defeitos nos motores e suas conexões.
   7.3. **Teste de comando sobre a placa de controle de velocidade (Cmod)**. A placa de FPGA Cmod S6 possui quatro LEDs e dois botões. O programa utilizado nela (**cmods6_motionlayer**) configura quatro controladores PID, quatro decodificadores de leitura de velocidade e quatro moduladores PWM para controle individual, paralelo e simultâneo dos quatro motores a uma taxa de amostragem de 200 Hz. Este programa configura a FPGA para receber as quatro velocidades de referência através de uma conexão serial UART da camada superior (TK1). Adicionalmente, o programa configura os botões para teste e verificação de erros. Um dos botões aciona um LED, enquanto o outro sobrepõe a velocidade de referência por um valor fixo. Isso permite enviar um comando que aciona todos os quatro motores com velocidade igual e constante enquanto pressionado. Assim, é possível detectar falhas na configuração da FPGA ou na conexão dos motores, sensores e no fornecimento de energia. Possibilita

também isolar erros da camada superior, que normalmente envia as velocidades de referência via serial.

7.4. O **software de teste de comando para a TK1** executa em seu ambiente Linux com linha de comando. Seu código é extremamente simples para eliminar dúvidas no isolamento de erros. Permite enviar velocidades arbitrárias nas direções X e Y e na rotação em torno de Z. Seu repositório git é obtido por:
git clone https://leialabufg@bitbucket.org/leialabufg/motiontester.git
Após compilar, execute o binário **./main**. Se não for passado nenhum argumento, o mesmo envia um comando padrão de velocidade para as quatro rodas. As opções de passagem de argumento são três:
A) **./main <velocidade em X>**
B) **./main <velocidade em X> <velocidade em Y>**
C) **./main <velocidade em X> <velocidade em Y> <velocidade R>**
Este software permite determinar se a comunicação serial e todos os elementos abaixo na hierarquia estão funcionando.

7.5. Estes três procedimentos auxiliam na busca por erros de construção do robô. Erros comuns são:
A) Conexões incorretas dos jumpers, jumpers invertidos.
B) Inversão das fases dos codificadores de rotação dos motores.
C) Inversão dos fios PWM e DIR.
D) Falhas nas soldas das placas.
E) Curto-circuito entre canais.
Testes de continuidade com multímetro são indicados para detectar falhas, especialmente as duas últimas. Assegure-se também de que nenhum outro canal além do GND está curto-circuitado à estrutura metálica do robô.
Importante: Erros de conexão e de curto-circuito como estes geralmente danificam componentes como placas de controle, placas FPGA e adaptadores serial FTDI. Muitas vezes esse fato dificulta o isolamento do problema original. Após muitas tentativas frustradas em isolar o problema, remova e teste componentes individualmente, começando pelos dispositivos maiores, como as placas de controle e FPGA.

ANEXO B – Artigo: Algoritmo rápido para extração de piso em tempo real a partir de nuvens de ponto estéreo não-organizadas.

Pattern Recognition Letters
journal homepage: www.elsevier.com

Fast algorithm for real-time ground extraction from unorganized stereo point clouds

Gilberto Antonio Marcon dos Santos[a], Victor Terra Ferrão[a], Cassio Dener Noronha Vinhal[a], Gelson da Cruz Junior[a]

[a] *LEIA – Laboratory for Education and Innovation on Automation*
School of Electrical, Mechanical and Computer Engineering
Federal University of Goiás, Goiânia, Brazil

ABSTRACT

This paper presents a fast, robust algorithm for ground extraction from unstructured point clouds obtained from stereo reconstruction. Unlike most point cloud segmentation approaches, our algorithm does not rely on 2.5D range image structures nor on any sensor information. All processes involved consider geometry only and do not depend on any reflectivity or color information. We propose applying a top-down 4-ary segmentation followed by an MLP segment-wise classification. This adaptive approach allows accurate differentiation between ground and obstacles in noisy point clouds of cluttered scenes. Real-time performance is achieved on a low cost embedded platform.

© 2016 Elsevier Ltd. All rights reserved.

1. Introduction

Depth perception is a powerful sensing source for autonomous vehicles and robots. Light detection and ranging (Lidar) scanners provide very accurate time-of-flight surface data. 3D Lidar scanners have driven most recent autonomous vehicles projects Levinson et al. (2011). However, device size, power consumption, and cost are still prohibitive for medium and small scale robotics projects. Unlike Lidar devices, time-of-flight cameras have no moving parts and provide depth data at lower costs, but distortions, low resolution and issues relating to moving objects and concavities in the scene still limit their use Foix et al. (2011). Structured light sensors provide depth information at even lower costs, but their performance is severely degraded under strong ambient illumination Gupta et al. (2013).

In contrast to these techniques, stereo vision offers a passive approach to depth sensing. Using ambient light, it requires less power to operate and offers affordable high resolution depth information. Dense stereo reconstruction requires more processing power than other methods, but modern algorithms are real-time capable with current hardware Van Der Mark and Gavrila (2006). It estimates depth based on stereo disparity, i.e., differences between left and right images. Physical limitations on the process establish accuracy boundaries. For the pinhole camera pair model, the depth error is approximated by

$$| \delta Z | \approx \frac{Z^2 \cdot \mu}{b \cdot f} \quad (1)$$

where Z represents the distance from the camera, μ the disparity error, b the baseline distance between the cameras, and f the camera focal distance Chang and Chatterjee (1992). δZ increases quadratically with distance, quickly degrading accuracy and thus dramatically limiting range. This is the main disadvantage of stereo reconstruction. Other disadvantages include dependency on environmental factors such as surface smoothness, illumination, and the presence of repetitive textures and/or occlusions. However, stereo cameras, as compared to active scanners, are cheaper, lighter, have no moving parts, and consume much less power.

Depth sensing methods produce point clouds representing the surrounding surfaces. Interpreting and extracting information from point clouds is an established field. Most studies focus on processing point clouds from Lidar sensors for navigation, localization and mapping. Algorithms for segmenting and interpreting points clouds from time-of-flight and structured light cameras are also common.

Stereo point clouds, however, are subject to several issues: for instance, high noise and data sparsity. Thus, applying methods made for active sensor point clouds to stereo point clouds often becomes infeasible. Some stereo point cloud segmentation algorithms are also subject to additional restrictions, such as operating on unorganized point clouds or on point clouds generated by real-time methods. This paper presents a robust method for segmenting unorganized stereo point clouds generated by real-time reconstruction. Figure 1 presents an example stereo point cloud from our test set.

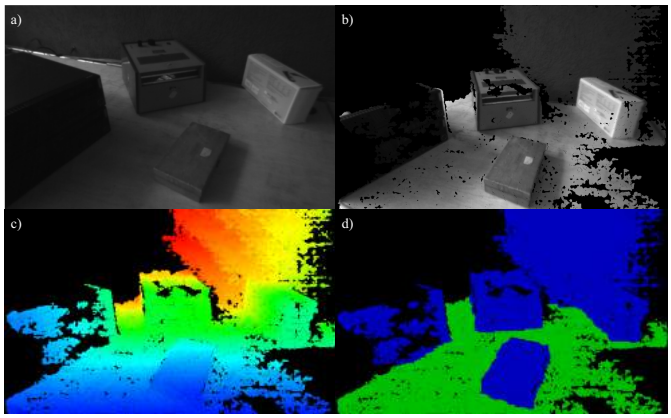


Fig. 1. Point cloud obtained with real-time stereo reconstruction. It portrays scattered objects on the floor and a wall on the background. (a) Original scene. (b) Luminous intensity point cloud. (c) Depth information, red is farther. (d) Hand labeled floor (green) for MLP training and accuracy testing.

2. Related work

Organized point clouds, also known as 2.5D range images, are depth-representing images where each pixel intensity indicates distance from the camera. Organized point clouds allow several geometric assumptions which facilitate processing and segmentation. Pixel adjacency often implies 3D point adjacency, making neighboring search much more efficient, for example. Recent methods, such as proposed those by Holz et al. (2011) and Zhang et al. (2010), have achieved high accuracy and low processing times, but can only operate on organized point clouds.

Unorganized point clouds, on the other hand, are unordered lists of points in *xyz* space and no neighboring information is implied. No geometric information can be inferred from the data structure and samples cannot be assumed to be orderly spaced. Algorithms for unorganized point clouds thus operate under much more challenging conditions.

A clear distinction also must be made between stereo point clouds obtained from off-line and from real-time methods. The former are significantly denser and more accurate than the later. Real-time performance point cloud reconstruction algorithms provide clouds that are typically sparser (fig. 1), thus imposing many more restrictions on the segmenting algorithms.

Many well established point cloud segmentation approaches are accurate and robust, but computationally expensive and slow. Model-based approaches, such as Schnabel et al. (2007), apply Monte Carlo processes in order to attempt to match predefined shapes; these require too much processing time to run in real-time. Top-down approaches usually rely on octrees, operating with these is costly Behley et al. (2015).

Fast state-of-the-art algorithms benefit from application-specific assumptions or data fusion. Others require precise active sensor point-clouds or specific data ordering. Bottom-up approaches are often graph-based: points generate graph nodes and connections between points generate graph edges. Similarities between node features are represented by edge weights and weight-based region growing algorithms are used. Fast bottom-up graph-based methods that rely only on geometric information include Klasing et al. (2008) and Moosmann et al. (2009), while Strom et al. (2010) and Schoenberg et al. (2010) use both geometric and color data. For building these graphs in real-time, implementations use sensor-specific data ordering and RGB camera data fusion. For general unorganized point clouds, slow neighboring search would be necessary, making these approaches too slow to operate in real-time.

Classic image processing algorithms are mature and time-efficient, but require dimensionality reduction and discretization to be applied to point clouds. A common strategy is to perform orthogonal projection onto the horizontal xy plane and to discretize Lidar cloud data into occupancy grids as in Hernandez and Marcotegui (2009) and Himmelsbach et al. (2009). Oniga and Nedevschi (2010) proposed a method for detecting obstacles on paved roads from stereo point clouds with real-time performance. This method consisted of discretizing the horizontal plane into a grid and then judging nodes based on point density and height. However, the method depended on many assumptions about vehicle pose, camera position, data density, and road shape; thus, it is not suitable for robot navigation in heterogeneous environments.

Other approaches perform perspective/cylindrical projection to achieve image-like uniform point density over the projection canvas Bewley and Upcroft (2013) then apply top-down or bottom-up image segmentation techniques. The use of bottom-up segmentation methods on these canvas, such as region growing, is still restricted to off-line stereo clouds due to numerous discontinuities found in real-time stereo clouds. Top-down segmentation approaches, such as those using quadtrees Hampp and Bormann (2013); Park et al. (2014); Marcon et al. (2016), have yielded successful results. 4-ary trees are simpler than octrees, performing faster and yet allowing top-down segmentation.

We propose, here, a fast ground extraction algorithm, that is capable of operating with low cost stereo cameras for robotic navigation and mapping. As an improvement on the method proposed by Marcon et al. (2016), the process also benefits from a top-down 4-ary tree segmentation. However, a multilayer perceptron (MLP) classification is used instead of a hard threshold to determine whether a tree node belongs to ground or obstacle. This makes the process more robust and also more applicable to noisy stereo point clouds.

The top-down 4-ary tree segmentation is in the spirit of Hampp and Bormann (2013); Park et al. (2014), but operates on an orthogonal xy projection instead of at perspective/cylindrical projection. A plane estimation process has been adopted in order to determine a good orthogonal projection direction. In contrast to Schnabel et al. (2007), this step is a simplified Monte Carlo search. Because it is just a preemptive step, the number of candidates can be dramatically reduced, and thus it does not affect time performance. Its top-down nature, associated with a MLP classification, makes it a robust method to missing data and high noise. It also does not depend on color information, data fusion, data structure ordering, or any assumed cloud organization.

The contribution of this paper is a robust, fast algorithm for ground extraction from unorganized stereo point clouds ob-

tained from real-time reconstruction methods. Our method also does not rely on any external sensor or pose assumption. It consists of a top-down adaptive segmentation over the orthogonal xy projection, followed by a MLP classification. Assuming that ground is a single non-self-overlapping surface, this dimensionality reduction benefits time performance without accuracy impairment. The area subdivision self-adapts to the cloud, focusing processing effort in detailed areas and enhancing separability of the MLP feature space.

3. Proposed algorithm

The work of Douillard et al. (2011) has experimentally proven that performing ground extraction prior to object segmentation dramatically reduces processing time without affecting overall segmentation accuracy. It facilitates object segmentation by enhancing separability. Based on this, our segmentation algorithm goal is to separate ground from obstacles.

Figure 2 presents the sequence of steps for our algorithm. The estimated error standard deviation for sensor noise, σ_e, is used as a threshold in many of these steps. For stereo point clouds, σ_e can be approximated by δZ as in Equation 1.

Our algorithm consists of four steps: (1) estimating the ground plane; (2) rotating the point cloud C to align the ground plane with the coordinate system xy plane; (3) adaptively subdividing the cloud using a 4-ary tree; and (4) running a MLP classification, for each tree node, into ground or not ground. The first step alone yields a rough ground extraction, as proposed by Konolige et al. (2009). However, floor unevenness and high noise conditions require further refinements in order to yield an accurate segmentation. Each step is described in detail in the following subsections.

3.1. Ground plane approximation and cloud rotation

A common approach for plane recognition in point clouds is to use the RANdom SAmple Consensus (RANSAC) algorithm, introduced by Fischler and Bolles (1981) as a robust regression method. RANSAC and other SAC variants have been successfully applied to point cloud plane extraction Schnabel et al. (2007); Konolige et al. (2009); Yang and Förstner (2010). Utilized specifically for ground plane segmentation from stereo clouds by Konolige et al. (2009), RANSAC finds the ground plane by considering non-ground points as outliers in the data set. This comes from assuming that the ground points will roughly constitute the largest plane in the scene, which is a fair assumption for mobile robotics.

Classic RANSAC has a specific stop condition that assumes the outlier ratio is known. However, when segmenting point clouds, every point that does not fit the plane model is considered an outlier. This includes every other object in the scene whose size and quantity is unknown before segmentation. Thus, the classic outlier ratio test becomes hardly valid and often leads to an unnecessary number of iterations. Our implementation follows the classic RANSAC work-flow, but instead of using the classic stop condition, we generate and evaluate a fixed number, c, of candidates. In addition, candidates are evaluated using the MSAC robust estimator from Torr and Zisserman (2000) because it provides good cost-accuracy trade-off:

$$\rho_{MSAC}(e_i) = \begin{cases} e_i^2 & \text{if } |e_i| < \sigma_e \\ \sigma_e^2 & \text{if } |e_i| \geq \sigma_e. \end{cases} \quad (2)$$

Clouds are preemptively decimated for improved execution time. The decimation ratio, d, and the number of candidate planes, c, are empirically defined parameters. Figure 3 presents a sensitivity analysis for these parameters. Note that d has little effect on the search results, but it has a major impact on execution time. Greater c values generate better search results up to $c = 1024$ but saturate past this range. Execution time increases exponentially for larger c values. Parameters $d = 100$ and $c = 1024$ were chosen as values providing a good accuracy-performance balance.

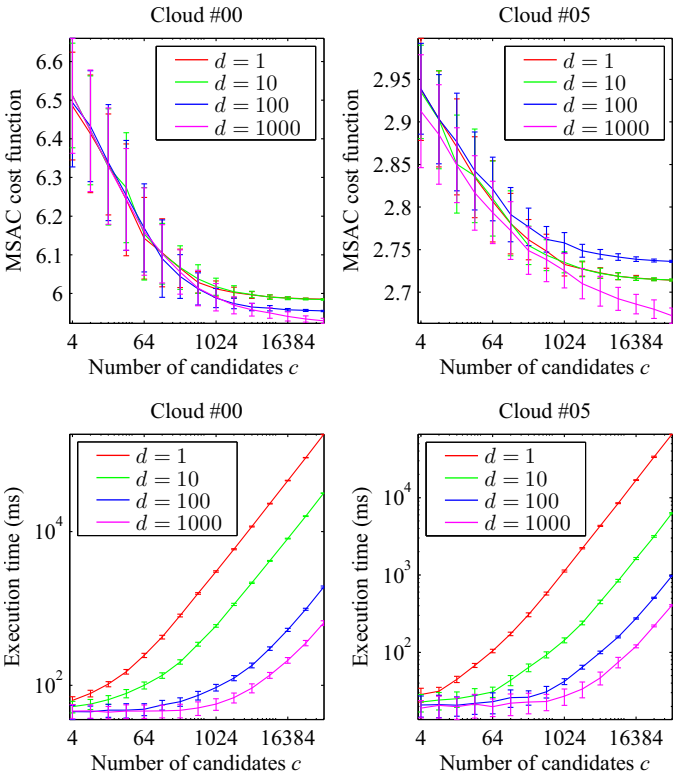


Fig. 3. Ground plane estimation performance for selected clouds. Each parameter set was run 100 times for statistical significance. Note that different clouds will present different cost ranges.

Assuming that no external sensor will provide orientation in respect to ground, we estimate the ground plane to rotate the cloud and make the floor parallel to the horizontal xy plane. This step is necessary for z values to represent height, which is utilized in the classification steps. Figure 4 presents a point cloud before and after rotation. Notice the camera at the origin of the coordinate system. Before rotation it points to the y axis.

3.2. 4-ary tree segmentation

After rotation, z values represent *height*. Subtracting ground plane height gives the approximate distance from the floor, d_f. Each tree node contains a point set L from C and a circumscribing rectangle, as in Figure 5; the root node initially lists all elements of C. The *height* standard deviation, σ_z, indicates

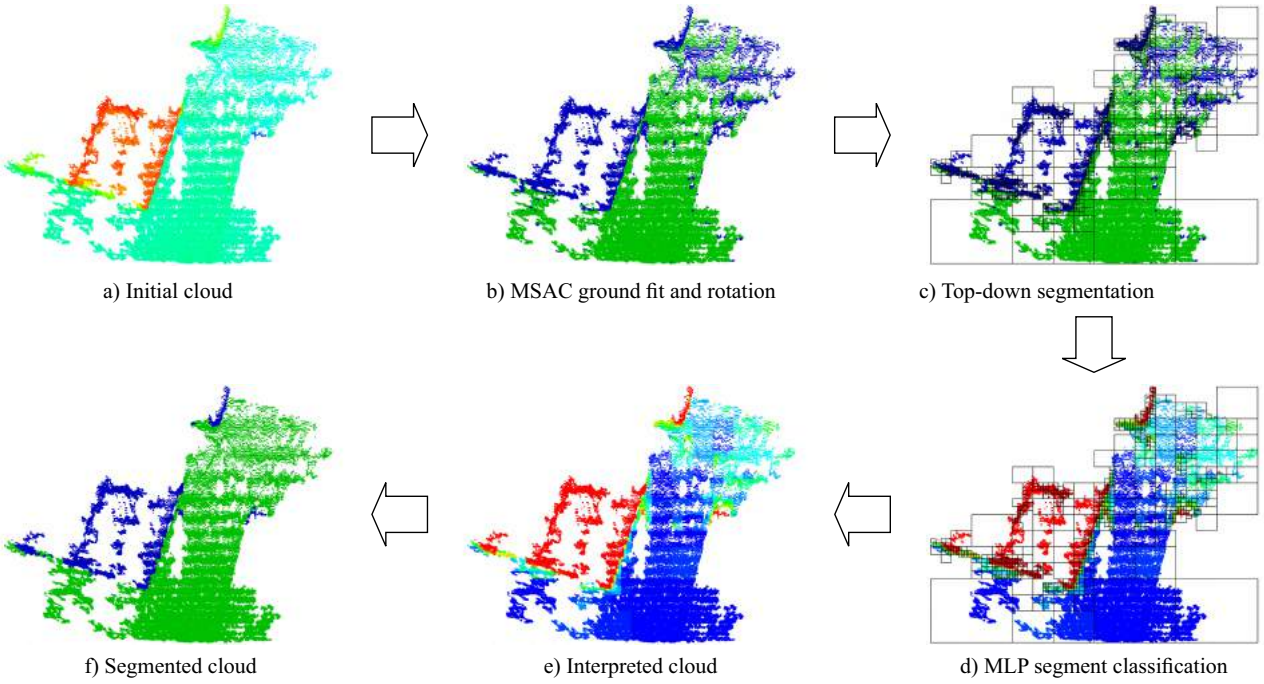


Fig. 2. Overview of the proposed algorithm. The clouds represent boxes scattered on the ground, viewed from top-down orthogonal projection. (a) Original cloud, color representing height. (b) Result of MSAC for approximating the ground plane followed by rotating the cloud to make the floor horizontal – green represents ground points. (c)After 4-ary tree adaptive segmentation. (d) MLP tree node classification. (e) Classification results, color represents certainty: red 100% obstacle and blue 100% ground. (f) Ground segmentation using thresholding over the MLP certainty outputs. Green represents ground, blue represents obstacles.

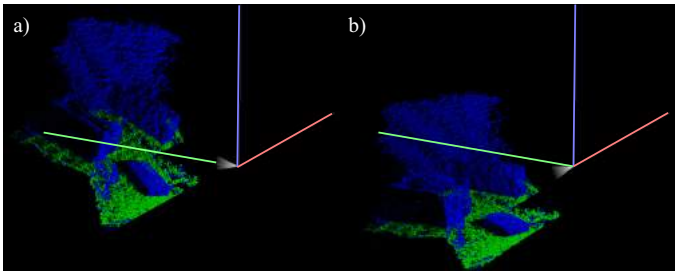


Fig. 4. Cloud rotation after ground plane estimation. (a) Before and (b) after rotation. The purpose of the rotation is to make the ground plane parallel to the coordinate system xy plane. Axis x, y and z are represented in red, green and blue, respectively. The gray cone represents the camera.

how flat (vertically homogeneous) a point set is:

$$\sigma_z = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(L_i^z - \overline{L^z})^2} \tag{3}$$

where L^z indicates the z values of the points in L.

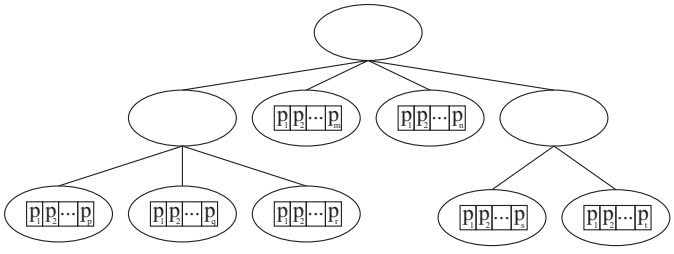


Fig. 5. Proposed 4-ary tree.

By performing a top-down 4-ary tree segmentation we isolate areas containing wide flat regions from areas that have vertical structures. Starting with a single rectangle that circumscribes all points, we recursively subdivide it according to σ_z. Rectangles with $\sigma_z > \sigma_e$ will be subdivided into four children, each covering a quarter of its area. Low σ_z nodes will not be subdivided since they are likely to constitute ground. High σ_z value nodes will continue to be subdivided in order to separate ground and non-ground nodes. Nodes with less than two points or a perimeter smaller than $4\sigma_e$ will halt subdivision; this is to avoid excessive segmentation. The algorithm will focus on complex regions, such as those containing obstacles, by finely subdividing them. Figure 6 presents an example 4-ary tree subdivision. After children are instantiated, L is cleared to avoid redundancy; thus, only leaf nodes will hold point lists L. Algorithm 1 shows the tree descent recursive call.

Data: Points list L and rectangle R.
Result: Adaptive point cloud subdivision.
if $\sigma_z > \sigma_e$ *and* $p > 4\sigma_e$ **then**
  Divide R into R_1,R_2,R_3, and R_4;
  Transfer points from L to L_1,L_2,L_3, and L_4;
  if $size(L_1) > 1$ **then**
    instantiate and descend $(L1, R1)$;
  if $size(L_2) > 1$ **then**
    instantiate and descend $(L2, R2)$;
  if $size(L_3) > 1$ **then**
    instantiate and descend $(L3, R3)$;
  if $size(L_4) > 1$ **then**
    instantiate and descend $(L4, R4)$;

Algorithm 1: Recursive 4-ary tree segmentation.

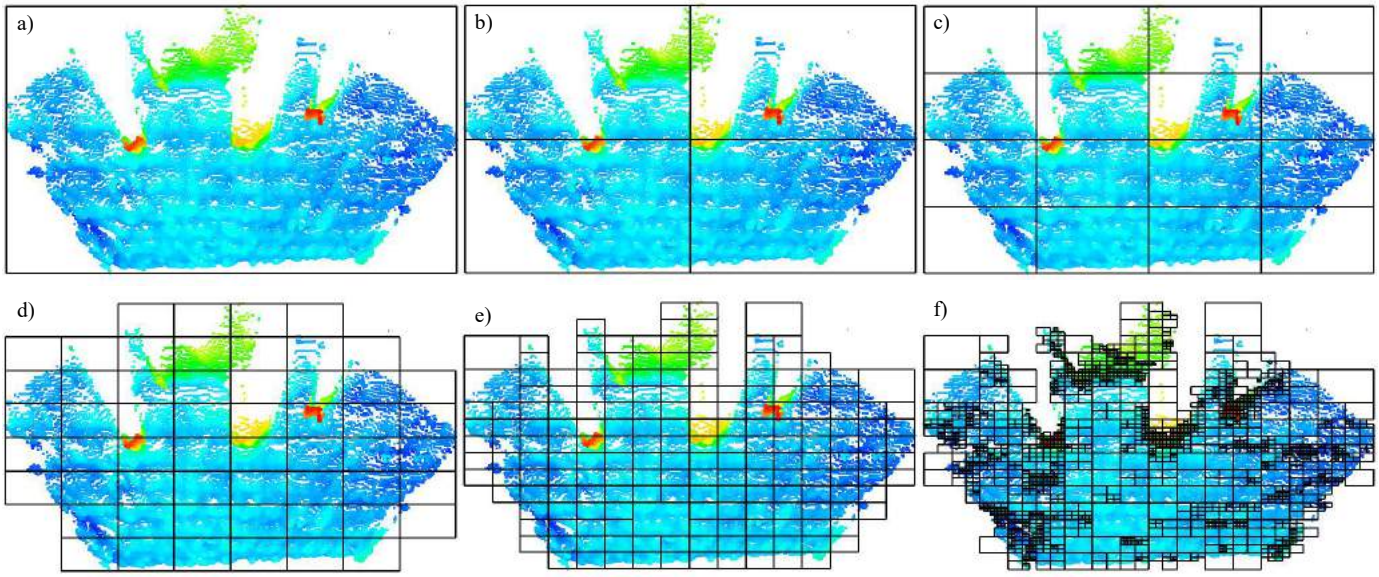


Fig. 6. Step-by-step (a-f) 4-ary tree subdivision. Color represents height: blue is lower, red is higher.

Sixteen cluttered scenes of objects scattered on the floor were used to provide typical cases of indoor obstacle negotiation. Figure 7 (b) presents the average number of tree nodes obtained from segmentation, from executing a thousand times. Note the high stability and repeatability of the subdivision process, demonstrated by the low deviation in number of nodes.

3.3. Segment classification

After the adaptive area subdivision, each tree node contains point sets which share similar features. A multilayer perceptron artificial neural network (MLP ANN) is used for classifying these nodes into ground or obstacle. Three features are fed to the MLP for this purpose: height standard deviation, σ_z; rectangle perimeter, p; and approximate distance from the floor, d_f.

The feature space and the training samples are represented in Figure 8. This training set was obtained from clouds #0 to #7 by manually labeling each point into ground and not ground as in Figure 1 (d). Note that the subdivision process that precedes the MLP classification eliminates nodes that are both wide and vertically heterogeneous. Therefore, the training samples are restricted to the areas where either p or σ_z are close to zero.

After the 4-ary subdivision, each cloud provides hundreds of training samples. The ground/non-ground ratio of each node is registered as the obstacle certainty for the training set; this is represented in Figure 9 (a). The classic error back-propagation algorithm is used for MLP training. After training, the MLP will generate results as represented in Figure 9 (b). These certainty outputs can be either thresholded or used in mapping and uncertainty reduction processes. By establishing a threshold one directly obtains a crisp segmentation as in Figure 2 (f).

4. Platform for testing

Code Laboratories DUO M is a low cost ultra-compact, configurable, high speed, wide field of view, integrated stereo camera. It was used for gathering all the data used in this study. The integrated firmware and driver API provided an encapsulated interface for image capture, stereo reconstruction and point cloud generation. It incorporates some high performance depth reconstruction algorithm implementations, including stereo block matching (BM) and semi-global stereo matching (SGM) Hirschmüller (2005). SGM is slower but more robust and denser than BM. For a better performance-quality trade-off we adopted SGM at 320x240 resolution. Figure 1 shows an example cloud obtained using this configuration.

For this camera pair, $b = 30mm$ and $f = 3mm$. At 320x240, the expected disparity error is $\mu = 12um$. By limiting depth to one meter – a reasonable range for small indoor robots, the expected depth error is at most 13 cm, calculated from eq. (1).

For real-time performance evaluation, a low cost embedded processing platform was adopted. The Nvidia Jetson TK1 is an entry level CPU-GPU embedded computer based on the ARM architecture. Capable of running a full-scale OS and yet consuming low power, it can be easily integrated to small mobile robots.

5. Validation and testing

Clouds were manually labeled to provide training samples and accuracy assessment ground truth (Figure 7). Each cloud provides around two thousand nodes; thus the training set consists of almost sixteen thousand samples (Figure 8).

The validation set comes from clouds #8 to #11. It was used for sensitivity analysis relating to the number of hidden layer (HL) neurons in the 1-15 range. Figure 10 presents the validation results. Increasing the number of HL neurons impacts classification time but does not benefits accuracy. This is due to the separability and low dimensionality of the feature space (Figure 8) – a product of the preceding area subdivision process. Having many HL neurons would not appreciably improve

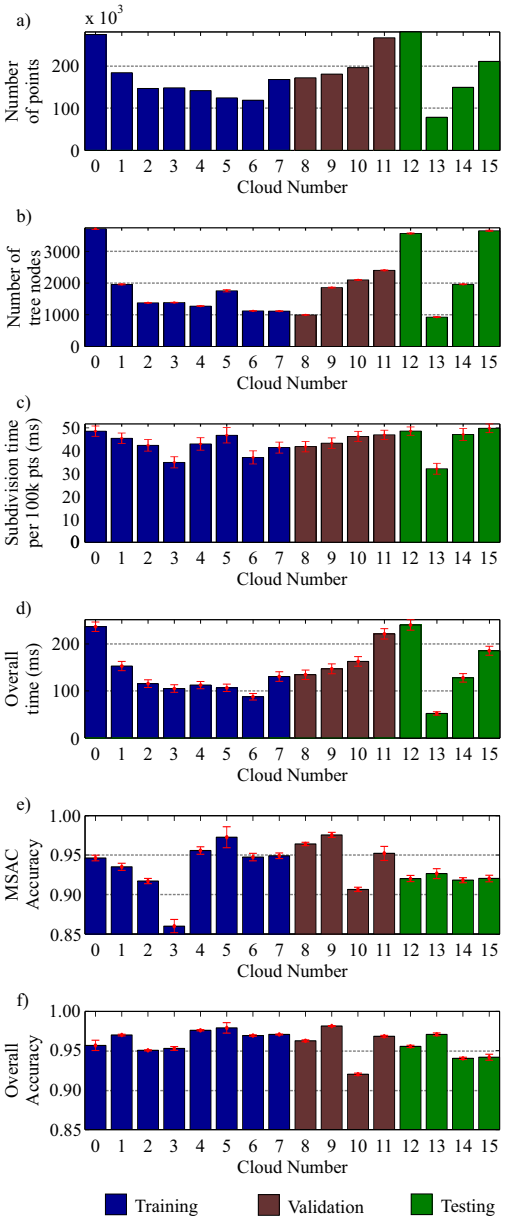


Fig. 7. Training, validating and testing sets with time and accuracy results.

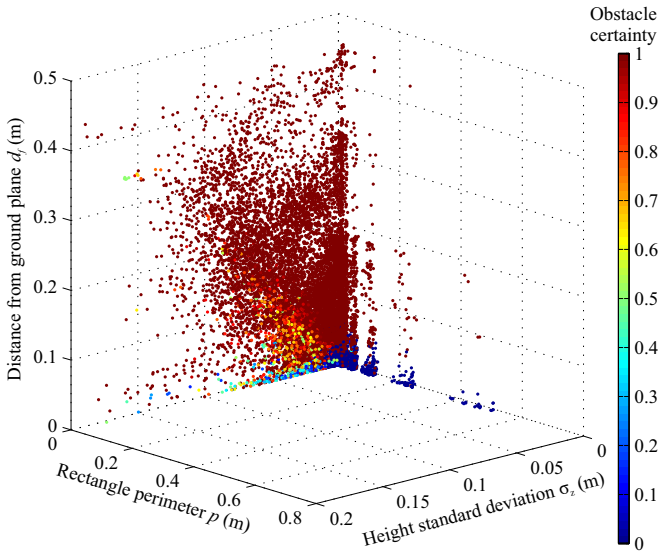


Fig. 8. Feature space and training samples. Red represents max obstacle certainty and blue represents max ground certainty. These features allow accurate ground-obstacle differentiation.

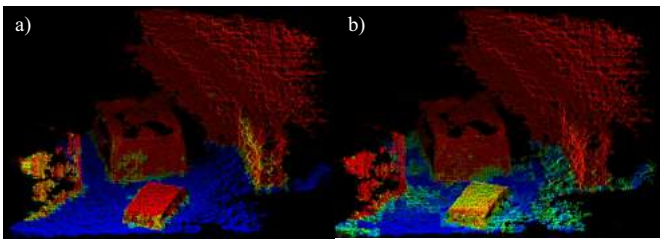


Fig. 9. MLP training set (a) and MLP output (b). Red represents max obstacle certainty and blue represents max ground certainty.

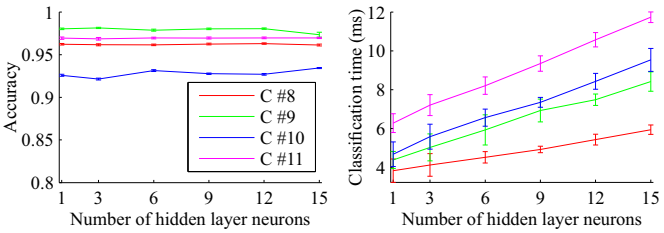


Fig. 10. Accuracy and classification time for varying number of HL neurons.

classification accuracy. Therefore, one HL neuron was adopted for the testing phase because it provides low computation time and appreciably accurate results.

Because no other algorithm in the literature operates under the same restrictions, it was not possible to provide a fair one-to-one performance comparison, except for the SAC approach proposed by Konolige et al. (2009). Figure 7 presents area subdivision time (c), overall processing time (d) and final accuracy (f) when running on the TK1 embedded board. Accuracy was calculated by thresholding MLP outputs and comparing these to the hand labeled ground truth. The threshold was set to 0.5 – outputs below 0.5 are considered ground. All tests scored above 90% accuracy, a good result for the noisy clouds used. The short execution times allow for real-time applications. Also the GPU cores available on this board were not used in our implementation. Both the Monte-Carlo search and the tree segmentation are suitable for parallelization; therefore, performance could be further improved by using these cores. Execution time is directly related to the number of cloud points, but accuracy depends on factors specific to the scene.

Figure 11 graphically presents classification results for the testing set (clouds #12 to #15), using the perspective projection. Note that most classification errors occur in transition areas from ground to obstacle. Due to noise, these 'gray' areas are hard to classify even for humans with prior information about the scene.

The SAC method for ground extraction proposed by Konolige et al. (2009) is one step of our algorithm (section 3.1); however, we use an enhanced RANSAC cost function. For comparison, accuracy was evaluated for the MSAC step alone (Figure 7 (e)), following the same procedure as that used for the overall

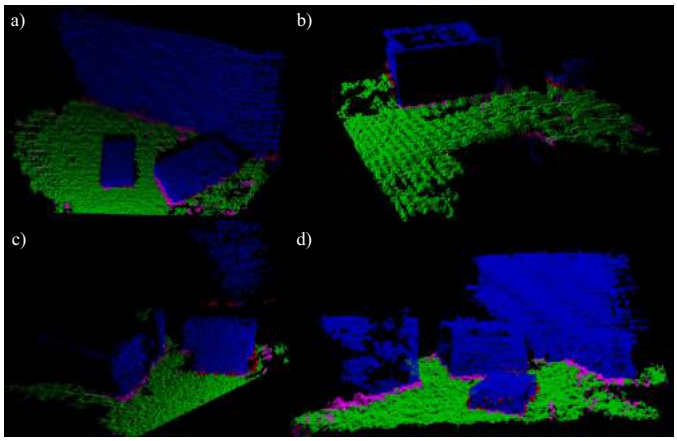


Fig. 11. Final results for the testing set: clouds #12 to #15 (a)-(d). Green and blue represent correctly-classified ground and non-ground, respectively. Pink and red represent misclassified ground and non-ground, respectively.

accuracy assessment. The complete method bestows a clear accuracy advantage in all tests. Figure 2 (b) and (f) geometrically demonstrate typical differences between the two segmentation methods results. MSAC does not adapts to local details, leaving several gaps over the detected floor that are slightly above or below the fitting plane. The 4-ary subdivision is a necessary refinement for adaptive-region ground segmentation.

6. Conclusions

This paper has presented a fast segmentation algorithm for stereo point clouds that classifies points into those belonging to ground and those which do not, also providing certainty values. Accuracy is quantitatively assessed and error cases are presented graphically. Most errors occur in areas where even a careful human labeler will encounter low certainty.

Performance is proven to be appropriate for real-time execution in low cost embedded systems. The proposed method provides a good, alternative, 3D sensing solution for low budget and small size robotics projects that cannot accommodate big and expensive active scanners. Future works will focus on exploiting the parallel nature of the algorithm to optimize performance, and also on integrating the algorithm within a localization and mapping framework.

Acknowledgments

This work has been supported by CAPES Foundation, Brazil.

References

Behley, J., Steinhage, V., Cremers, A., 2015. Efficient radius neighbor search in three-dimensional point clouds, in: Robotics and Automation (ICRA), 2015 IEEE International Conference on, pp. 3625–3630.

Bewley, A., Upcroft, B., 2013. Advantages of exploiting projection structure for segmenting dense 3d point clouds, in: Australian Conference on Robotics and Automation.

Chang, C., Chatterjee, S., 1992. Quantization error analysis in stereo vision, in: Signals, Systems and Computers, 1992. 1992 Conference Record of The Twenty-Sixth Asilomar Conference on, IEEE. pp. 1037–1041.

Douillard, B., Underwood, J., Kuntz, N., Vlaskine, V., Quadros, A., Morton, P., Frenkel, A., 2011. On the segmentation of 3d lidar point clouds, in: Robotics and Automation (ICRA), 2011 IEEE International Conference on, pp. 2798–2805.

Fischler, M.A., Bolles, R.C., 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM 24, 381–395.

Foix, S., Alenya, G., Torras, C., 2011. Lock-in time-of-flight (tof) cameras: a survey. Sensors Journal, IEEE 11, 1917–1926.

Gupta, M., Yin, Q., Nayar, S.K., 2013. Structured light in sunlight, in: Computer Vision (ICCV), 2013 IEEE International Conference on, IEEE. pp. 545–552.

Hampp, J., Bormann, R., 2013. Quadtree-based polynomial polygon fitting, in: Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on, pp. 4207–4213.

Hernandez, J., Marcotegui, B., 2009. Point cloud segmentation towards urban ground modeling, in: Urban Remote Sensing Event, 2009 Joint, pp. 1–5.

Himmelsbach, M., Luettel, T., Wuensche, H., 2009. Real-time object classification in 3d point clouds using point feature histograms, in: Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on, pp. 994–1000.

Hirschmüller, H., 2005. Accurate and efficient stereo processing by semi-global matching and mutual information, in: Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, IEEE. pp. 807–814.

Holz, D., Holzer, S., Rusu, R.B., Behnke, S., 2011. Real-time plane segmentation using rgb-d cameras, in: RoboCup 2011: robot soccer world cup XV. Springer, pp. 306–317.

Klasing, K., Wollherr, D., Buss, M., 2008. A clustering method for efficient segmentation of 3d laser data, in: Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on, pp. 4043–4048.

Konolige, K., Agrawal, M., Blas, M.R., Bolles, R.C., Gerkey, B., Solà, J., Sundaresan, A., 2009. Mapping, navigation, and learning for off-road traversal. Journal of Field Robotics 26, 88–113.

Levinson, J., Askeland, J., Becker, J., Dolson, J., Held, D., Kammel, S., Kolter, J.Z., Langer, D., Pink, O., Pratt, V., et al., 2011. Towards fully autonomous driving: Systems and algorithms, in: Intelligent Vehicles Symposium (IV), 2011 IEEE, IEEE. pp. 163–168.

Marcon, G.A., Ferrao, V.T., Vinhal, C.D.N., Cruz Junior, G., 2016. Performance analysis for a novel adaptive algorithm for real-time point cloud ground segmentation. The International Journal of Hybrid Intelligent Systems (IJHIS) 12:3.

Moosmann, F., Pink, O., Stiller, C., 2009. Segmentation of 3d lidar data in non-flat urban environments using a local convexity criterion, in: Intelligent Vehicles Symposium, 2009 IEEE, pp. 215–220.

Oniga, F., Nedevschi, S., 2010. Processing dense stereo data using elevation maps: Road surface, traffic isle, and obstacle detection. Vehicular Technology, IEEE Transactions on 59, 1172–1182.

Park, J., Choi, S., Yu, W., 2014. Quadtree sampling-based superpixels for 3d range data, in: Robotics and Automation (ICRA), 2014 IEEE International Conference on, pp. 5495–5501.

Schnabel, R., Wahl, R., Klein, R., 2007. Efficient ransac for point-cloud shape detection, in: Computer graphics forum, Wiley Online Library. pp. 214–226.

Schoenberg, J., Nathan, A., Campbell, M., 2010. Segmentation of dense range information in complex urban scenes, in: Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on, pp. 2033–2038.

Strom, J., Richardson, A., Olson, E., 2010. Graph-based segmentation for colored 3d laser point clouds, in: Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on, pp. 2131–2136.

Torr, P.H., Zisserman, A., 2000. Mlesac: A new robust estimator with application to estimating image geometry. Computer Vision and Image Understanding 78, 138–156.

Van Der Mark, W., Gavrila, D.M., 2006. Real-time dense stereo for intelligent vehicles. Intelligent Transportation Systems, IEEE Transactions on 7, 38–50.

Yang, M.Y., Förstner, W., 2010. Plane detection in point cloud data, in: Proceedings of the 2nd int conf on machine control guidance, Bonn, pp. 95–104.

Zhang, C., Wang, L., Yang, R., 2010. Semantic segmentation of urban scenes using dense depth maps, in: Computer Vision–ECCV 2010. Springer, pp. 708–721.