UNIVERSIDADE DE PASSO FUNDO

Matheus Maroso da Silva

SOFTWARE DE CONTROLE DE DESLOCAMENTO ROBÓTICO POR IMAGEM RGB

Passo Fundo

2018

Matheus Maroso da Silva

SOFTWARE DE CONTROLE DE DESLOCAMENTO ROBÓTICO POR IMAGEM RGB

Trabalho apresentado ao curso de Engenharia Elétrica, da Faculdade de Engenharia e Arquitetura, da Universidade de Passo Fundo, como requisito parcial para obtenção do grau de Engenheiro Eletricista, sob orientação do professor Me. Eng. Joan Michel Levandoski.

Passo Fundo

2018

Matheus Maroso da Silva

Software de controle de deslocamento robótico por imagem RGB

Trabalho apresentado ao curso de Engenharia Elétrica, da Faculdade de Engenharia e Arquitetura, da Universidade de Passo Fundo, como requisito parcial para obtenção do grau de Engenheiro Eletricista, sob orientação do professor Me. Eng. Joan Michel Levandoski.

Aprovado em ____ de _____ de _____.

BANCA EXAMINADORA

_____

Prof. Me. Eng. Joan Michel Levandoski - UPF

_____

Prof. Dr. Eng. Fernando Passold - UPF

_____

Prof. Dr. Eng. Jocarly Patrocínio de Souza - UPF

“Para toda ação haverá uma reação de igual intensidade e em sentido oposto”.

Isaac Newton

RESUMO

O processamento de imagens coloridas é uma área de estudo que possibilita inúmeras aplicações. Dentro desse contexto foi desenvolvido o projeto que utiliza de 5 cores: o vermelho, o verde, o azul, o amarelo e o laranja, como referência na locomoção de um protótipo robótico, neste caso foi utilizado o kit LEGO NXT 2.0. A captura da imagem para o processamento é feita pela câmera do Raspberry Pi 3 que a envia ao software MatLab onde é executado o programa desenvolvido, que faz a extração das coordenadas das cores, através destes dados é executado os comandos de locomoção.

Palavras-Chave: Processamento de Imagem. RGB. Raspberry Pi. MatLab. LEGO NXT.

ABSTRACT

The processing of colored images is an area of study that enables numerous applications. Within this context, the project was developed using 5 colors: red, green, blue, yellow and orange, as a reference in locomotion of a robotic prototype, in this case the LEGO NXT 2.0 kit was used. The capture of the image for processing is done by the camera of Raspberry Pi 3 that sends it to MatLab software where the developed program is executed, which extracts the coordinates of the colors, through this data the locomotion commands are executed.

Keywords: Image Processing. RGB. Raspberry Pi. MatLab. LEGO NXT.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE SIGLAS

RGB – *Red Green Blue*

R.U.R. – *Rossumovi Univerzální Roboti*

RIA – *Robotic Industries Association*

ISO – *International Organization for Standardiation*

IoT – *Internet of Things*

IIoT – *Industrial Internet of Things*

AGV – *Automatic Guided Vehicle*

VDI – *Verein Deutscher Ingenieure*

EM – Espectro Eletromagnético

SUMÁRIO

1 INTRODUÇÃO

Neste capítulo serão abordados os objetivos gerais e específicos, o contexto e a justificada do projeto, informações importantes que levaram a escolha do assunto e o que ele fará.

1.1 CONTEXTO

As 3 primeiras revoluções industriais trouxeram a produção em massa, as linhas de montagem, a eletricidade e a tecnologia da informação, elevando a renda dos trabalhadores e fazendo da competição tecnológica o cerne do desenvolvimento econômico. A quarta revolução industrial, que terá um impacto mais profundo e exponencial, se caracteriza, por um conjunto de tecnologias que permitem a fusão do mundo físico, digital e biológico. Os robôs dotados com um sistema de visão ganharam muito espaço nessa indústria, pois possuem maior flexibilidade, autonomia e certa inteligência artificial, obtida através da captura, do processamento e da interpretação de imagens e objetos, cujas posições e orientações passam a ser conhecidas via sistema computacional de apoio, associado ao controle do manipulador. São capazes de interagir com o meio de trabalho de forma a responder em tempo real a alterações operacionais.

O processamento de imagem é utilizado em várias áreas da indústria e do conhecimento atualmente, contudo o mesmo quase que totalmente trabalha só em tons de cinza, já a utilização de imagens coloridas gera maiores distinções e separações de objetos abrindo uma nova gama de possibilidades.

Pensando nisso, escolheu-se desenvolver o projeto junto a um veículo robótico, visto que a maioria utiliza de fitas magnéticas ou por meio de trilhos, surgiu a ideia de fazer a leitura por meio de imagem e executar esse deslocamento pelos dados obtidos nessa matriz.

1.2 OBJETIVO GERAL

Desenvolver um programa que processe imagens atráves de uma única câmera, buscando as cores RGB e retirando suas coordenadas, realizando assim a execução de comandos para guiar o protótipo movel até os pontos desejados.

1.3 OBJETIVOS ESPECÍFICOS

Os objetivos específicos do projeto são:

a) Mapear a área de implementação por meio de câmera alocada no ambiente;
b) Processar as imagens retiradas do local;
c) Controlar um protótipo a partir do processamento de imagem obtido;
d) Realizar testes para verificação do funcionamento do sistema.

1.4 JUSTIFICATIVA

O processamento de imagem colorida é uma área de estudo que há muito campo a ser explorado e poucos trabalhos realizados nessa área, grande parte trabalha com tons de cinza. Como as cores geram maiores distinções e separações de objetos se abre uma nova gama de possibilidades, como por exemplo, soluções para a agricultura, saúde e robótica. Na robótica surgiu a ideia de fazer o controle de locomoção de um AGV por câmera. O sistema agrega vantagens quando necessária a atualização ou mudança do layout fabril. Isto implica que não será preciso executar a troca de trilhos ou guias. De forma facilitada pode-se realizar a mudança dos pontos de entrega e recebimento de cargas. Com isto, pensou-se em utilizar desta tecnologia para fazer o controle de um protótipo robótico veicular buscando e guiando-se a partir de cores selecionadas.

2 REVISÃO DA LITERATURA

Para a compreensão do projeto e do seu funcionamento, é de fundamental importância apresentar conceitos que serviram para a sua concepção. Neste capítulo, serão apresentados esses conceitos.

2.1 ROBÔ

O termo Robô (Robot) foi utilizado pela primeira vez em 1921 por Karel Čapek escritor checo que mencionou tal vocábulo em sua peça dramática nomeada de "Robôs Universais de Rossum (R.U.R. – *Rossumovi Univerzální Roboti*)", onde usou da escrita para definir um autômato que se rebelou contra o ser humano. A palavra robô deriva de "robota" que tem origem eslava e significa "trabalho forçado" (ROMANO, 2002).

Na década de 40, o escritor e bioquímico Issac Asimov forneceu mais detalhes do que supostamente seria um robô, segundo ele a máquina teria duas características principais, uma aparência humana, e não possuir sentimentos. A programação feita pelos humanos definiria seu comportamento e nela haveria regras preestabelecidas que deveriam ser cumpridas. Além dessas informações Asimov criou o termo robótica que designou como sendo a ciência que estuda os robôs (ROMANO, 2002).

2.2 ROBÔTICA INDUSTRIAL

Com a descoberta da máquina a vapor por James Watt em meados do século XVIII teve-se o início da Primeira Revolução Industrial, que mudou um sistema baseado em manufatura para um processo mecânico. Já entre 1860 e 1900 é datado o início da Segunda Revolução Industrial onde com a expansão do mercado os países de primeiro mundo atualizaram os sistemas de produção obtendo, novas formas de energia como a elétrica e a derivada do petróleo, além da substituição do ferro para o aço. Posterior à data de 1900, já pensando em indústrias de larga escala, começou-se os estudos para a otimização das fábricas, chegando aos modelos de hoje em dia, com sistemas eletrônicos e com o auxílio de simuladores e programas computadorizados implantando assim, um sistema robótico garantindo mais precisão e velocidade, gerando uma produção em escala e caracterizando a Terceira Revolução Industrial (SELEME, Robson; SELEME, Roberto, 2013).

Grandes investimentos estão sendo realizados nas últimas décadas em robôs industriais devido às necessidades impostas pelo mercado de trabalho em obter um sistema de produção mais automatizado e dinâmico. Isto, em virtude das suas características de flexibilidade de programação e adaptação a sistemas integrados de manufatura (ROMANO, 2002).

A Associação de Robótica Industrial (RIA - *Robotic Industries Association*) definiu robô industrial como sendo um manipulador versátil reprogramável com o intuito de movimentar ferramentas, materiais, partes e peças especiais, de modo a seguir os movimentos predeterminados pela programação para execução do trabalho desejado (RIVIN, 1988).

Já a norma ISO (*International Organization for Standardiation*) 10218 concede uma definição mais complexa de robô industrial, "uma máquina manipuladora com vários graus de liberdade controlada automaticamente, reprogramável, multifuncional, que pode ter base fixa ou móvel para utilização em aplicações de automação industrial", é possível observar um desses modelos na Figura 1 (ROMANO, 2002, p. 15).

Figura 1 - Robô Industrial

Fonte: http://step-automation.com.br/2-1-industrial-robots/226185

Com o processo de automação das indústrias alguns objetivos surgiram para se executar a produção automatizada (BOUTEILLE at al., 1997):

- Redução de custos nos produtos fabricados, através de: diminuição do número de pessoas envolvidas na produção, aumento da produtividade, melhor aproveitamento da matéria prima e economia de energia;
- Melhorar as condições de trabalho, eliminando atividades perigosas ou nocivas ao ser humano;
- Melhora da qualidade dos produtos, por meio do controle mais eficaz do meio de produção;
- Realizar atividades de extrema dificuldade ou impossíveis para as pessoas, como a montagem de peças em miniatura, e a coordenação de movimentos complexos ou muito rápidas, por exemplo, no descolamento de materiais e objetos.

2.3 INDÚSTRIA 4.0

A ideia básica consiste em conectar máquinas, sistemas e ativos, criando redes inteligentes ao longo de toda produção podendo controlar o processo de forma automática, gerando assim capacidade e autonomia para agendar manutenções, prever falhas, e se adaptar aos requisitos e mudanças não planejadas na produção (SILVEIRA, 2017).

Seis princípios são necessários para o desenvolvimento e implantação da indústria 4.0, são eles (SILVEIRA, 2017):

- Capacidade de operação em tempo real: aquisição e tratamento de dados de forma rápida quase que instantânea, que permite tomada de decisões em tempo real;
- Virtualização: simulações e sistemas supervisórios em maiores escalas tendo cópias virtuais das fábricas inteligentes, o que permitirá a rastreabilidade e monitoramento remoto de todos os processos devido aos inúmeros sensores que são espalhados ao longo de empresa;
- Descentralização: a tomada de decisões poderá ser feita pelo sistema cyber-físico dependendo das necessidades da produção em tempo real. As máquinas não receberão apenas comandos, elas poderão fornecer informações e dados do ciclo de trabalho;

- Orientação a serviços: com a utilização de arquiteturas de software orientadas a serviços junto com o conceito de *Internet of Services*;
- Modularidade: maior flexibilidade na alteração das tarefas das máquinas, fazendo a produção de acordo com a demanda, podendo acoplar e desacoplar módulos dependendo da necessidade.

A indústria 4.0 já é uma realidade e isso se deve a várias áreas de tecnologia e desenvolvimento, entre elas podem-se citar três de grande importância: a Internet das Coisas, *Big Data Analytics*, e a Segurança (SILVEIRA, 2017).

2.3.1 Internet das Coisas

A Internet das Coisas (IoT – *Internet of Things*) é a técnica que permite conectar informações de dispositivos em geral ligados à rede. A digitalização destes dados pode ser de máquinas, processos e dispositivos. Esses dados complementam a camada operacional de uma planta industrial, a interconexão dos dados e sistemas da empresa, permitem formar o ecossistema cibernético, onde se obtém a interoperação completa e total da planta industrial, chamando assim de planta digital (VENTURELLI, 2017).

No processo de digitalização da produção industrial existem alguns cuidados que devem ser tomados para obter um desempenho aceitável, é necessário digitalizar os movimentos dos ativos, documentos e cenários para o planejamento e controle da qualidade, além de fazer a conexão da logística, dos fornecedores e dos suprimentos permitindo uma gestão em tempo real (VENTURELLI, 2017).

A digitalização de dados da indústria é um desafio por apresentar alguns problemas que precisam ser solucionados como criar uma rede de informações complementar na produção que permita planejar e monitorar a produção e manutenção em tempo real, conectar redes independentes, como logística, fornecedores, laboratórios nas redes industriais, estabelecendo padronização e segurança da informação nas redes de IoT na indústria (VENTURELLI, 2017).

A ideia de conectar qualquer dispositivo que consiga gerar informações e possa se conectar a um serviço de nuvem (cloud), é denominado de Internet das Coisas e pode ser usado tanto em casas, hospitais, nos esportes e empresas (VENTURELLI, 2017).

Contudo nas empresas elas recebem outro nome que é Internet Industrial das Coisas (IIoT - *Industrial Internet of Things*), onde executam a mesma função da IoT, apenas por algumas diferenças: poder se comunicar com o fornecedor em tempo real para fazer a análise da

qualidade do produto, além de ajudar na logística empresarial de entrada e saída de materiais controlando e analisando a produção (VENTURELLI, 2017).

Com a implementação desses métodos é visível ganhos em várias áreas da indústria, como (VENTURELLI, 2017):

- Redução de operações ou paradas;
- Melhoria do uso do ativo;
- Redução do custo do ciclo do ativo;
- Melhoria da produção;
- Aumento da rapidez na tomada de decisões;
- Oportunidade para novos negócios;
- Permitir venda ou compra de produtos como serviço.

2.3.2 Big Data Analytics

Todas as informações geradas a partir dos sensores nas indústrias vão para o *Big Data* que é uma estrutura de análise de dados e tomada de decisões em tempo real e sem intermediários, possibilitando ações autônomas no processo, sem a necessidade de interferência humana, através do aprendizado de máquina (*machine learning*), visando sempre o resultado final que é apontar os melhores caminhos dentro do ecossistema produtivo (VENTURELLI, 2017).

No início os primeiros controles só apoiavam a tomada de decisões do operador, pois era ele que detinha o conhecimento para execução da tarefa. Todavia com a evolução das redes, podem-se gravar as informações do processo e analisá-las, tomando assim decisões com base nos dados conhecidos e gerados no local (VENTURELLI, 2017).

Todos os setores de uma indústria são ligados ou tem conexão com um sinal de internet, a informação gerada através deles é depositada na nuvem da empresa que é uma parte do *Big Data* possibilitando assim fazer várias ações intermediárias como a análise de dados, de cenários, fazer projeções, planejamentos, a análise de qualidade e prognóstico. Tudo que permita fazer resoluções (VENTURELLI, 2017).

Com a implantação do sistema alguns benefícios são destacados como (VENTURELLI, 2017):

- Diminuição de operadores: o sistema tomará decisões, operações de melhor desempenho, segurança de planta e economia de energia;
- Fim do planejamento reativo: o sistema que será virtualizado, realimentando o processo que sempre estará em tempo real nos indicadores para toma de decisões;
- Todo o sistema será preditivo: manutenção, risco, aproveitamento e atuará no processo para conhecimento.

De forma geral o *Big Data* é um serviço dentro da Indústria 4.0 que compõe um cibersistema, onde as informações adquiridas em toda a indústria estejam armazenadas na nuvem e esta transmita as informações ao conjunto para executar as ferramentas de mineração de dados e aprendizado de máquinas gerando assim resultados (VENTURELLI, 2017).

2.3.3 Segurança

A conectividade que se espera e se quer no sistema vem acompanhada de uma forte ameaça que as empresas negligenciam que é a falta de segurança de seus dados. É implantado um forte sistema de segurança físico no ambiente de trabalho, mas não no meio computacional que rege com mais força o futuro da instituição (VENTURELLI, 2017).

Algumas ações básicas podem ser citadas para uma implementação inicial de segurança, tanto no nível físico como lógico, são (VENTURELLI, 2017):

- Autenticação de usuários e equipamentos;
- Controle de acesso: físico e lógico;
- Detecção de intrusão: físico e lógico;
- Criptografia de dados;
- Assinatura digital;
- Isolamento e/ou segregação de ativos;
- Varredura de vírus;
- Monitoramento de atividade sistema/rede;
- Segurança perimetral de planta.

2.4 AGV

O Veículo Guiado Automaticamente (AGV – *Automatic Guided Vehicle*) é um robô dedicado ao transporte de materiais e a transferência ou armazenagem de objetos e itens dentro de uma indústria que não necessitam de um operador humano para guiá-lo. Tem tamanho reduzido, e são ideais para ambientes que possuem fluxo repetitivo e deslocamento de produtos entre pontos previamente determinados devido ao seu transporte seguro e eficaz (NOGUEIRA; TEIXEIRA; FREITAS, 2015) (SILVA; QUEIROZ; CURY, 2010).

Tem como objetivo poupar tempo na execução de atividades, redução de custos na produção e otimização da área produtiva da indústria. Conseguem se comunicar uns com os outros para melhor aproveitamento das rotas, além de solicitar informações para outras máquinas e equipamentos podendo ser instalado em qualquer ambiente (AGVS, 2013).

Existem diferentes modelos que dependem do local e da aplicação em que serão subjugados, podendo trabalhar nas indústrias automotivas, alimentícias e farmacêuticas, os mais comuns são (AGVS, 2013):

- Rebocadores: são similares ao transporte realizado por trens, são utilizados para transportar vagões que podem ser uma variedade de racks sobre rodas, suas principais características são sistemas de locomoção com alta tração e restrição de raio das curvas do percurso para executar o transporte de comboios. São muito populares na indústria automobilística que nela tem como principal atuação o transporte de kits de peças na produção. Observa-se na Figura 2 um modelo de AGV rebocador.

Figura 2 - AGV Rebocador

Fonte: http://www.agvs.com.br/agv-rebocador.html

- Carregadores: o transporte realizado por esse tipo é semelhante à dos caminhões, eles são utilizados para transporte de cargas unitárias podendo ser uma simples caixa até um rack complexo, sua principal característica é poder circular em lugares com pouco espaço para a execução de manobras podendo até mesmo girar em seu próprio eixo, contudo para a perfeita realização do trabalho a carga deverá estar em uma base rolante ou em um suporte tipo mesa para que o robô entre sob elas para levantar a superfície e fazer o transporte, na Figura 3 observamos um AGV carregador.

Figura 3 - AGV Carregadores

Fonte: http://www.agvs.com.br/agv-carregador.html

Algumas vantagens ressaltam a utilização do AGV, são elas (AGVS, 2013):

- Precisão e segurança de funcionamento;
- Previsibilidade, os percursos são sempre os mesmos;
- Eliminação de operações que não agregam valor ao produto;
- Padronização do processo;
- Possibilidade de trabalho 24 horas por dia;
- Implementação simples e rápida de novos processos;
- Redução do nível de ruídos em comparação à movimentação com rebocadores;

- Redução de custos de operação em relação aos processos convencionais de transporte industrial;
- Contribuição para preservação do meio ambiente, zero emissões de poluentes.

No ramo automotivo os AGVs são responsáveis pelo reboque de carrinhos com peças do armazém até a linha de produção, auxiliam na montagem transportando o produto por diversas estações, onde os operários vão dando forma ao objeto, e fazem a transferência entre estações ou áreas do processo produtivo dentro da indústria (AGVS, 2013).

Na indústria alimentícia ele é usado para fazer o transporte da matéria prima dos estoques até as máquinas e posteriormente do produto final até os locais de expedição (AGVS, 2013).

Há também AGVs em hospitais executando o transporte de medicamentos, alimentos, roupas de cama e uniformes até as lavanderias e movendo as macas entre os leitos (AGVS, 2013).

Entre todas as aplicações demonstradas a que mais se destaca é no setor de logística na manipulação e organização de armazéns podendo ser dinâmicos ou estáticos, fazendo o transporte de matéria prima ou produto acabado entre pontos de recebimento e o processo de industrialização, posteriormente para pontos de expedição, executa também a locomoção de cargas específicas os quais não podem ser feitas através de equipamentos comuns como empilhadeiras e rebocadores, por exemplo, produtos de grande volume ou peso, e aqueles que precisam de um maior cuidado ou delicadeza. Um exemplo de empresa que usa de AGVs para logística é a Amazon® (AGVS, 2013).

Atualmente, no mercado se destacam cinco tipos de orientações diferentes, estas podem ser escolhidas pela empresa para implantação do robô, são eles (AGVS, 2013):

- Óptico: onde sensores iram detectar uma faixa branca entre duas pretas, podendo estas ser pintadas ou fitas podem ser utilizadas no piso para fazer a demarcação, é a opção mais barata de implementação e é recomendada para áreas de pouco tráfego;
- Indutivo: sensores detectam uma faixa metálica no chão está por sua vez, pode ser uma fita de metal ou chapas com o recorte do percurso planejado;
- Indutivo por frequência: este modelo conta com uma antena que identifica o sinal emitido por um cabo que é colocado em um corte de 20 mm abaixo da superfície, tal aplicação é recomendada em áreas de altíssimo tráfego tanto de empilhadeiras quanto de máquinas de arraste, onde há a possibilidade de danificação de faixas pintadas ou

tiras metálicas, contudo para esses sensores é necessário um gerador de frequência que deve ser instalado próximo ao percurso;

- Magnética: a detecção de uma faixa magnética é o que ocorre neste modelo, mas novamente é preciso ser um local de pouco tráfego de empilhadeiras;
- Laser: um sensor laser é fixado no AGV que se orienta por pontos reflexivos que são fixados em pontos estratégicos como, colunas e paredes. Devido a complexidade dos equipamentos utilizados esse meio de orientação é a mais cara dos outros quatro apresentados, todavia seu uso é recomentado para áreas de tráfego intenso onde não é possível cortar o piso.

O uso constante e elevado de AGVs nas indústrias fez com que se criassem normas e padrões para garantir sua segurança e funcionalidade. A norma VDI 2510 afirma que a velocidade máxima permitida é de 1 m/s ou 3,6 Km/h, segundo a norma regulamentado NR-26 (26.1.5.3 Amarelo (126.004-9/12)) a cor amarela deverá ser predominante neles. Ainda, é recomendado ter sensores de obstáculo para evitar colisões com pessoas, máquinas ou estruturas, para-choques são interessantes no caso de houver alguma falha e os sensores não executarem devidamente suas funções, para não causarem grandes danos e nem trazerem grande risco aos funcionários do local, e o uso de sinalizadores também é uma forma de avisar caso algo esteja errado ou que o veículo está passando pelo local (AGVS, 2017).

2.5 PROCESSAMENTO DIGITAL DE IMAGENS

A definição de imagem pode ser obtida por uma função bidimensional $f\ (x, y)$, onde x e y são coordenadas espaciais no plano, e a amplitude de f em qualquer par de coordenadas (x, y) é nomeada de intensidade ou nível de cinza da imagem nesse ponto. Denomina-se de imagem digital quantidades finitas e discretas de x, y e f. Com o auxílio de um computador digital pode-se processar imagens digitais que são formadas por um número finito de elementos cada um com uma localização e com um valor específico, essas medidas podem ser nomeadas como elementos pictóricos, elementos de imagem, pels ou *pixels*, onde *pixel* é a palavra mais utilizada para definir uma imagem digital, e a esse campo recebe o nome de processamento digital de imagens (GONZALEZ e WOODS, 2010).

Segundo Marques Filho e Vieira Neto (1999), um pixel *p*, com coordenadas *(x,y)*, possui quatro vizinhos verticais e horizontais, podendo ser definidas as seguintes coordenadas para os

mesmos, *(x+1, y), (x-1, y), (x, y+1)* e *(x, y-1),* estes por sua vez designão $N_4(p)$ também chamados de “4-vizinhança” (MARQUES FILHO E VIEIRA NETO, 1999).

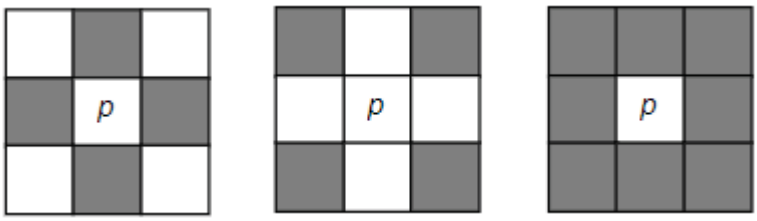
Existem ainda os quatro vizinhos diagonais de *p* que possuem as coordenadas *(x-1, y-1), (x-1, y+1), (x+1, y-1)* e *(x+1, y+1),* formando o conjunto $N_d(p)$ (MARQUES FILHO E VIEIRA NETO, 1999).

A “8-vizinhança” de *p* é definida como (MARQUES FILHO E VIEIRA NETO, 1999):

$$N_8(p) = N_4(p) \cup N_d(p) \qquad (1)$$

É possível observar os tipos de vizinhanças na Figura 4.

Figura 4 - Conceitos de 4-vizinhança, vizinhança diagonal e 8-vizinhança



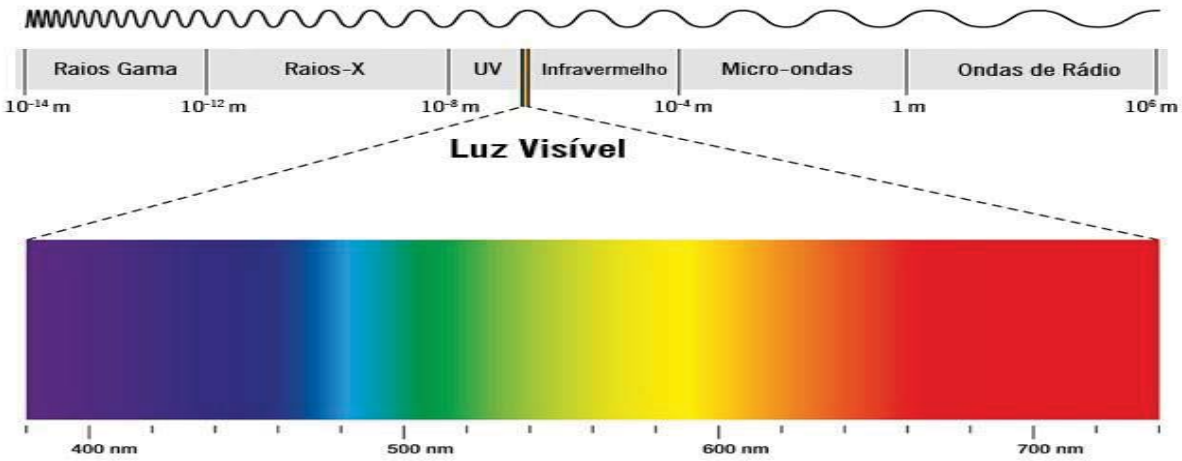
Fonte: Marques Filho e Vieira Neto (1999, p.26)

Dos cinco sentidos que os humanos possuem a visão sem dúvida é o mais avançado e o mais importante devido ao poder que as imagens exercem sobre a percepção humana, contudo as pessoas são limitadas pela banda visual do espectro eletromagnético (EM) apresentado na Figura 5, já os aparelhos de imagens atuais não possuem esse limite e podem variar a sua faixa de percepção desde ondas gama até ondas de rádio, influenciando e abrindo campos de estudo para as mais amplas aplicações, no Quadro 1 é mostrado a diferença entre o sistema visual humano e artificial disponível ao público, visto que é possível produzir câmeras diferentes dependendo da aplicação, todavia seu custo é elevado dificultando o acesso. (GONZALEZ e WOODS, 2010).

Não existe praticamente mais nenhuma área de empreendimento técnico hoje em dia que não utilize o processamento digital de imagens de alguma forma, podem-se citar exemplos como os raios-X, microscopia ótica, detecção de objetos em indústrias, radiografias, entre outras, todas utilizam das ondas eletromagnéticas que podem ser interpretadas como ondas senoidais, tendo vários comprimentos, que propagam, ou podem ser vistas como um fluxo de partículas sem massa, se movendo na velocidade da luz e em seu próprio padrão ondulatório,

as partículas sem massa contém energia, que se denomina fóton (GONZALEZ e WOODS, 2010).

Figura 5 - Espectro Eletromagnético



Fonte: https://www.todamateria.com.br/espectro-eletromagnetico/

Quadro 1 – Diferenças entre o Sistema Visual Humano e Artificial

	Sistema Visual Humano	**Sistema de Visão Artificial**
Espectro	Limitado à faixa de luz visível (300nm a 700nm) do espectro de ondas eletromagnéticas.	Pode operar em praticamente todo o espectro de radiações eletromagnéticas, dos raios X ao infravermelho.
Flexibilidade	Extremamente flexível, capaz de se adaptar a diferentes tarefas e condições de trabalho.	Normalmente inflexível. Apresenta bom desempenho apenas na tarefa para qual foi projetado.
Habilidade	Pode estabelecer estimativas relativamente precisas em assuntos subjetivos.	Pode efetuar medições exatas, baseadas em contagem de pixels, e, portanto, dependentes da imagem digitalizada.
Cor	Possui capacidade de interpretação subjetiva de cores.	Mede objetivamente os valores das componentes R, G e B para determinação de cor.
Sensibilidade	Capaz de se adaptar a diferentes condições de luminosidade,	Sensível ao nível e padrão de iluminação, bem como à distinção

	característica físicas da superfície do objeto e distância do objeto. Limitado na distinção de muitos níveis diferentes de cinza, simultaneamente.	em relação ao objeto e suas características físicas. Pode trabalhar com centenas de tons de cinza, conforme o projeto do digitalizador.
Tempo de resposta	Em torno de 0,1s.	Depende de aspectos de hardware, podendo ser da ordem de 0,001s.
2D e 3D	Pode executar tarefas 3D e com múltiplos comprimentos de onda (dentro do espectro visível) facilmente.	Executa tarefas 2D com relativa facilidade, mas é lento e limitado em tarefas 3D.
Percepção	Percebe variações de brilho em escala logarítmica. A interpretação subjetiva de brilho depende da área ao redor do objeto considerado.	Pode perceber brilho em escala linear ou logarítmica.

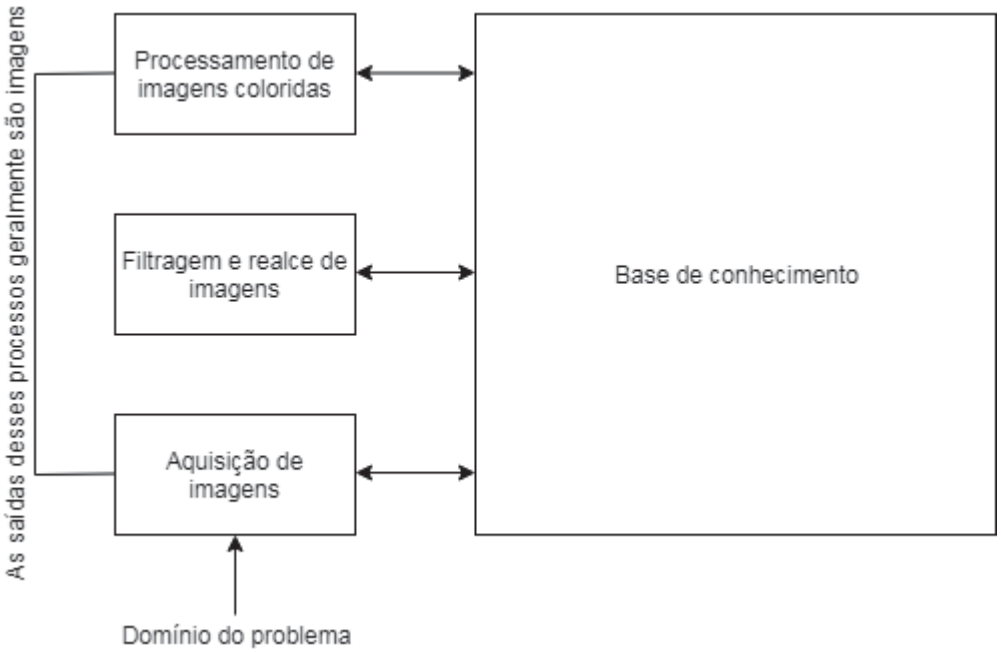
Fonte: Foresti (2006, p. 6)

Alguns passos são fundamentais para o processamento digital de imagem apresentados na Figura 6. O primeiro deles e o que faz os outros funcionaram é a aquisição de imagem que pode ser algo simples como receber uma imagem que já está em formato digital, mas em geral envolve um pré-processamento, como redimensionamento (GONZALEZ e WOODS, 2010).

Já o realce de imagem é o processo de manipular ela para adequá-la a uma aplicação específica, nota-se que a utilização da nova gravura depende da orientação em que se vai dar ela, um processo que realça radiografias não é a melhor abordagem para realçar imagens de satélites (GONZALEZ e WOODS, 2010).

A utilização da cor no processamento de imagens é motivada por dois fatores principais. O primeiro, a cor é um poderoso descritor que muitas vezes simplifica a indentificação do objeto e sua extração de uma cena. Em segundo lugar, os seres humanos são capazes de distinguir milhares de tons e intesidades de cor em comparação com apenas duas dúzias de tons de cinza (GONZALEZ e WOODS, 2010).

Figura 6 - Passos fundamentais em processamento digital de imagens

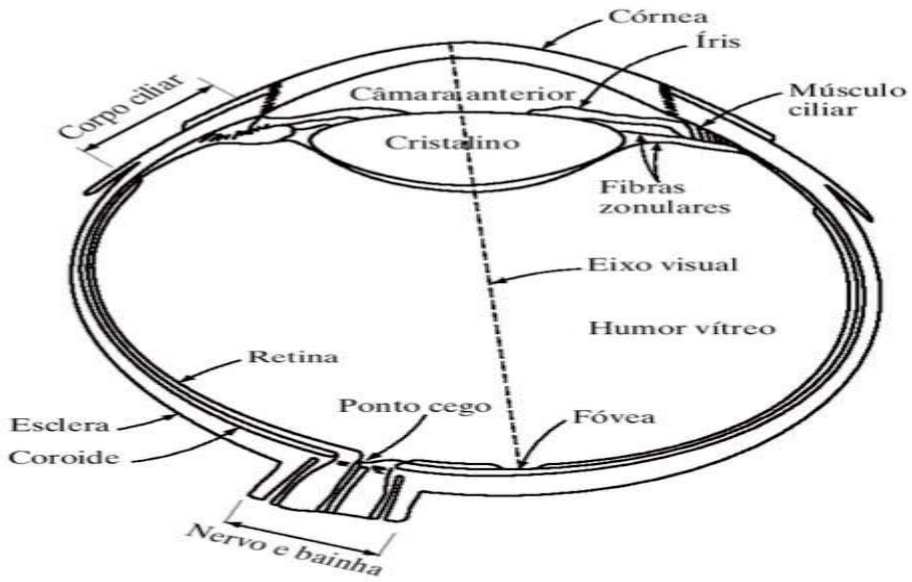


Fonte: Adaptado de Gonzales e Woods (2010, p. 16)

2.5.1 Processamento de Imagens Coloridas

O olho humano (Figura 7) é praticamente uma esfera, com um diâmetro médio de aproximadamente 20 mm. É revestido por três membranas: a córnea, a esclera, a coroide e a retina (GONZALEZ e WOODS, 2010).

Figura 7 - Diagrama simplificado de um corte transversal do olho humano



Fonte: Gonzales e Woods (2010, p. 23)

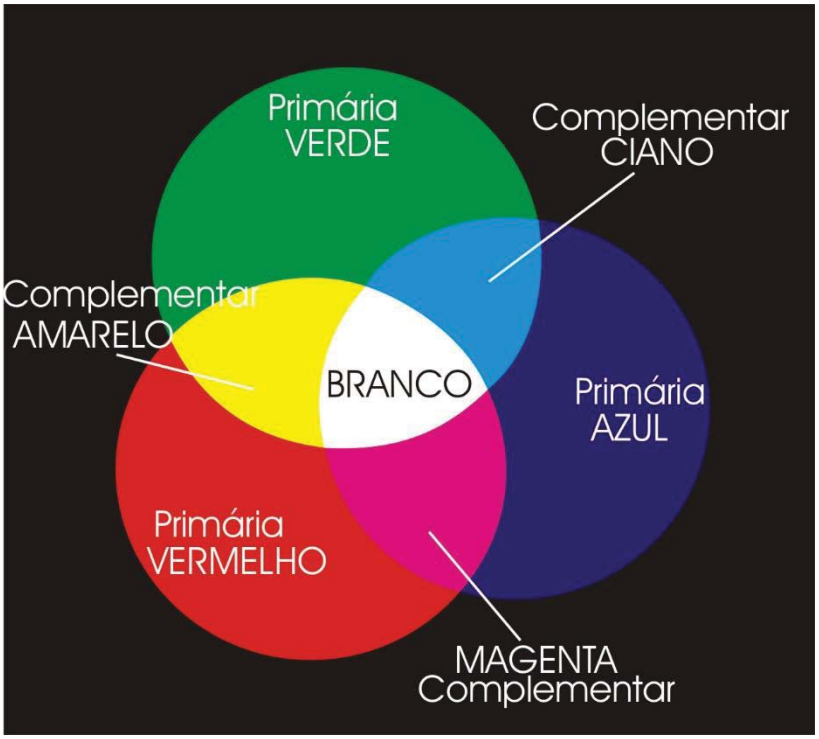
A retina é a membrana mais interna do olho, quando o mesmo está adequadamente focalizado, a luz de um objeto externo forma uma imagem na película. A visão de padrões é obtida pela distribuição de receptores discretos de luz, que são os cones e os bastonetes, ao longo da superfície da camada (GONZALEZ e WOODS, 2010).

Cada olho possui cerca de 6 a 7 milhões de cones. Eles se localizam principalmente na porção central da retina, chamada de fóvea, e são muito sensíveis à cor. Os humanos podem distinguir pequenos detalhes graças a eles, em grande parte porque cada um deles está conectado à sua própria terminação nervosa (GONZALEZ e WOODS, 2010).

Dos tantos milhões de cones citados eles podem ser divididos em três principais categorias de sensoriamento: vermelho, verde e azul, isto se dá ao nível de sensibilidade nas respectivas cores com percentuais de 65 %, 33 % e 2 % (GONZALEZ e WOODS, 2010).

Em virtude dessas características de absorção do olho humano, as cores são vistas como combinações das chamadas cores primárias: vermelho (R, de *red*), verde (G, de *green*) e azul (B, de *blue*), formando o conhecido RGB. Contudo nenhuma cor pode ser chamada isoladamente de vermelho, verde ou azul. Mas ao misturar as mesmas em diversas proporções de intesidade, pode ser geradas novas cores como mostra a Figura 8, todavia não é possível fazer todas as pigmentações existentes com elas (GONZALEZ e WOODS, 2010).

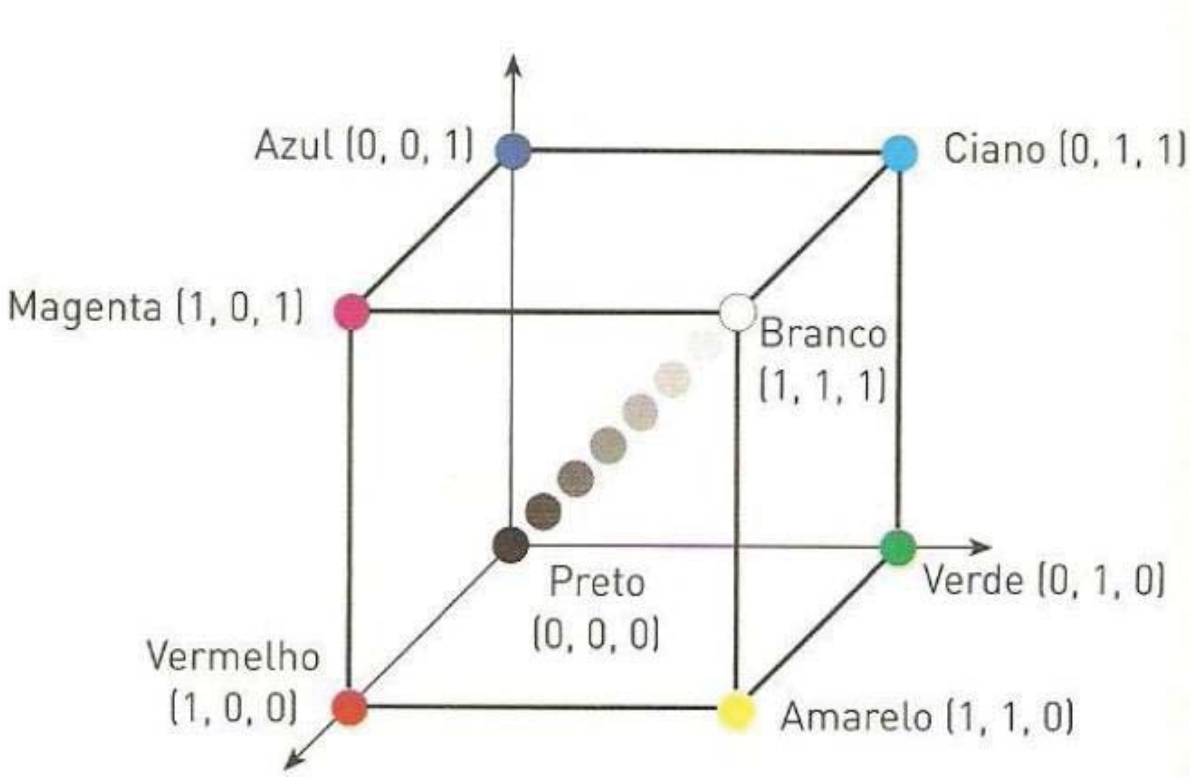
Figura 8 - Cores primárias aditivas de misturas de luz



Fonte: Adaptado de Gonzales e Woods (2010, p. 261)

No modelo RGB, cada cor aparece em seus componentes espectrais primários de vermelho, verde e azul, por sua vez o mesmo é baseado em um sistema de coordenadas cartesianas. O subespaço de cores de interesse é o cubo apresentado na Figura 9, no qual os valores RGB primários estão em três vértices; as cores segundárias ciano, magenta e amarelo, estão em outros três vértices mais distântes da origem. Nesse modelo, a escala de cinza (pontos de valores RGB iguais) estende-se do preto até o branco ao longo do segmento de reta que une esses dois pontos. As diferentes cores nesse modelo são pontos no cubo ou dentro dele e são definidas por vetores que se estendem a partir da origem. Por conveniência, assume-se que todos os valores de cor foram normalizados, de forma que o cubo possui valores unitários (GONZALEZ e WOODS, 2010).

Figura 9 - Cubo de cores RGB



Fonte: Adaptado de Gonzales e Woods (2010, p. 265)

2.6 MATLAB

O *Matlab* é um acrónimo resultante da junção dos termos ***Mat****rix* e ***Lab****oratory*, sendo uma ferramenta informática, interativa e de alto desempenho que foi desenvolvida originalmente no início dos anos 1970 visando à resolução de problemas na manipulação de

matrizes, cálculos de álgebra e de análise numérica. Possui uma linguagem de programação própria, um ambiente com excelentes capacidades gráficas e é vasto em seus conjuntos de funções que são separados por áreas de cálculo científico, denominadas *toolboxes* (MORAIS; VIEIRA, 2006).

Entre os *toolboxes* mais relevantes podemos destacar: o processamento de imagem, que é utilizado para fazer a manipulação e leitura de imagens, a comunicação, que pode fazer a modelização dos diferentes componentes presentes em sistemas de comunicação, e as redes neurais, que podem fazer a simulação e projeto de redes neurais (MORAIS; VIEIRA, 2006).

O elemento de dados básico do *software* é a matriz, podendo essas ser criadas sem dimensionamento prévio, além de contar com um grande conjunto de funções internas (*built-in functions)* possibilitando a resolução de problemas computacionais de forma mais rápida, simples e compacta (MORAIS; VIEIRA, 2006).

Por ter uma linguagem de programação própria algumas características devem ser ressaltadas, como (*MATHWORKS*, R2017b):

- Linguagem de alto nível para computação científica e de engenharia;
- Ambiente de trabalho atento para a exploração iterativa, design e resolução de problemas;
- Gráficos para visualização de dados e ferramentas para a criação personalizada dos mesmos;
- Aplicativos para encaixa de curvas, classificação de dados, análise de sinal, ajuste do sistema de controle e muitas outras tarefas;
- Caixas de ferramentas adicionais para uma ampla gama de aplicações científicas e de engenharia;
- Ferramentas para criação de aplicativos com interfaces de usuário personalizadas;
- Interfaces para C/C++, Java®, .NET, Python, SQL, Hadoop, e Microsoft® Excel®;
- Opções de implantação de royalties para *MATLAB* programas de compartilhamento com os usuários finais.

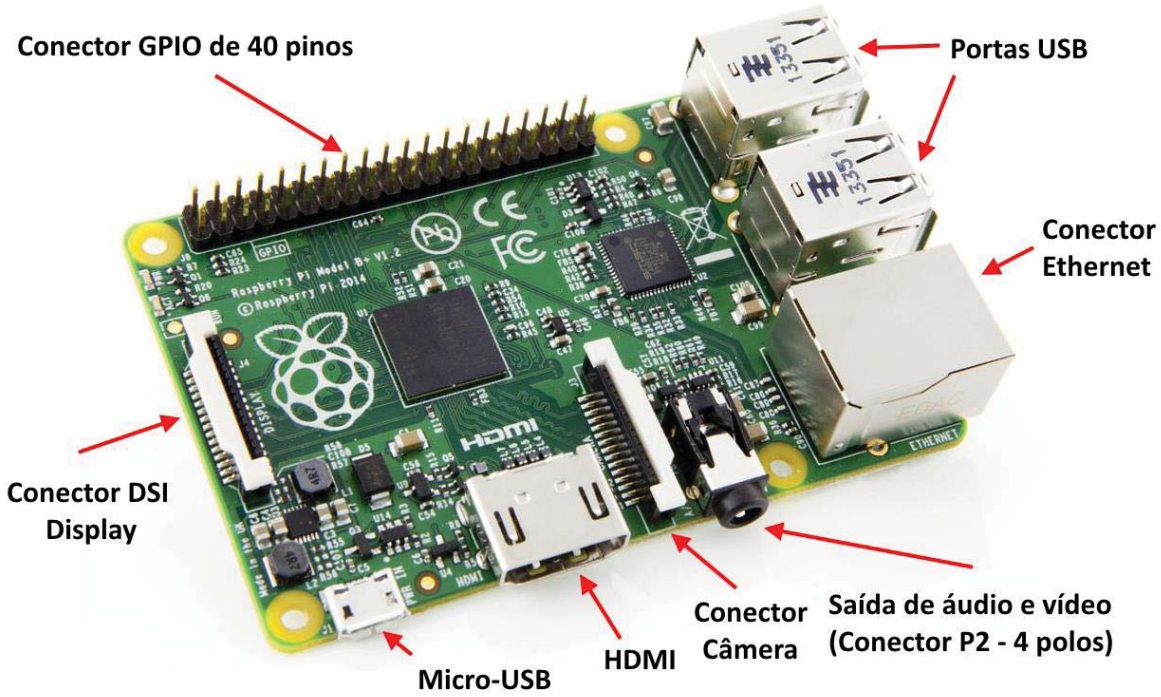
2.7 RASPBERRY PI

O Raspberry Pi é um mini computador do tamanho de um cartão de crédito e com um preço extremamente acessível que executa o Linux como sistema operacional, mais

precisamente a versão Debian, chamada Raspbian própria para ele, um sistema completo contando com um pacote de edição de texto, capacidade de reprodução de vídeo, jogos e todo o resto que um sistema operacional possui (MONK, 2013).

Ele conta com: portas USB, conector para Ethernet, saída de áudio e vídeo, conector para câmera, saída HDMI, entrada para micro-USB, conector DSI display e conector GPIO com 40 pinos como mostra a Figura 10 (THOMSEN, 2014).

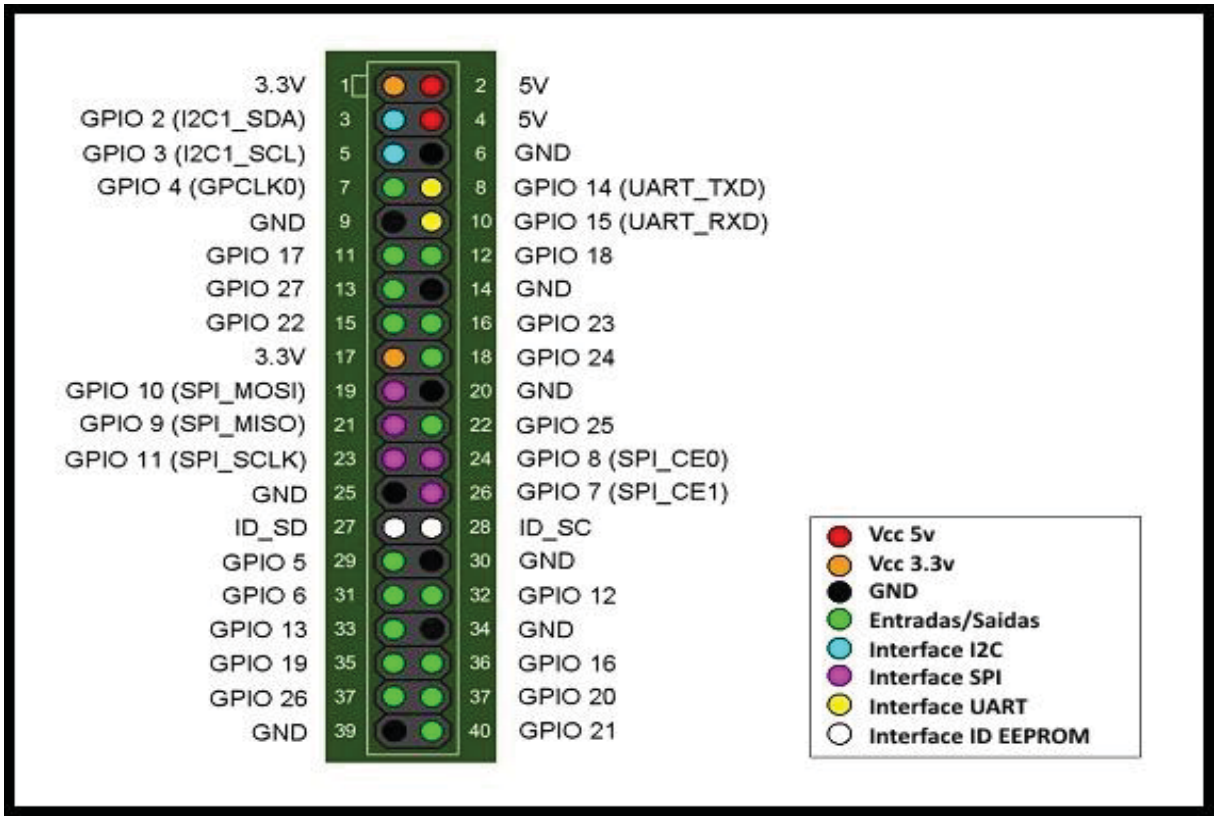
Figura 10 - Raspberry Pi portas de entrada e saída



Fonte: https://www.filipeflop.com/blog/tutorial-raspberry-pi-linux/

Pode-se observar na Figura 11 a que cada pino do conector GPIO corresponde.

Figura 11 - Pinos do conector GPIO



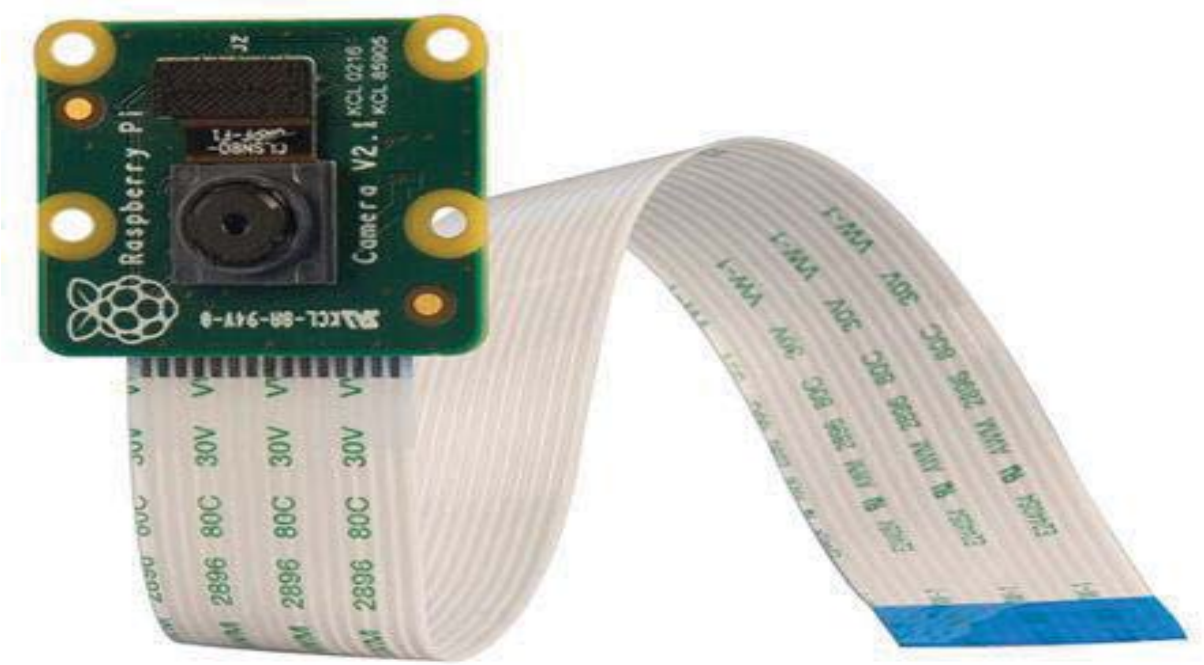
Fonte: https://www.filipeflop.com/blog/tutorial-raspberry-pi-linux/

O Raspberry Pi 3 Model B+ custa 35 dólares e possui uma configuração interna bastante abrangente e robusta, ele possui (FORSTER, 2016):

- Processador Quad-core 64-bit ARM Cortex A53 com clock de 1.2 GHz;
- Em torno de 50% mais rápido que o Raspberry Pi 2;
- 802.11n Wireless LAN;
- Bluetooth 4.1 (incluindo Bluetooth Low Energy);
- 400 MHz VideoCore IV multimídia;
- Memória de 1 GB LPDDR2-900 SDRAM (900 MHz).

A Câmera V2 de 8MP será utilizada no conjunto é de uso próprio para a família Raspberry com encaixe já determinado e com compatibilidade garantida, além de conseguir captar vídeos de 1080p30, 720p60, 640x480p90 e obter uma imagem de até 3280x2464 pixels, a mesma é mostrada na Figura 12.

Figura 12 - Câmera Raspberry Pi V2 8MP



Fonte: https://www.raspberrypi-spy.co.uk/2016/04/8-megapixel-raspberry-pi-camera-module-v2/

3 DESENVOLVIMENTO DO PROJETO

Todo o processo de desenvolvimento foi feito no Laboratório de Projetos Avançados (LAPA) localizado no prédio H2 da Universidade de Passo Fundo no curso de Engenharia Elétrica.

Antes de iniciar o detalhamento do projeto é necessário fazer uma observação. Como o custo de um AGV é elevado, optou-se por utilizar para meios de estudo o LEGO Mindstorms NXT 2.0 disponível no almoxarifado do curso.

O NXT disponível possui as seguintes configurações (LEGO, 2018):

- Um microprocessador interno de 32-bit;
- Portas de entrada e 3 de saídas;
- Comunicação USB e Bluetooth;
- 3 servos motores;
- Sensores, 1 ultrassônico, 2 de toque e 1 sensor de cor.

O kit LEGO com os sensores pode ser observado na Figura 13.

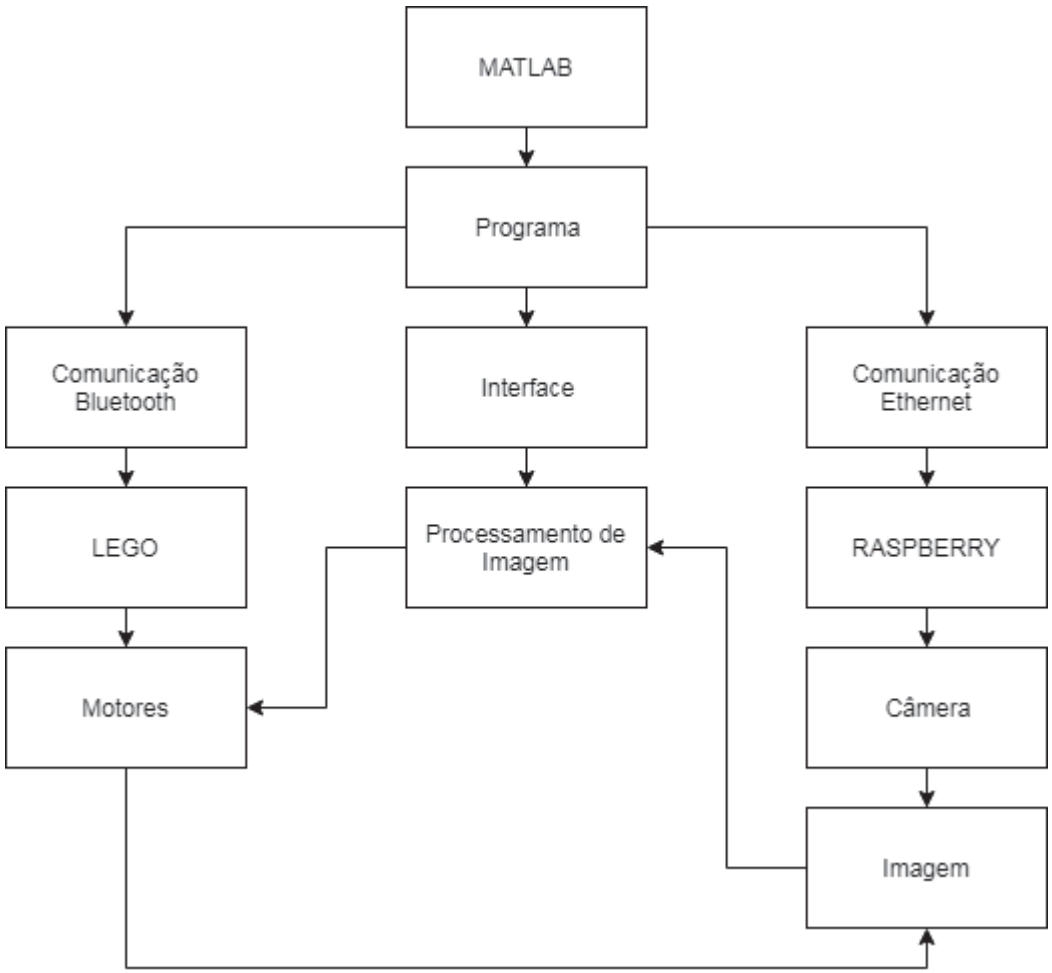
Figura 13 - Sensores do Kit LEGO MINDSTORMS NXT 2.0

Fonte: https://shop.lego.com/en-US/LEGO-MINDSTORMS-NXT-2-0-8547

O diagrama de blocos da Figura 14 mostra o funcionamento do processo desenvolvido. Inicialmente é feita a comunicação entre todos os dispositivos, posteriormente é tirada a foto pelo Raspberry sendo a mesma processada pelo MatLab e logo após os comandos de deslocamento são enviados até o LEGO.

É levado em conta que por se tratar de um projeto de envio de dados, recebimento e ação existe um atraso (*delay*), para não sobrepor informações podendo ocasionar erros de precisão.

Figura 14 - Diagrama de blocos do projeto



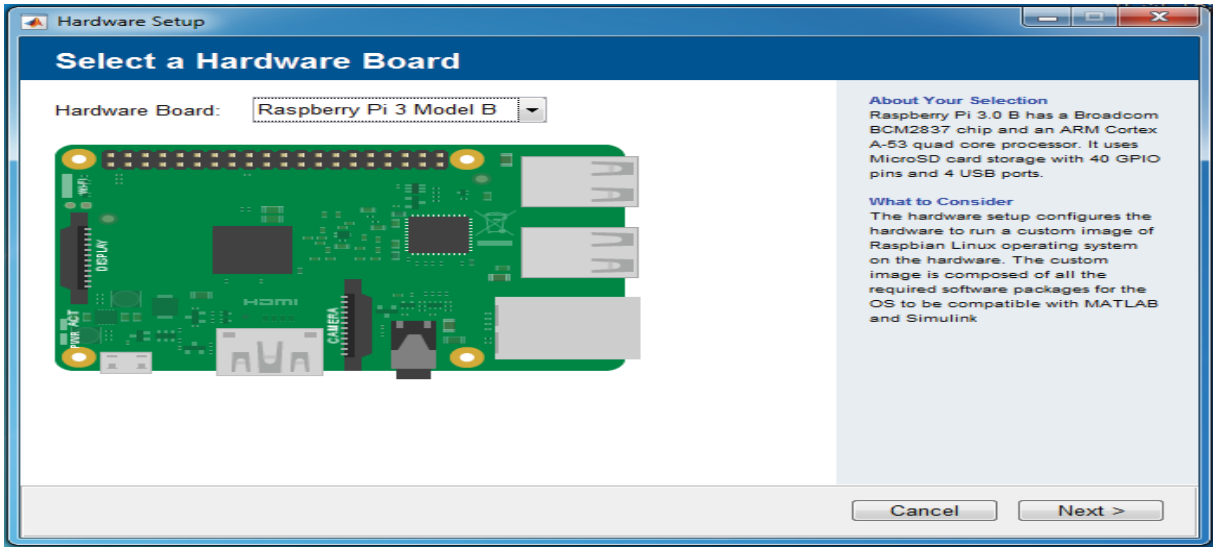
Fonte: Próprio Autor

Para tornar o programa viável é necessário antes ter o MatLab instalado. A versão R2017b ou uma mais atualizada seria o recomendado devido aos drivers de utilização, caso contrário é preciso instalar o TDM64-GCC-4.9.2. Este arquivo é utilizado para liberar o download dos drivers no MatLab, se não, não é permitido fazer o donwload.

Posterior ao passo citado é possível fazer o download do *hardware support packages* do Raspberry Pi dentro do programa MatLab, onde ele libera a compatibilidade e o acesso aos

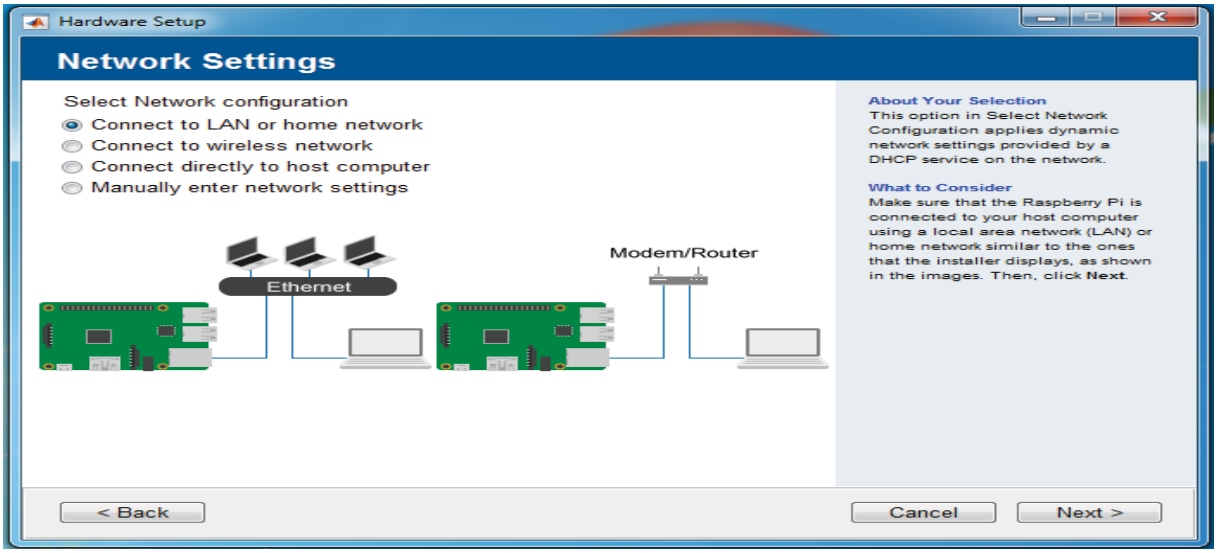
comandos para controlar o mesmo, para tal é necessário configurar o cartão de memória do microcomputador pela janela mostrada na Figura 15 do MatLab, na mesma é ajustado o IP para conexão que pode ser escolhido entre quatro opções, por redes distintas, por Wi-Fi e Ethernet, para estas duas opções é fundamental que ambos os dispositivos estejam conectados na mesma rede e a última é por ligação direta (Figura 16), a forma de acesso escolhida foi a que possui a menor taxa de interferência e melhor tempo de resposta, ou seja, via conector RJ45. Feito este processo, basta configurá-lo como mostra o APÊNDICE A, a partir daí todos os comandos são dados pelo Matlab.

Figura 15 - Configurando Raspberry Pi 3 Model B MatLab



Fonte: Próprio Autor

Figura 16 - Configurações de Rede Raspberry

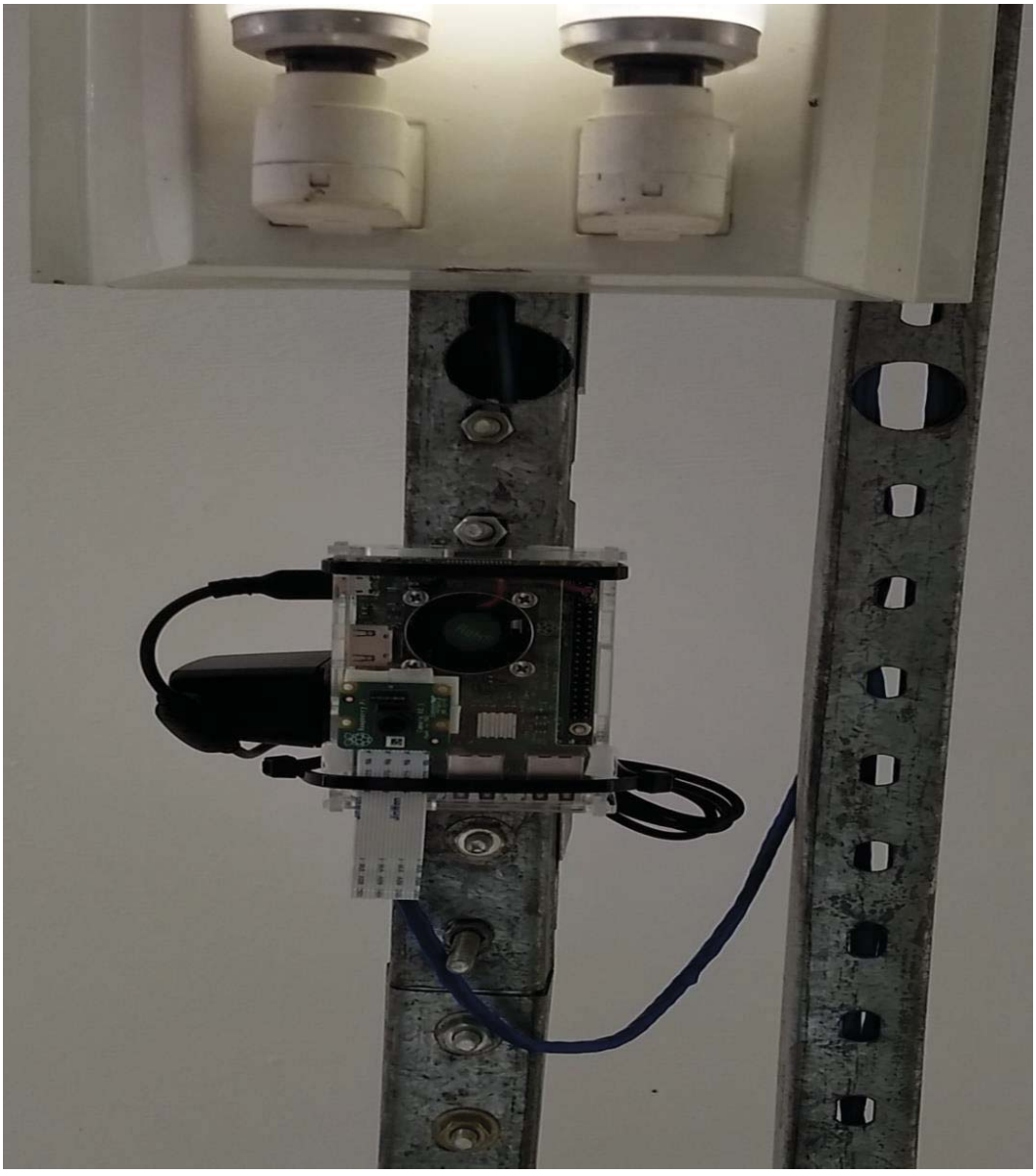


Fonte : Próprio Autor

O LEGO Mindstorm NXT 2.0 utilizado, por sua vez não possuí um *package* para baixar como o Raspberry, sendo necessário fazer o download dos drivers do dispositivo e os adicionar manualmente dentro do MatLab, assim liberando o comunicação e os comandos para o seu funcionamento. A versão 3.0 do LEGO possui compatibilidade, sendo necessário apenas fazer o donwload do seu *package* no MatLab.

A Câmera V2 de 8MP e o Raspberry Pi 3 Model B+ estão localizados a 2,7 metros do chão (Figura 17), fixado no perfilado que segura as lâmpadas do laboratório que possui dimensões 38x38 mm. Foi escolhido esse local pelo fato de o mesmo poder ser usado como suporte, além dos cabos de energia e Ethernet passarem dentro ele, facilitando assim a alimentação e a ligação do microcomputador com a rede.

Figura 17 - Raspberry com a câmera instalada no local

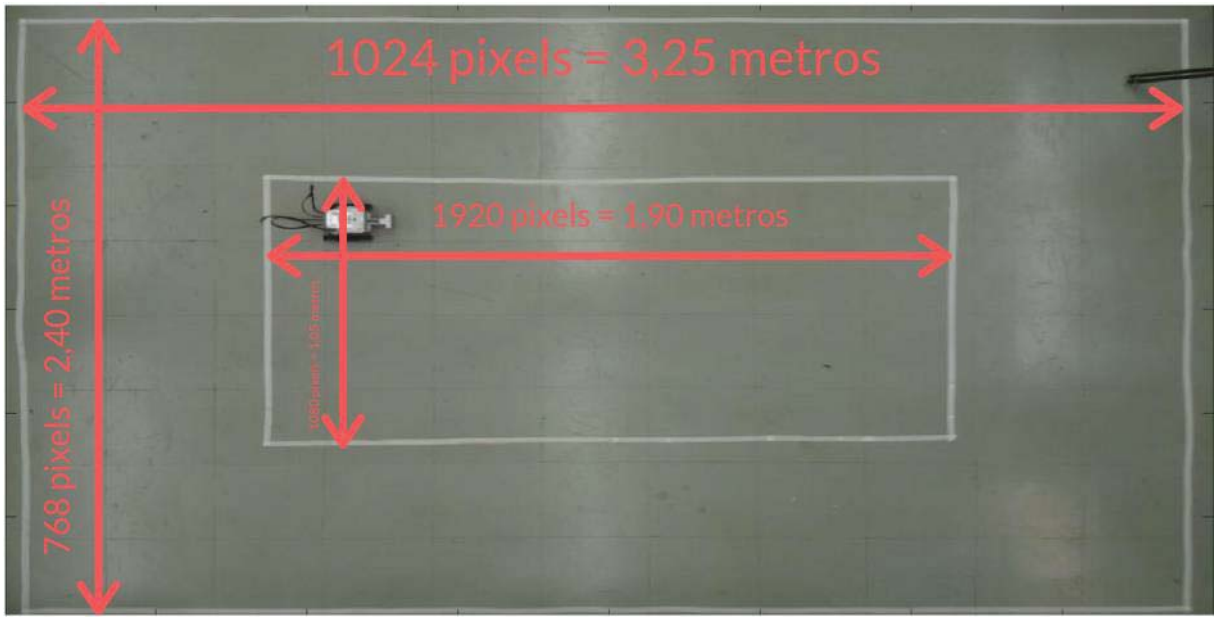


Fonte: Próprio Autor

Com todos os periféricos acionados e ativados, é feita a conexão entre eles. O microcomputador faz uma comunicação Ethernet com o computador, e o mesmo faz uma ligação Bluetooth com o LEGO como mostra o APÊNDICE A.

Após as conexões de troca e recebimento de dados feitos, o próximo passo é a configuração da resolução da câmera como apresenta o APÊNDICE B. No caso é uma definição de 1024 x 768 pixels, abrangendo uma área de 3,25 x 2,40 metros, se a mesma for alterada para valores menores do que o utilizado, ele manterá o mesmo tamanho físico, mudando apenas a qualidade da foto, caso fosse maior, por exemplo, 1920 x 1080 pixels, seria de 1,90 x 1,05 metros, diminuiria o tamanho dá área de atuação, mas melhoraria a resolução da imagem. Para obter uma área maior de teste foi escolhido a melhor resolução para o maior tamanho disponível de espaço físico, ou seja, 1024 x 768. É possível observar a diferença na Figura 18.

Figura 18 - Relação pixel metro



Fonte: Próprio Autor

Com a resolução definida, é alterado os valores de saturação, brilho e nitidez presentes na câmera além de ser aplicado um filtro anti-ruído. Todos estes ajustes são implementados diretamente na programação do Raspberry que tem como entrada de dados a câmera (APÊNDICE B). Também presente no apêndice, está a forma como os motores do LEGO foram configuradas.

Uma vez concluído esse processo de configurações e ajustes, é enviado o comando da retirada da foto, que por sua vez executa 4 fotos em sequência e guarda apenas a última, assim

eliminando resquícios de imagem. Com a mesma recebida pelo computador ela passa por um filtro de retirada de cores, onde separa a imagem 3D e a transforma em 3 diferentes figuras separando uma dimensão para cada canal, onde a camada 1 tem predominantemente a cor vermelha, a 2 tem como majoritária o tom verde e posteriormente a 3 tem o azul, nota-se a programação envolvida nesse processo no APÊNDICE C, e o resultado obtido no APÊNDICE D. Notou-se, que o primeiro filtro mudou a tonalidade da imagem para tons de cinza, e as cores desejadas ficaram com níveis de saturação de coloração branca, lembrando que quanto mais próximo do valor 0 mais escura ficará a imagem e quanto mais próximo de 255 mais clara ela ficará.

É possível observar na Figura 19, como são apresentados os valores da imagem no seu formato de matriz, observa-se que existem valores na faixa de 57, 74, 83, 90, 100, 110 e 120, eles por sua vez, são cores diferentes ou iguais com um nível de intesidade diferente gerando algumas leves alterações nos valores.

Figura 19 - Matriz da Imagem

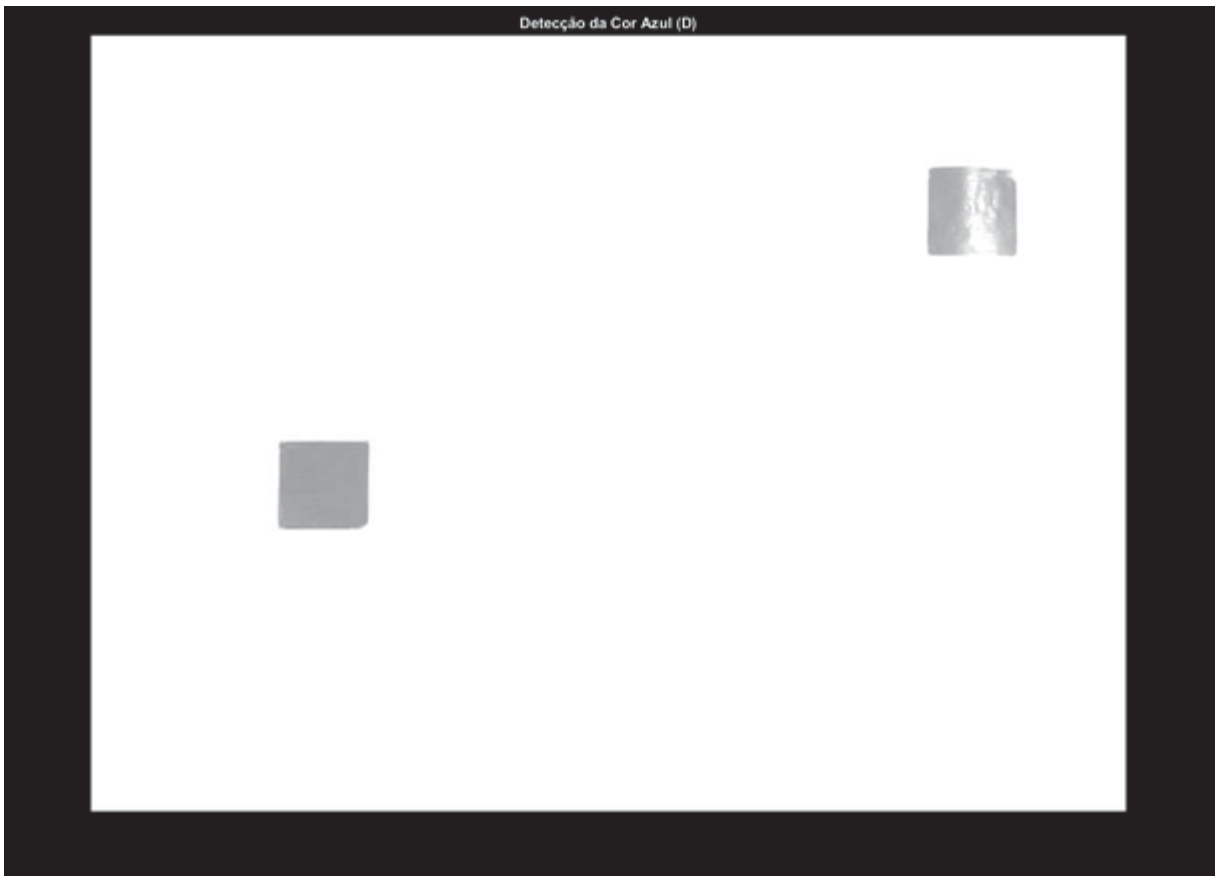
113	114	113	117	116	118	119	121	121	123	123	120	122
120	123	121	118	117	120	119	122	121	124	125	127	129
116	119	117	122	120	123	123	126	125	128	128	129	131
124	126	125	125	123	126	125	129	128	131	131	132	134
105	108	106	108	106	109	108	112	111	114	114	121	125
62	63	62	70	69	71	71	74	74	76	76	85	90
49	51	49	49	47	50	49	53	52	55	55	60	66
57	58	58	60	61	62	63	64	65	66	67	70	77
73	74	74	76	75	74	75	76	78	80	81	90	96
84	85	85	88	87	87	87	88	90	92	94	108	113
86	86	87	90	89	89	89	91	93	95	97	102	104
83	84	84	86	86	86	86	88	90	93	94	102	102
82	82	82	83	82	82	83	85	87	90	92	97	96
82	83	83	82	82	82	83	85	88	90	92	91	92
82	82	82	83	82	83	84	86	89	92	93	102	106
81	80	80	82	82	82	83	85	88	91	93	99	105
82	82	83	82	83	83	83	84	87	91	95	102	108
82	82	83	82	83	83	83	84	87	91	95	100	105
81	82	82	82	83	83	83	84	87	91	95	101	105
81	82	82	82	83	83	83	84	87	91	95	105	108
80	81	82	82	83	83	83	84	87	91	95	108	110

Fonte: Próprio Autor

Com os dados obtidos, e baseando-se do modelo aditivo de espaço de cores RGB de Thomas Young (1773-1829) foi desenvolvido os filtros de realce mostrados no APÊNDICE C, e ao mesmo tempo sua resposta pode ser observada no APÊNDICE E, observa-se que as áreas que não eram das cores, vermelha, verde e azul, ficaram com a cor preta, já as três citadas ficaram em tons de cinza, ganhando um destaque importante na imagem e no desenvolvimento

do projeto, a cor azul pode ser observada na Figura 20. Para a melhor visualização e impressão foi feita a inversão das cores das imagens.

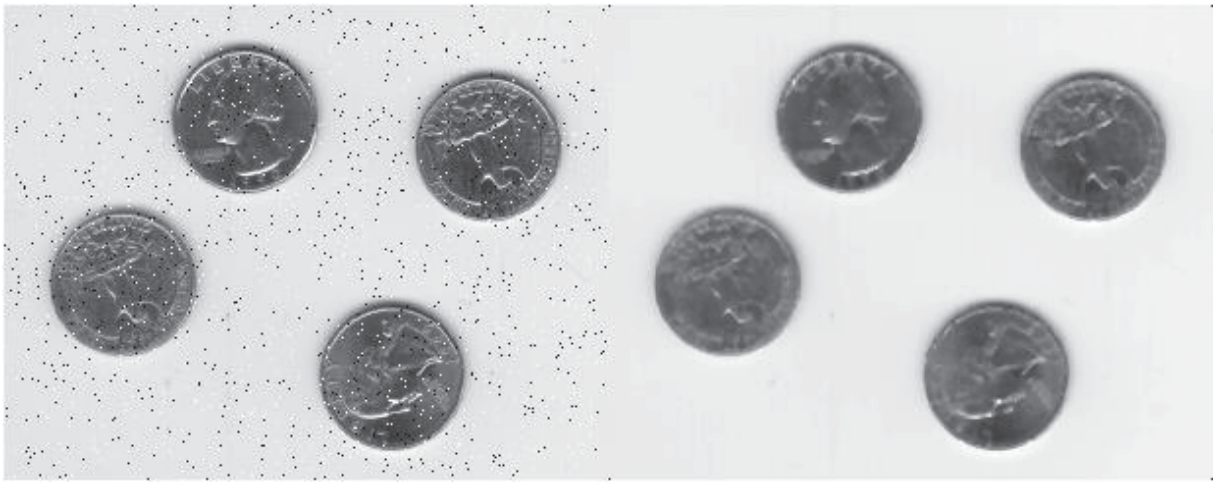
Figura 20 - Cor Azul com Filtro de Realce



Fonte: Próprio Autor

É utilizado o filtro *medfilt2* observado no APÊNDICE C que só pode ser executado em imagens 2D devido as suas características matemáticas que são observadas na seção secundária 2.5 Processamento de Imagem sobre autoria de Marques Filho e Vieira Neto (1999). Esse processo utiliza da 8-*neighbours*, uma matriz 3x3, fazendo a média dos valores dos pixels reduzindo assim pequenos ruídos e imperfeições contidas na imagem. É observado um exemplo de antes e depois na Figura 21, onde é observado que os pontos pretos pequenos são ruídos presentes na imagem, todavia neste exemplo a grande quantidade de pontos serve para demonstrar a eficiência do filtro. (MATLAB, 2018).

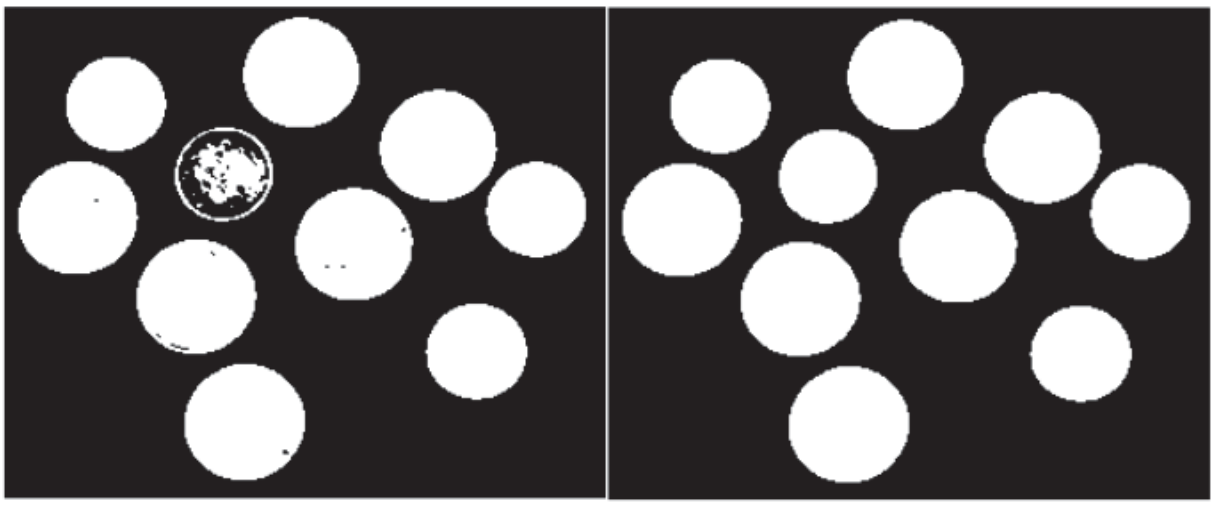
Figura 21 - Filtro medfilt2, antes e depois



Fonte: https://www.mathworks.com/help/images/ref/medfilt2.html

Junto ao filtro citado anteriormente é utilizado o *imfill* que preenche os buracos e regiões existentes na imagem. É considerada uma abertura, um conjunto de pixels de fundo que não podem ser alcançados através do preenchimento do plano de fundo da borda da imagem. Este por sua vez utiliza da mesma teoria aplicada no *medfilt2*, é observado um exemplo teórico na Figura 22 (MATLAB, 2018).

Figura 22 - Filtro imfill, antes e depois



Fonte: https://www.mathworks.com/help/images/ref/imfill.html#buo3g4l-1-conn

A resposta da utilização dos filtros desenvolvidos até este ponto pode ser observada no APÊNDICE F, no mesmo, nota-se que o destaque da cor cinza não é muito significativo para a visão humana, não chamando a devida atenção e podendo camuflar erros existentes muito próximos da faixa de cor preta, por esse motivo foi feita a saturação das cores diferentes de

preto, de uma forma simples, mas eficaz. Foi realizada a multiplicação de seus valores até chegar a 255 que é a cor branca (saturação), como a tonalidade preta tem valor de 0, o mesmo não é afetado pelo processo, assim ressaltando apenas a área de interesse, que pode ser visualizada no APÊNDICE G. Uma amostra de ambos pode ser visualizado na Figura 23 e 24, respectivamente a cor verde sem a saturação e com saturação. Para a melhor visualização e impressão foi feita a inversão das cores das imagens.

Figura 23 - Cor verde não saturada

Fonte: Próprio Autor

Figura 24 - Cor verde saturada

Fonte: Próprio Autor

Observou-se antes que em uma imagem quase que totalmente preta não se destacou a área cinza, mas os valores da matriz por ela gerada seriam diferentes de 0 causando erro na hora da leitura e execução do programa. Com este ajuste de tons, é por nós humanos observado com mais ênfase a imagem.

Devido a necessidade de mais detalhes e diferenciações, foi desenvolvido novos filtros de cores. A tonalidade amarela e laranja, se juntaram as já existentes, vermelha, verde e azul. Aumentando assim o nível de dados obtidos com apenas uma imagem.

Pode-se observar o código de detecção das novas cores no APÊNDICE H. Junto do mesmo é observado que existem compensações que precisam ser feitas nas outras pigmentações para existir a devida separação, como mostrado no verde e no vermelho. De mesma forma, é necessário eliminar valores ruidosos existentes, para isso é feita uma subtração de matrizes fazendo com que valores inferiores não desejados se tornem zero. Com os ajustes feitos, é observado na Figura 25 a resposta parcial, e no APÊNDICE I a total.

Figura 25 - Cores amarela e laranja

Fonte: Próprio Autor

Com os últimos ajustes das detecções das cores na imagem é feita a extração das informações nela contida. Com o comando *logical* do MatLab é iniciado a retiradas dos dados, o mesmo por sua vez transforma cada valor diferente de zero da matriz imagem em um valor verdadeiro (*true*) ou 1, já se existirem valores iguais a zero eles recebem falso (*false*) ou no caso 0 (Figura 26).

Figura 26 - Matriz True e False

FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
FALSE	FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE
FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE

Fonte: Próprio Autor

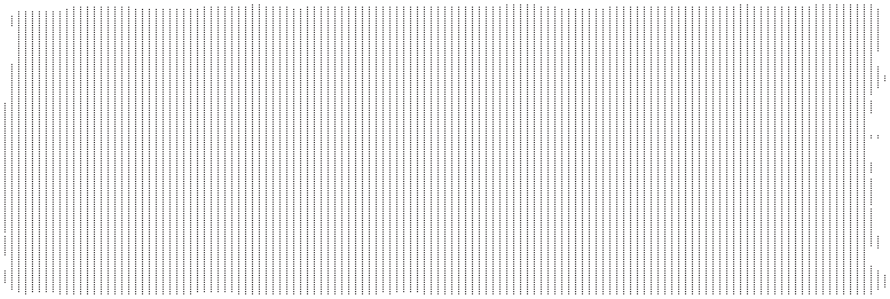
Devido ao formato do arquivo que fica depois de utilizar o comando *logical* é multiplicado o mesmo por uma matriz com valor unitário, obtendo a resposta mostrada na Figura 27, com isso é possível utilizar de outra instrução nomeada *find*, a mesma acha todos os valores iguais a 1, nas suas respectivas coordenadas de *x* e *y*, gerando a posição, o tamanho e a área das cores em pixels. E na Figura 28 é observado a imagem gerada a partir da matriz númerica, nela é possível observar as dimensões do quadrado azul vistos pelo Excel.

Figura 27 - Matriz de 0 e 1

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	1	1	1	1
0	0	0	1	1	1	1	1	1	1
0	0	1	1	1	1	1	1	1	1
0	0	1	1	1	1	1	1	1	1
0	0	1	1	1	1	1	1	1	1
0	0	1	1	1	1	1	1	1	1
0	0	1	1	1	1	1	1	1	1
0	0	0	1	1	1	1	1	1	1
0	0	0	1	1	1	1	1	1	1
0	0	0	1	1	1	1	1	1	1
0	0	0	1	1	1	1	1	1	1

Fonte: Próprio Autor

Figura 28 - Imagem Númerica



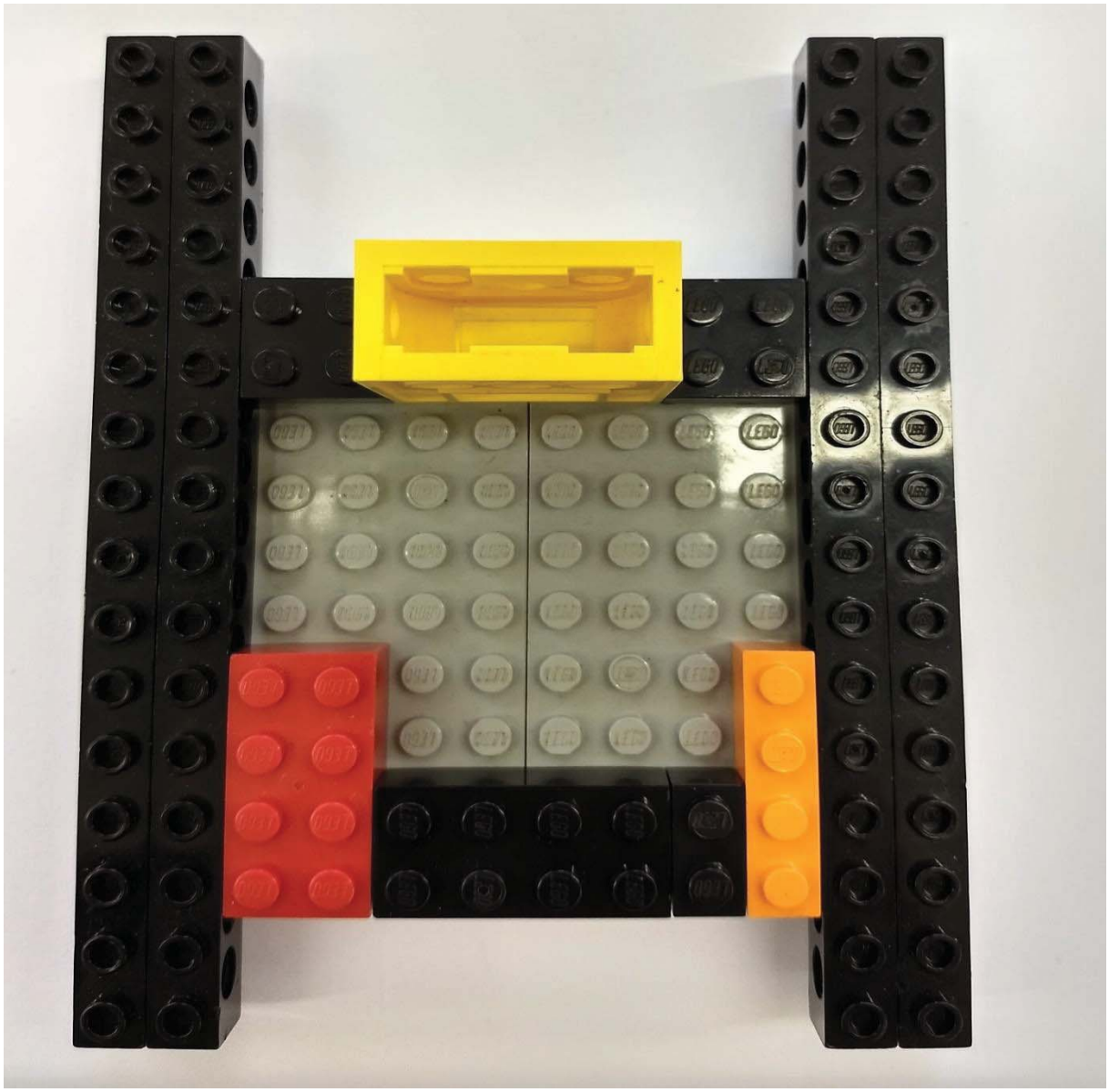
Fonte: Próprio Autor

Os dados tirados da Figura 28 mostram que, as áreas com valores de 0 são da cor branca, e os que possuem o número 1 recebem a coloração preta, é interessante vilualizar a imagem desta forma apresentada, pois é possível identificar onde existem ruídos e os pontos exatos da matriz, visto que a mesma é obtida usando o programa Excel, que fornece os dados exatos de colunas e linhas.

Com a função *find* é possível dar sequência aos outros procedimentos, pois o comando possibilita achar o ponto máximo, mínimo e o ponto médio (APÊNDICE J). Com os pontos encontrados é possível encontrar a distância total em pixels, tanto em *x* quanto em *y*, a área que ocupa toda uma cor, e com esses dados é possível saber a posição exata do protótipo e qual o seu destino.

O primeiro passo a se definir em um protótipo veicular é onde será a frente, a traseira, a esquerda e a direita, visto que temos que ter sentido e direção para fazê-lo se locomover, pois no espaço o mesmo se comporta como um vetor. Os pontos foram adotados da seguinte maneira, o amarelo é a frente, o vermelho é a esquerda e o laranja é a direita, como mostra a Figura 29, sabe-se que a traseira é o ponto central da distância entre o laranja e o vermelho.

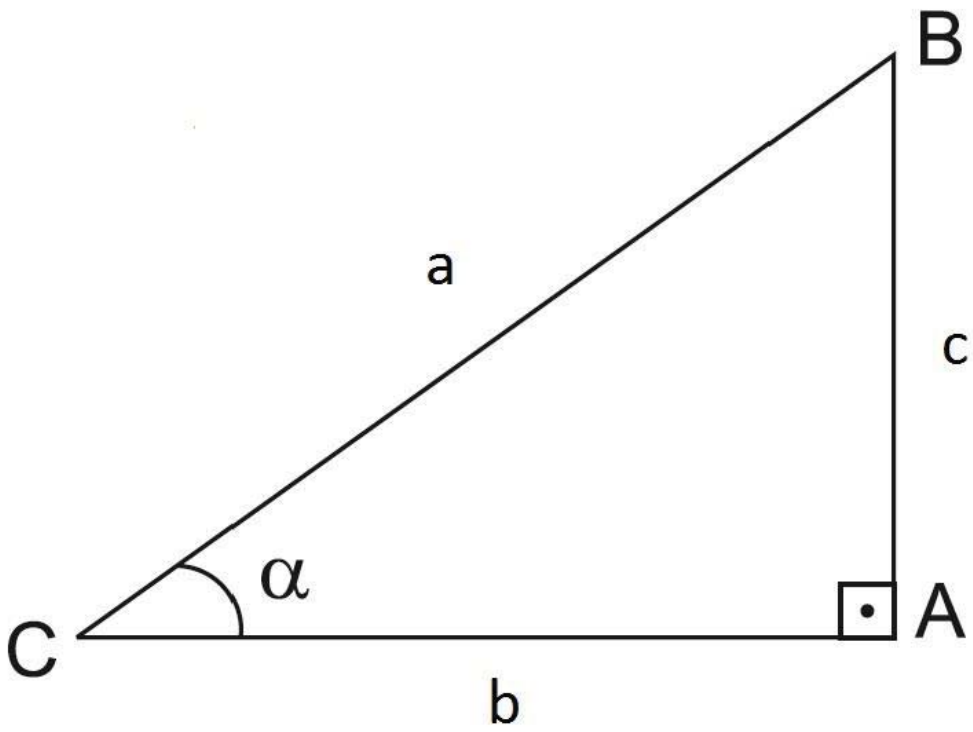
Figura 29 - Cores do LEGO



Fonte: Próprio Autor

Definidos os quatro pontos guias é necessário saber como o mesmo fará para girar no próprio eixo ajustando assim sua direção e sentido, estimando a orientação do objeto à ser detectado. A resposta foi a encontrada utilizando do triângulo retângulo (Figura 30) e suas equações matemáticas (2) e (3).

Figura 30 - Triângulo Retângulo



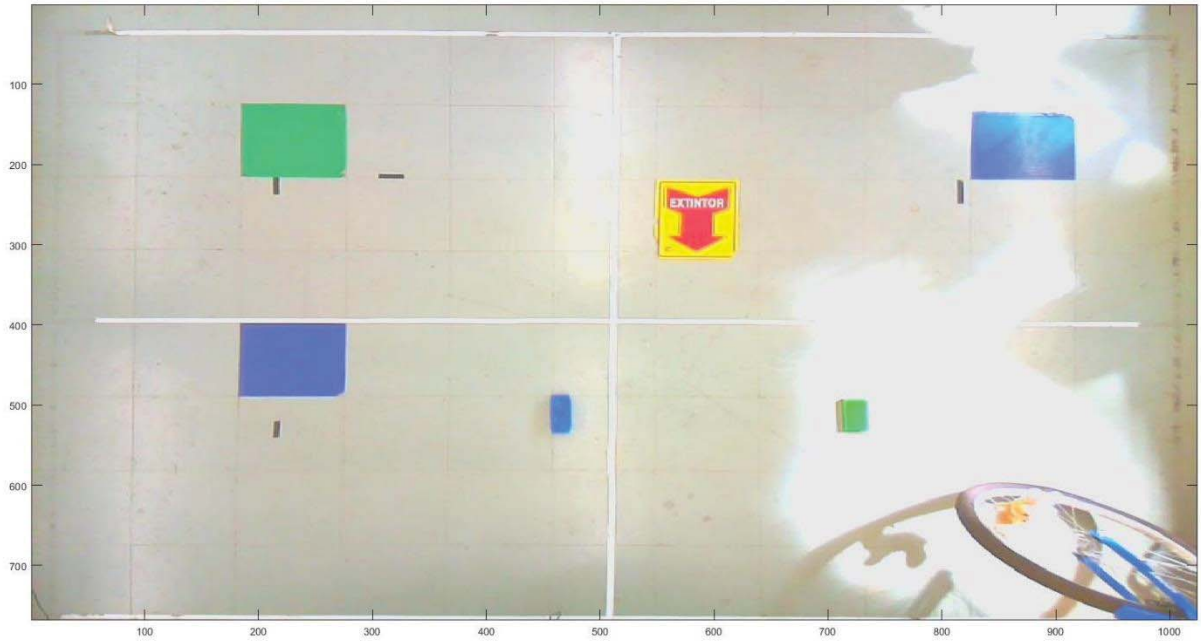
Fonte: Próprio Autor

$$a = \sqrt{b^2 + c^2} \tag{2}$$

$$\propto = atan\left({}^{c}/_{b}\right) \tag{3}$$

Como a matriz da imagem é dada de cima para baixo (eixo X) e da esquerda para a direita (eixo Y) como mostra a Figura 31, a cor verde está mais próxima de zero, ela seria o ponto B do triângulo retângulo, e as colorações laranja e vermelha seriam o ponto C, visto que é calculado o $\propto$ das duas tonalidades (APÊNDICE K) e analisado qual dos dois é maior para tomar a decisão de rotação correta. O MatLab entrega os valores de ângulos na forma de radianos, por isso esses números encontrados são divididos por 0,0174533, devolvendo um resultado em graus, ficando assim, mais nítido aos olhos humanos.

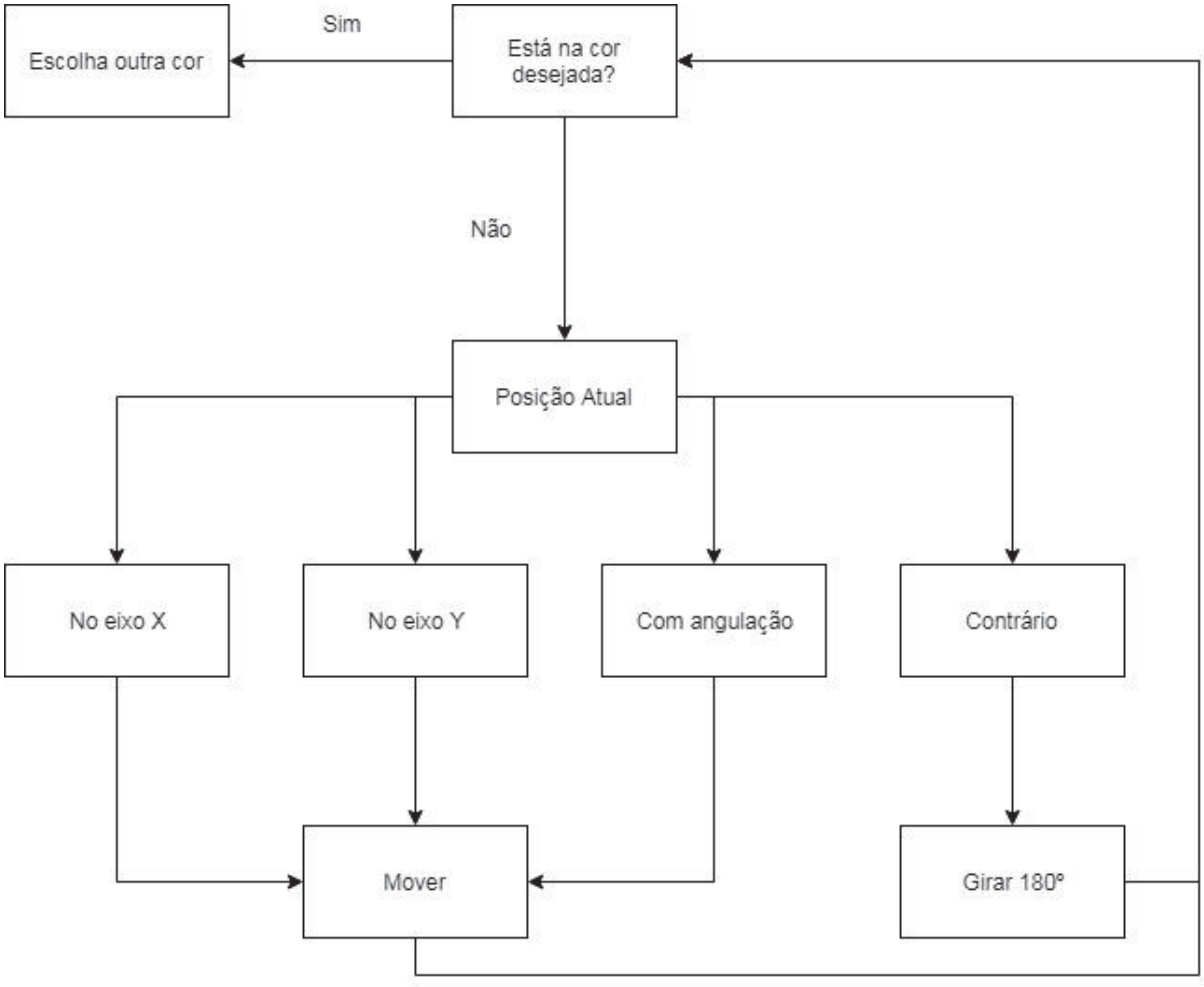
Figura 31 - Matriz da Imagem



Fonte: Próprio Autor

É possível observar no diagrama de decisões (Figura 32) o funcionamento das escolhas para ocorrer o deslocamento. Primeiro é observado se o módulo robótico já não está em cima da cor desejada, caso não esteja ele executa a localização da posição atual e escolhe um dos quatro caminhos possíveis. Caso ele esteja reto ao eixo X ou Y do destino, apenas percorrerá a distância acionando os dois motores, por outro lado se estiver em uma locação diagonal, executará apenas um dos motores para acertar os eixos, a quarta opção é detecção se o mesmo está virado de forma contrária ao ponto de chegada, sendo assim, é girado 180º. Observa-se que após cada movimento é retornado a mesma pergunta, caso a mesma permaneça com a resposta negativa é executado o programa novamente até chegar no ponto desejado. O código de programação para executar tais tarefas está localizado no APÊNDICE L.

Figura 32 – Fluxograma do Processamento de Imagem Realizado



Fonte: Próprio Autor

No exemplo feito (Quadro 2) é observado os valores obtidos com a execução do programa, com o mesmo é adquirido, as coordenadas referêntes de cada objeto e seus ângulos de inclinação em relação a cor verde, dependendo dos valores resultantes é executado uma das quatro ações já comentadas anteriormente.

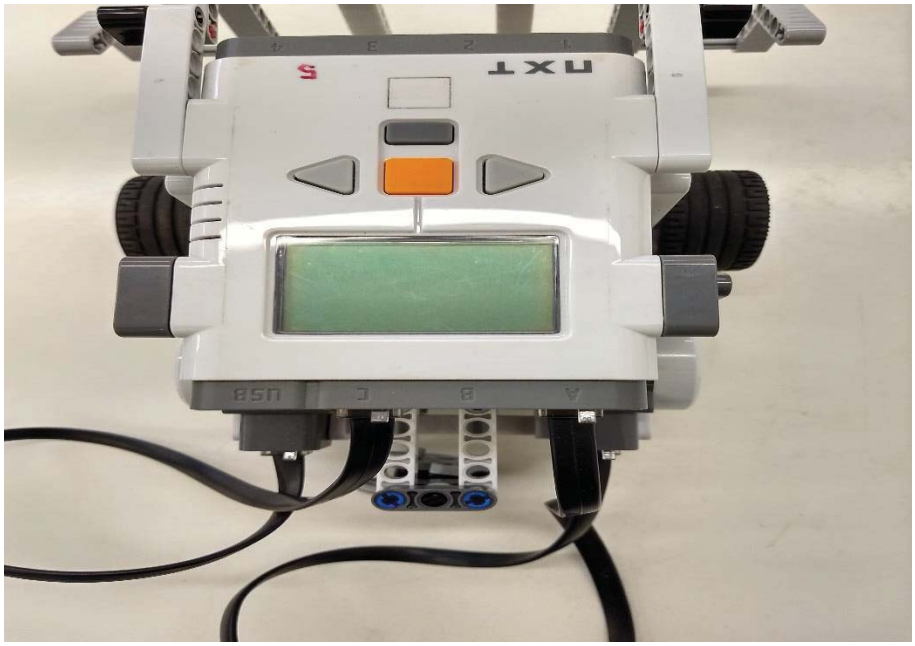
Neste caso em expecífico, como o ângulo LVG é 18, portanto é maior que o YG que é 17, é ativando o motor A (Figura 33), fazendo com que o protótipo gire para a esquerda.

Quadro 2 – Teste Cor Verde

RED		DIFERENÇAS	PONTO MÉDIO	COORDENADAS	DISTÂNCIA	
yRMIN	799	13	7	806	VERMELHO - VERDE	
yRMAX	812				Y	574
xRMIN	345	18	9	354	X	182
xRMAX	363				LARANJA - VERDE	
GREEN					Y	575
yGMIN	189	85	43	232	X	178
yGMAX	274					
xGMIN	128	88	44	172	CATETOS	
xGMAX	216				OPOSTO	
BLUE					XOG	178
yBMIN	186	838	419	605	XRG	182
yBMAX	1024				XYG	170
xBMIN	137	631	316	453	XLVG	186
xBMAX	768				ADJACENTE	
YELLOW					YOG	575
yYMIN	790	6	3	793	YRG	574
yYMAX	796				YYG	561
xYMIN	338	8	4	342	YLVG	575
xYMAX	346					
ORANGE					ÂNGULOS	
yOMIN	799	16	8	807	LVG	18
yOMAX	815				YG	17
xOMIN	343	14	7	350	OG	17
xOMAX	357				RG	18
TRASEIRA					ÁREA VERDE	
y		1		807	CALCULADA	7480
x		4		358	REAL	7019

Fonte: Próprio Autor

Figura 33 - Posição dos Motores do LEGO



Fonte: Próprio Autor

Como os servos motores do LEGO não aceitam números negativos e decimais, apenas números inteiros foi feito testes para descobrir qual era o valor necessário para girar determinados ângulos, chegando aos seguintes resultados (Quadro 3).

Quadro 3 – Relação Valor Ângulo

VALOR NÚMERICO	ÂNGULOº
220	45º
480	90º
700	135º
950	180º
1150	225º
1390	270º
1600	315º
1870	360º

Fonte: Próprio Autor

Com o auxílio dos valores apresentados é possível determinar o quanto é necessário entregar de números inteiros ao servo para ele girar os respectivos graus. Com base no Quadro 2 e 3 é feito os seguintes cálculos:

$$RotacaoAnguloLVG = {(AnguloLVG * 220)}/{45} \quad (4)$$

$$RotacaoAnguloLVG = {(18 * 220)}/{45} \quad (5)$$

$$RotacaoAnguloLVG = 88 \quad (6)$$

Portanto o valor a ser recebido no servo motor A será de 88 fazendo com que o LEGO gire 18º.

Para a opção de alinhamento horizontal e vertical é entregue o número da distância entre os pontos de saída e de chegada multiplicados por 3 visto que caso contrário, o mesmo iria andar muito pouco além de quando estiver quase chegando ao destino não conseguir se mover adequadamente, pois os valores ficariam na casa de 10 à 20, fazendo com o que servo gire muito pouco, demorando muito para concluir a tarefa.

Após a detecção de que o protótipo está com a posição invertida é colocado o valor de 950 fazendo com que o mesmo gire 180º arrumando assim seu sentido.

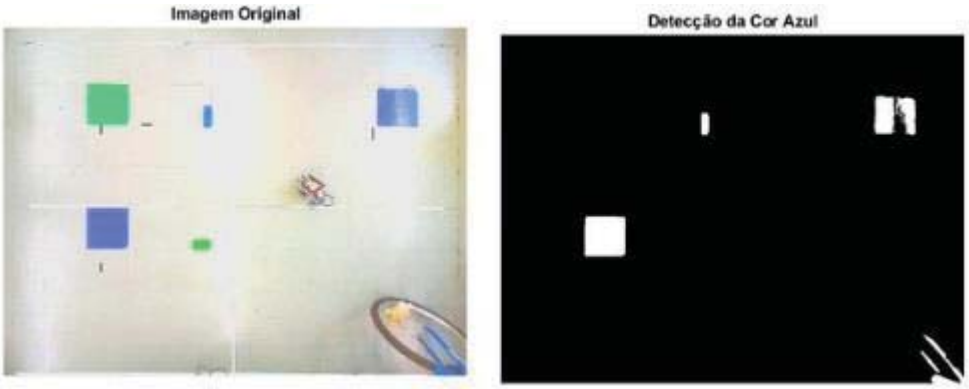
4 RESULTADOS E DISCUSSÕES

A realização dos testes foi feita totalmente no Laboratório de Projetos Avançados (LAPA), onde foram observados erros e acertos que levaram a conclusão do projeto.

Uma das dificuldades mais marcantes encontradas na realização do projeto foi fazer os filtros das cores funcionarem adequadamente, a primeira tentativa foi transformar a matriz gerada em vetores e a partir deles fazer a manipulação dos valores contidos nela, o que não resultou em uma resposta satisfatória. O problema foi resolvido de outra maneira, através da manipulação da imagem em si, está por fim foi a resposta encontrada para superar o obstáculo, como mostra o trabalho realizado.

Outra adversidade encontrada no decorrer da atividade foi a iluminação em abundância ocasionada por lâmpadas e por dias muito claros, gerando saturações de cores causando erros nas aplicações dos filtros, como mostra o exemplo da Figura 34, onde a cor azul foi prejudicada devido a uma lâmpada fluorescente ficar posicionada bem acima dela. Todavia como este problema apresentado é de extrema dificuldade de resolução, foi deixado de lado, visto que o software funcionava adequadamente na parte da tarde e da noite, apresentando problemas apenas no turno da manhã.

Figura 34 - Erro causado pela saturação

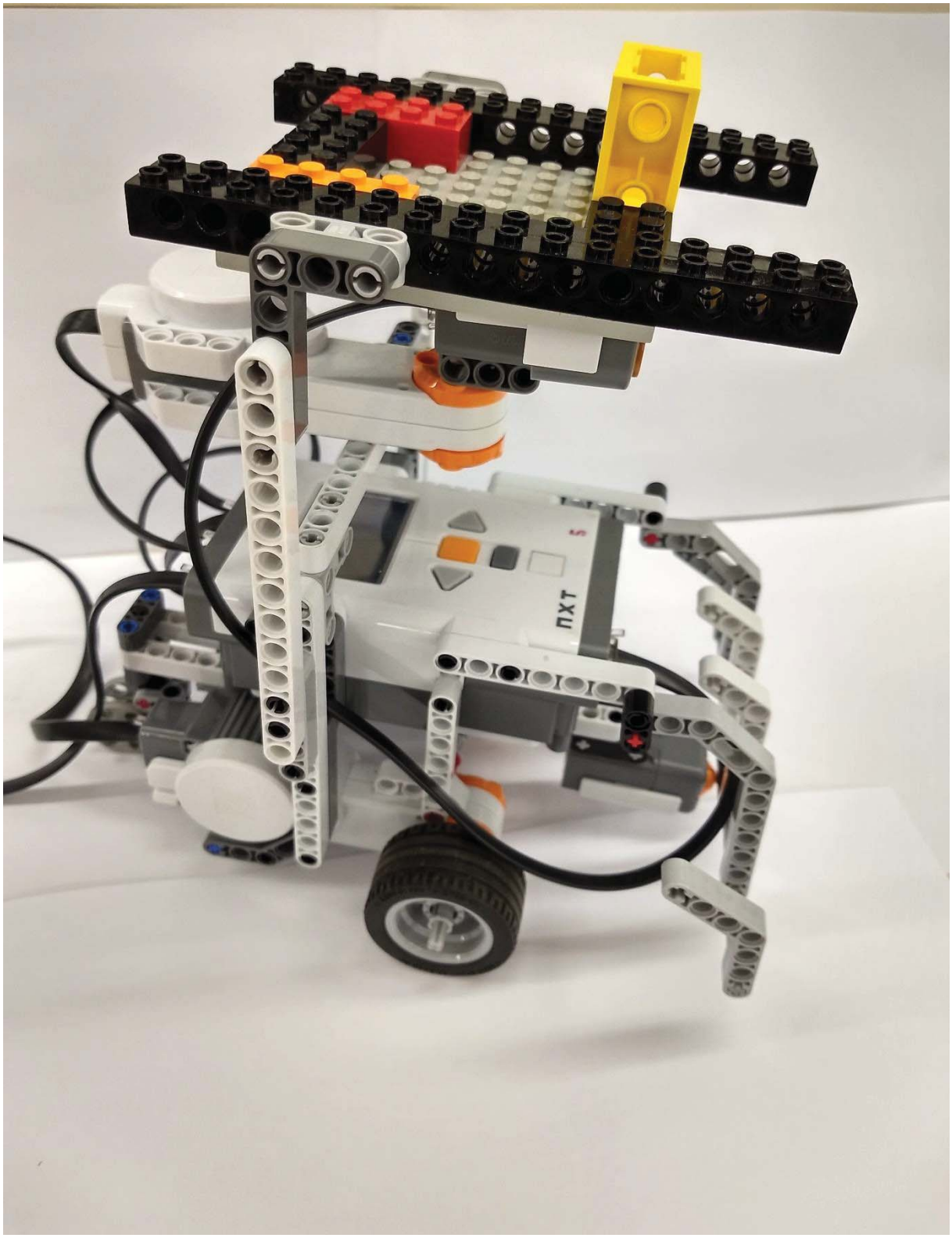


Fonte: Próprio Autor

Outro inconveniente encontrado foi, por se tratar de um sistema mecânico LEGO (Figura 35), pela comunicação ser por Bluetooth, além do tempo para tirar a foto, um atraso (*delay*) para ocorrer a resolução de suas atividades, visto que o mesmo precisa terminar a primeira ação para receber a segunda, caso contrário ele guarda na memória os outros comandos, gerando

acumulo de dados ocasionando erros, e a não precisão necessária para executar as tarefas requeridas.

Figura 35 - Protótipo LEGO

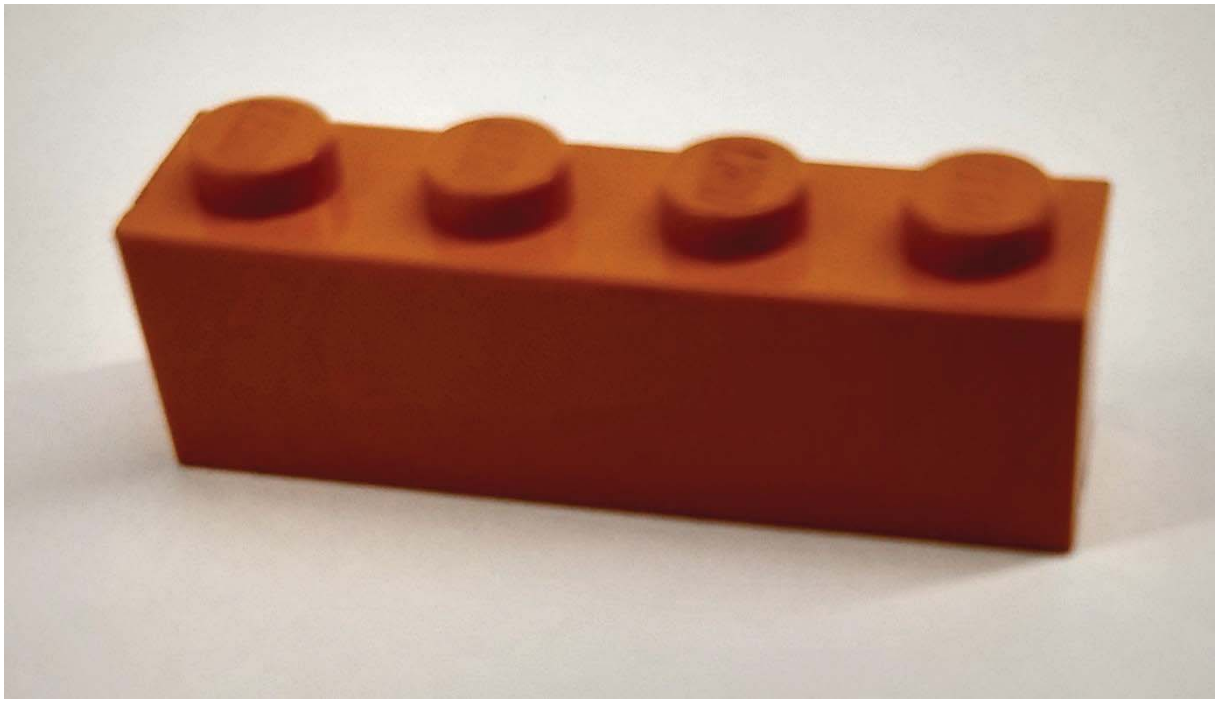


Fonte: Próprio Autor

É possível ver no APÊNDICE M a detecção de todas as cores que foram desenvolvidas, e é observado que os objetos que não possuem tais colorações não são contados, idealmente seria necessário acoplar algum sensor de presença no LEGO para encontrar o que as imagens ignoram.

A câmera escolhida e os filtros utilizados mostram-se muito eficazes, pois são capazes de encontrar até mesmo uma peça de LEGO muito pequena mostrada na Figura 36, a mesma possui 31,91mm de comprimento e 7,90 mm de largura.

Figura 36 - Peça de LEGO laranja

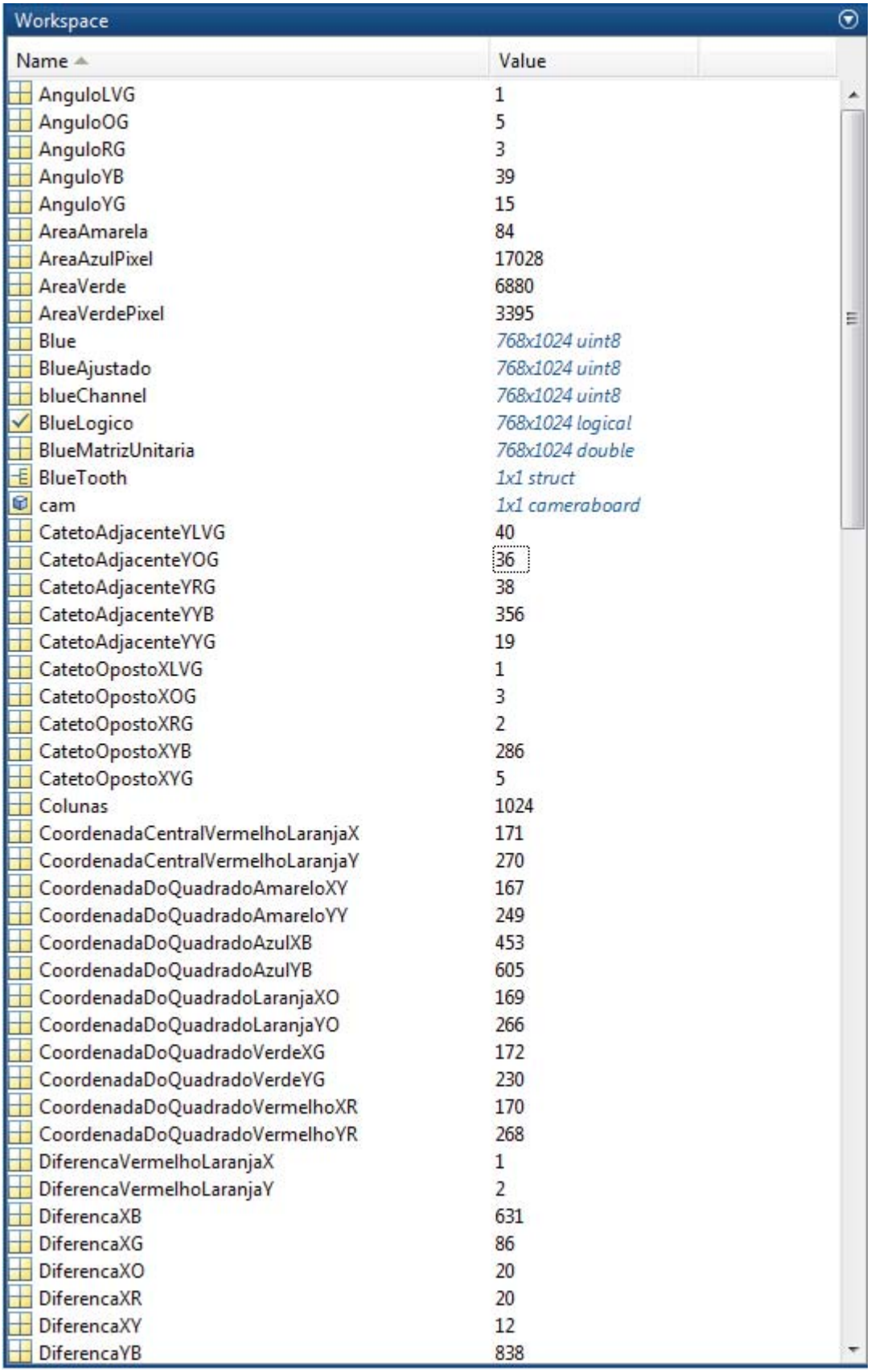


Fonte: Próprio Autor

É possível observar nos APÊNDICES N, O, P, Q, R, S e T o deslocamento do LEGO até chegar na área verde.

E na Figura 37 é observado uma parte dos valores matemáticos obtidos no processo, nele existem os valores de ângulos entre as cores, a área das cores de forma matemática, e a mesma de forma real pela contagem de todos os pixels da imagem, as distâncias entre os pontos, seus valores máximos e mínimos, as coordenadas x e y dos objetos entre outras informações necessárias ao seu funcionamento.

Figura 37 - Workspace MatLab



Fonte: Próprio Autor

Um dado importante é que o tamanho dos quadrados (pontos de destinos) presentes nas fotos (Figura 38) é de 30 cm x 30 cm, e gera uma distância de pixels de 90 x 90, foi feito desta forma por ser uma área maior do que a do protótipo e possível ser detectada se o mesmo estiver sobre ela, além de que a programação entende que quando a área escolhida está com menos de 50 % do valor original, o LEGO está sobre ela.

Figura 38 - Quadrado azul com suas dimensões

Fonte: Próprio Autor

5 CONSIDERAÇÕES FINAIS

A proposta de um programa para identificação de cores e envio de informações para um protótipo robótico móvel foi atingida. A maior dificuldade enfrentada foi no desenvolvimento dos filtros de cor, visto que para a sua correta utilização não era permitido nenhum tipo de ruído, além da luminosidade do ambiente que pode variar no decorrer do dia mudando a tonalidade da imagem capturada, contudo o mesmo se mostrou muito útil e eficiente detectando 5 cores diferentes, estas por sua vez podem ser de qualquer tipo de objeto contanto que a cor dele seja válida.

Para solucionar os problemas envolvidos com os filtros foi demandado muito tempo, pois além de não encontrar material suficiente sobre o assunto, existem muitas diferenças do mundo ideal para o real, como ruídos, mal funcionamento e alteração do ambiente, o que acarretou na falta de tempo hábil para implementação dos sensores no protótipo robótico, o mesmo por sua vez possui o de toque e o ultrassônico que auxiliariam no desempenho de evitar obstáculos, os detectores já foram estudados e é possível sua implementação junto ao MatLab.

Apesar dos problemas citados, o programa se comporta de forma satisfatória ao realizar sua função, sendo possível ainda ser utilizado do código escrito para outras aplicações, como na identificação de frutas maduras de não maduras, detecção de automóveis pelas cores, além de monitorar o movimento de automóveis em estacionamentos.

REFERÊNCIAS

AGVS, Automated Guided Vehicle Solutions. **Veículos autoguiados**: Virtual Books, 2013. Disponível em: < http://www.agvs.com.br/index.html>. Acesso em: 23 outubro 2017, 21:33:05.

BOUTEILLE, Daniel et al. ***Les Automatismes Programables***. 2. ed. Toulouse: Cépaduès-éditions, 1997.

FORESTI, Renan Luís. **Sistema de visão robótica para reconhecimento de contornos de componentes na aplicação de processos industriais**: Virtual Books, 2006. Disponível em: < http://www.lume.ufrgs.br/bitstream/handle/10183/11169/000607423.pdf?sequence=1>. Acesso em: 29 outubro 2017, 20:21:25.

FORSTER, Guz. **O novo Raspberry Pi 3:** mesmo preço, muitas melhorias: Virtual Books, 2016. Disponível em: <http://www.experimentoria.com.br/o-novo-raspberry-pi-3-mesmo-preco-muitas-melhorias/>. Acesso em: 25 outubro 2017, 19:47:55.

GONZALEZ, Rafael C.; WOODS, Richard E.. **Processamento digital de imagens**. 3. ed. São Paulo: Pearson Education, Inc., 2011.

LEGO. **LEGO® MINDSTORMS® NXT 2.0:** Virtual Books, 2018. Disponível em: <https://shop.lego.com/en-US/LEGO-MINDSTORMS-NXT-2-0-8547>. Acesso em: 28 novembro 2018, 15:37:30.

MATHWORKS. **MATLAB Product Description**: Virtual Books, 2017. Disponível em: <http://www.mathworks.com/help/matlab/learn_matlab/product-description.html>. Acesso em: 25 outubro 2017, 20:12:10.

MATT. **8-megapixel Raspberry Pi Camera Module v2**: Virtual Books, 2016. Disponível em: < https://www.raspberrypi-spy.co.uk/2016/04/8-megapixel-raspberry-pi-camera-module-v2/>. Acesso em: 06 novembro 2017, 19:45:10.

MONK, Simon. **Programando o Raspberry Pi:** Primeiros Passos com Python: Virtual Books, 2013. Disponível em: <https://s3.novatec.com.br/capitulos/capitulo-9788575223574.pdf>. Acesso em: 25 outubro 2017, 19:42:45.

MORAIS, Vagner; VIEIRA, Cláudio. **MATLAB 7&6**. 3. ed. Lisboa: Abril, 2006.

NOGUEIRA, Alan Douglas; TEIXEIRA, Vinicius; FREITAS Wilhan Rangel de. **Aplicação de robótica em processos de transporte industrial**: Virtual Books, 2015. Disponível em: <http://revista.unilins.edu.br/index.php/cognitio/article/viewFile/221/215>. Acesso em: 23 outubro 2017, 20:54:25.

RIVIN, Eugene I.. ***Mechanical Design of Robots***. 1. ed. New York: McGraw-Hill Inc, 1988.

ROMANO, Vitor Ferreira. **Robótica Industrial**: aplicação na indústria de manufatura e de processos: Virtual Books, 2002. Disponível em: <https://pt.scribd.com/doc/211992278/Livro-Robotica-Industrial-pdf>. Acesso em: 23 outubro 2017, 20:20:45.

SELEME, Robson; SELEME, Roberto Bohlen. **Automação da produção**. 1. ed. Curitiba: InterSaberes, 2013.

SILVA, Yuri G.; QUEIROZ, Max Hering de; CURY, José E. R.. **Síntese e implementação de controle supervisório modular local para um sistema de agv**: Virtual Books, 2010. Disponível em: <https://www.researchgate.net/publication/265879163_SINTESE_E_IMPLEMENTACAO_DE_CONTROLE_SUPERVISORIO_MODULAR_LOCAL_PARA_UM_SISTEMA_DE_AGV>. Acesso em: 23 outubro 2017, 21:08:25.

SILVEIRA, Cristiano Bertulucci. **O Que é Indústria 4.0 e Como Ela Vai Impactar o Mundo**: Virtual Books, 2017. Disponível em: < https://www.citisystems.com.br/industria-4-0/>. Acesso em: 25 outubro 2017, 20:03:40.

STEP, Shanghai STEP Electric Corporation. **Robôs Industriais**: Virtual Books, 1995. Disponível em < http://step-automation.com.br/2-1-industrial-robots/226185>. Acesso em: 06 novembro 2017, 19:42:25.

THOMSEN, Adilson. **Primeiros passos com o Raspberry Pi**: Virtual Books, 2014. Disponível em: < https://www.filipeflop.com/blog/tutorial-raspberry-pi-linux/>. Acesso em: 25 outubro 2017, 19:53:15.

VENTURELLI, Márcio. **Indústria 4.0**: Virtual Books, 2017. Disponível em: < https://www.automacaoindustrial.info/industria-4-0/>. Acesso em: 25 outubro 2017, 19:58:50.

APÊNDICE A – CONFIGURAÇÕES DE CONEXÃO RASPBERRY E LEGO

```
%% Configurações dos Dispositivos
%
   %% Raspberry PI 3

   % Limpa o Command Window.
   clc
   % Limpa toda a Workspace.
   clear all
   % Fecha todas as janelas abertas.
   close all

   % Ativa RPI como a variável de conexão com o Raspberry.
   rpi = raspi();

   disp(' Bem Vindo ao Programa de Reconhecimento de Cores RGB para Controle Robótico ');

   %% LEGO NXT 2.0

   disp(' Estabelecendo conexão ao LEGO NXT 2.0 ');

   % Limpa as variáveis do NXT.
   COM_CloseNXT all
   % Arquivo que permite a conexão via BlueTooth.
   BlueTooth = COM_OpenNXT('bluetooth-example-windows64.ini');
   COM_SetDefaultNXT(BlueTooth);

   disp(' Conexão estabelecida ao LEGO NXT 2.0 ');
```

APÊNDICE B – CONFIGURAÇÕES DA CÂMERA E MOTORES DO LEGO

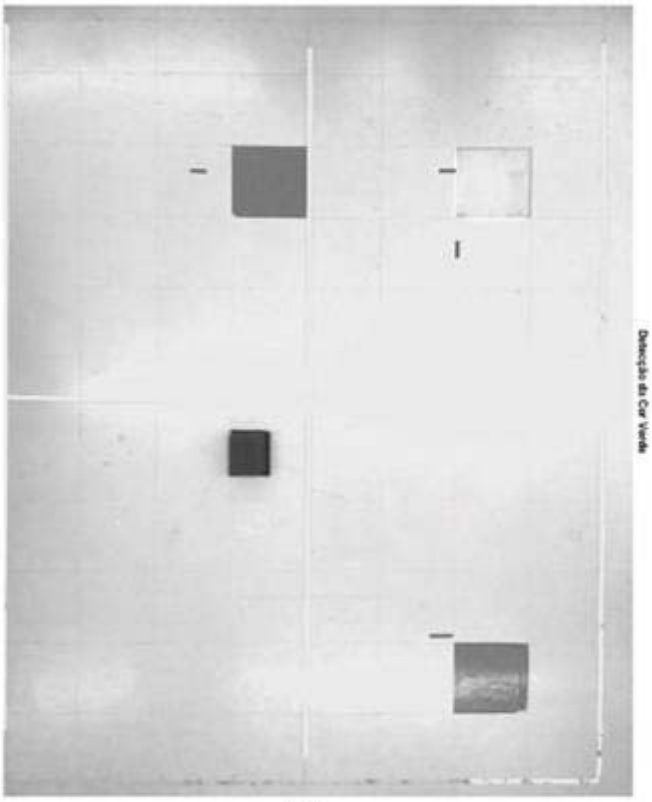
```
% % Câmera do Raspberry

    disp(' Aplicando as configurações da câmera... ');

    % Configuração da resolução da câmera.
    cam = cameraboard(rpi,'Resolution','1024x768');
    % Configura a saturação da câmera.
    cam.Saturation = 100;
    % Configura o brilho da câmera.
    cam.Brightness = 70;
    % Configura a qualidade da câmera.
    cam.Sharpness = 100;
    % Filtro para retirada de ruído próprio da câmera.
    cam.ImageEffect = 'denoise';

%% Motores do Lego

    % Motor A

    % Seleciona os motores a serem utilizados
    MotorA = NXTMotor('A');
    % Potência total em percentual dos motores variando de 0(mínimo) até 100(máximo)
    MotorA.Power = 100;
    MotorA.SpeedRegulation = false;
    MotorA.SmoothStart = true;

    % Motor C

    MotorC = NXTMotor('C');
    MotorC.Power = 100;
    MotorC.SpeedRegulation = false;
    MotorC.SmoothStart = true;

    % Motor AC

    MotorAC = NXTMotor('AC');
    MotorAC.Power = 100;
    MotorAC.SpeedRegulation = false;
```

```
MotorAC.SmoothStart = true;
```

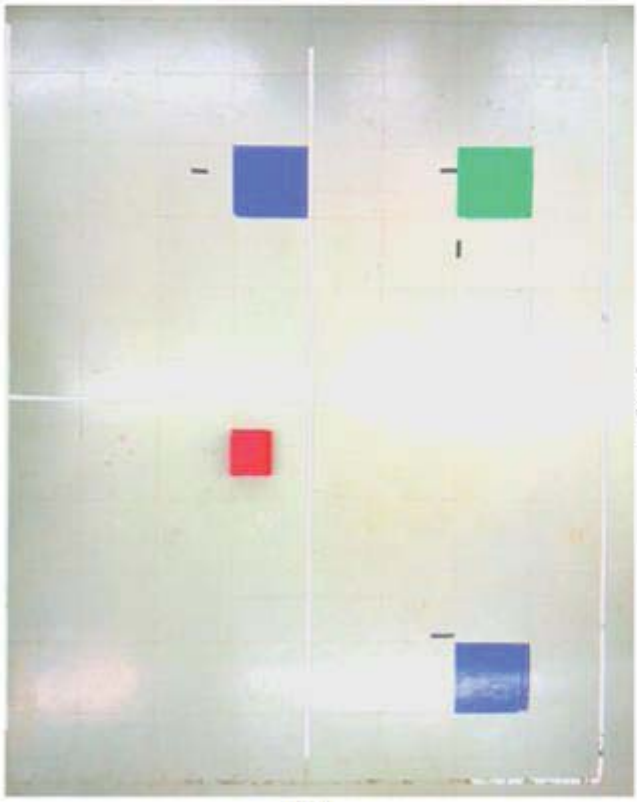
APÊNDICE C – FOTO, PRIMEIRO FILTRO, FILTRO DE REALCE E FILTRO DE CORREÇÃO

```
%% Foto

   % Tira 4 fotos.
   for i = 1:4
      img = snapshot(cam);
      drawnow;
   end

%% Primeiro Filtro

   rgbImage = img;
   % Separa a imagem criada em 3 com prioridades de cores em: vermelho,
   % verde e azul.
   redChannel = rgbImage(:, :, 1);
   greenChannel = rgbImage(:, :, 2);
   blueChannel = rgbImage(:, :, 3);

%% Filtro de Realce

   % Filtro implementado para fazer a exclusão das cores não desejadas.
   Red = (redChannel - (blueChannel ./ 2 + greenChannel .* 2));
   Green = (greenChannel - (redChannel ./ 2 + blueChannel));
   Blue = (blueChannel - (redChannel ./ 2 + greenChannel));

%% Filtro MEDFILT2 e IMFILL

   RedAjustado = imfill((medfilt2(Red) .* 20), 8, 'holes');

   BlueAjustado = imfill((medfilt2(Blue) .* 10), 8 , 'holes');

   GreenAjustado = imfill((medfilt2(GreenUm) .* 50), 8, 'holes');
```

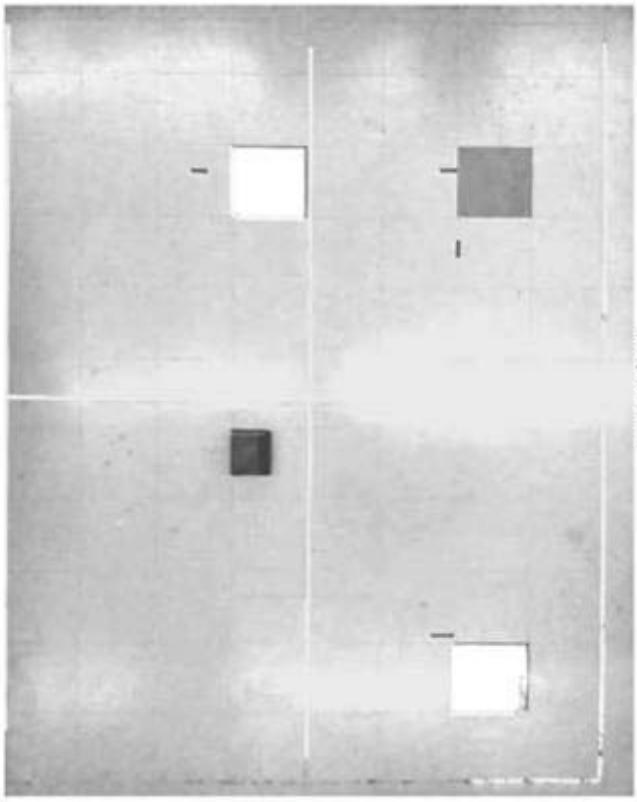
APÊNDICE D – PRIMEIRO FILTRO E SUAS RESPOSTAS



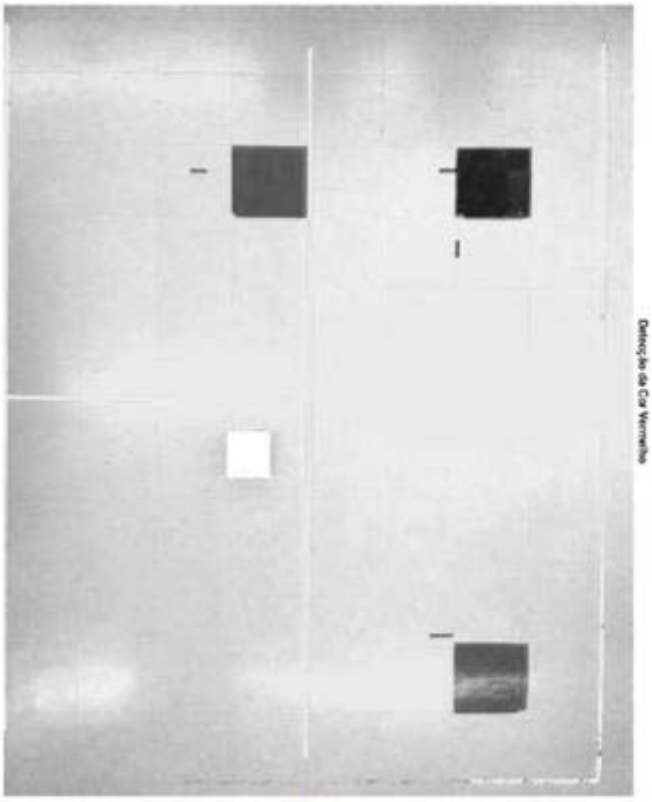
(c)



(a)

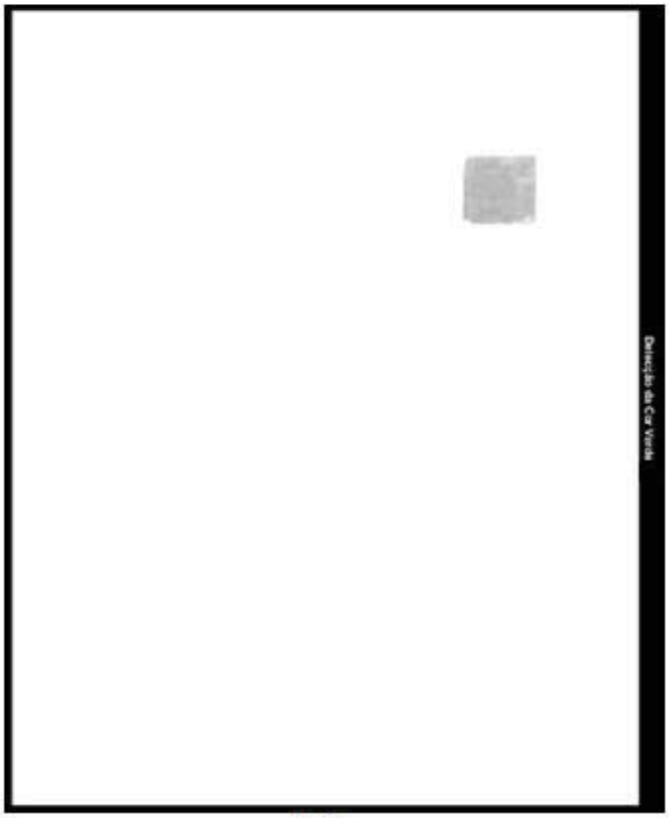


(d)

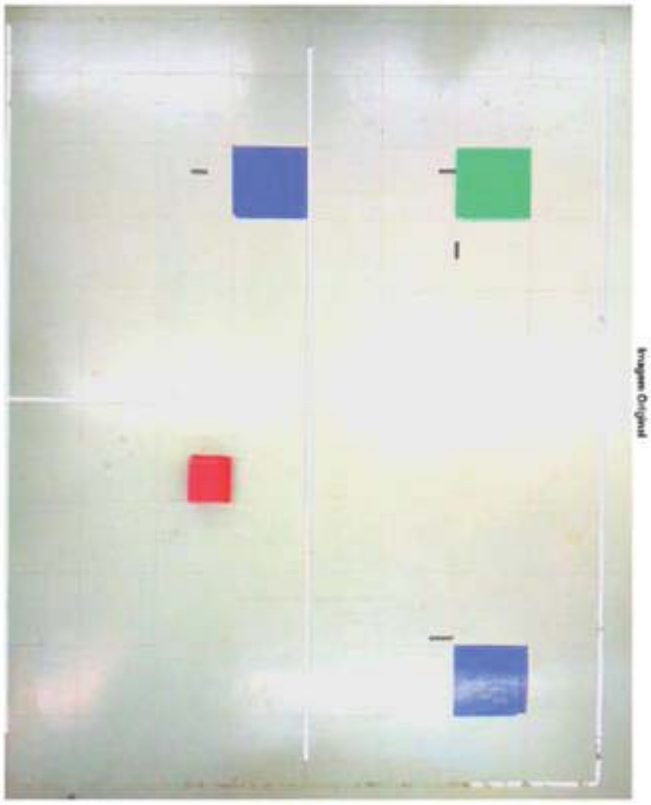


(b)

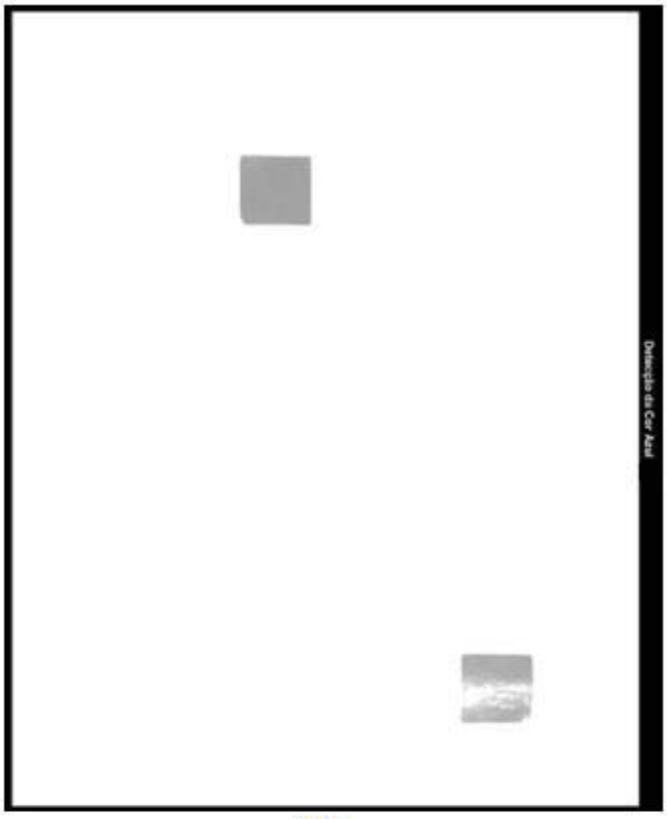
APÊNDICE E – RESPOSTA AOS FILTROS DE REALCE



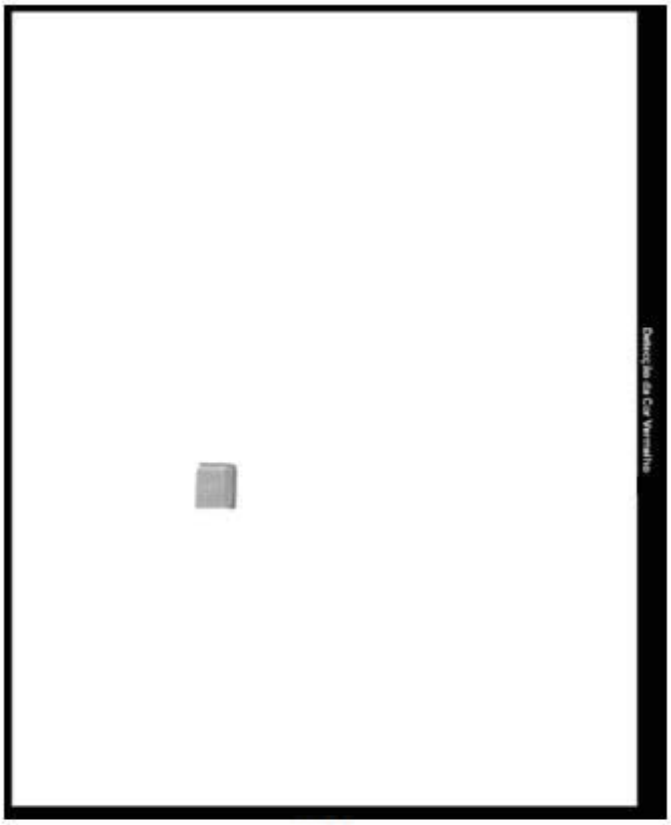
(c)



(a)

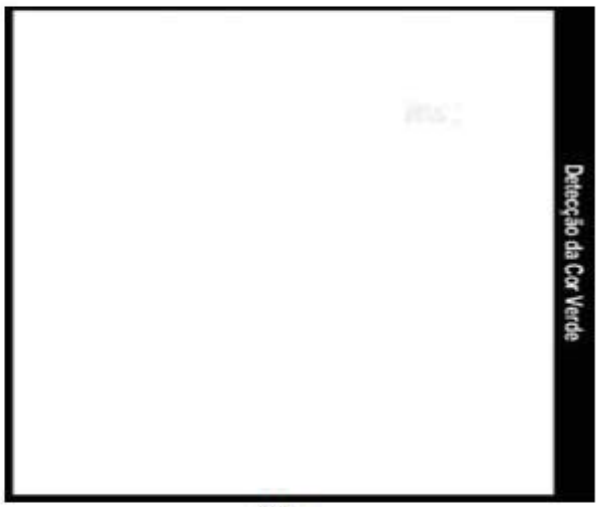


(d)

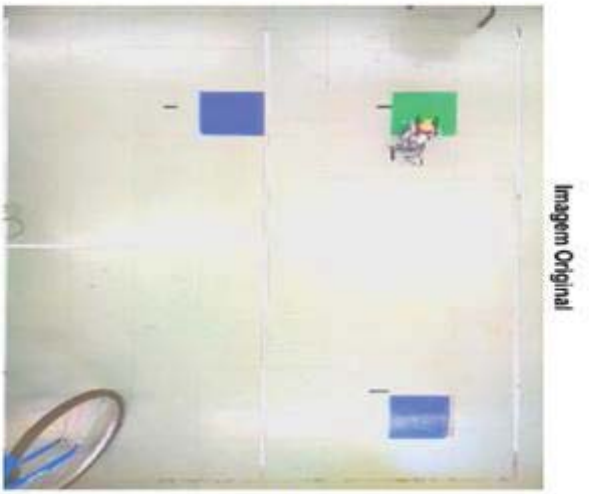


(b)

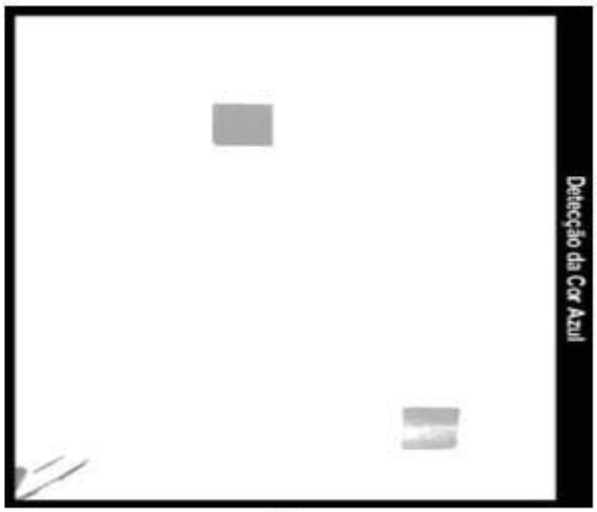
APÊNDICE F – RESPOSTA AOS FILTROS MEDFILT2 E IMFILL



(c)



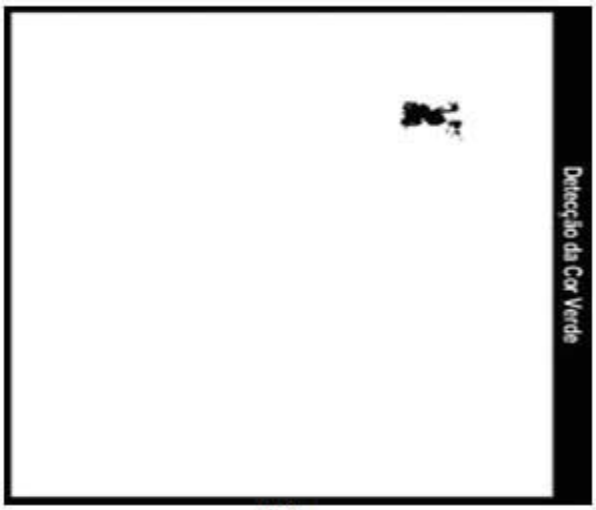
(a)



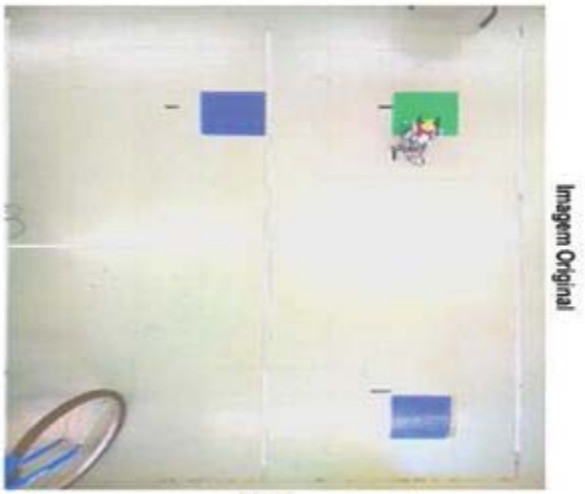
(d)

(b)

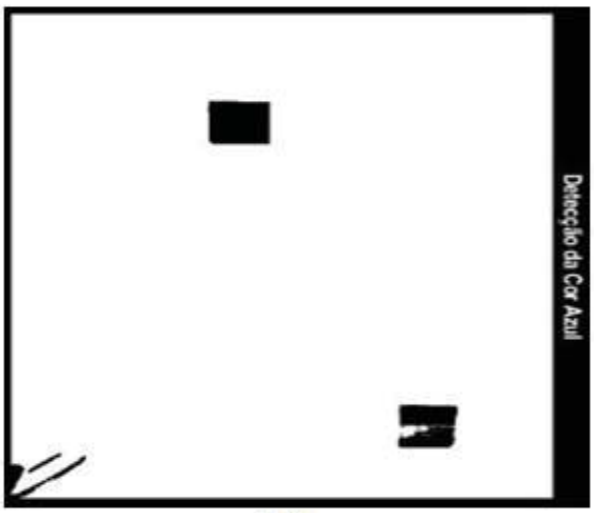
APÊNDICE G – RESPOSTA A SATURAÇÃO DA COR CINZA



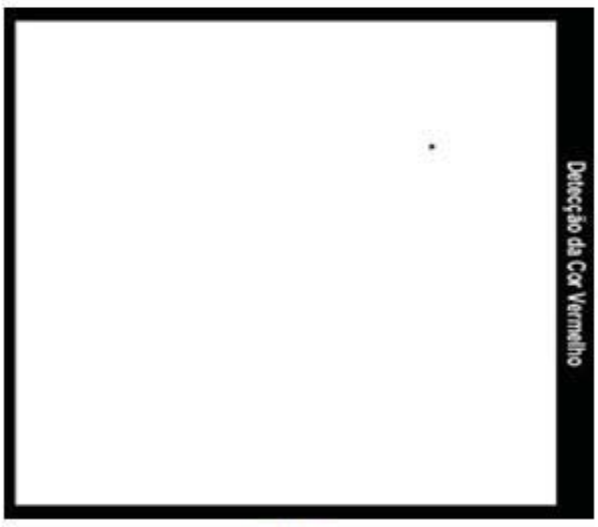
(c)



(a)



(d)

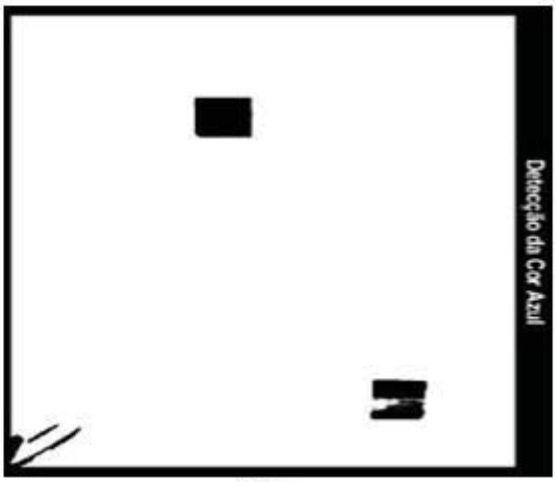


(b)

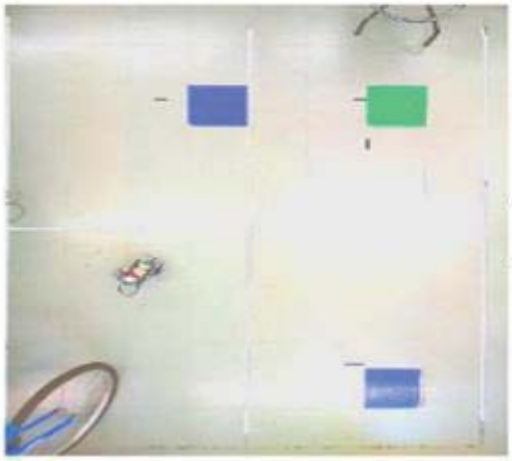
APÊNDICE H – DETECÇÃO DE NOVAS CORES

```
%% Filtro de cor AMARELO

    Yellow = (greenChannel - (redChannel ./ 2 + blueChannel .* 2));

   GreenUm = ((Green - Yellow) - uint8(50 .* ones(768,1024)));

    YellowUm = medfilt2(Yellow) - GreenAjustado;

    YellowAjustado = imfill(medfilt2(YellowUm) .* 100, 'holes');

%% Filtro de cor LARANJA

        Orange = (((redChannel - (blueChannel ./ 2 + greenChannel .* 1.4)) - RedAjustado) - YellowAjustado);

    OrangeUm = (Orange .* 20 - uint8(20 .* ones(768,1024)));

    OrangeAjustado = imfill(medfilt2(OrangeUm) .* 50, 'holes');

    RedAjustado2 = (RedAjustado - OrangeAjustado);
    OrangeAjustado2 = (RedAjustado2 - OrangeAjustado - YellowAjustado);
    YellowAjustado2 = (YellowAjustado - OrangeAjustado2 - RedAjustado2);
```

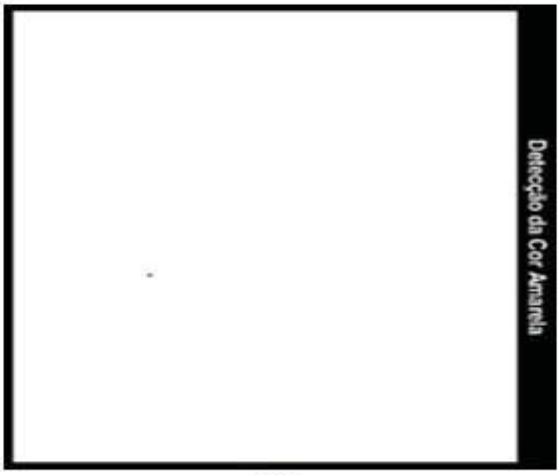
APÊNDICE I – RESPOSTA AS NOVAS CORES



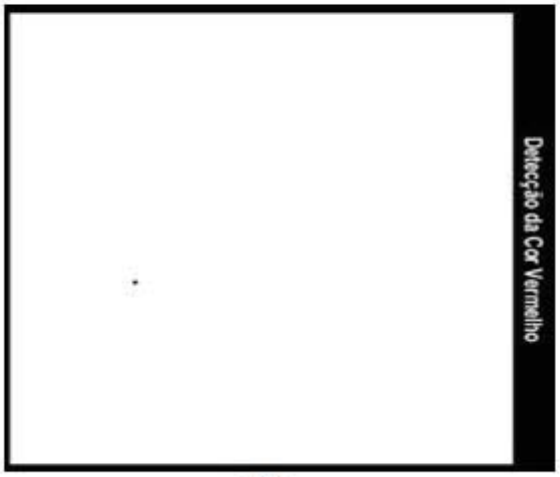
(d)



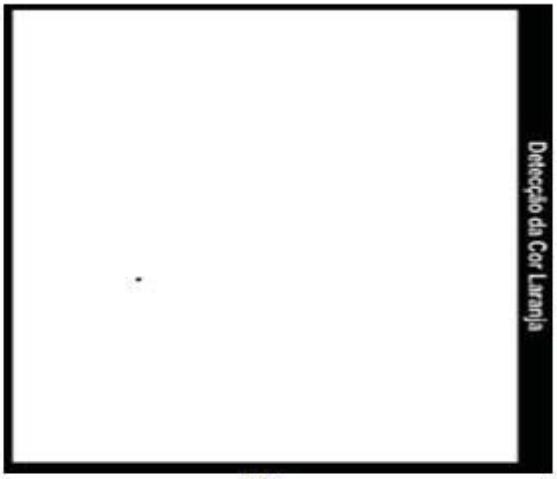
(a)



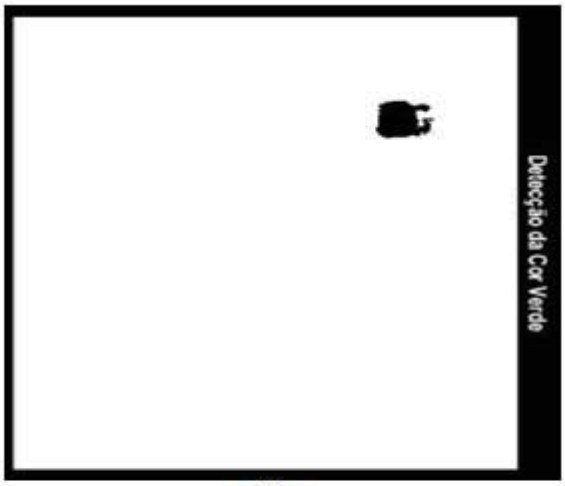
(e)



(b)



(f)



(c)

APÊNDICE J – RETIRADA DE DADOS NÚMERICOS

```
% Acha os valores equivalentes a 1 na matrizes e os guarda em coordenadas
% de X e Y.

[xR,yR] = find(RedMatrizUnitaria == 1);
[xG,yG] = find(GreenMatrizUnitaria == 1);
[xB,yB] = find(BlueMatrizUnitaria == 1);
[xY,yY] = find(YellowMatrizUnitaria == 1);
[xO,yO] = find(OrangeMatrizUnitaria == 1);

% Guarda o valor mínimo e máximo numa variável.

yRMIN = min(yR)
yRMAX = max(yR)
xRMIN = min(xR)
xRMAX = max(xR)

yGMIN = min(yG)
yGMAX = max(yG)
xGMIN = min(xG)
xGMAX = max(xG)

yBMIN = min(yB)
yBMAX = max(yB)
xBMIN = min(xB)
xBMAX = max(xB)

yYMIN = min(yY)
yYMAX = max(yY)
xYMIN = min(xY)
xYMAX = max(xY)

yOMIN = min(yO)
yOMAX = max(yO)
xOMIN = min(xO)
xOMAX = max(xO)

% Faz a diferença dos pontos máximos e mínimos obtidos anteriormente.
```

DiferencaXR = (xRMAX - xRMIN)
DiferencaYR = (yRMAX - yRMIN)

DiferencaXG = (xGMAX - xGMIN)
DiferencaYG = (yGMAX - yGMIN)

DiferencaXB = (xBMAX - xBMIN)
DiferencaYB = (yBMAX - yBMIN)

DiferencaXY = (xYMAX - xYMIN)
DiferencaYY = (yYMAX - yYMIN)

DiferencaXO = (xOMAX - xOMIN)
DiferencaYO = (yOMAX - yOMIN)

% Ponto médio truncado(round).

PontoMedioXR = round(DiferencaXR / 2)
PontoMedioYR = round(DiferencaYR / 2)

PontoMedioXG = round(DiferencaXG / 2)
PontoMedioYG = round(DiferencaYG / 2)

PontoMedioXB = round(DiferencaXB / 2)
PontoMedioYB = round(DiferencaYB / 2)

PontoMedioXY = round(DiferencaXY / 2)
PontoMedioYY = round(DiferencaYY / 2)

PontoMedioXO = round(DiferencaXO / 2)
PontoMedioYO = round(DiferencaYO / 2)

% Ponto inicial mais o ponto médio é igual as coordenadas das cores.

CoordenadaDoQuadradoVermelhoXR = xRMIN + PontoMedioXR
CoordenadaDoQuadradoVermelhoYR = yRMIN + PontoMedioYR

CoordenadaDoQuadradoVerdeXG = xGMIN + PontoMedioXG
CoordenadaDoQuadradoVerdeYG = yGMIN + PontoMedioYG

CoordenadaDoQuadradoAzulXB = xBMIN + PontoMedioXB
CoordenadaDoQuadradoAzulYB = yBMIN + PontoMedioYB

CoordenadaDoQuadradoAmareloXY = xYMIN + PontoMedioXY
CoordenadaDoQuadradoAmareloYY = yYMIN + PontoMedioYY

CoordenadaDoQuadradoLaranjaXO = xOMIN + PontoMedioXO
CoordenadaDoQuadradoLaranjaYO = yOMIN + PontoMedioYO

APÊNDICE K – CÁLCULO DO ÂNGULO

% Distância das cores.

DistanciaVermelhoVerdeX = abs(CoordenadaDoQuadradoVermelhoXR - CoordenadaDoQuadradoVerdeXG)
DistanciaVermelhoVerdeY = abs(CoordenadaDoQuadradoVermelhoYR - CoordenadaDoQuadradoVerdeYG)

DistanciaLaranjaVerdeX = abs(CoordenadaDoQuadradoLaranjaXO - CoordenadaDoQuadradoVerdeXG)
DistanciaLaranjaVerdeY = abs(CoordenadaDoQuadradoLaranjaYO - CoordenadaDoQuadradoVerdeYG)

% Catetos para ter o ângulo fazendo o triãngulo retângulo.

CatetoOpostoXOG = abs(CoordenadaDoQuadradoLaranjaXO - CoordenadaDoQuadradoVerdeXG)
CatetoAdjacenteYOG = abs(CoordenadaDoQuadradoLaranjaYO - CoordenadaDoQuadradoVerdeYG)

CatetoOpostoXRG = abs(CoordenadaDoQuadradoVermelhoXR - CoordenadaDoQuadradoVerdeXG)
CatetoAdjacenteYRG = abs(CoordenadaDoQuadradoVermelhoYR - CoordenadaDoQuadradoVerdeYG)

CatetoOpostoXYG = abs(CoordenadaDoQuadradoAmareloXY - CoordenadaDoQuadradoVerdeXG)
CatetoAdjacenteYYG = abs(CoordenadaDoQuadradoAmareloYY - CoordenadaDoQuadradoVerdeYG)

CatetoOpostoXLVG = abs(CoordenadaCentralVermelhoLaranjaX - CoordenadaDoQuadradoVerdeXG)
CatetoAdjacenteYLVG = abs(CoordenadaCentralVermelhoLaranjaY - CoordenadaDoQuadradoVerdeYG)

CatetoOpostoXYB = abs(CoordenadaDoQuadradoAmareloXY - CoordenadaDoQuadradoAzulXB)
CatetoAdjacenteYYB = abs(CoordenadaDoQuadradoAmareloYY - CoordenadaDoQuadradoAzulYB)

% Somatório das cores para saber, qual a maior distância.

SomatorioLaranja = CoordenadaDoQuadradoLaranjaXO + CoordenadaDoQuadradoLaranjaYO
SomatorioVermelho = CoordenadaDoQuadradoVermelhoXR + CoordenadaDoQuadradoVermelhoYR

% Ângulos do triângulo retângulo

AnguloLVG = round(atan(CatetoOpostoXLVG / CatetoAdjacenteYLVG) / 0.0174533)
AnguloYG = round(atan(CatetoOpostoXYG / CatetoAdjacenteYYG) / 0.0174533)
AnguloOG = round(atan(CatetoOpostoXOG / CatetoAdjacenteYOG) / 0.0174533) *% 0.0174533 = 1º / Transforma de radianos para graus o valor obtido*
AnguloRG = round(atan(CatetoOpostoXRG / CatetoAdjacenteYRG) / 0.0174533)
AnguloYB = round(atan(CatetoOpostoXYB / CatetoAdjacenteYYB) / 0.0174533)

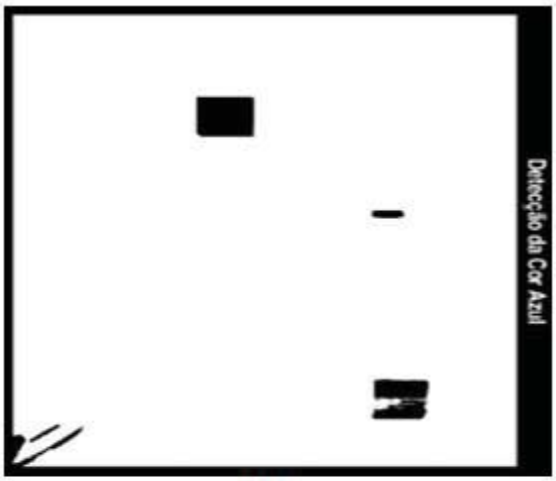
```
% Áreas para verificar a chegada até o destino.

AreaVerde = (DiferencaXG * DiferencaYG)
AreaAmarela = (DiferencaXY * DiferencaYY)

AreaVerdePixel = sum(sum(GreenMatrizUnitaria))
AreaAzulPixel = sum(sum(BlueMatrizUnitaria))
```

APÊNDICE L – CONDIÇÕES DE EXECUÇÃO

```
% Valores maior que do VERDE.

% Rotação 180º

if (AnguloYB > 80) && (AnguloLVG > 80) && (CoordenadaDoQuadradoAmareloXY > CoordenadaCentralVermelhoLaranjaX)
   MotorA.TachoLimit = 950;
   MotorA.SendToNXT();
   pause(3)
   Imagem
end

if (AnguloYB < 10) && (AnguloLVG < 10) && (CoordenadaDoQuadradoAmareloYY > CoordenadaCentralVermelhoLaranjaY)
   MotorA.TachoLimit = 950;
   MotorA.SendToNXT();
   pause(3)
   Imagem
end

% Ângulos

if (AnguloYB > AnguloLVG)
   RotacaoAnguloLVG = (AnguloLVG * 220) / 45;
   MotorC.TachoLimit = round(RotacaoAnguloLVG);
   MotorC.SendToNXT();
   pause(3);
   Imagem
end

if (AnguloYB < AnguloLVG)
   RotacaoAnguloLVG = (AnguloLVG * 220) / 45;
   MotorA.TachoLimit = round(RotacaoAnguloLVG);
   MotorA.SendToNXT();
   pause(3);
   Imagem
end
```

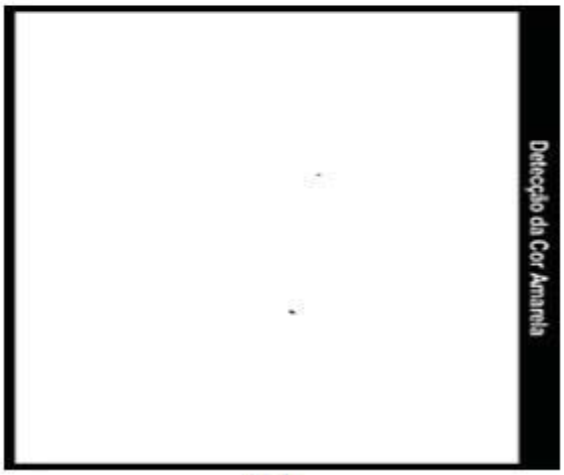
```
% Eixo X - Vertical

if (CoordenadaDoQuadradoAmareloXY > CoordenadaDoQuadradoAzulXB)
   DistanciaX = CoordenadaDoQuadradoAmareloXY - CoordenadaDoQuadradoAzulXB;
   MotorAC.TachoLimit = round(DistanciaX * 3);
   MotorAC.SendToNXT();
   pause(3)
   Imagem
end

% Eixo Y - Horizontal

if (CoordenadaDoQuadradoAmareloYY > CoordenadaDoQuadradoAzulYB)
   DistanciaY = CoordenadaDoQuadradoAmareloYY - CoordenadaDoQuadradoAzulYB;
   MotorAC.TachoLimit = round(DistanciaY * 3);
   MotorAC.SendToNXT();
   pause(3)
   Imagem
end
```

APÊNDICE M – TESTE DE RECONHECIMENTO DAS CORES

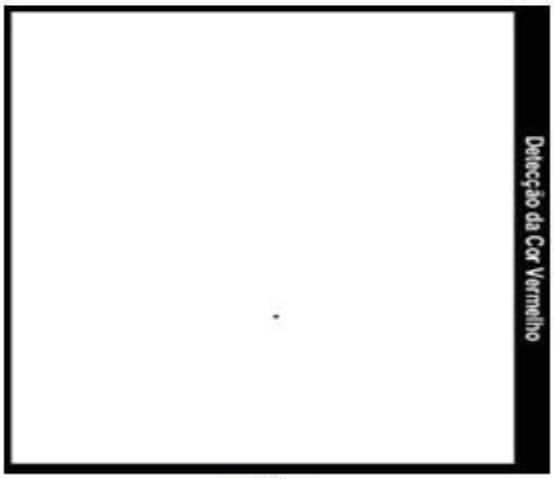


(d)

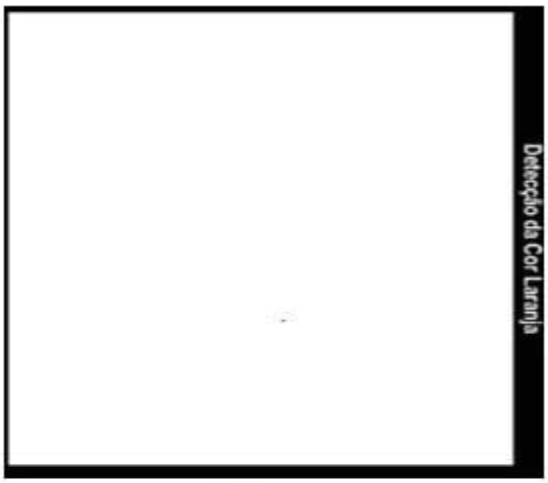
(a)



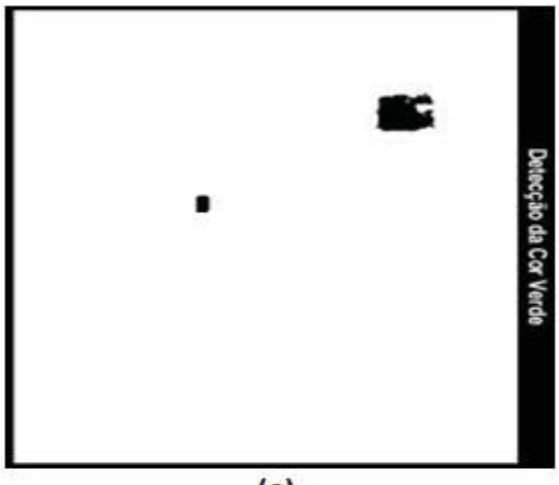
(e)



(b)

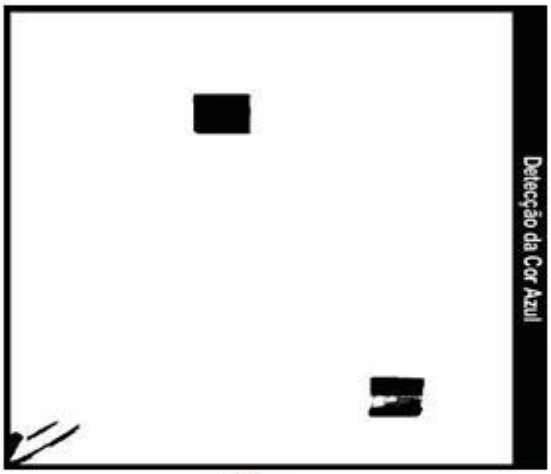


(f)

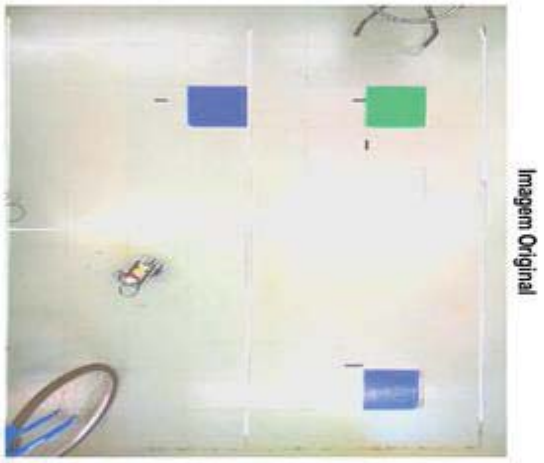


(c)

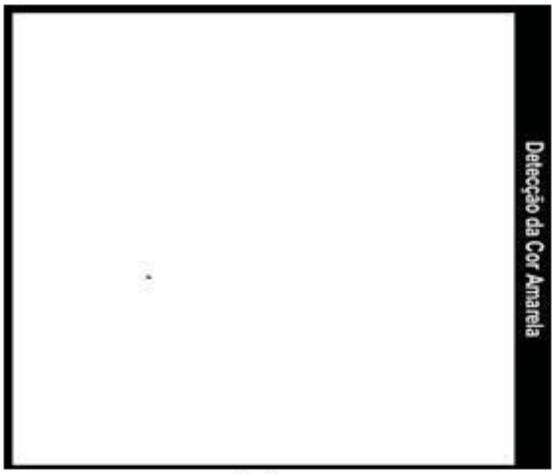
APÊNDICE N – TESTE DE MOBILIDADE PARTE 1



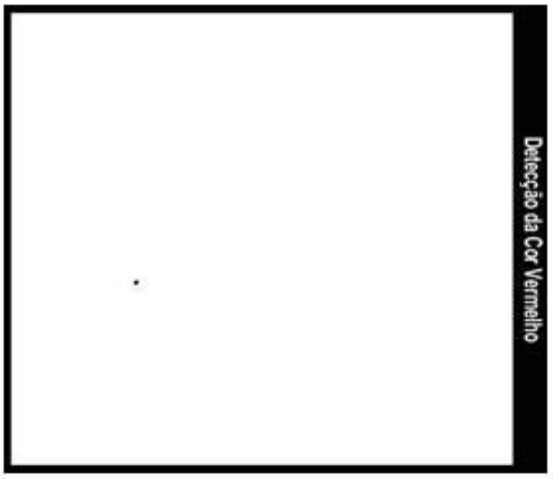
(d)



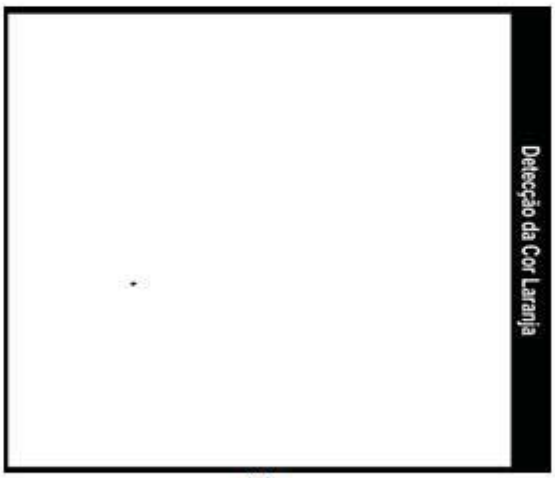
(a)



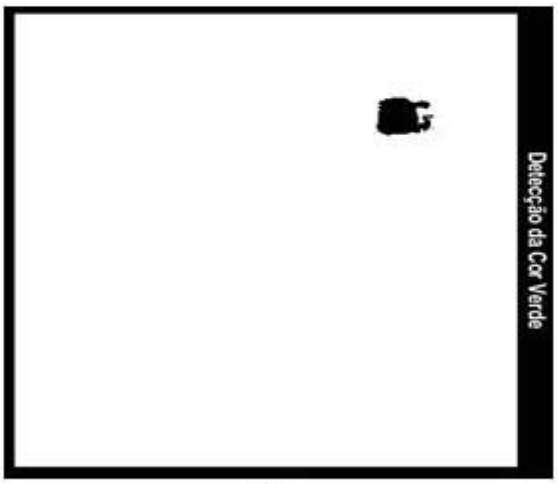
(e)



(b)

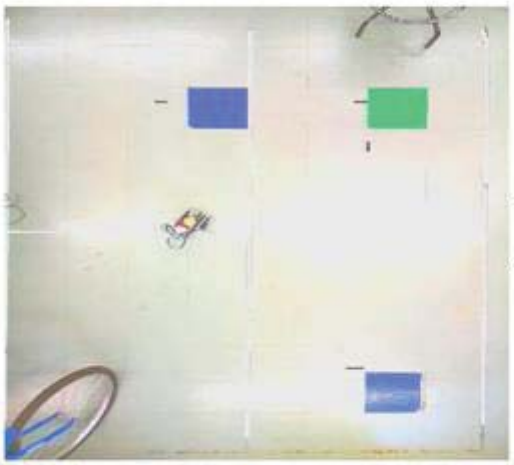


(f)

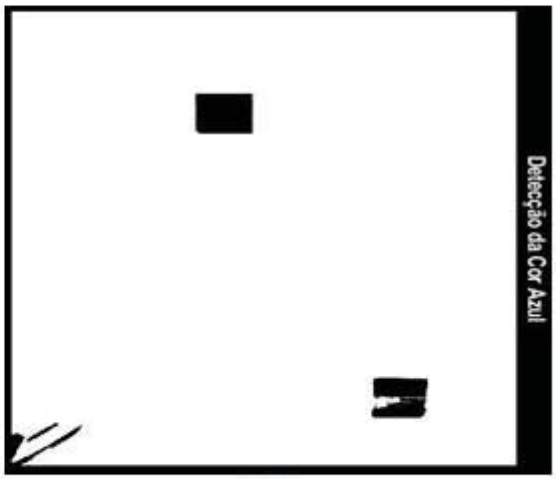


(c)

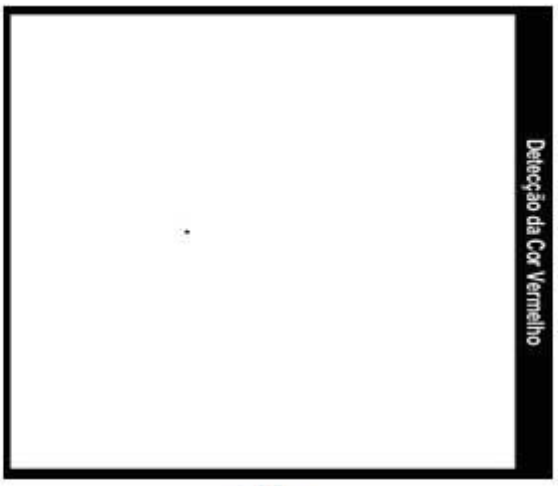
APÊNDICE O – TESTE DE MOBILIDADE PARTE 2



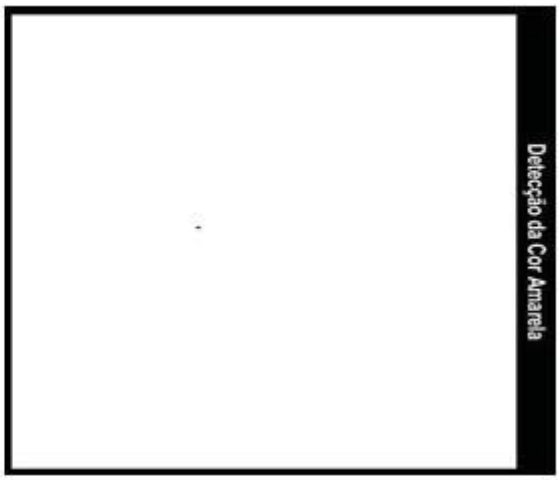
(a)



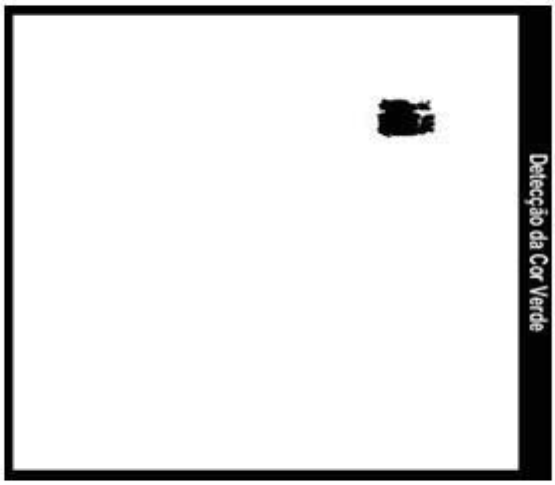
(e)



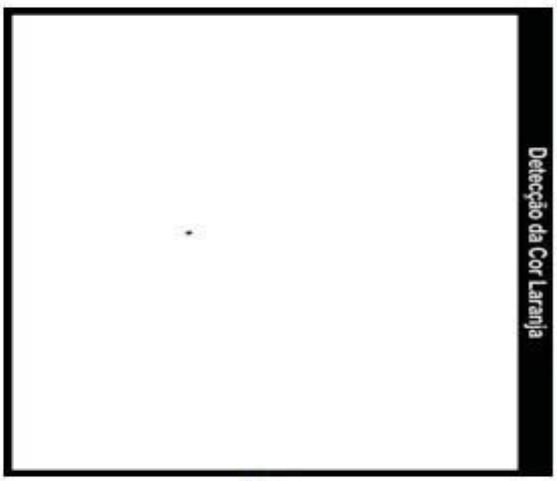
(b)



(e)

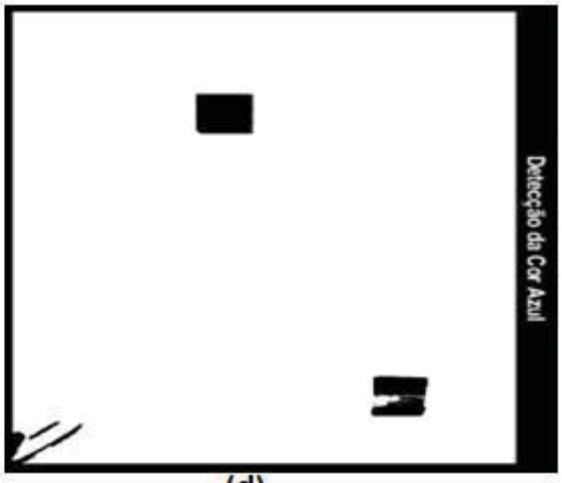


(c)

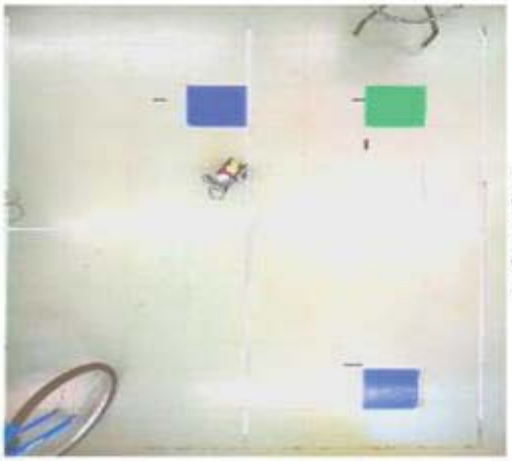


(f)

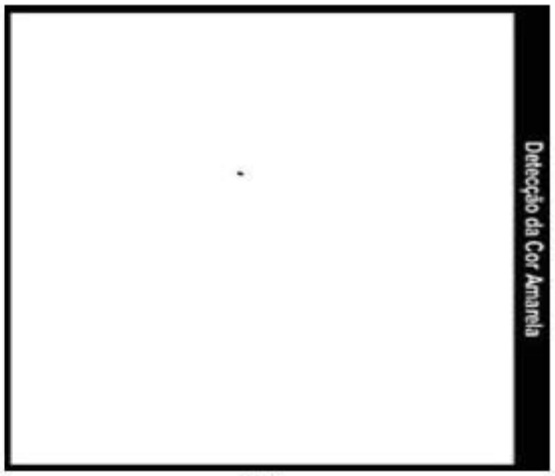
APÊNDICE P – TESTE DE MOBILIDADE PARTE 3



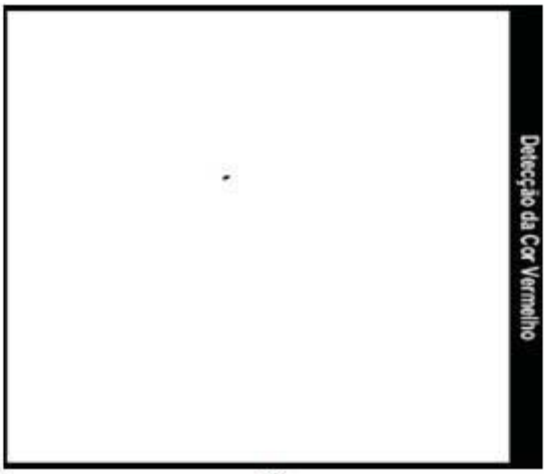
(d)



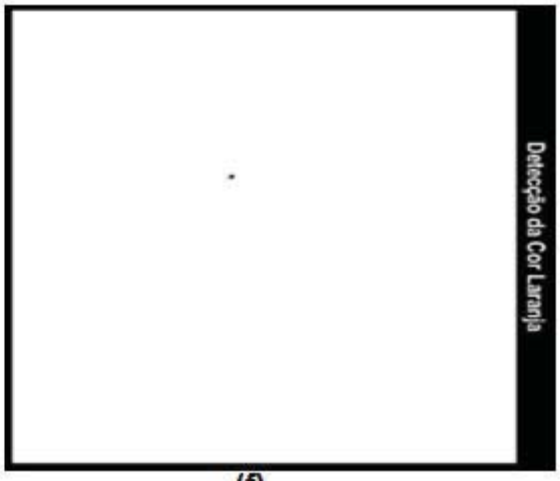
(a)



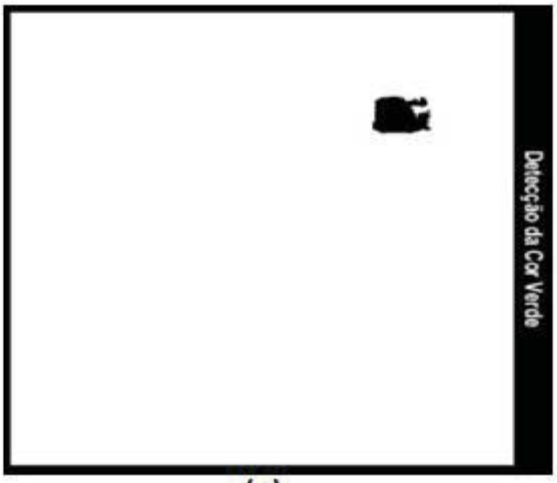
(e)



(b)

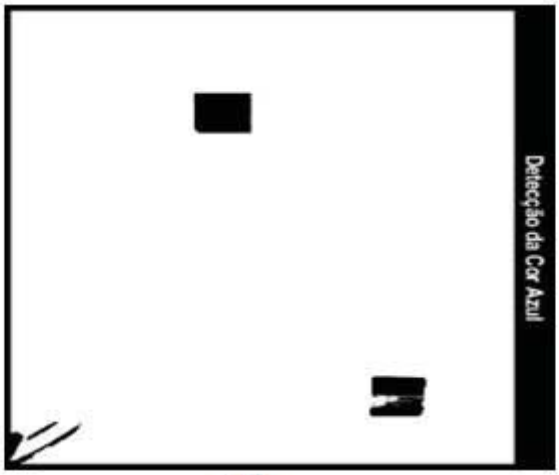


(f)

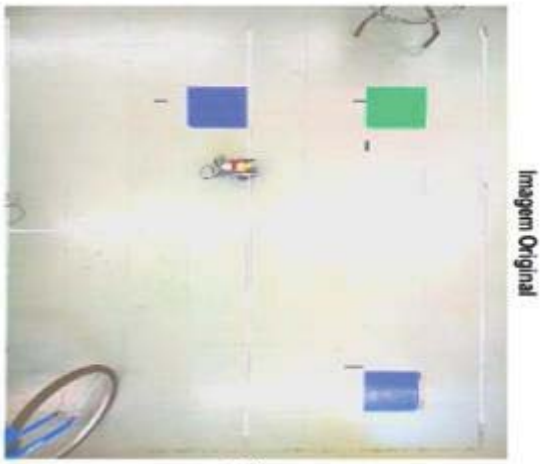


(c)

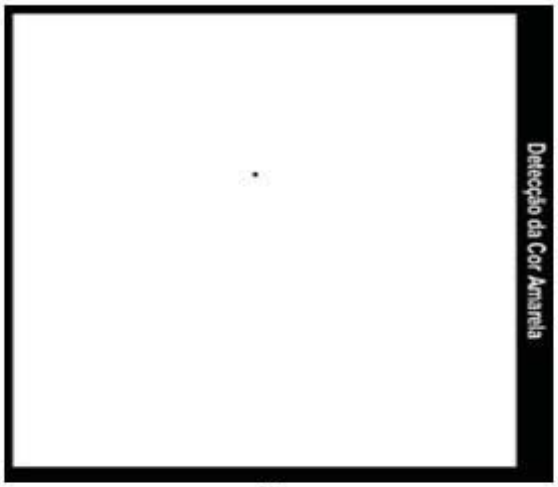
APÊNDICE Q – TESTE DE MOBILIDADE PARTE 4



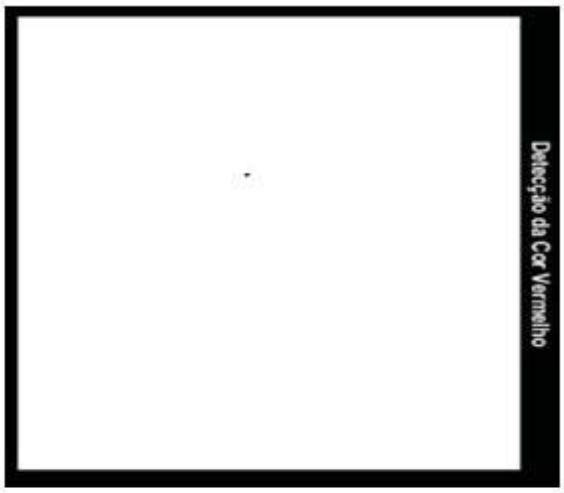
(d)



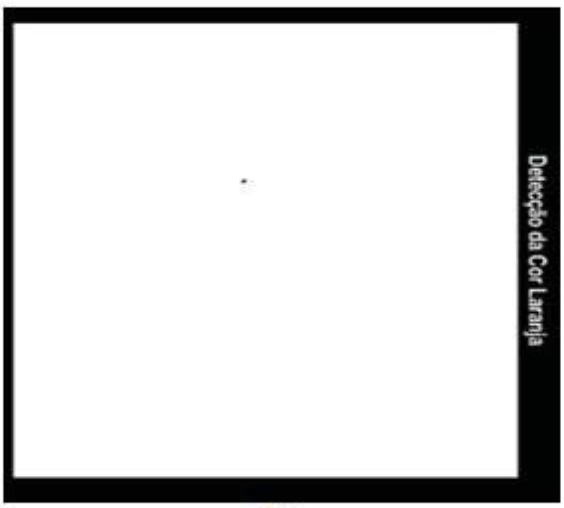
(a)



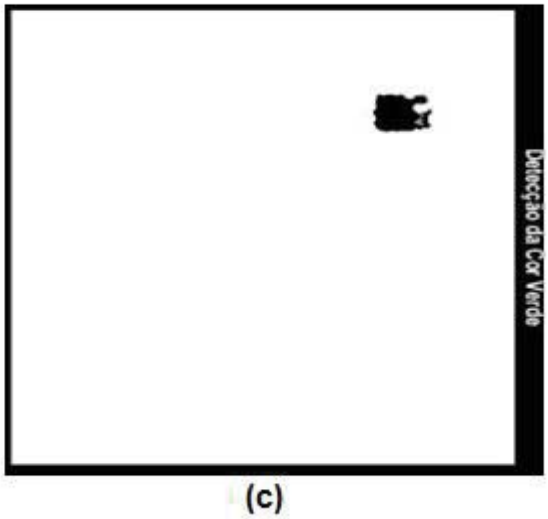
(e)



(b)

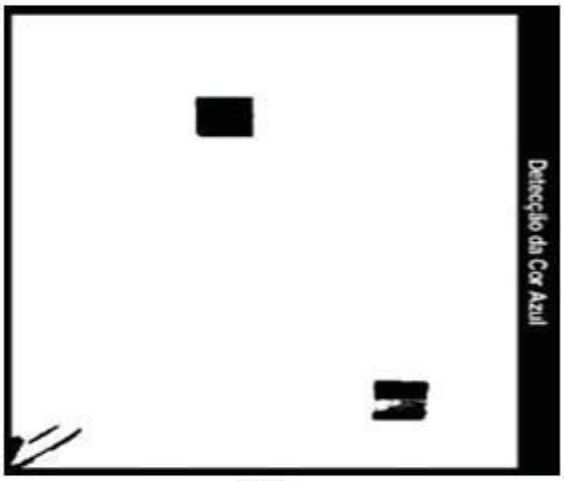


(f)

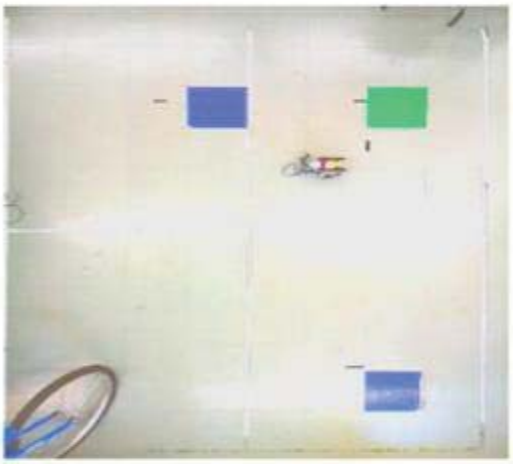


(c)

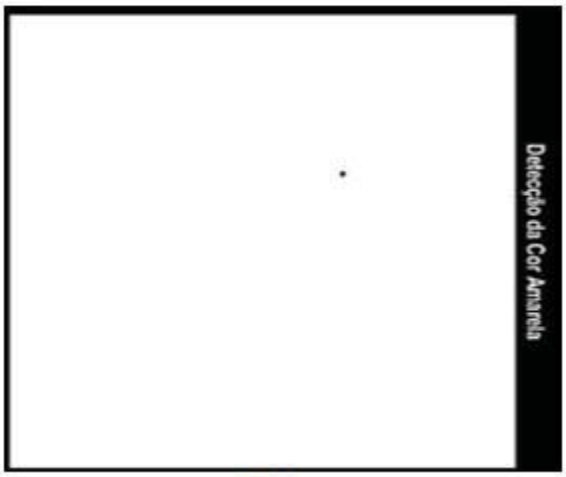
APÊNDICE R – TESTE DE MOBILIDADE PARTE 5



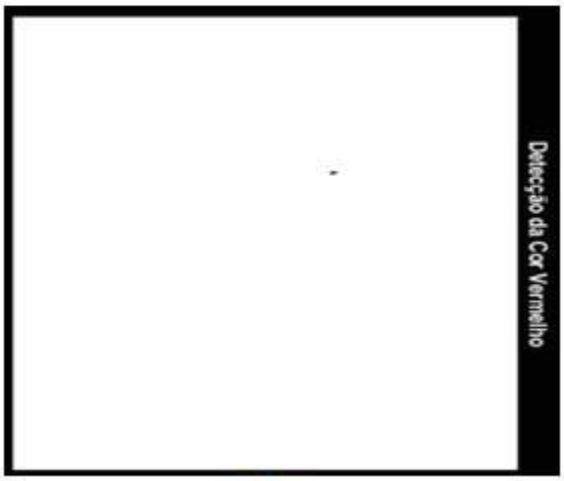
(d)



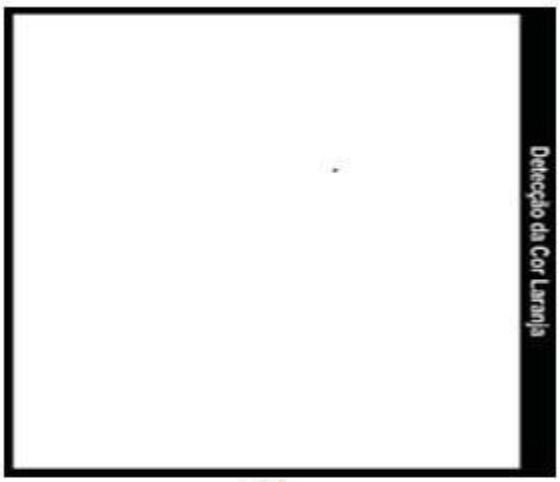
(a)



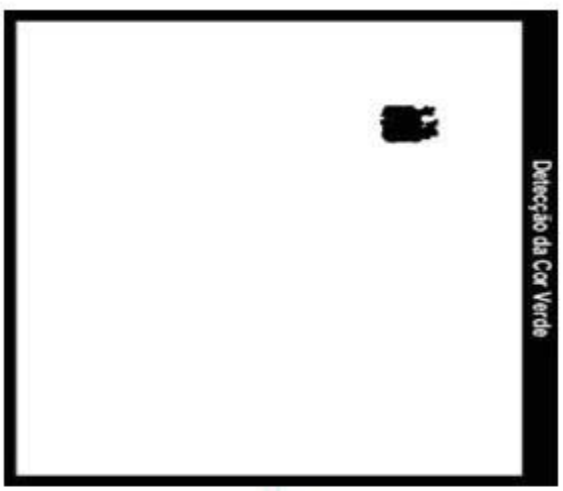
(e)



(b)

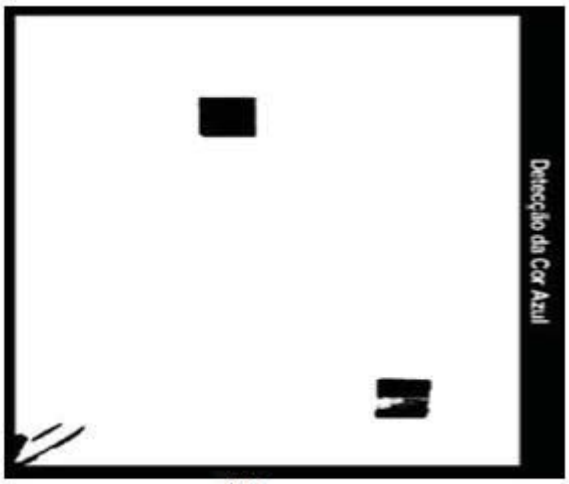


(f)

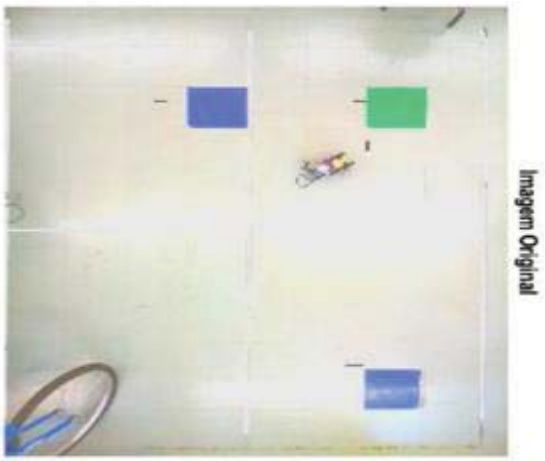


(c)

APÊNDICE S – TESTE DE MOBILIDADE PARTE 6

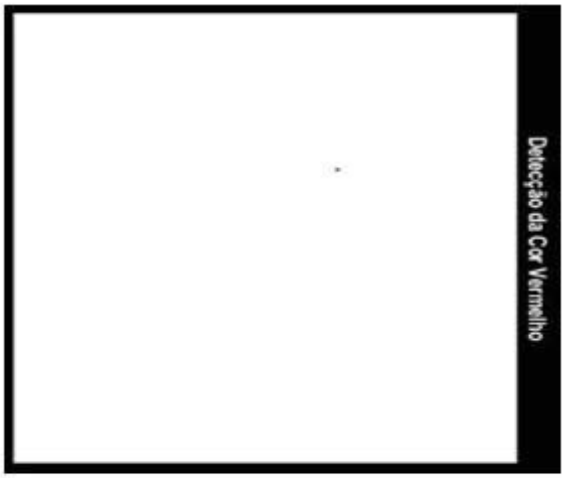


(d)

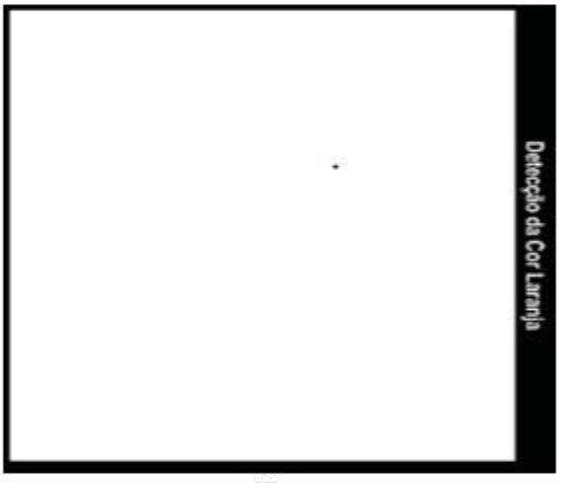


(a)

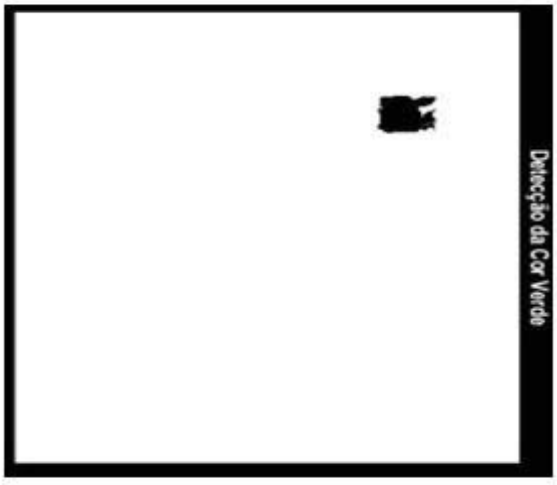
(e)



(b)

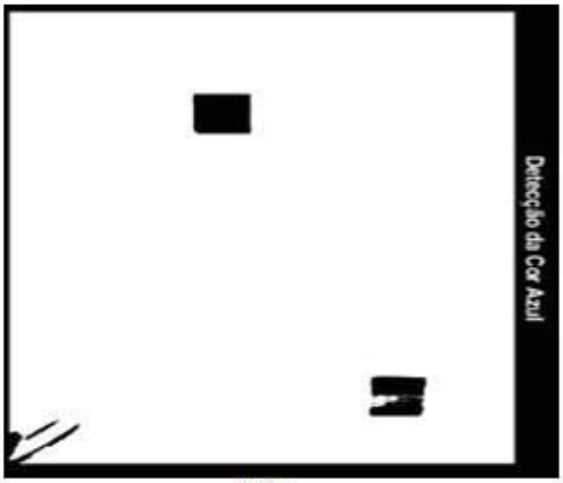


(f)

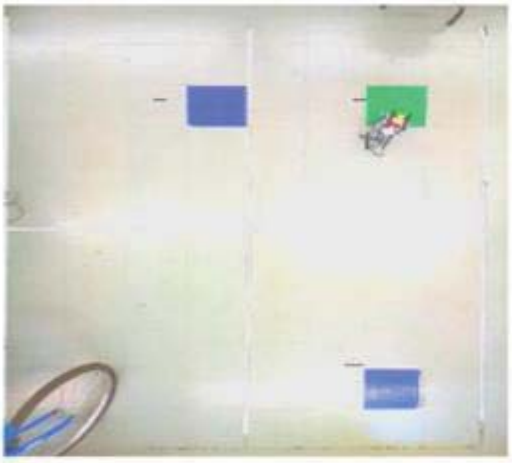


(c)

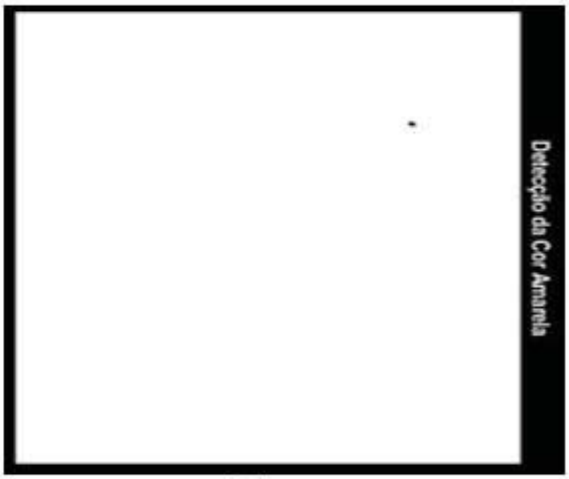
APÊNDICE T – TESTE DE MOBILIDADE PARTE 7



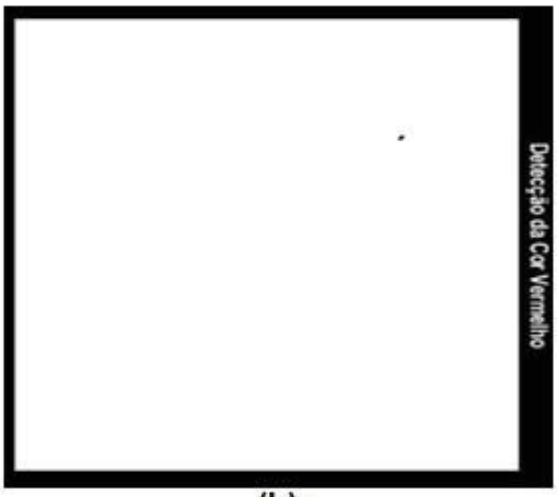
(d)



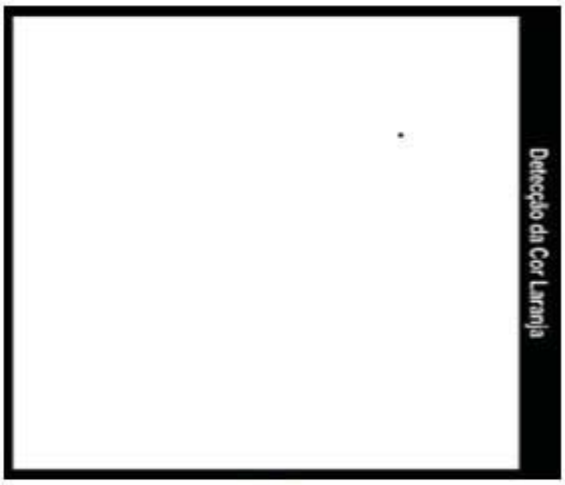
(a)



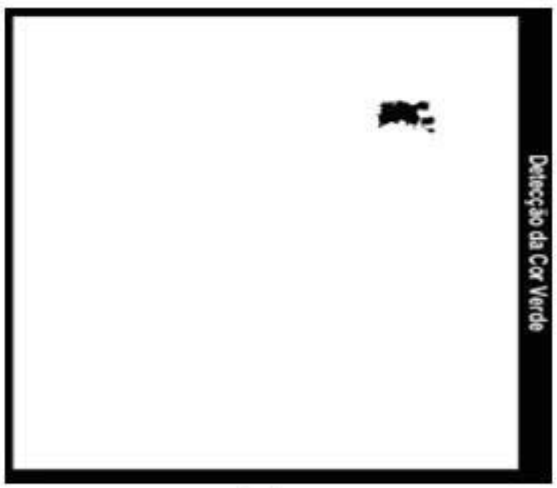
(e)



(b)



(f)



(c)