OSCAR ERNESTO ROJAS ROJAS

Implementação de um Sistema de Medição de Ângulos para Alinhamento da Direção Veicular Usando Visão Computacional

20/2013

CAMPINAS
2013

UNICAMP

UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA MECÂNICA

OSCAR ERNESTO ROJAS ROJAS

Implementação de um Sistema de Medição de Ângulos para Alinhamento da Direção Veicular Usando Visão Computacional

Orientador: Prof. Dr. Paulo Roberto Gardel Kurka

Dissertação de Mestrado apresentada à Faculdade de Engenharia Mecânica da Universidade Estadual de Campinas para obtenção do título de Mestre em Engenharia Mecânica, na Área de Mecânica dos Sólidos e Projeto Mecânico.

ESTE EXEMPLAR CORRESPONDE À VERSÃO FINAL DA DISSERTAÇÃO DEFENDIDA PELO ALUNO OSCAR ERNESTO ROJAS ROJAS E ORIENTADA PELO PROF. DR. PAULO ROBERTO GARDEL KURKA

ASSINATURA DO ORIENTADOR

CAMPINAS
2013

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA E ARQUITETURA - BAE - UNICAMP

Rojas Rojas, Oscar Ernesto
R638i Implementação de um sistema de medição de ângulos para alinhamento da direção veicular usando visão computacional / Oscar Ernesto Rojas Rojas. --Campinas, SP: [s.n.], 2013.

Orientador: Paulo Roberto Gardel Kurka.
Dissertação de Mestrado - Universidade Estadual de Campinas, Faculdade de Engenharia Mecânica.

1. Programação orientada a objetos (Computação). 2. Visão artificial. 3. Visão por computador. 4. Visão por computador - Aplicações industriais. 5. Automóveis - Molas e suspensão. I. Kurka, Paulo Roberto Gardel, 1958-. II. Universidade Estadual de Campinas. Faculdade de Engenharia Mecânica. III. Título.

Título em Inglês: Implementation of an angle measurement system for vehicular steering alignment using computer vision
Palavras-chave em Inglês: Object-oriented programming (Computer science), Artificial vision, Computer Vision, Computer Vision - Industrial, Automobiles - Springs and suspension
Área de concentração: Mecânica dos Sólidos e Projeto Mecânico
Titulação: Mestre em Engenharia Mecânica
Banca examinadora: Antonio Celso Fonseca de Arruda, Clésio Luis Tozzi
Data da defesa: 15-02-2013
Programa de Pós Graduação: Engenharia Mecânica

UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA MECÂNICA

COMISSÃO DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA

DEPARTAMENTO DE PROJETO MECÂNICO

DEFESA DE MESTRADO ACADEMICO

Implementação de um Sistema de Medição de Ângulos para Alinhamento da Direção Veicular Usando Visão Computacional

Autor: Oscar Ernesto Rojas Rojas

Orientador: Prof. Dr. Paulo Roberto Gardel Kurka

A Banca Examinadora composta pelos membros abaixo aprovou esta dissertação:

Prof. Dr Paulo Roberto Gardel Kurka, Presidente
DPM/FEM/UNICAMP

Prof. Dr Antonio Celso Fonseca de Arruda
DEP/FEM/UNICAMP

Prof. Dr Clésio Luis Tozzi
DCA /FEEC/UNICAMP

Campinas, 15 de Fevereiro de 2013

Dedico este trabalho a minha mãe Clara Inés Rojas e a minha irmã Laura Rojas pelo apoio, carinho e compreensão.

Agradecimentos

Agradeço primeiramente a Deus pela oportunidade de realizar este trabalho.

Ao meu orientador Prof. Dr. Paulo Roberto Gardel Kurka, pela confiança, apoio e dedicação ao longo desse período.

A minha família, minha mãe e minha irmã pela paciência, compreensão e apoio.

Aos amigos e companheiros de trabalho: Fabiano Bargos, Caio Rodrigues, Jorge Suzuki, Carlos Mingoto e Jaime Delgado pela colaboração e amizade durante esse período.

Especial agradecimento a Jaime Izuka por toda a sua ajuda, colaboração, apoio e amizade. Uma pessoa com quem você sabe que sempre pode contar.

A Gilberto Luis Valente Costa, o primeiro brasileiro que conheci no Brasil. Uma pessoa que admiro e quero muito. Obrigado por fazer nossa estadia no Brasil ainda melhor.

A Paola Gonzalez Ramos pelo apoio, amizade e compreensão. Ajudou-me muito nesse período e tudo haveria sido muito mais difícil sem a sua presença, muito obrigado.

Aos amigos da Colômbia por todos os momentos de lazer e distração: Camilo Gordillo, Jenny Lombo, Manuel Arcila, Camilo Ariza, Andrés Puerto, Liz Katherine, Miguel Angel, Veronica Sierra, Diana Martinez e German Castañeda.

A Juliana Momoe e Miriam Dominguez por sua valiosa companhia e momentos de muita alegria.

A todas as pessoas que fizeram parte desse período e fizeram de minha estadia no Brasil uma experiência muito boa.

A CAPES - Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – pelo indispensável apoio financeiro.

"Não somos o que sabemos. Somos o
que estamos dispostos a aprender".
(Council on Ideas)

Resumo

Este trabalho tem como finalidade a implementação de um método de medição de ângulos de alinhamento de direção veicular, baseado em imagens estereoscópicas. São desenvolvidas soluções de processamento e análise de imagens, bem como a sua integração em um programa gerenciador da tarefa de medição dos ângulos. A implementação do programa de gerenciamento é feita utilizando os conceitos *model-view-control* (MVC) e programação orientada a objetos. Utilizam-se os pacotes de código livre *framework C++ Qt®*, *Armadillo*, *OpenCV* e *DOxygen*. São apresentados resultados de operação do sistema utilizando imagens virtuais e reais.

Palavras-chave: Alinhamento de carros, Estéreo visão, Programação Orientada a Objetos, MVC, Visão Computacional, Processamento de imagens.

Abstract

The main objective of this work is the implementation of a measuring methodology to obtain the angles of alignment of the steering mechanism of a car, based on stereoscopic images. Solutions for image processing and analysis are proposed and implemented in the form of an integrated operating software. Implementation of the software is done using the MVC (Model-View-Controller) and OOP (Object Oriented Programming) concepts. Free software packages are used, such as the *Qt*® *C++* Framework, Armadillo, OpenCV and *DOxygen.* Results from the use of the operating software are presented, using virtual and real images.

Keywords: Car alignment, Stereovision, Object Oriented Programming, MVC, Computer Vision, Image Processing.

Lista de Ilustrações

Lista de Tabelas

Lista de Abreviaturas e Siglas

Abreviações

MVC	- Model View Controller
kpi	- King Pin Inclination
MEMS	- Micro Electro Mechanical Systems
3D	- Três dimensões
HSV	- Hue Saturaion Value
RGB	- Red Green Blue
PCA	- Principal Component Analysis
POO	- Programação Orientada a Objetos
UML	- Unified Modeling Language
GUI	- Graphical User Interface

SUMÁRIO

1 INTRODUÇÃO

Câmeras de vídeo tem encontrado um crescente uso em aplicações industriais, no setor educativo, em desenvolvimentos de pesquisa, e mais recentemente, no campo do entretenimento. Os dispositivos baseados em câmeras são usados com mais frequência para medir distâncias, detectar movimento, reconstruir ambientes de navegação robótica, além de muitas outras aplicações (ALMEIDA, MARTINS e FLEURY, 2007), (BIANCHI, 2007). Novas tecnologias de fabricação e miniaturização permitem cada vez mais o acesso a dispositivos de aquisição de imagem com melhor qualidade e maior velocidade de transferência de dados. Hoje em dia, por exemplo, podem-se encontrar celulares com câmeras integradas de 12 milhões de *pixels* (XPERIA, 2012) e a tendência é de compactar um número cada vez maior de *pixels* em menores espaços físicos.

A configuração chamada de sistema estéreo (SHAPIRO e STOCKMAN, 2001, p. 446) consiste na utilização de duas ou mais câmeras, com a ajuda de ferramentas de processamento de imagens e visão computacional. Um sistema estéreo permite, a partir do número mínimo de duas imagens, obter a localização de pontos no espaço tridimensional. Ou seja, pode-se reconstruir qualquer geometria tridimensional a partir de duas imagens dessa geometria. Isso possibilita o desenvolvimento de diversos sistemas de mapeamento, utilizados em robótica (SOUZA e GONÇALVES, 2011), sistemas de medição de distâncias e reconstrução de superfícies, utilizados no campo da engenharia civil (JURJO, MAGLUTA, *et al.*, 2007) e sistemas de criação de mapas de profundidade e reconhecimento de gestos corporais, utilizados em dispositivos de entretenimento (KINECT, 2010).

Este trabalho explora as possibilidades de reconstrução e medida dos ângulos de alinhamento das rodas de um carro, utilizando sistemas de visão estereoscópicos.

O processo de alinhamento consiste em ajustar as relações entre os componentes da suspensão, da direção e das rodas do veículo. Tais relações são dadas, essencialmente, através de medidas angulares (HAGERMAN, 2011). O alinhamento veicular é importante por garantir a segurança da estabilidade de direção, evitando ainda o desgaste irregular nos pneus. Atualmente

existem diversos equipamentos para realizar alinhamento das rodas, mas tais equipamentos podem ser muito complexos ou caros. Alguns deles estão baseados no uso de fios, dispositivos complexos de fixação na roda, fitas métricas, projetores e sensores de luz, goniômetros etc... Equipamentos mais modernos baseiam-se no uso de padrões fixos nas rodas e visão computacional. Tais equipamentos, no entanto, são de alto custo e muitas vezes de difícil operação. Justifica-se, portanto, a necessidade de se desenvolver um sistema barato, não invasivo, confiável e de uso simples, que permita realizar a tarefa do alinhamento em pouco tempo e de forma precisa. No presente trabalho, as medidas dos ângulos de alinhamento são feitas utilizando apenas técnicas de visão estereoscópica e processamento das imagens das rodas. Um *software* de alinhamento de uso fácil e indutivo é aqui desenvolvido, utilizando o sistema operacional *Linux*, podendo também ser usado em *Microsoft® Windows®* e em *Mac®*. A programação usada utiliza ferramentas de *software* livre, como o *framework* Qt®, e está desenvolvida usando o paradigma de programação orientada a objetos e o padrão MVC (*Model-View-Controller*).

1.1 Trabalhos correlatos

Atualmente pode-se encontrar no mercado vários tipos de equipamento para medir os ângulos de alinhamento das rodas num carro (SNAP-ON DO BRASIL, 2004). As soluções presentes podem-se dividir em duas categorias: medidores baseados em painéis a laser e medidores baseados em sistemas digitais ou computadorizados.

Os sistemas baseados em painéis a laser são os mais baratos e tradicionais. Esses sistemas utilizam grampos ou garras presos às rodas assim como laser e um painel corrediço com marcações de escala. Na Figura 1.1 apresenta-se o grampo utilizado na roda para medição com laser.

Figura 1.1: Grampo preso na roda (EBERLIN, 2008)

A emissão laser é refletida num painel com escalas de medida colocado na frente do veículo, possibilitando a leitura da medida dos ângulos de alinhamento das rodas dianteiras com boa precisão. A Figura 1.2 apresenta o painel utilizado neste tipo de calibração. A Figura 1.3 apresenta os equipamentos, o painel e a configuração do sistema para fazer a medida dos ângulos.

Figura 1.2: Painel de calibração (FENIXCAR)

Figura 1.3: Alinhamento feito com painéis e laser. (FENIXCAR)

Uma das desvantagens deste tipo de sistema é a grande quantidade de e componentes necessárias para a sua operação. A Figura 1.4 apresenta os componentes de um sistema de alinhamento baseado no uso de *laser*. Este equipamento é comercializado no Brasil pela Fenix Car® Equipamentos Automotivos (FENIXCAR) a um custo de R$ 6.980,00 (Junho 2012). Além do preço e do grande número de peças, o seu uso não é intuitivo, necessitando que o operador realize um treinamento adicional, o que aumenta o custo final do sistema.

Figura 1.4: Componentes para alinhamento com laser (FENIXCAR)

O segundo tipo de medidores são baseados em sistemas digitais ou computadorizados. Neste grupo pode-se encontrar os que usam clinômetros baseado em *MEMS (Micro-Electro-Mechanical Systems)*. Os clinômetros tipo *MEMS* são sensores de inclinação contidos em um circuito eletrônico densamente integrado com dimensões na ordem de micras. Esses tipos de clinômetros utilizam uma massa de prova que se movimenta dependendo do ângulo de inclinação e é usado em medidores portáteis. O sensor mede ângulos devido à influência da gravidade.

Outra forma de medir os ângulos é usando visão computacional. Estes equipamentos usam processamento de imagens para localizar padrões fixados nas rodas. A maior desvantagem deste tipo de sistemas é seu alto custo. Na Figura 1.5 é apresentado um dos sistemas mais modernos o VISUALINER 3D® da empresa Snap-on® (VISUALINER) que é comercializado no Brasil pela Sun Equipamentos®. Este equipamento baseia-se em visão computacional e em processamento de imagens para detectar padrões fixados nas rodas e realizar a medição dos ângulos de alinhamento.

Figura 1.5: O sistema de alinhamento VISUALINER 3D®

Outra empresa que trabalha com equipamentos deste tipo é a BOSCH®. Esta empresa, além de comercializar alinhadores baseados em visão computacional, também comercializa balanceadores e desmontadores de rodas. O principal produto da empresa, no campo de alinhamento, é o alinhador FWA 4630 (BOSCH). A Figura 1.6 apresenta os padrões utilizados por esse produto, assim como os emissores de luz laser, dispositivo verde na imagem.

Figura 1.6: Alinhador de rodas FWA 4630

A Figura 1.8 apresenta o esquema de uso do dispositivo. A informação é apresentada na tela de um computador.

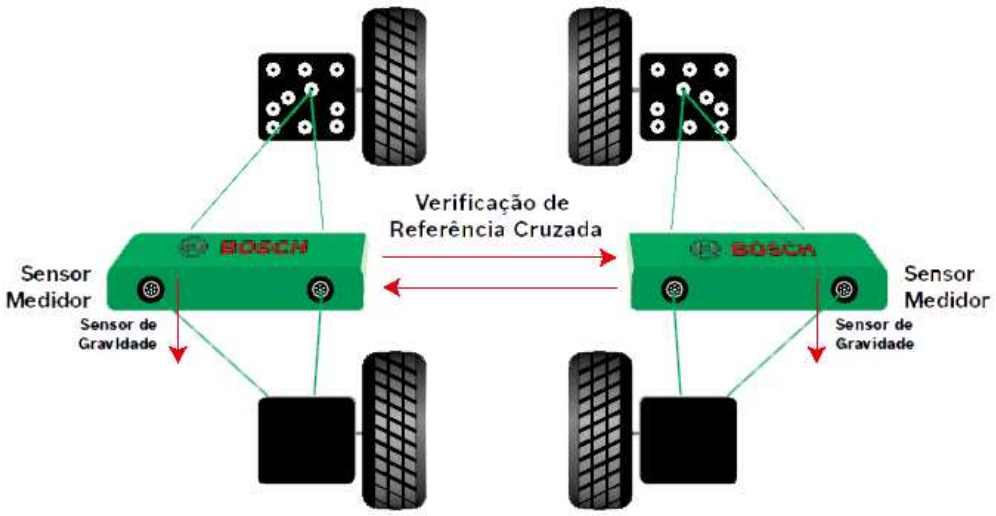

Figura 1.7: Esquema de funcionamento do alinhador FWA 4630 (BOSCH)

Uma desvantagem dos sistemas apresentados até agora é que todos utilizam padrões nas quatro rodas, isto pode causar danos na calota ou o padrão pode não se fixar bem na roda e, além disso, aumenta o número de componentes do sistema e, por conseguinte, o custo.

Um sistema mais avançado que não utiliza padrões fixos nas rodas foi apresentado para patente nos Estados Unidos no ano 2005 (GMBH, 2005). O sistema foi construído no ano 2007, mas foi comercializado por um breve período de tempo e depois retirado do mercado por razões internas. A Figura 1.8 apresenta este sistema. O conceito semelhante de sistema de baixo custo e de fácil manipulação para medição dos ângulos de alinhamento foi desenvolvido e analisado no trabalho de Carlos Mingoto (MINGOTO, 2012), originando uma patente de propriedade intelectual (KURKA e MINGOTO, 2011). A implementação do sistema idealizado por Mingoto, tornando-o mais próximo de um produto com viabilidade operacional e comercial é a principal motivação do presente trabalho, permitindo que se estabeleçam os objetivos específicos de trabalho descritos a seguir.

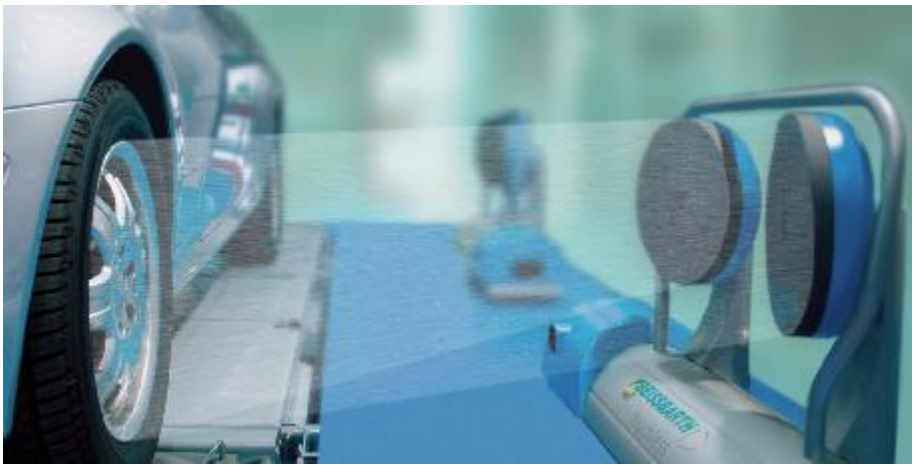

Figura 1.8: Sistema de alinhamento sem padrões nas rodas

1.2 Objetivos

O objetivo geral desse projeto é desenvolver um programa que permita a medição dos dois ângulos mais importantes de alinhamento da direção veicular, a saber: o ângulo de *toe* e o ângulo de *camber*, usando visão computacional. O programa deve ser desenvolvido usando *software* livre e de fácil uso. Nesse sentido, os objetivos específicos são:

- Desenvolver o programa usando a linguagem de programação C++ e com ferramentas de *software* livre.

- Desenhar uma interface de usuário fácil de usar, intuitiva e de fácil assimilação.

- Fazer uso de ferramentas de programação que permitam que o programa se possa executar em os Sistemas Operativos mais usados como Microsoft® Windows®, MacOS® e Linux®.

1.3 Organização do texto

O presente trabalho esta dividido em sete capítulos. No segundo capítulo são apresentados os parâmetros de alinhamento, isto é: os ângulos a serem medidos e corrigidos. O terceiro capítulo apresenta o conceito de sistema estéreo, calibração de câmeras e reconstrução tridimensional. O quarto capítulo apresenta uma visão geral do sistema desenvolvido pelo grupo de pesquisa do laboratório e em detalhe o processo de medição dos ângulos de alinhamento. São tratados diversos temas como o processo de calibração, a obtenção de pontos correlatos, a reconstrução da roda e finalmente obtenção dos ângulos de alinhamento. O quinto capítulo apresenta o *software* desenvolvido, suas etapas e como utilizá-las. O sexto capítulo apresenta a arquitetura do *software* bem como o paradigma de programação, as bibliotecas e os pacotes utilizado. O sétimo capítulo apresenta os resultados, tanto virtuais como reais, do processo de calibração e medidas dos ângulos. Finalmente no oitavo capítulo é feita uma discussão dos resultados, e são apresentadas sugestões para futuros trabalhos.

2 PARÂMETROS DE ALINHAMENTO

Os sistemas de alinhamento são dispositivos utilizados para medir e modificar as relações dos componentes envolvidos no sistema de direção e suspensão no carro. Em primeiro lugar para realizar o alinhamento do carro é necessário medir os chamados ângulos de alinhamento.

No processo de alinhamento são medidos, principalmente, três ângulos: o *toe* ou ângulo de convergência/divergência, o *camber* ou ângulo de cambagem, e o *caster*. O processo de alinhamento consiste em verificar e corrigir os valores destes ângulos para que estejam entre os limites fornecidos pelo fabricante.

Nas seções seguintes descrevem-se detalhadamente os três ângulos, bem como a sua importância no processo de alinhamento.

2.1 Ângulos de alinhamento

A Figura 2.1 apresenta a roda de um carro com os três ângulos mais importantes, a saber, o *toe*, o *camber* e o *caster*. Para cada um destes ângulos define-se uma região positiva e outra negativa.

Figura 2.1: Ângulos mais importantes no alinhamento

2.1.1 Ângulo de convergência/divergência (*toe*)

Este é o ângulo que forma a roda dianteira com a linha de movimento do carro visto de cima. A linha de movimento é definida pelas rodas traseiras. Este ângulo é apresentado na Figura 2. com a cor azul. A Figura 2.2 apresenta este ângulo como α_1e α_2. Um desvio muito grande neste ângulo resulta um desgaste prematuro do pneu além de puxar o carro para um dos lados, diminuindo a estabilidade e tornando mais difícil a dirigibilidade do veículo.

Têm-se três casos para o ângulo α em cada roda: se o eixo de movimento da roda dianteira é paralelo ao eixo de movimento da roda traseira, então a roda está alinhada e o ângulo de *toe* é zero, Figura 2.2.a. Se o eixo de movimento da roda dianteira não é paralelo ao eixo de movimento do carro então a roda possui convergência (*toe* positivo), conforme ilustrado na Figura 2.2.b, ou divergência (*toe* negativo), conforme ilustrado na Figura 2.2.c.

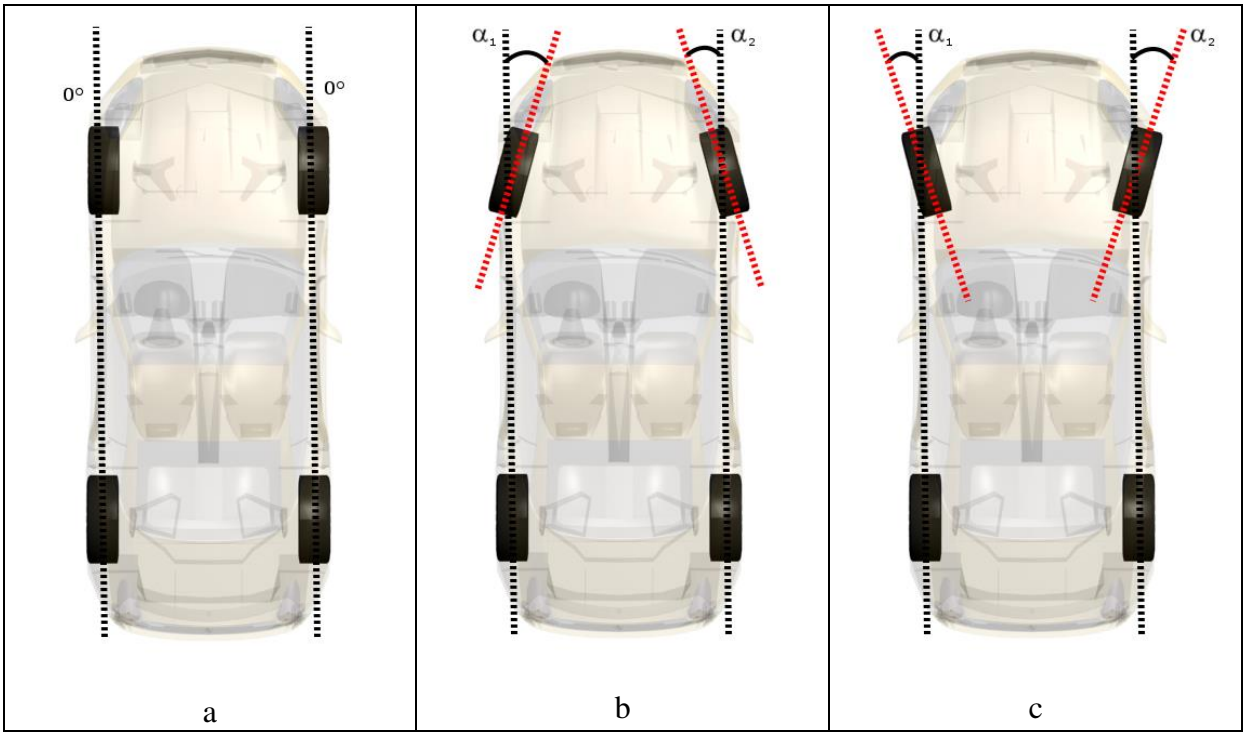

Figura 2.2: Ângulo de *toe*

Outra maneira de conhecer se as rodas dianteiras têm convergência ou divergência é pela diferença de distâncias entre os centros da parte anterior e posterior das rodas vista desde cima. A Figura 2.3 apresenta este tipo de medida. Os pontos A_1 e A_2 representam os pontos médios da parte anterior das rodas; os pontos B1 e B2 representam os pontos médios da parte posterior. Assim a diferença entre os segmentos de reta $\overline{A_1A_2}$ e $\overline{B_1B_2}$, fornce uma estimativa do grau de convergência ou divergência das rodas. Se a diferença é positiva então as rodas divergem, se a diferença é negativa, então as rodas convergem, se a diferença é zero, então as rodas estão alinhadas.

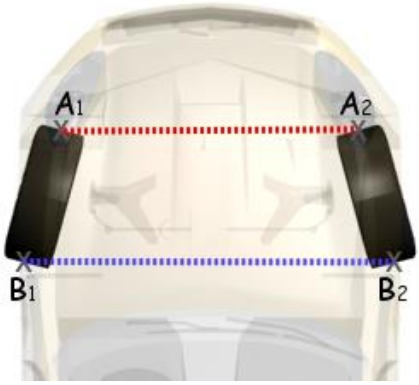

Figura 2.3: Medida da convergência/divergência por distâncias.

Os fabricantes de carros recomendam que, dependendo do tipo de tração do carro, o processo de alinhamento deve deixar um pequeno ângulo de *toe* quando o carro está em repouso. No caso do carro possuir tração dianteira é recomendável deixar um pequeno ângulo de divergência devido a que, em marcha, as rodas dianteiras tendem a convergir. No caso do carro de tração traseira, as rodas tendem a divergir quando o veículo está em marcha, por isso é recomendável deixar um pequeno ângulo de convergência.

2.1.2 Ângulo de *camber*

O ângulo de *camber* é o ângulo formado quando a roda, vista de frente, está inclinada com respeito a uma linha vertical. O ângulo de *camber* pode ser positivo, negativo ou nulo. Na Figura 2.4 apresentam-se as três possibilidades. Quando a roda possui ângulo de *camber* zero, Figura 2.4.a, o pneu não apresenta desgaste irregular. Quando a roda possui ângulo de *camber* negativo, Figura 2.4.b, existe a tendência de um desgaste na parte interna do pneu. Quando a roda possui *camber* positivo, Figura 2.4.c existe a tendência de um desgaste na parte externa do pneu. O ângulo de *camber* nulo proporciona estabilidade ao carro, principalmente nas curvas.

O ângulo de inclinação de *camber* nos veículos em geral, não possui regulagem de ajuste, o que significa que é difícil de alterar. Tal inclinação se apresenta como consequência de uma

colisão ou solavancos fortes. Para corrigir este ângulo são necessários procedimentos de desentortamento da suspensão, executados por ferramentas e operadores especializados.

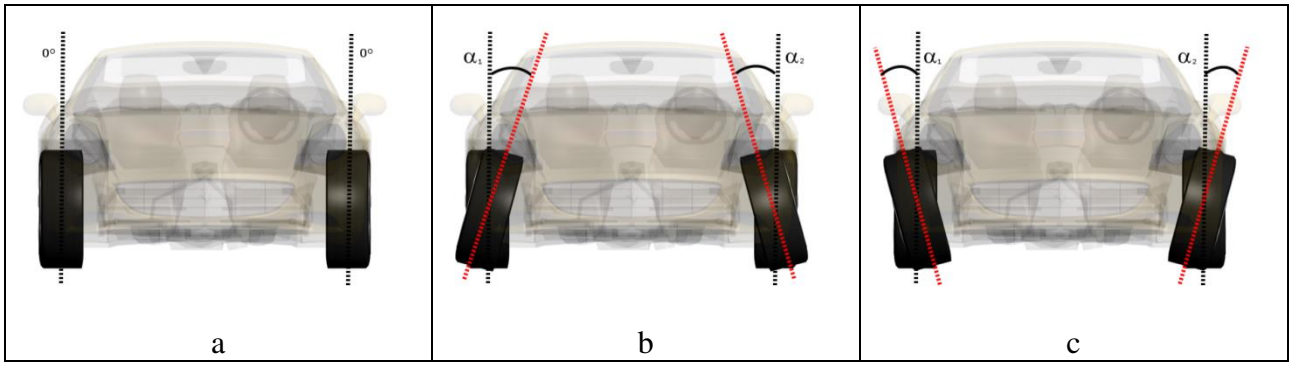

Figura 2.4: Ângulo de *camber*

Na Figura 2.5 apresenta-se um pneu com desgaste causado por um ângulo de *camber* grande. Este tipo de desgaste não permite que o pneu tenha o máximo de contato com o chão, tornando o carro instável nas curvas.

Figura 2.5: Pneu com desgaste devido a *camber*.

2.1.3 Ângulo de *caster*

O ângulo de *caster* é obtido olhando a roda de lado. É o ângulo formado entre a linha vertical e a linha gerada pelos pontos de ancoragem da suspensão. Este ângulo proporciona firmeza na direção em linha reta e ajuda a retornar as rodas na posição certa após uma curva.

A Figura 2.6.a apresenta a roda com um ângulo de *caster* nulo. A Figura 2.6.b apresenta a roda com ângulo de *caster* negativo. Um ângulo de *caster* nulo ou negativo é indesejado devido à instabilidade causada no sistema direcional do veículo.

A Figura 2.6.c apresenta um ângulo de *caster* positivo o que é geralmente desejável. Um ângulo de *caster* levemente positivo permite um "ataque" da roda aos obstáculos, porém um ângulo positivo exagerado causa um volante pesado e difícil de manipular.

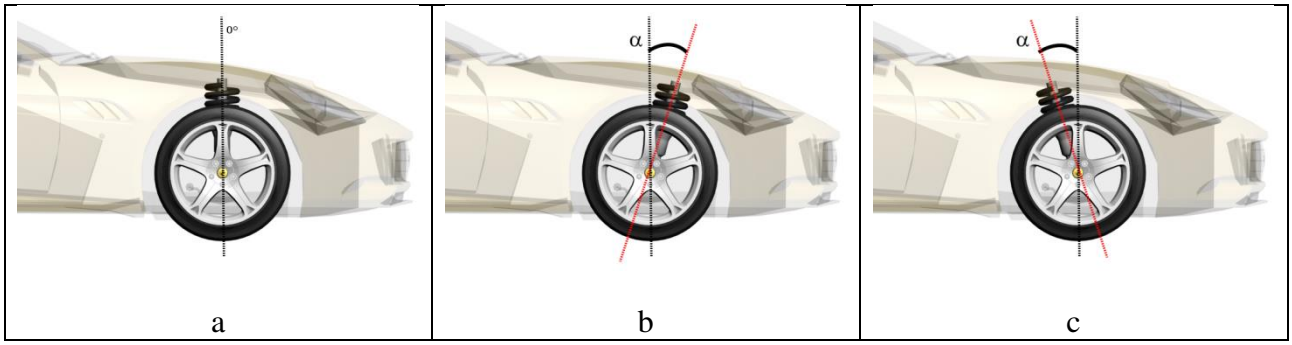

Figura 2.6: Ângulo de *caster*

O sistema de alinhamento apresentado neste trabalho realiza a medida de dois dos três ângulos apresentados nesta seção: o ângulo de *toe* e o ângulo de *camber*, pois estes ângulos são obtidos diretamente com o contorno da roda.

2.1.4 Sequência de medição dos ângulos e obtenção do *caster*

Devido ao fato de que o sistema consegue medir os dois ângulos mais importantes, é necessário fazer um procedimento para obter o valor pratico do terceiro ângulo: o *caster*, baseado na medida da cambagem depois um pequeno e arbitrário esterçamento da roda Em (KURKA e MINGOTO, 2011) e (JANUARY, 2007) é descrita a sequência de medidas e o processo de obtenção do ângulo de *caster*.

2.2 Correção dos ângulos de alinhamento

Após a obtenção das magnitudes de cada ângulo é necessária a correção dos mesmos. Esta correção necessita ferramentas especiais e conhecimentos do sistema de suspensão do carro. O procedimento realizado por um mecânico consiste em ajustar os braços de direção de cada uma das rodas dianteiras. Os braços de direção são barras com comprimento ajustável que vão desde a caixa de direção até a roda e que são ajustáveis por meio de porcas. Os braços de direção possuem dois terminais, cada uma num extremo da barra. Estes terminais contêm porcas que permitem fazer o ajuste do comprimento da barra.

É importante lembrar que os ângulos de *toe* e de *camber* não podem ser zero quando o carro está estático. Eles devem possuir um valor pequeno que permita que, em movimento, os ângulos de *toe* e de *camber* fiquem próximos de zero.

No capítulo seguinte apresenta-se o conceito de visão estéreo assim como a calibração de câmeras e a realização da reconstrução tridimensional.

3 SISTEMAS ESTÉREOS

A habilidade do cérebro de encontrar as diferenças (*disparidade*) entre as imagens de um mesmo objeto, observado pelos dois olhos, permite que se tenha uma noção precisa das dimensões de largura, altura e profundidade de tal objeto. A concepção e implementação de algoritmos que imitem essa habilidade visual é conhecido como visão estéreo (FORSYTH e PONCE, 2002, p. 336).

A visão estéreo é amplamente usada em tarefas como a navegação visual de robôs, cartografia, reconhecimento aéreo e fotometria (Ibid., p. 337).

Para se obter a profundidade de um ponto no espaço usando duas câmeras, é necessário possuir informações sobre a posição de cada uma das câmeras. O primeiro passo é determinar, através de calibração os parâmetros intrínsecos (internos) e extrínsecos (externos) de cada câmera. O segundo passo é determinar as coordenadas da projeção em cada imagem do mesmo ponto no espaço. Isso é chamado de problema de correlação. Por último é necessário localizar o ponto onde ocorre a intersecção de dois raios e calcular as coordenadas tridimensionais desse ponto.

A Figura 3. apresenta um sistema estéreo. O ponto ***P*** no espaço é projetado como *p* na câmera um e *p'* na câmera dois. O ponto *O* é o centro óptico (foco) da câmera um e o ponto *O'* é o foco da câmera dois.

Na seção seguinte apresenta-se o processo de calibração das câmeras. Na seção 3.2 é apresentado o problema de correlação de pontos e o algoritmo de reconstrução tridimensional.

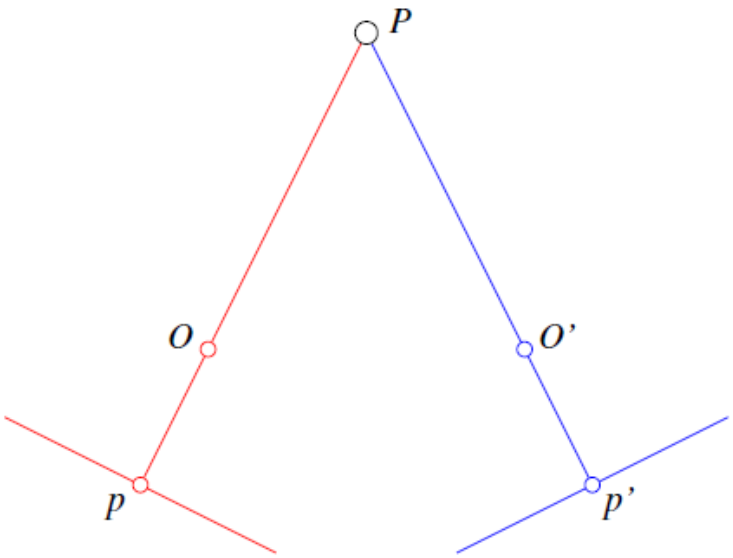

Figura 3.1: Sistema estéreo (FORSYTH e PONCE, 2002)

3.1 Algoritmo de calibração de câmeras.

A calibração de câmeras é o processo que busca estimar os valores dos parâmetros intrínsecos e extrínsecos do modelo da câmera. A ideia principal detrás da calibração é obter os valores das equações que ligam um conjunto de coordenadas 3D conhecidas e suas projeções na imagem (TRUCCO e VERRI, 1998, p. 124)

Vários são os algoritmos de calibração existentes, apresentados em diversos trabalhos como os de Ahmed (AHMED, HEMAYED e FARAG, 1999), Batista (BATISTA, 1996) (BATISTA, ARAÚJO e DE ALMEIDA, 1998) e Zhang (ZHANG, 2000). Uma característica comum de tais métodos é o emprego de objetos padrão de calibração como caixas ou tabuleiros quadriculados. Mais recentemente desenvolveram-se também algoritmos de calibração que não precisam de padrões (FAUGERAS, QUAN e STRUM, 2000). No entanto um método de calibração robusto, de uso popular, é o descrito no artigo (TSAI, 1987). Nesse método, um tabuleiro plano e quadriculado de dimensões conhecidas é usado como padrão. Na presente dissertação utiliza-se uma variação de um método de calibração desenvolvido no laboratório de

Processamento de Sinais da Faculdade de Engenharia Mecânica, que utiliza como padrão uma caixa comum. Tal método foi fruto de artigo aceito para publicação no *Journal of the Brazilian Society of Mechanical Sciences and Engineering* (KURKA, DELGADO, *et al.*, 2012). A variação do método de calibração proposta neste trabalho consiste em substituir a caixa comum por uma caixa vazada com esferas de diferentes cores em cada vértice, conforme apresentado na Figura 3.2.

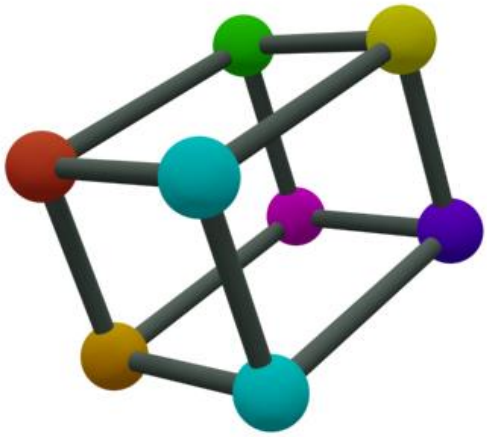

Figura 3.2: Caixa vazada de calibração

Os dados de entrada para o algoritmo de calibração são as dimensões da caixa e as coordenadas de cada um dos vértices na imagem.

3.1.1 Algoritmo de procura de círculos

Nesta seção apresenta-se o algoritmo utilizado para obter a localização das projeções das esferas na imagem RGB da caixa mostrada na Figura 3.2. Esta informação é usada para localizar de maneira mais precisa os vértices e arestas da caixa. O procedimento desenvolvido aqui, para

localização das projeções das esferas, foi fruto de trabalho apresentado no Terceiro Simpósio de Processamento de Sinais (ROJAS e KURKA, 2012).

O primeiro passo depois da obtenção da imagem da câmera em modelo RGB é convertê-la ao modelo HSV (*Hue-Saturation-Value*) ou Tonalidade-Saturação-Valor. O modelo de cores HSV fornece uma representação intuitiva da cor, aproximando-a da forma como os humanos a percebem e a manipulam (MANJUNATH, 2001). A conversão do modelo RGB ao HSV não é linear, mas é reversível.

Uma representação do modelo HSV é uma pirâmide hexagonal invertida, como ilustrado na Figura 3.3, utilizando coordenadas cilíndricas onde o eixo vertical representa o valor (V) da cor. A tonalidade (H) é definida como um ângulo entre 0 e 360°. A saturação (S) é o grau de pureza da cor, e é medida como a distância radial desde o eixo central até a superfície externa da pirâmide, atingindo valores desde 0 até 1 (SURAL, 2002).

A vantagem de usar o padrão HSV, ao invés do RGB, é que o canal H contém a informação da cor, independentemente da luminosidade da cena capturada na imagem.

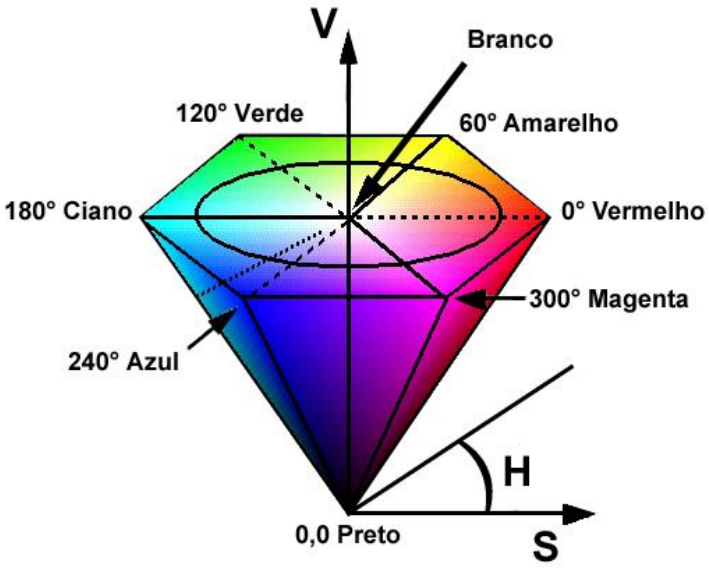

Figura 3.3: Representação geométrica do modelo HSV.

O processo de conversão RGB/HSV é feito de acordo com os critérios abaixo.

Seja $MAX = \max\{R, G, B\}$ o valor máximo entre as componentes R, G e B de um *pixel* e seja $MIN = \min\{R, G, B\}$ o valor mínimo das componentes R, G e B do mesmo *pixel*. Assim, a conversão de um *pixel* do sistema RGB ao sistema HSV é obtida de acordo com as Equações (3.1), (3.2) e (3.3)

$$H = \begin{cases} não\ definido & se\ MAX = MIN \\ 60° \times \dfrac{G - B}{MAX - MIN} + 0° & se\ MAX = R\ e\ G \geq B \\ 60° \times \dfrac{G - B}{MAX - MIN} + 360° & se\ MAX = R\ e\ G < B \\ 60° \times \dfrac{B - R}{MAX - MIN} + 120° & se\ MAX = G \\ 60° \times \dfrac{R - G}{MAX - MIN} + 240° & se\ MAX = B \end{cases}, \tag{3.1}$$

$$S = \begin{cases} 0 & se\ MAX = 0 \\ 1 - \dfrac{MIN}{MAX} & em\ outro\ caso' \end{cases} \tag{3.2}$$

$$V = MAX. \tag{3.3}$$

A imagem no modelo HSV é separada em cada um de seus componentes, assim tem-se o canal H, canal S e canal V em imagens separadas. É aplicada uma função de umbral (*threshold*) nos canais S e V. Eliminam-se, do canal S, as regiões muito brancas da imagem, e, no canal V, eliminam-se as regiões muito pretas.

As duas imagens obtidas do passo anterior são unidas por meio da operação binaria AND, onde o resultado é uma imagem binaria que é multiplicada com a imagem do canal H. A imagem resultante da multiplicação está em tons de cinza e contém a informação das cores que possuem valores de saturação alta. São feitas sete operações de umbral nesta imagem, uma para cada cor

procurada, os limites do umbral são os valores mínimos e máximos de cada cor no canal H. Assim, por exemplo, para procurar as regiões de cor amarela, da Figura 3.3, o valor mínimo da operação de umbral é de aproximadamente 45° e o valor máximo é de aproximadamente 75°. Depois de feita a operação de umbral com os valores de todas as cores, são obtidas sete imagens binarias, uma imagem para cada cor. Para cada imagem obtida do passo anterior é utilizado um algoritmo de procura de regiões conectadas (*blobs*) onde são separadas as diferentes zonas da imagem. Uma filtragem onde são excluídas regiões muito pequenas também é feita neste passo. Em cada uma das sete imagens obtém-se um número diferente de *blobs,* e cada um desses *blobs* serve como entrada para o passo seguinte do processamento. Um fluxograma do algoritmo de procura dos círculos é apresentado na Figura 3.4.O algoritmo de identificação de círculos é descrito na seção 3.1.1.1. Finalmente obtêm-se os valores dos raios e as coordenadas dos centros das regiões que são classificadas como círculos.

.

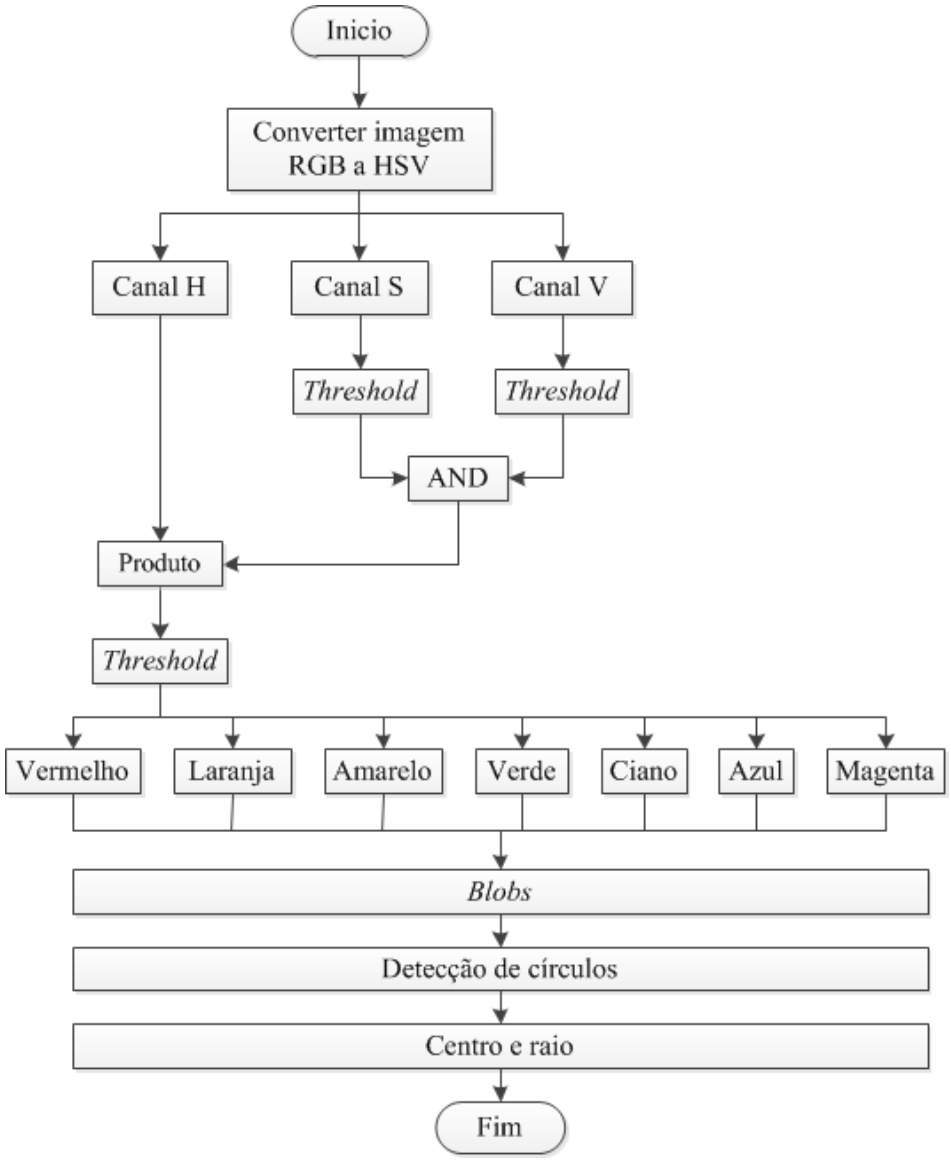

Figura 3.4: Algoritmo de procura de círculos

3.1.1.1 Algoritmo de identificação de círculos.

O algoritmo de identificação de círculos é desenvolvido devido à necessidade de classificar a um conjunto de *pixels* conectados (*blob*) como círculos ou não.

A entrada do algoritmo são os *blobs* obtidos após aplicar o algoritmo de procura de elementos conectados em cada uma das sete imagens. Para cada *blob* aplica-se um algoritmo de procura de bordas. O algoritmo de Canny (CANNY, 1986) é usado no presente caso. Em seguida são obtidas as coordenadas *i* e *j* de cada *pixel* branco na imagem. As coordenadas são armazenadas em dois vetores, coluna $\boldsymbol{a}$ e $\boldsymbol{b}$. Depois é calculado o centroide (média aritmética simples) da região de acordo com a Equação 3.4

$$\bar{a} = \frac{1}{n}\sum_{i=1}^{n} a_i, \qquad \bar{b} = \frac{1}{n}\sum_{i=1}^{n} b_i. \tag{3.4}$$

Em seguida é calculada a distância entre o centroide e cada um dos *pixels* do seu perímetro, equação 3.5. A distância média é calculada, de acordo com a equação 3.6

$$d_i = \sqrt{(a_i - \bar{a})^2 + (b_i - \bar{b})^2}, \tag{3.5}$$

$$\bar{d} = \frac{1}{n}\sum_{i=1}^{n} d_i. \tag{3.6}$$

Finalmente é calculada a variância das distâncias obtidas, conforme expresso na Equação 3.7, e se o valor é menor do que um limite arbitrado de 10, o algoritmo determina que a região é um círculo, e calcula o seu raio $\bar{d}$ e o centro $\bar{a}, \bar{b}$

$$s^2 = \frac{1}{n}\sum_{i=1}^{n} (d_i - \bar{d})^2. \tag{3.7}$$

O diagrama do fluxo do algoritmo é apresentado na Figura 3.5.

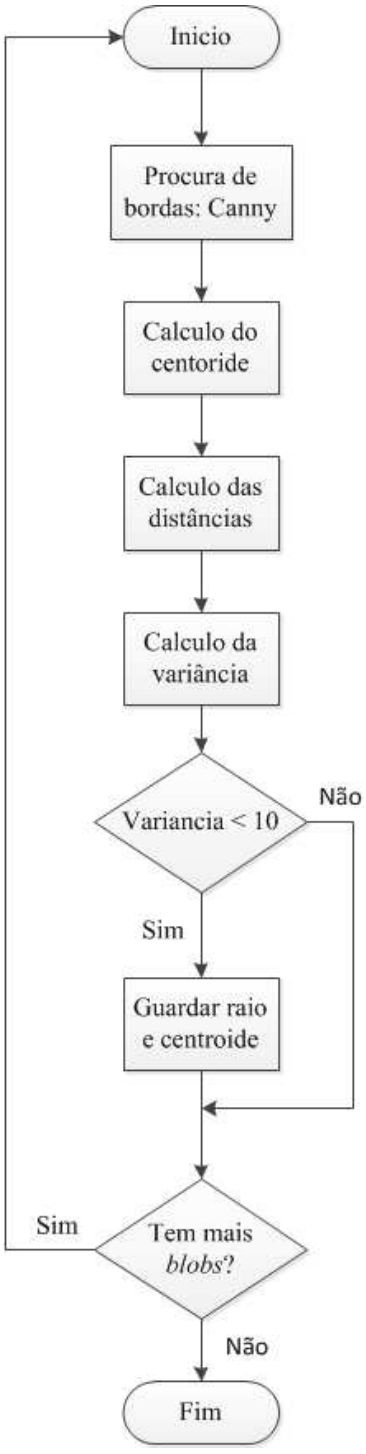

Figura 3.5: Algoritmo de identificação de círculos.

3.1.2 Algoritmo de procura dos vértices usando os lados da caixa

Pode acontecer que o centro de um círculo, como foi calculado na seção anterior, não seja a localização exata do vértice da caixa. Por conseguinte é necessário complementar a informação utilizando as arestas ou lados da mesma. Os centros dos círculos obtidos são utilizados para estimar a localização das barras que os unem e restringir sua procura na imagem. Nesta seção é apresentado o algoritmo que procura as barras que unem as esferas e formam os lados da caixa.

O primeiro passo é determinar os centros de duas esferas que estejam conectadas através de uma aresta da caixa. Os algoritmos de procura de círculos e de arestas são testados com a imagem da Figura 3.6. Nesta imagem a caixa é modelada num entorno virtual com um fundo e com uma superfície horizontal, tornando assim, o teste mais próximo da realidade.

A explicação do algoritmo de obtenção das arestas da caixa é apresentada para uma única aresta, mas é o mesmo aplicado para as 11 demais arestas.

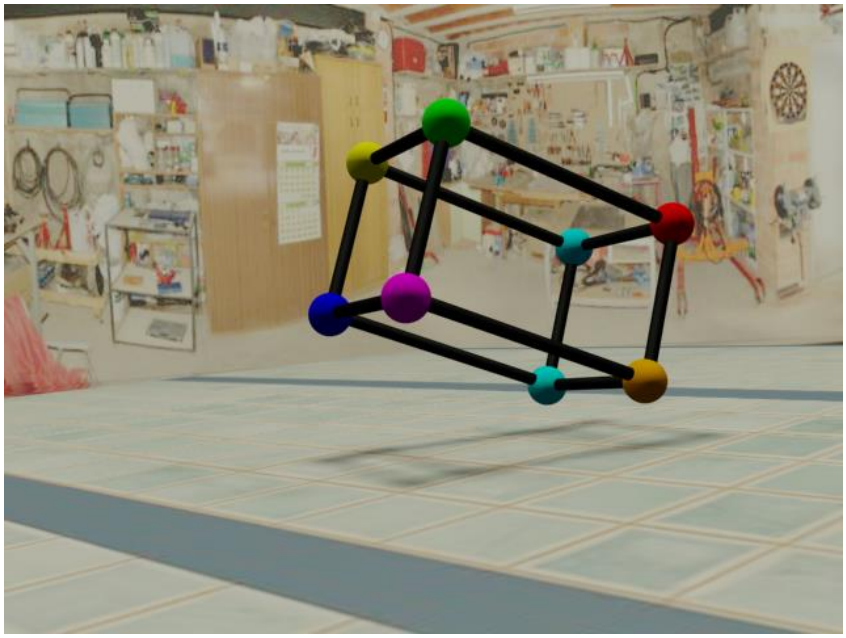

Figura 3.6: Imagem virtual para teste dos algoritmos.

A Figura 3.7 apresenta a imagem de entrada em escala de cinza e a localização dos centros das esferas (em branco) obtidas usando o algoritmo da Figura 3.4.

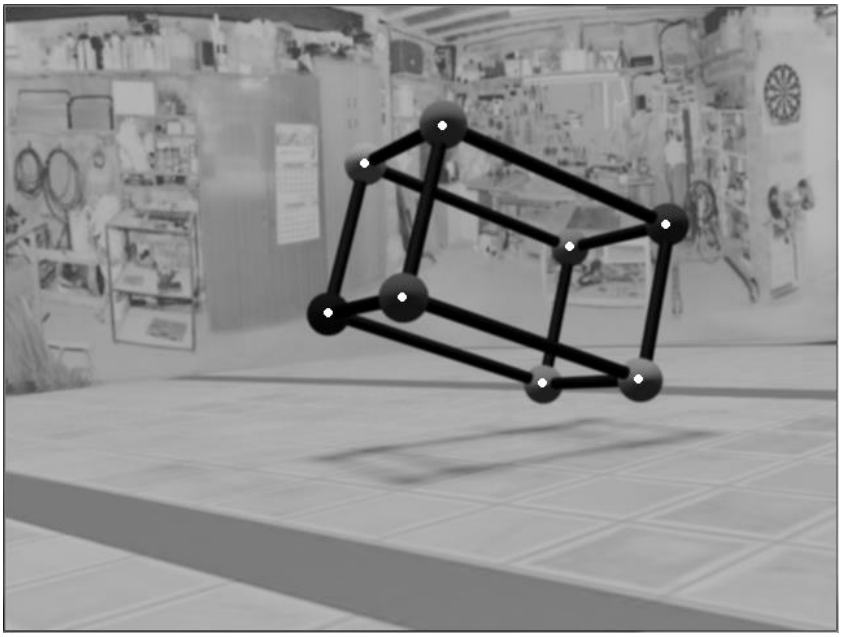

Figura 3.7: Imagem virtual com os centros dos círculos identificados.

As arestas da caixa devem ser pretas pois assim podem ser facilmente segmentadas utilizando a informação do canal V. O canal V, no modelo de cores HSV, possui informação sobre a obscuridade da cor. Quanto mais baixo é o seu valor, mais escura é a cor. Assim pode-se realizar uma operação de umbral no canal V e obter os *pixels* mais escuros da imagem. A Figura 3.8 mostra o resultado de aplicação deste umbral. Se o valor do *pixel* é maior do que 31 então o seu valor é estabelecido em 255. Em seguida é aplicado o algoritmo de procura de bordas de Canny à imagem resultante. A Figura 3.9 mostra a detecção das bordas da imagem da Figura 3.8.

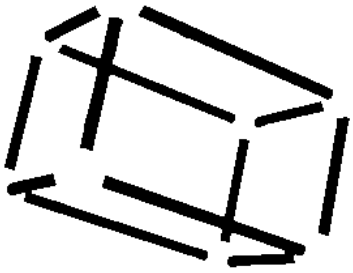

Figura 3.8: Operação de umbral no canal V para um valor de V > 31.

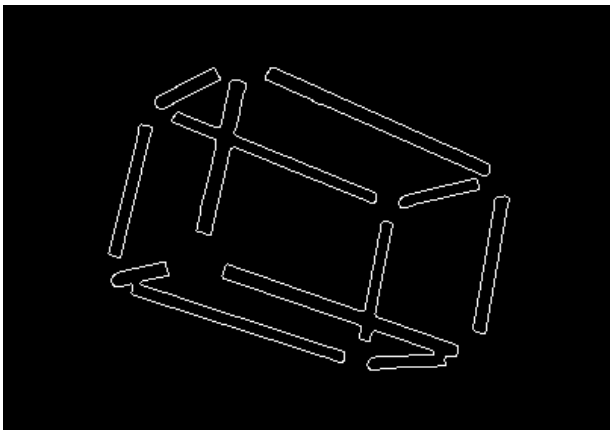

Figura 3.9: Bordas das arestas da caixa.

Para este exemplo calcula-se a equação da reta que une a esfera de cor vermelha à esfera de cor verde da Figura 3.6. É utilizada a informação dos dois centros para calcular a equação da reta $j = m * i + b$ de acordo com a Equação 3.8

$$m = \frac{j_2 - j_1}{i_2 - i_1}, \qquad b = j_1 - m * i_1. \tag{3.8}$$

A Tabela 3. mostra os valores dos centros dos círculos assim como os coeficientes da reta que passa por eles. A Figura 3.10 apresenta a linha obtida.

Tabela 3.1: Dados da reta estimada.

	Vermelho	Verde
i	508	337
j	167	91
m	0,44	
b	-56,52	

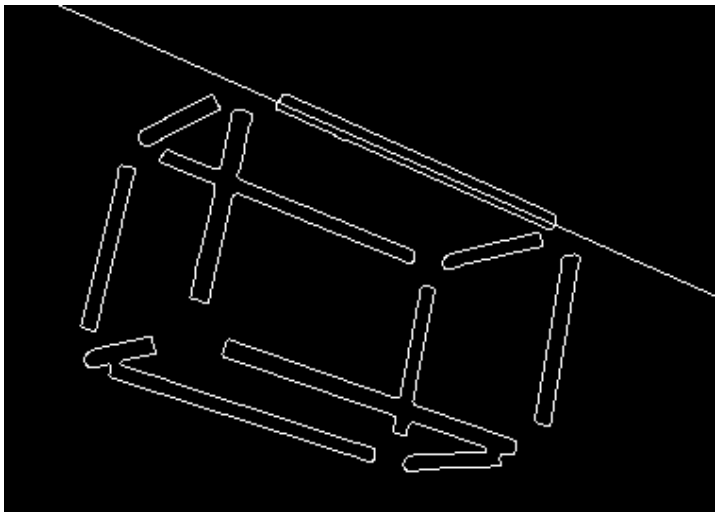

Figura 3.10: Reta que une os centros das esferas vermelha e verde

O algoritmo para encontrar as bordas superior e inferior da aresta procurada é explicado através da Figura 3.11. O ponto vermelho representa o centro do circulo vermelho. A linha pontilhada representa a reta estimada que une os centros dos circulos vermelho e verde. Os quadrados azuis representam pontos sobre a reta estimada que estão separados uma distância Δx. A partir de cada quadrado azul é realizado um deslocamento para acima e para abaixo, representado pelos pontos verdes. As coordenadas do primeiro *pixel* branco localizado, representados por triângulos vermelhos, são armazenadas em dois vetores, um para os pontos da linha superior e outro para os pontos da linha inferior.

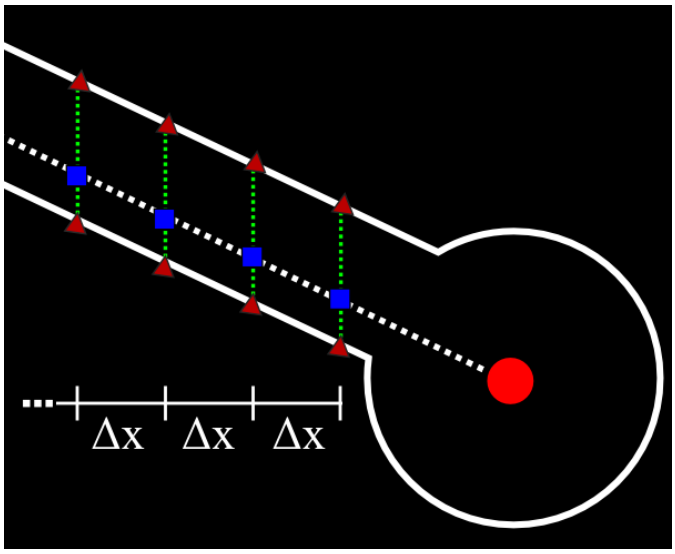

Figura 3.11: Procura das retas superior e inferior

Após obter as coordenadas dos pontos é realizada uma regressão linear para obter a linha que melhor se ajuste a cada conjunto de dados. Assim por exemplo as coordenadas *i* e *j* dos pontos na linha superior são armazenadas nos vetores $\boldsymbol{a}$ e $\boldsymbol{b}$ respectivamente.

Tem-se um número *n* de pontos, então os coeficientes *m* e *b* da reta $j = m * i + b$ são calculados a partir da Equação 3.9 e Equação 3.10

$$m = \frac{n * \sum_{i=1}^{n} a_i b_i - \sum_{i=1}^{n} a_i \sum_{i=1}^{n} b_i}{n * \sum_{i=1}^{n} (a_i^2) - \left(\sum_{i=1}^{n} a_i\right)^2}, \tag{3.9}$$

$$b = \frac{\sum_{i=1}^{n} b_i - m * \sum_{i=1}^{n} a_i}{n}. \tag{3.10}$$

Depois de calcular os coeficientes da linha superior e inferior calculam-se os coeficientes da linha média. Equação 3.11 e Equação 3.12

$$\overline{m} = \frac{m_1 + m_2}{2}, \tag{3.11}$$

$$\bar{b} = \frac{b_1 + b_2}{2}. \tag{3.12}$$

A Figura 3.12 apresenta, na cor vermelha, a linha estimada que une os centros dos círculos vermelho e verde. A linha superior (m_1, b_1) e a linha inferior (m_2, b_2), da aresta procurada, são mostradas em cor verde. Finalmente a linha média $(\bar{m}, \bar{b})$ é mostrada em cor azul. Os coeficientes da linha média são armazenados para serem usados para determinar a posição dos vértices da caixa.

Figura 3.12: Linha azul que une os círculos vermelho e verde usando a informação do aresta da caixa.

A Figura 3.13 apresenta a imagem resultante após aplicar o mesmo processo de calculo dos coeficientes da reta que une a esfera de cor vermelha a de cor laranja.

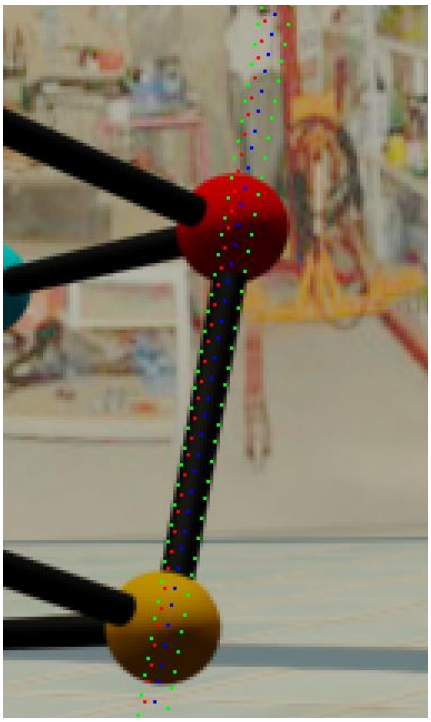

Figura 3.13: Linha resultante azul que une o circulo de cor vermelha e laranja.

Em seguida é calculada a intersecção da linha azul da Figura 3.12 com a linha azul da Figura 3.13. O ponto onde as duas retas se interceptam é uma localização mais precisa do vértice da caixa onde esta a esfera de cor vermelha. A Figura 3.14 apresenta em cor azul a localização do vértice obtido usando só a informação do círculo e em cor vermelha usando a informação das arestas da caixa. Pode-se verificar uma melhor localização dos vértices da caixa usando a informação adicional das arestas da mesma.

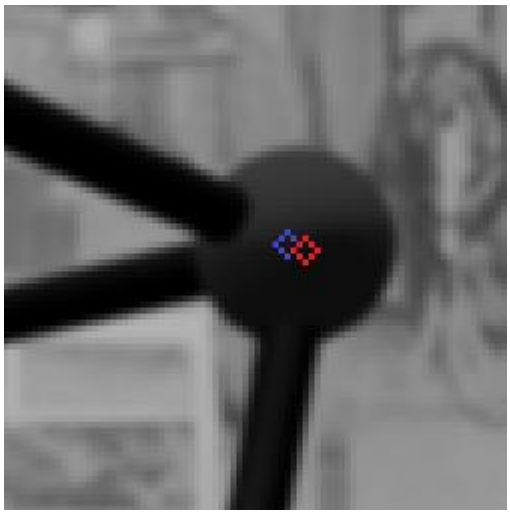

Figura 3.14: Em azul, o vértice calculado usando a informação do círculo, em vermelha, o mesmo vértice usando a informação dos lados da caixa.

Finalmente a Figura 3.15 apresenta a caixa de calibração com os vértices localizados. Com as coordenadas dos vértices da caixa e com as dimensões reais dela é feita a calibração usando o algoritmo proposto em (KURKA, DELGADO, *et al.*, 2012)

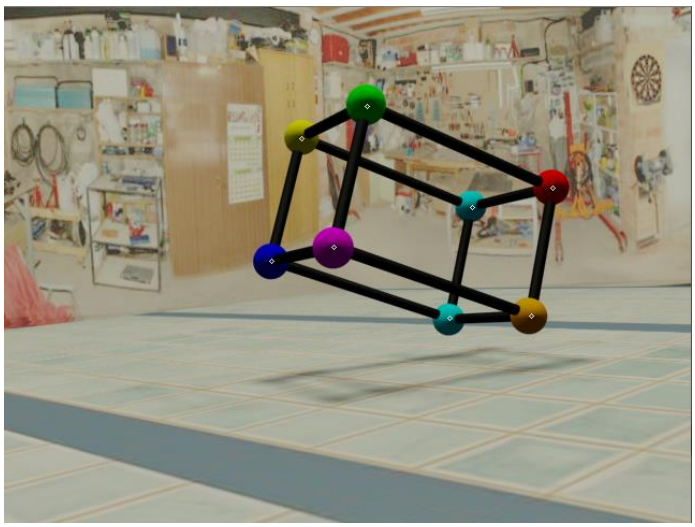

Figura 3.15: Caixa de calibração com os vértices localizados.

Por meio do algoritmo de calibração pode-se obter a chamada matriz de projeção. A matriz de projeção é uma matriz de três linhas e quatro colunas que contém os parâmetros intrínsecos e extrínsecos da câmera e permite obter as coordenadas de imagem de um ponto no espaço. A Equação 3.13 apresenta esta matriz

$$\begin{bmatrix} \rho i \\ \rho j \\ \rho \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \tag{3.13}$$

3.2 Reconstrução tridimensional

Com a informação obtida no processo de calibração e conhecendo as coordenadas 3D de um ponto qualquer do sistema de coordenadas globais, é possível determinar as coordenadas onde o ponto será projetado na imagem, Equação 3.13. A reconstrução 3D procura fazer o oposto: conhecer as coordenadas de um ponto no espaço a partir da sua projeção na imagem. Já que isso não é possível por falta de informação de profundidade, é necessário obter as projeções do mesmo ponto de duas imagens desde que, sejam posições diferentes e que depois seja realizado um processo de triangulação.

O problema de reconstrução se divide, então, em duas etapas, na primeira etapa se devem localizar os pontos correlatos, ou seja, as projeções de um ponto no espaço em duas imagens tomadas de posições diferentes. A segunda etapa é determinar as coordenadas 3D do ponto por meio de uma triangulação.

Na seção seguinte apresenta-se o algoritmo de procura de pontos correlatos.

3.2.1 Procura de pontos correlatos.

O primeiro passo do algoritmo de reconstrução é localizar os pontos **x** e **x'**, Figura 3.16, chamados pontos correlatos. Esses pontos na imagem são a projeção do mesmo ponto **X** no espaço. Neste capitulo apresenta-se o algoritmo de procura de pontos correlatos usando a geometria epipolar.

A geometria epipolar permite relacionar os focos das duas câmeras, **C** e **C'**, um ponto no espaço **X** e sua projeção, **x** e **x'** em cada uma das imagens. Os pontos **e** e **e'** são chamados os epipolos das duas câmeras. O epipolo **e'** é a imagem (virtual) do foco **C** da primeira câmera na imagem da segunda câmera. Igualmente o epipolo **e** é a imagem (virtual) do foco **C'** da segunda câmera na imagem da primeira.

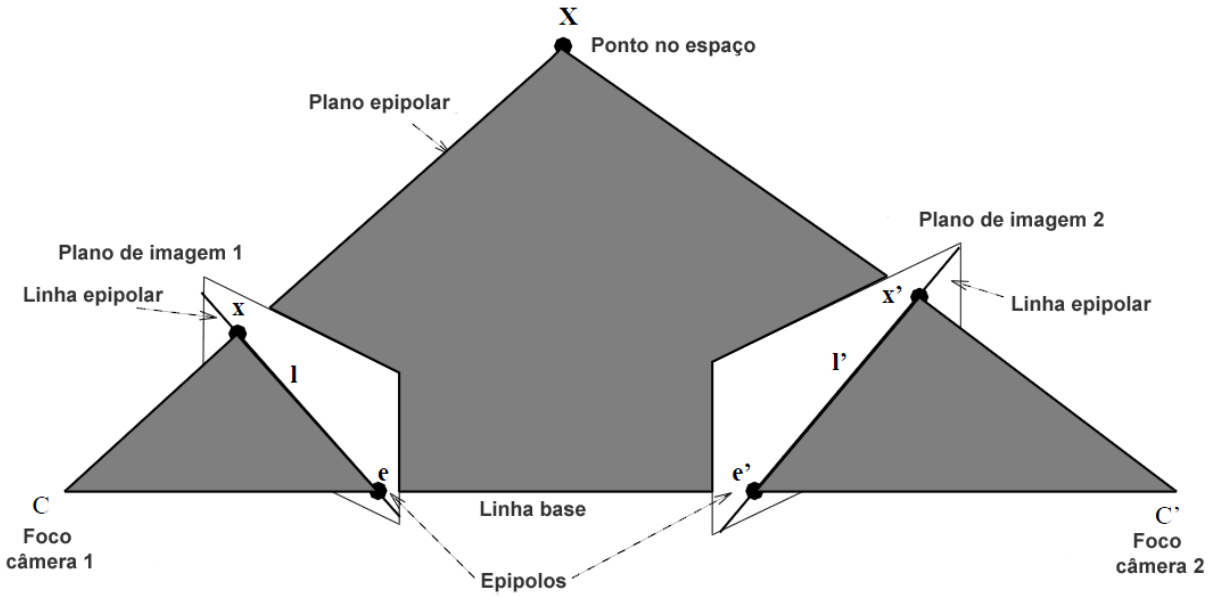

Figura 3.16: Geometria epipolar (HUYNH, 2011)

Se **x** e **x'** são imagens do mesmo ponto **X**, então **x'** pertence à linha epipolar **l'** associada com **x**. Esta restrição é fundamental na visão estéreo.

As linhas epipolares podem ser usadas para localizar pontos correlatos nos sistemas de visão estéreos. As linhas são obtidas a partir dos parâmetros de calibração do sistema estéreo. Na Figura 3.17 têm-se duas imagens de um sistema estéreo calibrado. Utilizando as matrizes de projeção e os epipolos pode se obter, para cada ponto sobre a imagem da esquerda, uma linha epipolar correspondente na imagem da direita. Na Figura 3.18 é mostrada a roda um e um ponto sobre ela, assim como a linha epipolar gerada na imagem da direita. O ponto correlato na imagem da direita localiza-se sobre a linha epipolar gerada, fornecendo uma menor região de procura do ponto na imagem.

Figura 3.17: Imagens obtidas de um sistema estéreo.

Figura 3.18: Ponto azul sobre a roda da imagem esquerda que gera uma linha epipolar branca na imagem direita.

O algoritmo de localização dos pontos correlatos é apresentado na secção 4.4. Após se obter os pontos correlatos é necessário fazer uma triangulação para calcular as coordenadas do ponto no espaço. Na próxima seção discute-se o problema de triangulação.

3.2.2 Triangulação e obtenção das coordenadas no espaço

Dado um sistema estéreo com câmeras calibradas e com dois pontos coincidentes **p** e **p'**, Figura 3.19, o problema da reconstrução recai na localização do ponto no espaço onde o raio $L_1 = C_1\boldsymbol{p}$ se cruza com o raio $L_2 = C_2\boldsymbol{p}'$. Na pratica, os raios L_1 e L_2 nunca se cruzam devido a erros na calibração e na localização dos pontos coincidentes. Deve-se calcular o ponto médio do vetor que minimiza a distância entre os raios L_1 e L_2. Esse tipo de reconstrução é chamado reconstrução por triangulação.

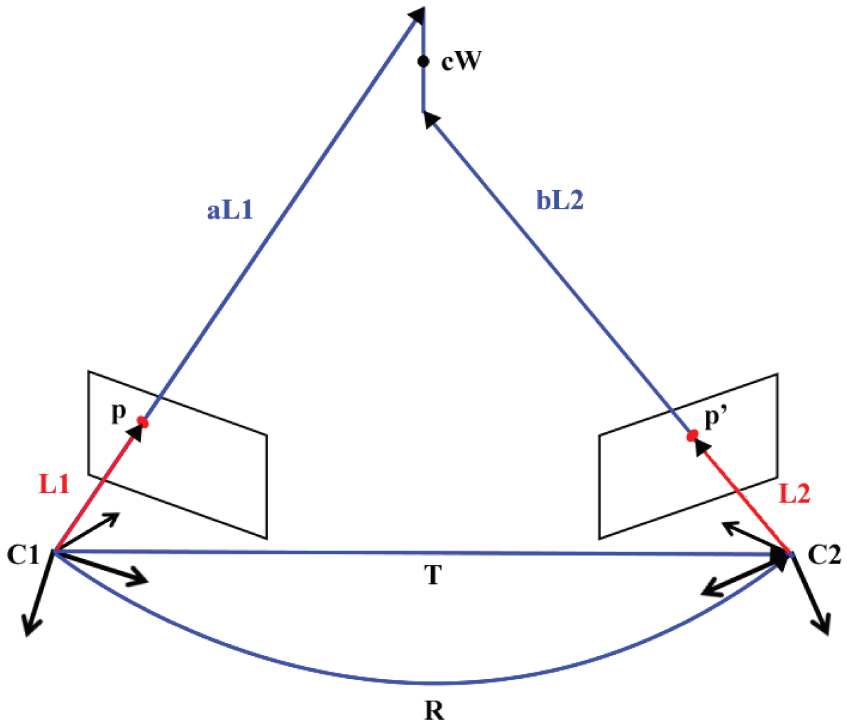

Figura 3.19: Reconstrução por triangulação.

As coordenadas do ponto no espaço podem ser determinadas no sistema de coordenadas globais, ou no sistema de coordenadas de qualquer uma das duas câmeras. Para obter as coordenadas deve-se resolver o sistema apresentado na Equação 3.14

$$a\boldsymbol{p} - b\mathbf{R}^{T}\boldsymbol{p}' + c(\boldsymbol{p} \times \mathbf{R}^{T}\boldsymbol{p}') = \boldsymbol{T}. \tag{3.14}$$

Onde $\mathbf{R}$ é a matriz de rotação do sistema estéreo e $\boldsymbol{T}$ o vetor de translação. Tanto $\mathbf{R}$ quanto $\boldsymbol{T}$ definem a posição da câmera esquerda com respeito à câmera da direita, ou vice versa (TRUCCO e VERRI, 1998, p. 162-163)

4 SISTEMA DE MEDIÇÃO DE ÂNGULOS DE ALINHAMENTO DESENVOLVIDO NO LABORATORIO

A concepção do sistema de medição de ângulos de alinhamento, bem como testes iniciais em ambiente real e virtual, foi elaborada na tese de doutorado de Carlos Mingoto (MINGOTO, 2012), no Laboratório de Processamento de Sinais da Faculdade de Engenharia Mecânica da UNICAMP. O presente trabalho implementa os principais conceitos desenvolvidos na tese de Mingoto, na forma de um programa de computador para usuário final (*front end*), integrado aos periféricos de aquisição de imagem (câmeras). O sistema integrado é capaz de medir os ângulos de *toe* e *camber* das duas rodas dianteiras de um veículo, comparando-os com a orientação padrão média das rodas traseiras. A medição é feita por meio de visão computacional, utilizando câmeras do tipo *webcams*, processamento de imagens e reconstrução 3D.

Neste projeto os componentes físicos utilizados são: um computador, oito câmeras *Microsoft® LifeCam® Studio®* apresentada na Figura 4., dois cubos de calibração como o apresentado na Figura 3.2, um de tamanho maior do que o outro, e quatro estruturas (pedestais) para sustentação de duas câmeras, Figura 4.2. Cada uma das estruturas é colocada ao lado de cada roda do veículo de tal forma que as câmeras captem a imagem completa da roda.

Figura 4.1: Câmeras utilizadas no sistema

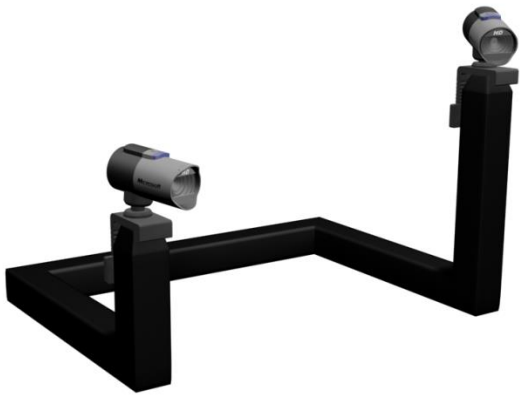

Figura 4.2: Pedestal utilizado para obtenção de imagens em cada roda

O uso do sistema de calibração inicia com o posicionamento de cada um dos pedestais em locais que permitam a obtenção das imagens de cada roda do veículo, conforme ilustrado na Figura 4.4. O posicionamento dos pedestais deve ser feito inicialmente de forma a permitir que as imagens das rodas de uma variedade de veículos que sejam estacionados no ambiente de medição, sejam sempre visíveis por todas as câmeras. O segundo passo que é realizado apenas na primeira vez em que os pedestais com as câmeras são posicionados no ambiente de medição, consiste na calibração global das quatro câmeras de referência (definição de câmeras de referência abaixo) do sistema. O terceiro passo, realizado também apenas uma vez, consiste nas calibrações locais de cada par de câmeras de cada pedestal. A quarta etapa é o posicionamento do veículo com as rodas visíveis por todas as câmeras. O quinto passo consiste no cálculo de correlação de pontos nas imagens tomadas por cada par de câmeras e a reconstrução da posição de cada roda num espaço 3-D. O sexto passo trata de obter os vetores normais a cada roda reconstruída no passo anterior. O sétimo passo consiste em realizar a comparação dos vetores normais das rodas dianteiras e traseiras, e obtendo finalmente as medidas dos ângulos de alinhamento.

Na Figura 4.3 é apresentado o algoritmo geral de obtenção dos ângulos

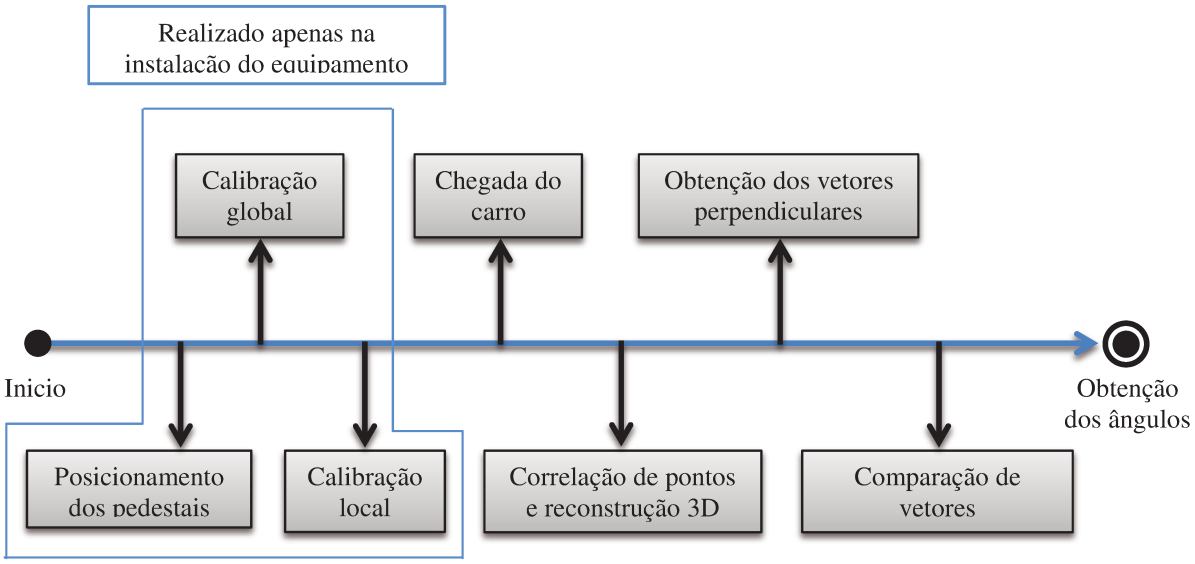

Figura 4.3: Algoritmo de medição dos ângulos de alinhamento

4.1 Posicionamento das estruturas e configuração física do sistema

A Figura 4.4 apresenta uma visão superior do carro e com os pedestais posicionados e suas câmeras numeradas. Para uma melhor explicação do sistema, as câmeras são divididas em duas categorias: câmeras de precisão e câmeras de referência. As câmeras de precisão estão posicionadas de tal forma que possam obter uma imagem total da roda, com a maior proximidade possível. Na Figura 4.4 as câmeras de precisão possuem numeração impar. A precisão do sistema depende da resolução das câmeras utilizadas e da área que a roda ocupa na imagem. Quanto maior a área ocupada pela roda na imagem, maior é a precisão de medida angular (KURKA e MINGOTO, 2012).

As câmeras pares na Figura 4.4 são chamadas de referência. Estas câmeras devem ser capazes de captar a imagem da roda, e ao mesmo tempo, para efeito de calibração global, visualizar a região central do espaço onde o veículo é posicionado. A Seção 4.2 apresenta os detalhes do procedimento. de calibração global.

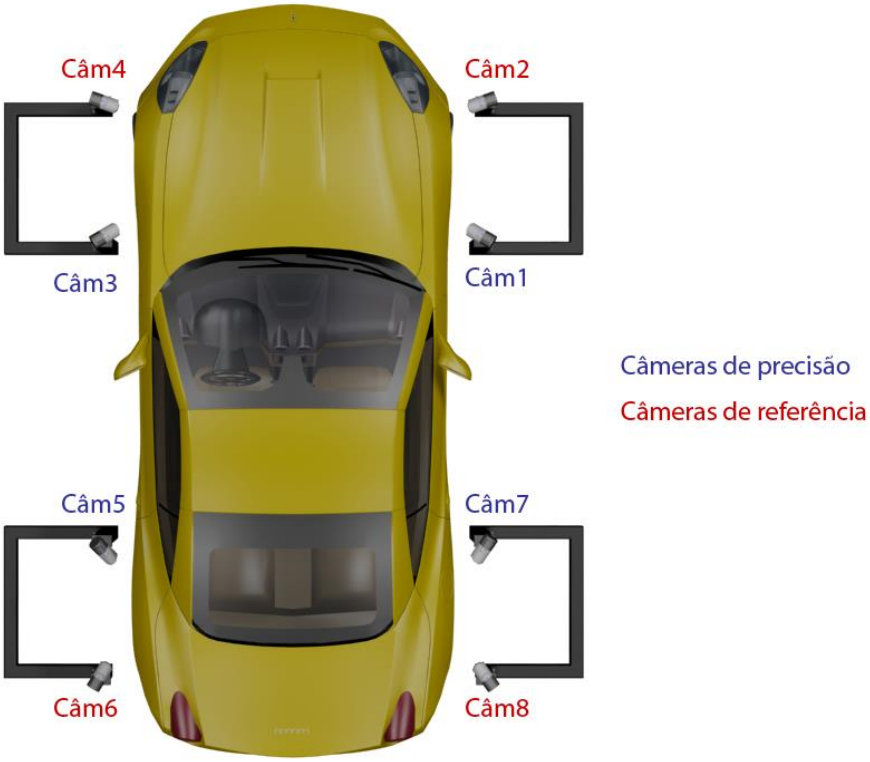

Figura 4.4: Configuração física do sistema

4.2 Calibração global

Arbitra-se o sistema global primário das medidas, como sendo o sistema de referências da câmera 2 . Para que os ângulos de alinhamento sejam expressos no sistema de coordenadas da câmera 2 é necessário obter as medidas de translação e rotação das três câmeras de referência restantes. A calibração é feita conforme descrito no capítulo 3, utilizando-se a caixa de calibração global apresentada à Figura 4.5. A caixa define o sistema de referência de calibração de "mundo" cuja origem é a esfera de cor vermelha. O eixo X de calibração é paralelo à aresta que une a esfera vermelha à esfera de cor ciano. O eixo Y é paralelo à aresta que une a esfera vermelha à esfera de cor amarela. Finalmente o eixo Z é paralelo à aresta que une a esfera vermelha à esfera de cor verde.

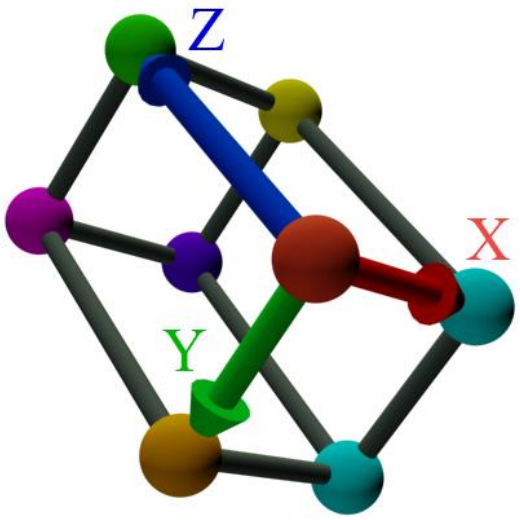

Figura 4.5: Caixa de calibração

Na Figura 4.6 é apresentado o processo de alinhamento global, onde são obtidas as matrizes de rotação e os vetores de translação das câmeras de referência com respeito ao sistema de referência do mundo. O primeiro subscrito na nomenclatura das matrizes de rotação e nos vetores de translação é o número da câmera que está sendo calibrada; o segundo subscrito é a primeira letra de *World* (mundo), por se referir a medidas feitas a partir do sistema de referências físico da caixa de calibração global.

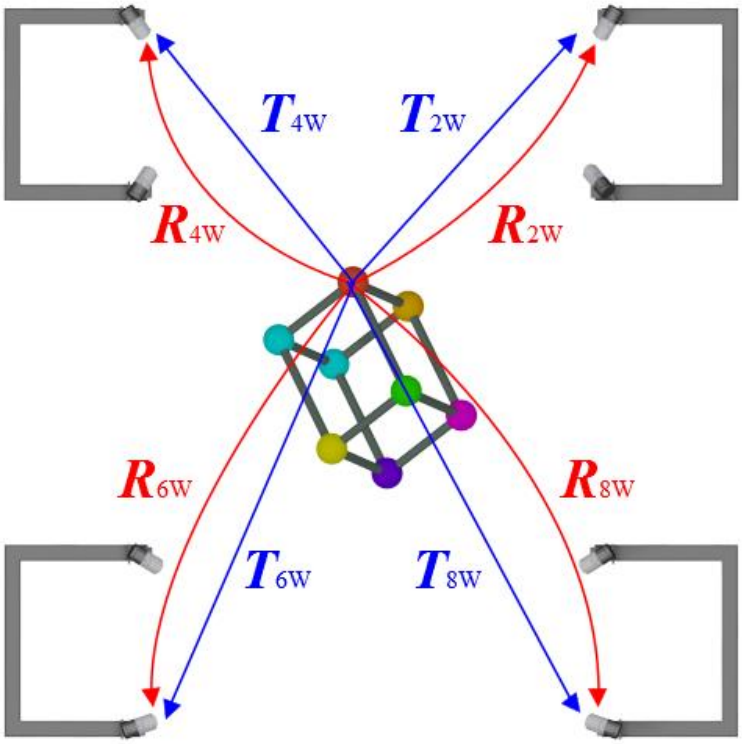

Figura 4.6: Calibração global

4.3 Calibração local

Na calibração local são calibrados os quatro pares de câmeras dos pedestais, Figura 4.7. Neste processo uma caixa menor de calibração local é posicionada sucessivamente à frente de cada pedestal. No software desenvolvido, o sequencia de calibração local é realizado na seguinte ordem: câmeras 1-2, 3-4, 5-6 e 7-8.

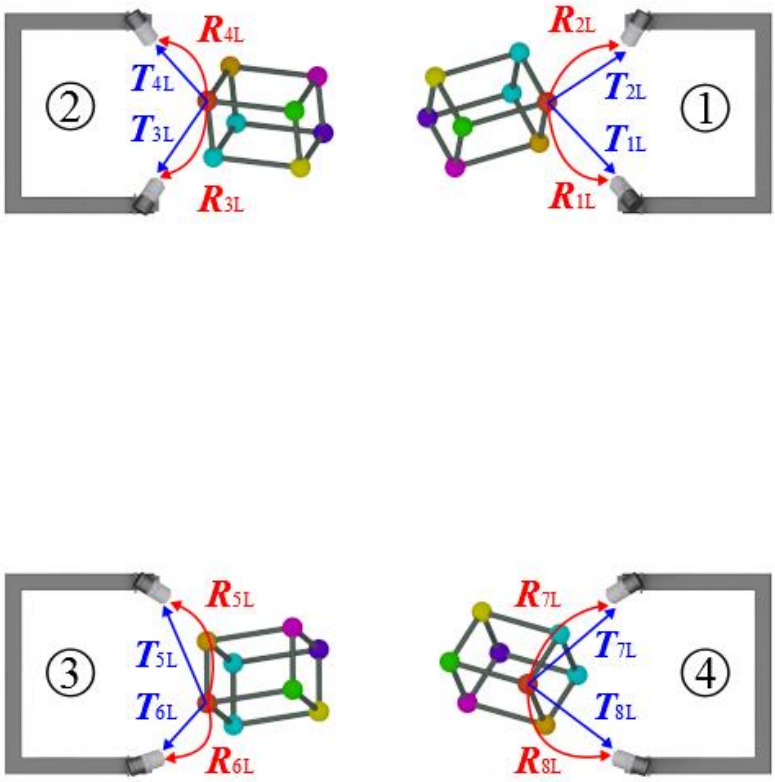

Figura 4.7: Calibração local

As matrizes de rotação e os vetores de translação têm como primeiro subscrito o número da câmera e como segundo subscrito a letra *L* de local, que diferencia as matrizes e os vetores obtidos na calibração global. Nesta calibração são levados em contas tanto os parâmetros intrínsecos quanto os extrínsecos de cada câmera.

Após realizadas as calibrações global e local, o sistema está pronto para receber os veículos corretamente posicionados, de forma que as imagens de suas rodas sejam captadas por todas as câmeras.

4.4 Correlação de pontos

Nessa seção é apresentado o algoritmo de correlação de pontos das imagens captadas em um pedestal. O processo ocorre de forma similar para os 4 pedestais do sistema.

A Figura 4.8 apresenta as imagens de uma roda virtual obtidas por um sistemas de câmeras calibradas do pedestal traseiro direito, utilizando a ferramenta de modelagem 3Ds MAX® (3DSMAX). A imagem da esquerda é a obtida pela câmera de referência e a da direita pela câmera de precisão.

Figura 4.8: Imagens da roda virtual obtidas com câmeras calibradas

Após obter as imagens de cada câmera é necessário identificar a roda dentro de cada imagem, para isso é utilizado um algoritmo de extração do contorno e obtenção dos parâmetros que definem a elipse da roda em cada imagem separadamente.

A elipse é uma seção cônica que obedece à Equação 4.1

$$Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0. \quad (4.1)$$

Onde $A, B, C, D, E\ e\ F$ são chamadas parâmetros da seção cônica. No caso da elipse deve-se cumprir a desigualdade da equação 4.2.

$$B^2 - 4AC < 0 \quad (4.2)$$

Existem pesquisas desenvolvidas para obter os parâmetros da elipse conhecendo pontos no seu contorno. Em (MCLAUGHLIN, 1998) é utilizada a transformada de Hough para detectar elipses numa imagem. Em (XIE e JI, 2002) é apresentado um algoritmo de detecção de elipses onde não é utilizada a informação da curvatura nem do contorno, só a informação do braço menor da elipse. Mas existem algoritmos mais simples que usam o erro quadrado mínimo das distancias para obter os parâmetros da elipse que melhor se ajustem aos pontos na imagem. Neste trabalho foi utilizado o algoritmo proposto por Mingoto (MINGOTO, 2012) para a resolução do sistema (4.1) e a obtenção dos parâmetros da elipse. Nesse algoritmo também é expressa a elipse em termos de seu centro (Xc, Yc), a longitude do braço maior (a), a longitude do braço menor (b) e o ângulo de rotação θ com respeito ao referencial global (X,Y), como apresentada na Figura 4.9.

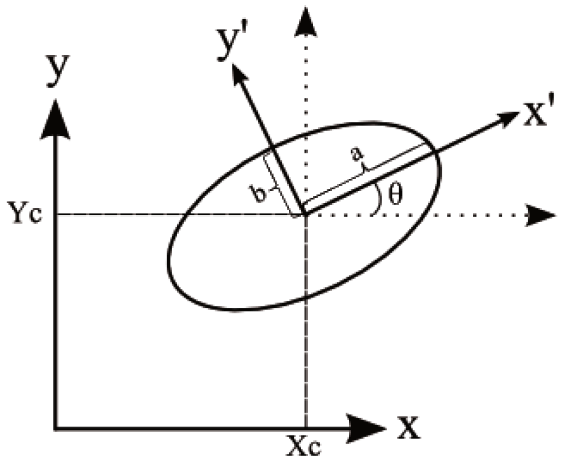

Figura 4.9: Parâmetros da elipse.

O algoritmo completo é apresentado na Figura 4.10. O primeiro passo é converter a imagem no sistema RGB à escala de cinza. A seguir é aplicado um algoritmo de procura de bordas (CANNY, 1986). O passo seguinte é aplicar um algoritmo de segmentação de elementos conectados chamados *blobs* (CHANG, CHEN e LU, 2004). A área de cada *blob* é calculada. Se a área do *blob* é menor do que 40% da área da imagem, este é descartado. O valor de 40% foi

obtido de forma experimental por apresentar o maior índice de filtragens corretas. Se a área do *blob* é superior a 40% da área da imagem então este é armazenado. Após analisar todos os *blobs* da imagem calcula-se aquele com a maior área e obtém-se o seu contorno para se obter os parâmetros da elipse.

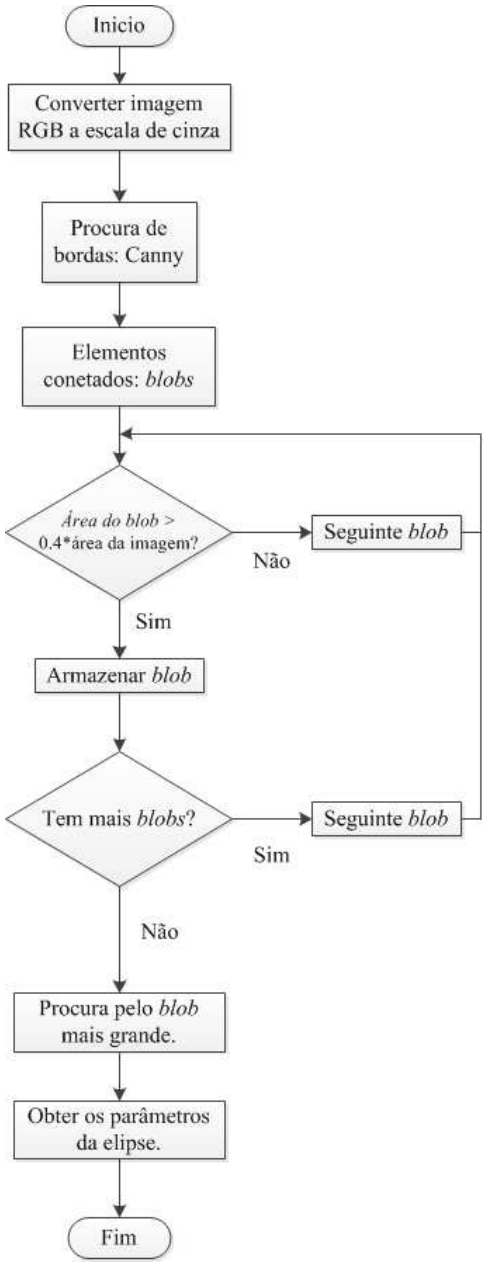

Figura 4.10: Algoritmo de extração do contorno da roda.

A Figura 4.11 apresenta as imagens com as elipses obtidas em cor azul. O numero de pontos sobre a elipse é um parâmetro ajustável. Assim podem-se obter mais pontos sobre o contorno e, por conseguinte obter mais pontos correlatos.

Figura 4.11: Imagens das rodas com as elipses obtidas em azul.

Com a parametrização das elipses e a geometria epipolar das imagens é possível fazer a procura dos pontos correlatos. O processo começa com a escolha de um ponto sobre a elipse em uma das duas imagens, por exemplo, da imagem esquerda na Figura 4.11. É calculada a equação da linha epipolar deste ponto na imagem da direita e depois são calculados os dois pontos de interseção entre a linha e a equação da elipse na imagem da direita.

A Figura 4.12 apresenta o processo de obtenção de dois pontos correlatos. O ponto escolhido é mostrado com um círculo vermelho na imagem esquerda, a linha epipolar é uma reta inclinada mostrada em branco e os dois pontos de interseção com a elipse da imagem direita são mostrados em vermelho. Se a coordenada de imagem *i* do ponto escolhido na roda esquerda é maior do que a coordenada *i* do centro dessa elipse então o ponto correlato certo na elipse da direita é aquele com o valor mais alto de *i.* Casso contrario é o ponto com o menor valor *i*.

Figura 4.12: Ponto escolhido na imagem esquerda em circulo vermelho, reta epipolar em branco e pontos de interseção com a elipse da imagem direita em vermelho.

Ao realizar o processo descrito, em todo o contorno da elipse, obtêm-se os pontos correlatos apresentados na Figura 4.13.

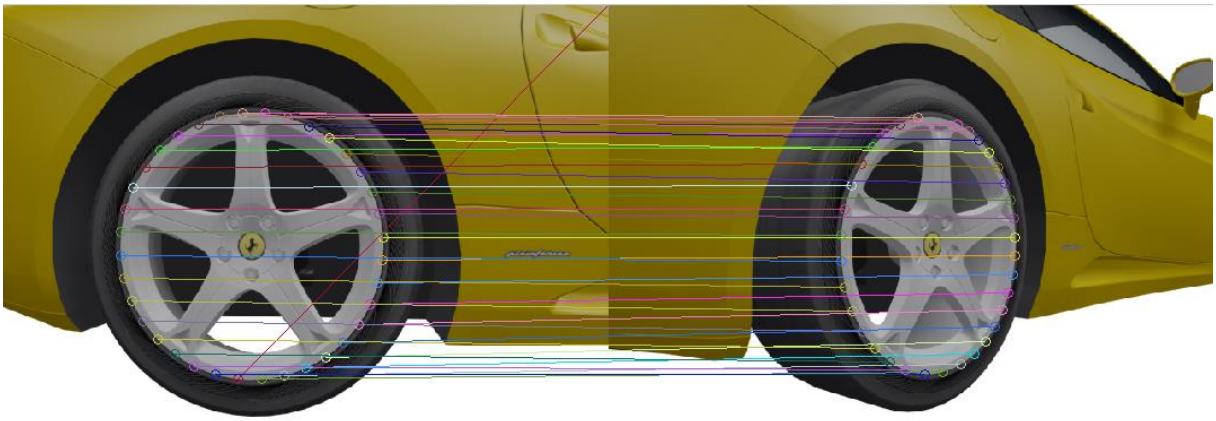

Figura 4.13: Pontos correlatos nas duas imagens

4.5 Reconstrução 3D

Após obter os pontos correlatos pode-se aplicar o algoritmo de reconstrução apresentado na Seção 3.2.2e obter as coordenadas do circulo com respeito à câmera de referência do pedestal que contém as duas câmeras. A Figura 4.14 apresenta a reconstrução com respeito a câmera de referência.

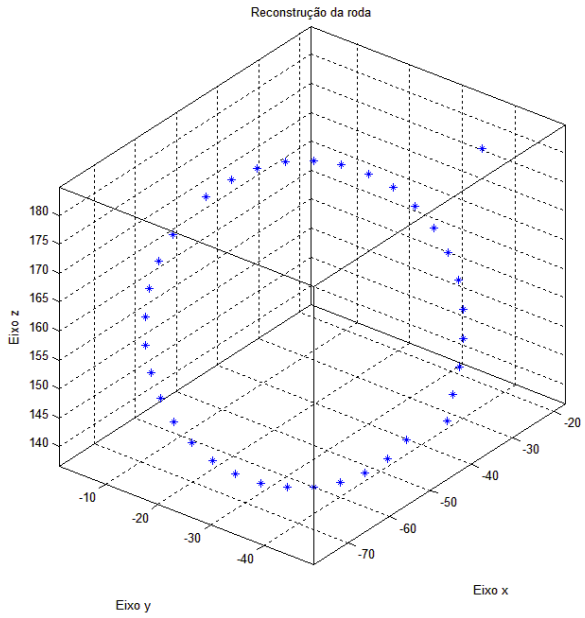

Figura 4.14: Reconstrução da roda

Pontos mais próximos à borda superior e inferior na elipse da esquerda não apresentam uma boa correspondência na imagem da direita, devido a erros de aproximações angulares de ângulos de pequena magnitude. Para solucionar esse problema são obtidos os pontos com maior e menor valor na coordenada vertical *j* da elipse da esquerda, apresentados como X na Figura 4.15. Logo é calculada a distancia (d) entre eles. Depois são descartados os pontos que estejam numa distancia 10% por abaixo e 10% por acima da distancia calculada entre os dois extremos. Na Figura 4.15 os pontos descartados são os que estão dentro da franja cinza.

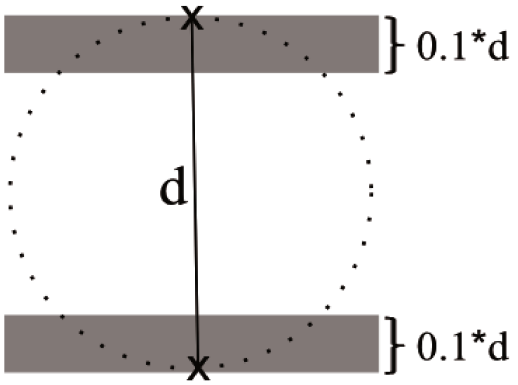

Figura 4.15: Seleção de pontos correlatos

Na Figura 4.16 é apresentada a correlação de pontos e a reconstrução 3D descartando os pontos nos extremos superior e inferior das elipses.

O processo de procura dos pontos correlatos e reconstrução 3D é realizado para cada roda do carro, assim ao final obtém-se as coordenadas tridimensionais de cada roda expressadas no sistema de coordenadas da câmera de referência de cada sistema estéreo.

Na seção seguinte trasladam-se os pontos reconstruídos ao sistema de coordenadas da câmera dois e aplicar o algoritmo de PCA.

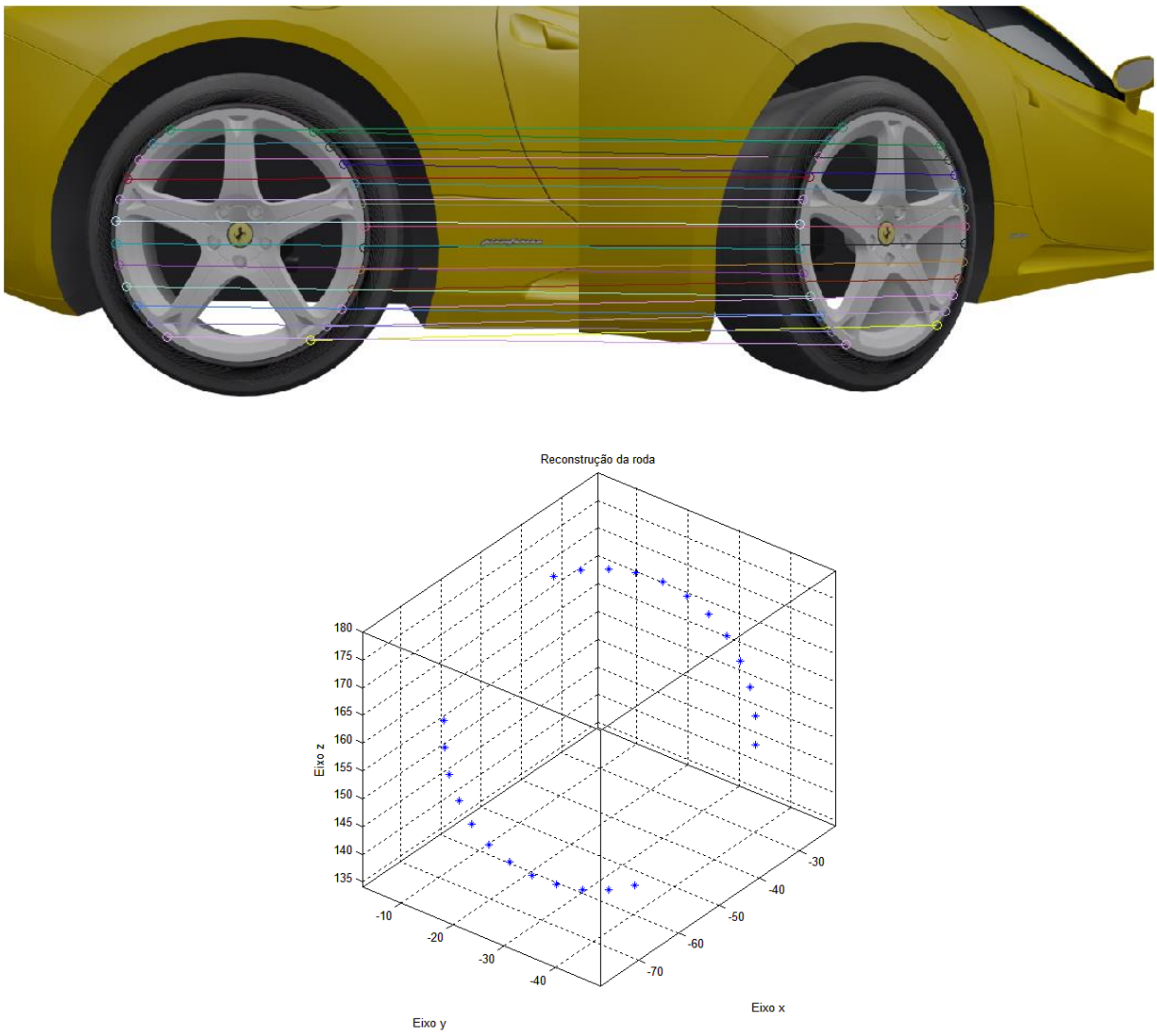

Figura 4.16: Pontos correlatos e reconstrução 3D sem pontos descartados.

4.6 Obtenção dos vetores perpendiculares

Após obter os pontos correlatos e a reconstrução tridimensional obtém-se os vetores perpendiculares à superfície da roda. Como as coordenadas das rodas reconstruídas estão expressas no sistema de coordenadas das câmeras de referência em cada sistema estéreo, é

necessário expressar essas coordenadas no referencial da câmera dois para medir os ângulos num mesmo sistema de coordenadas.

Com a informação da calibração global são obtidas as matrizes de rotação e os vetores de translação de cada câmera de referencia (Câm4, Câm6 e Câm8 da Figura 4.4) com respeito ao referencial da Câm2. Após expressar as coordenadas dos pontos de cada roda reconstruída no mesmo referencial é aplicado o algoritmo de PCA (*Principal-Component-Analysis*).

O PCA é uma das técnicas estatísticas mais usadas em processamento de sinais (ABADPOUR, 2005). Ela serve, sobretudo, para redução dimensional de dados (GONZALEZ, SCHWARTZ e PEDRINI, 2012) e na compressão de imagens RGB (MUDROVÁ e PROCHÁZKA, 2005). Neste trabalho ela é usada para determinar o vetor perpendicular ao plano gerado por o conjunto de pontos obtidos na reconstrução. A Figura 4.17 apresenta o vetor perpendicular à superfície em azul, o vetor vermelho é a componente na direção *X* e o verde na direção *Y*. Para a obtenção dos ângulos de alinhamento só é necessária a informação do vetor perpendicular (azul) e o centro do circulo.

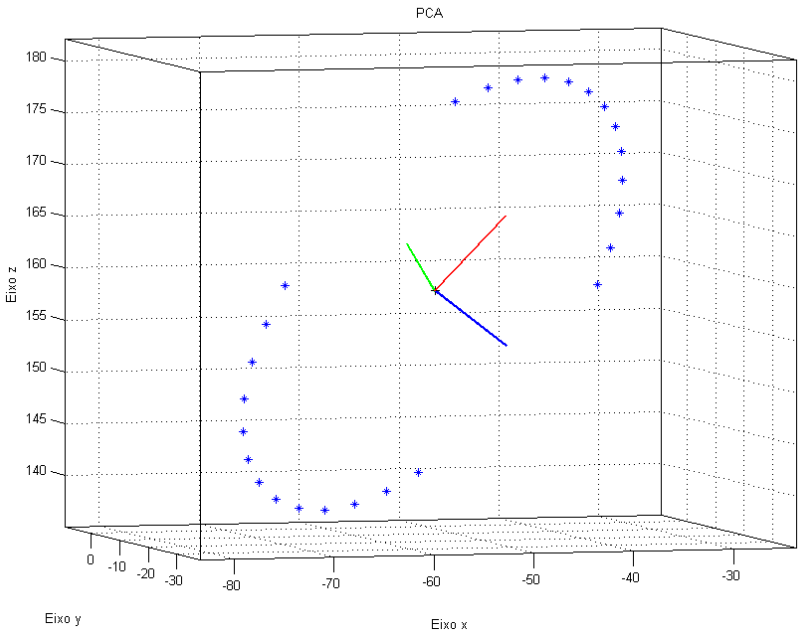

Figura 4.17: Analise dos componentes principais (PCA)

4.7 Comparação dos vetores e obtenção dos ângulos

Para ilustrar o processo de obtenção dos ângulos de alinhamento utiliza-se a situação hipotética da Figura 4.18, onde são geradas virtualmente quatro rodas e após aplicar o PCA obtêm-se os vetores perpendiculares a elas apresentadas em azul.

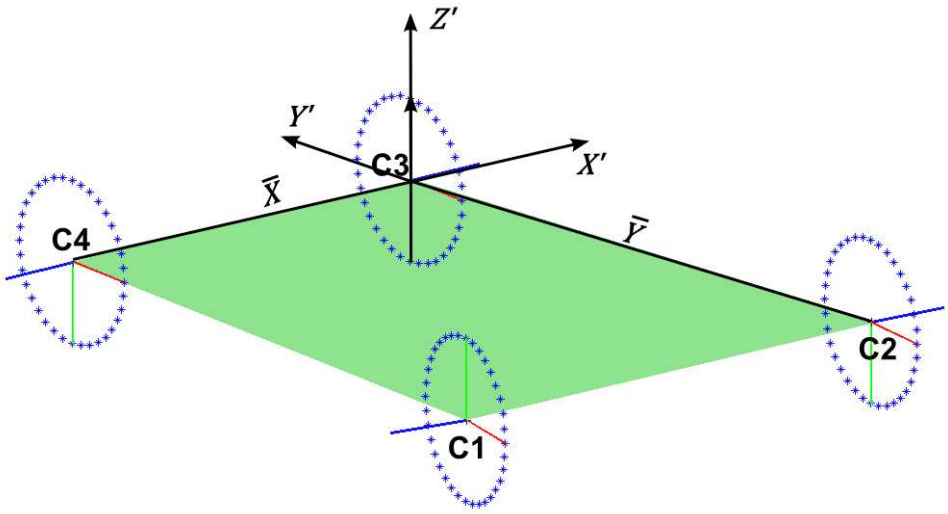

Figura 4.18: Calculo do sistema de referência do carro.

O processo começa com a obtenção do vetor $\bar{X}$ a partir das coordenadas do centro das rodas três e quatro, **C3** e **C4** na figura. Calcula-se seu vetor unitário X'. Em seguida é calculado o vetor $\bar{Y}$ entre os centros das rodas **C2** e **C3**. Depois é calculado o produto vetorial entre os vetores X' e $\bar{Y}$, para se obter o vetor vertical Z' de acordo com a Equação 4.3

$$Z' = X' \times \bar{Y}, \tag{4.3}$$

o vetor Z' é perpendicular ao plano formado pelos centros das rodas.

O próximo passo consiste em realizar o produto vetorial entre os vetores $\boldsymbol{Z'}$ e $\boldsymbol{X'}$, para obter o vetor de direção de movimentação do veículo, $\boldsymbol{Y'}$, perpendicular a eles, de acordo com a Equação 4.4

$$\boldsymbol{Y'} = \boldsymbol{Z'} \times \boldsymbol{X'}. \tag{4.4}$$

Os vetores $\boldsymbol{X'}$, $\boldsymbol{Y'}$ e $\boldsymbol{Z'}$ são tornados unitários, gerando o sistema de coordenadas do veículo $\hat{\boldsymbol{x}}$, $\hat{\boldsymbol{y}}$ e $\hat{\boldsymbol{z}}$ que é o sistema com respeito ao qual se obtém os ângulos de *toe* e *camber* de cada roda dianteira. Assim os vetores perpendiculares das rodas **C1, C2, C3 e C4** são transladados a origem deste sistema. A Figura 4.19 apresenta os vetores **V3** e **V4** paralelos ao eixo $\hat{\boldsymbol{x}}$ que representam os vetores perpendiculares às rodas três e quatro respetivamente. O vetor **V2** representa o vetor perpendicular à roda dois e o vetor **V1** representa o vetor perpendicular à roda um.

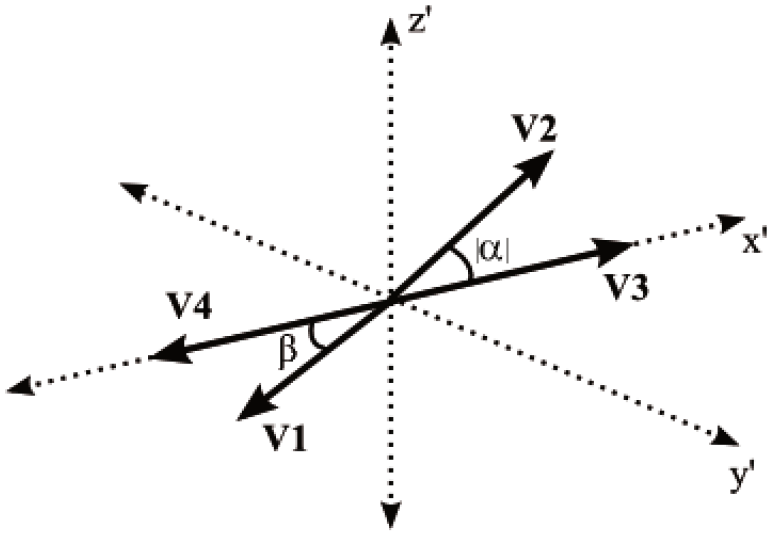

Figura 4.19: Vetores no sistema de referência do carro.

Seja $\boldsymbol{V2_{xy}}$ o vetor com as componentes *x'* e *y'* do vetor **V2**, é dizer a projeção do vetor **V2** no plano x'-y', Figura 4.20. O ângulo *toe* da segunda roda, α_{toe} é o ângulo formado entre o vetor $\boldsymbol{V2_{xy}}$ e o vetor **V3** de acordo com a Equação 4.5

$$\alpha_{toe} = \cos^{-1}\left(\frac{\boldsymbol{V2_{xy}} \cdot \boldsymbol{V3}}{|\boldsymbol{V2_{xy}}||\boldsymbol{V3}|}\right). \tag{4.5}$$

De forma similar, seja $\boldsymbol{V2_{xz}}$ o vetor com as componentes *x'* e *z'* do vetor **V2**, ou seja a projeção do vetor **V2** no plano x'-z'. O ângulo de *camber* da segunda roda, α_{camber} é o ângulo formado entre o vetor $\boldsymbol{V2_{xz}}$ e o vetor **V3** de acordo com a Equação 4.6

$$\alpha_{camber} = \cos^{-1}\left(\frac{\boldsymbol{V2_{xz}} \cdot \boldsymbol{V3}}{|\boldsymbol{V2_{xz}}||\boldsymbol{V3}|}\right). \tag{4.6}$$

De forma similar são obtidos os ângulos de alinhamento da roda um, β_{toe} e β_{camber} a partir dos vetores $\boldsymbol{V1_{xy}}, \boldsymbol{V1_{xz}}$ e **V4** usando as Equações 4.5 e 4.6 como é apresentado na Figura 4.20

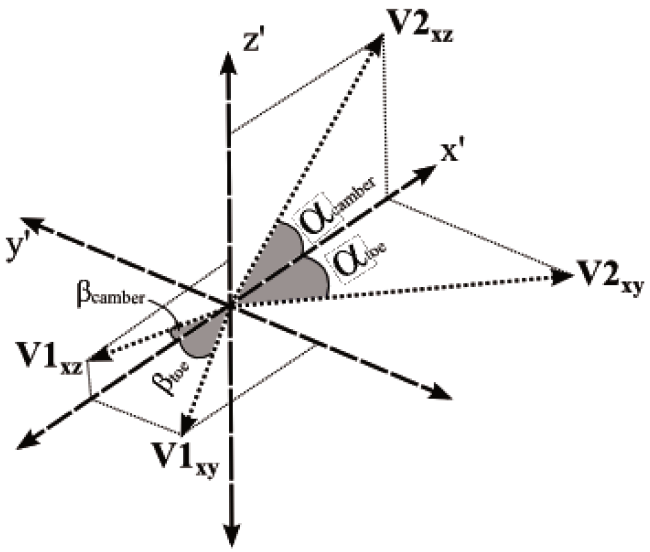

Figura 4.20: Ângulos de alinhamento para as rodas dianteiras.

Neste capítulo foi apresentado o sistema de alinhamento e foram detalhadas cada uma das etapas do processo, começando com o posicionamento do equipamento, calibração de câmeras, obtenção de pontos correlatos e reconstrução 3D. Foi apresentado o algoritmo de obtenção dos ângulos de alinhamento *toe* e *camber* das duas rodas dianteiras. No seguinte capítulo apresenta-se e *software* desenvolvido, as suas etapas e uso geral.

5 SOFTWARE DE ALINHAMENTO

Neste capítulo apresenta-se o software desenvolvido que implementa o algoritmo de medida dos ângulos apresentado no Capítulo 4. Foram utilizadas imagens virtuais geradas na plataforma 3ds Max (3DSMAX). O *software* aqui apresentado esta dividido em três partes. Na primeira parte é feita a configuração inicial do sistema. Na segunda parte é levado a cabo o processo de calibração. Finalmente, na terceira parte são reconstruídas as rodas e são obtidos os ângulos de alinhamento.

A Figura 5.21 apresenta a tela inicial do programa. É pedido para o usuário que escolha qual procedimento quer fazer. Se for a primeira vez que o programa é executado, é necessário fazer a configuração inicial. Assim se o usuário deseja fazer a calibração ou a medição dos ângulos sem ter feita a configuração, é apresentada uma mensagem.

Figura 5.21: Tela inicial do programa.

5.1 Configuração

O processo de configuração consta de duas partes, na primeira parte o usuário deve identificar quais câmeras serão utilizadas durante o processo, isso é necessário devido a que o computador pode ter *webcams* conectadas que não fazem parte do sistema de medidas dos ângulos. Na segunda parte o usuário assigna as posições de cada câmera de acordo com o diagrama da Figura 4.4.

A Figura 5.22 apresenta a tela mostrada ao usuário quando ele escolhe fazer a configuração. Aqui o usuário determina a partir dos dois botões de rádio "Sim" ou "Não", se a câmera atualmente apresentada será usada no sistema. Os botões azuis em forma de setas ao lado da imagem da câmera permitem selecionar a câmera anterior ou a seguinte, na sequencia de câmeras conectadas ao computador. Após selecionar as oito câmeras o usuário prime o botão "Finalizar" e passa à segunda parte do processo de configuração.

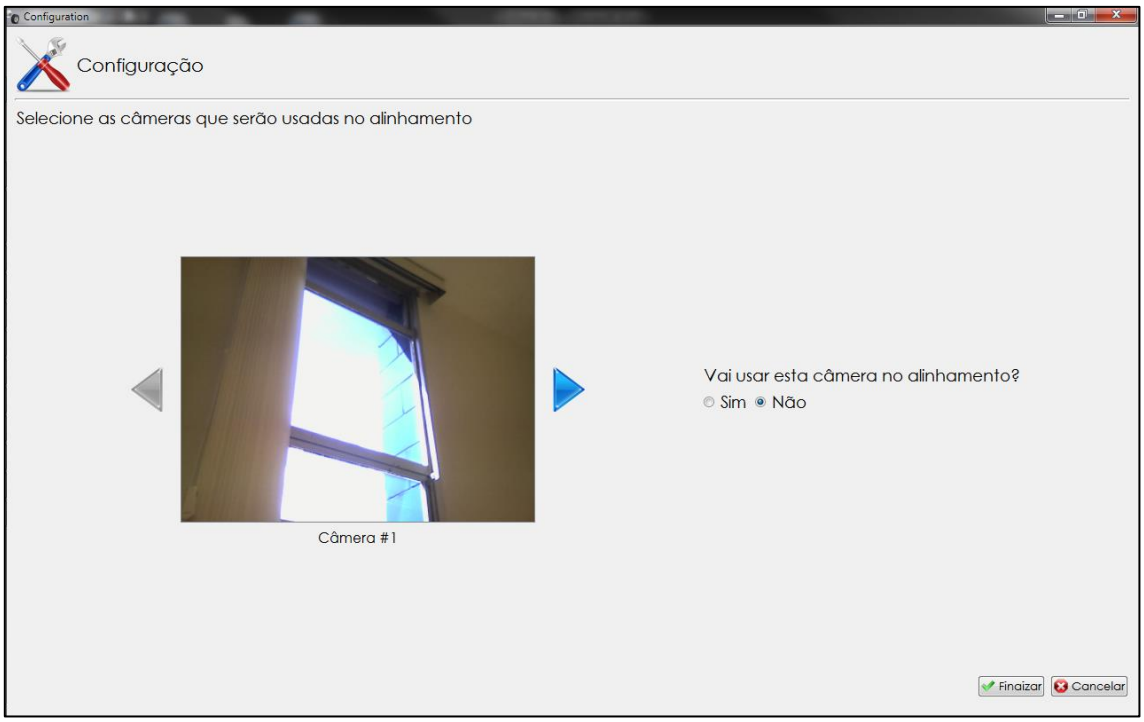

Figura 5.22: Tela da primeira parte da configuração.

Na segunda parte da configuração, Figura 5.23, é pedido ao usuário assignar a cada câmera um número de acordo com a imagem do carro na direita. Isto é feito para determinar a posição de cada câmera no sistema.

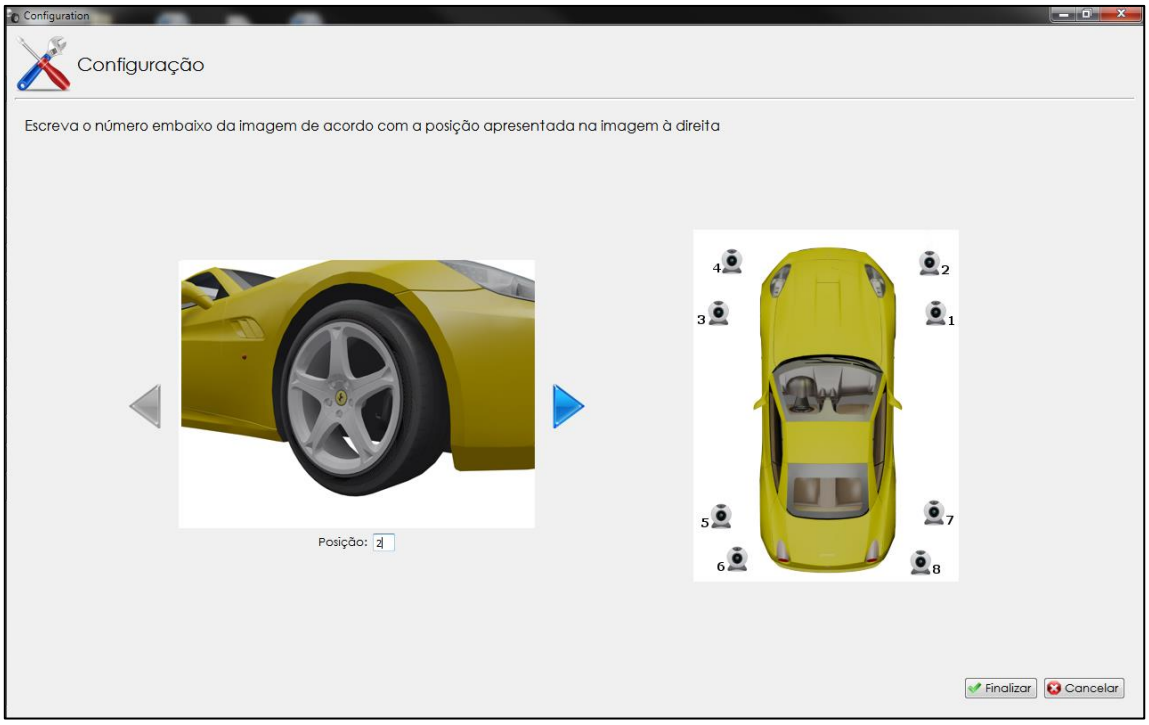

Figura 5.23: Tela da segunda parte da configuração.

Quando o usuário acaba o processo de numerar as câmeras e após premir o botão "Finalizar" é criado o arquivo ***config_cams.xml*** com a informação obtida no processo de configuração. Se a criação do arquivo é completada corretamente é apresentada uma mensagem como mostra a Figura 5.24.

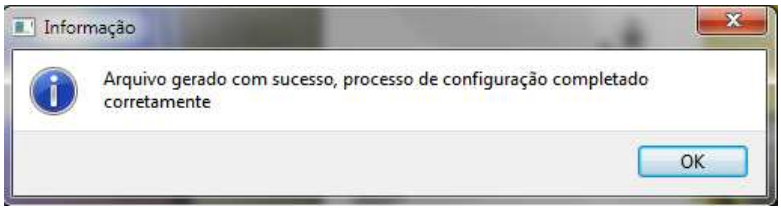

Figura 5.24: Confirmação da criação do arquivo ***config_cams.xml***

5.2 Calibração

O processo de calibração é dividido em duas partes. Na primeira parte são obtidos os valores no canal H das cores das esferas na caixa no sistema HSV. Na segunda parte são obtidos os parâmetros intrínsecos e extrínsecos das câmeras. O processo de obtenção dos parâmetros de calibração é feito tanto para a calibração global quanto a calibração local (seção 4.2 e 4.3).

Após ser selecionado o processo de calibração na tela inicial, é verificado que o arquivo ***config_cams.xml*** exista e que tem permissão de leitura, após isso é apresentada a tela de calibração, Figura 5.25.

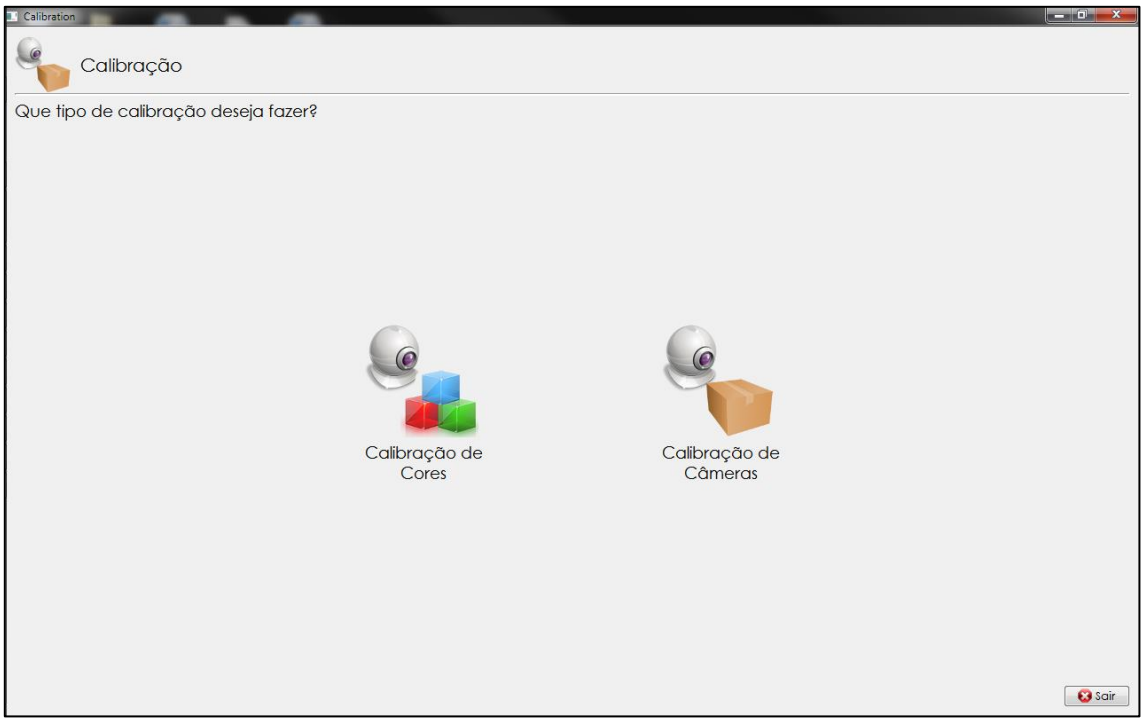

Figura 5.25: Tela inicial de calibração

Após o usuário premir o botão de “Calibração de Cores” é apresentada a janela da Figura 5.26. O programa inicia a câmera número um do sistema, de acordo com a numeração da Figura 4.4, e obtém as suas imagens. Neste exemplo é utilizada uma imagem fixa da caixa, mas o usuário verá o que esteja obtendo a câmera. Nesta parte é pedido para o usuário premir sobre cada cor na ordem que está na lista da direita. O usuário primeiro deve premir a esfera vermelha da caixa. O algoritmo intenta determinar se a cor premida é a vermelha, se for assim então é trocado o sinal de erro por o sinal de “OK” e é pedido para o usuário premir na esfera laranja e o processo se repete até que sejam completadas as sete cores. Se alguma cor não é reconhecida é apresentada uma mensagem ao usuário onde pede que mude os valores limites das cores.

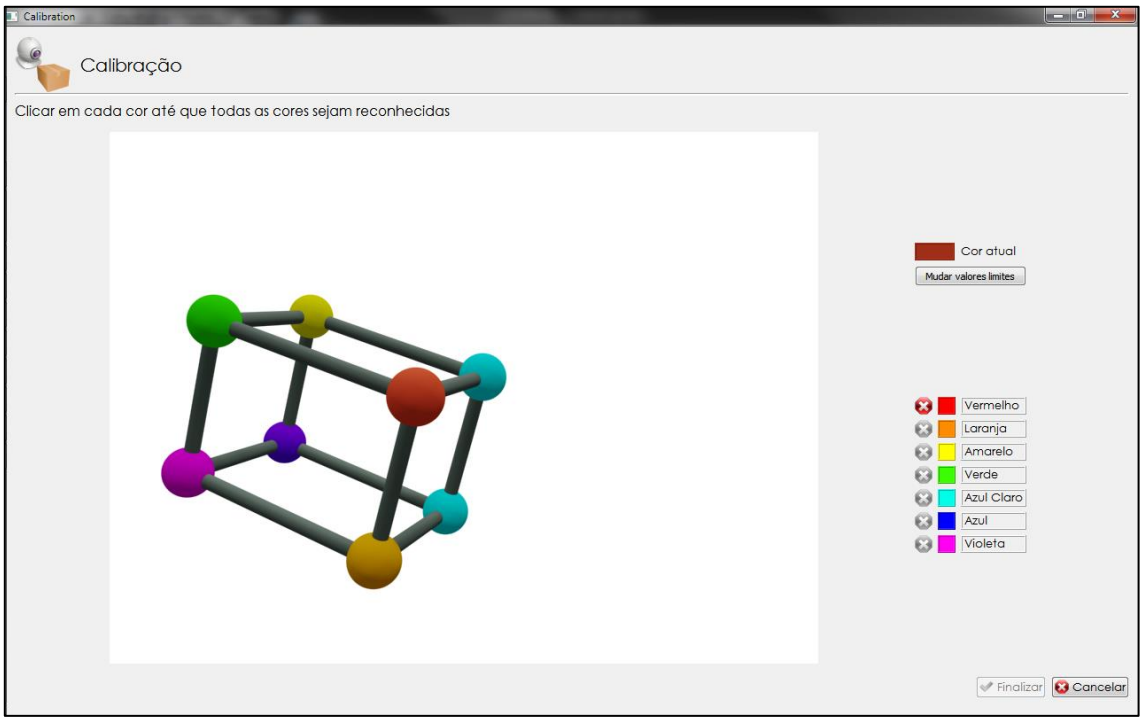

Figura 5.26: Tela de calibração de cores.

Se o usuário premir o botão “Mudar valores limites” abre-se uma nova janela onde o usuário pode selecionar a cor na lista da direita e modificar os valores mínimo e máximo no canal H até que na imagem só apareça a esfera da cor selecionada na imagem, Figura 5.27. Após salvar e finalizar o processo essa janela é fechada e o programa retorna à tela da Figura 5.26 onde o usuário deve tentar de novo completar as cores na ordem da lista. Se todas as cores são reconhecidas corretamente é gerado o arquivo ***config_colors.xml*** e o usuário é notificado.

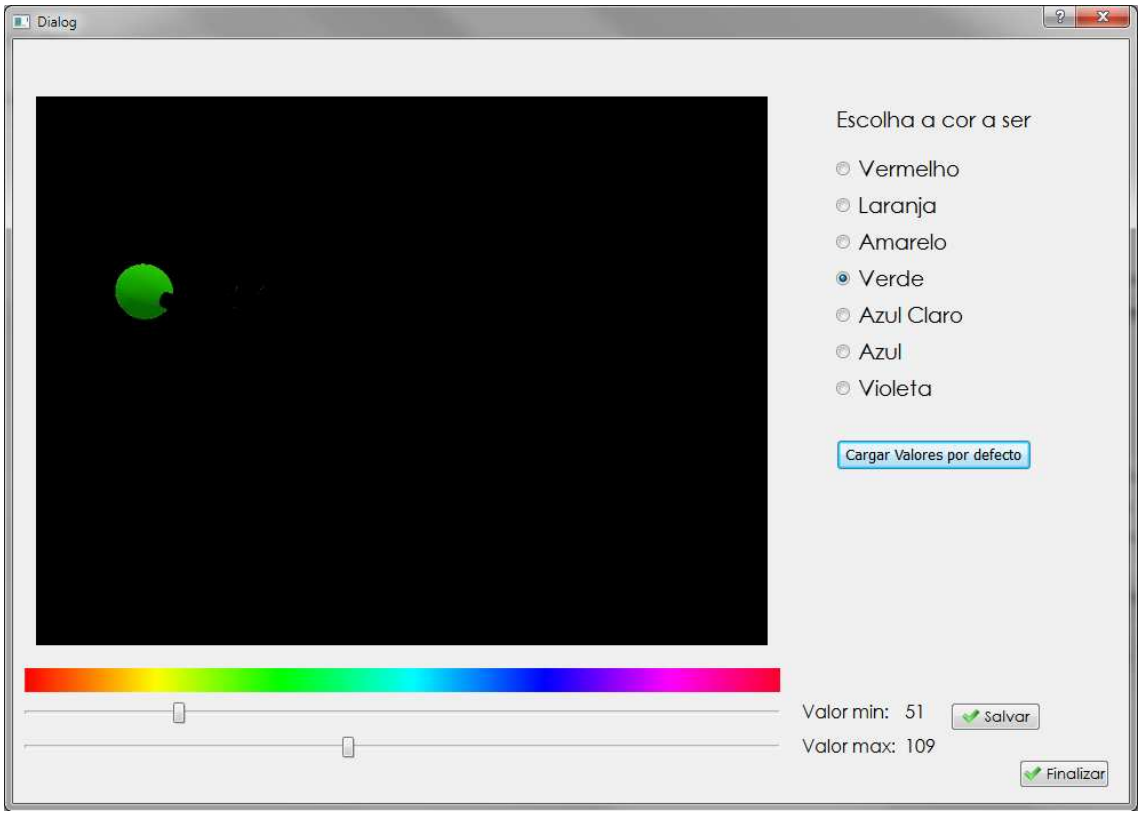

Figura 5.27: Janela para mudar os valores limites das cores.

Caso o usuário prima o botão “Calibração de Câmeras” na tela da Figura 5.25 o programa inicia as câmeras de referência, ou seja, as câmeras 2, 4, 6 e 8, e estabelece a resolução determinada pelo usuário. Se todo o processo de inicialização das câmeras finaliza corretamente é apresentada a tela da Figura 5.28, onde são mostradas as quatro câmeras e o usuário deve colocar a caixa de calibração de tal forma que as oitos esferas sejam visíveis pelas quatro câmeras ao mesmo tempo. Quando a caixa este localizada corretamente o usuário deve premir o botão “Iniciar” para iniciar o processo de calibração. Se o processo termina corretamente é habilitado o botão “Seguinte” para continuar com a calibração local. A Figura 5.29 apresenta o momento em que o processo de calibração global termina corretamente. Caso alguma esfera não for localizada ou exista algum erro no processo de calibração, o usuário é notificado através de uma mensagem e ele deve reposicionar o cubo e tentar de novo.

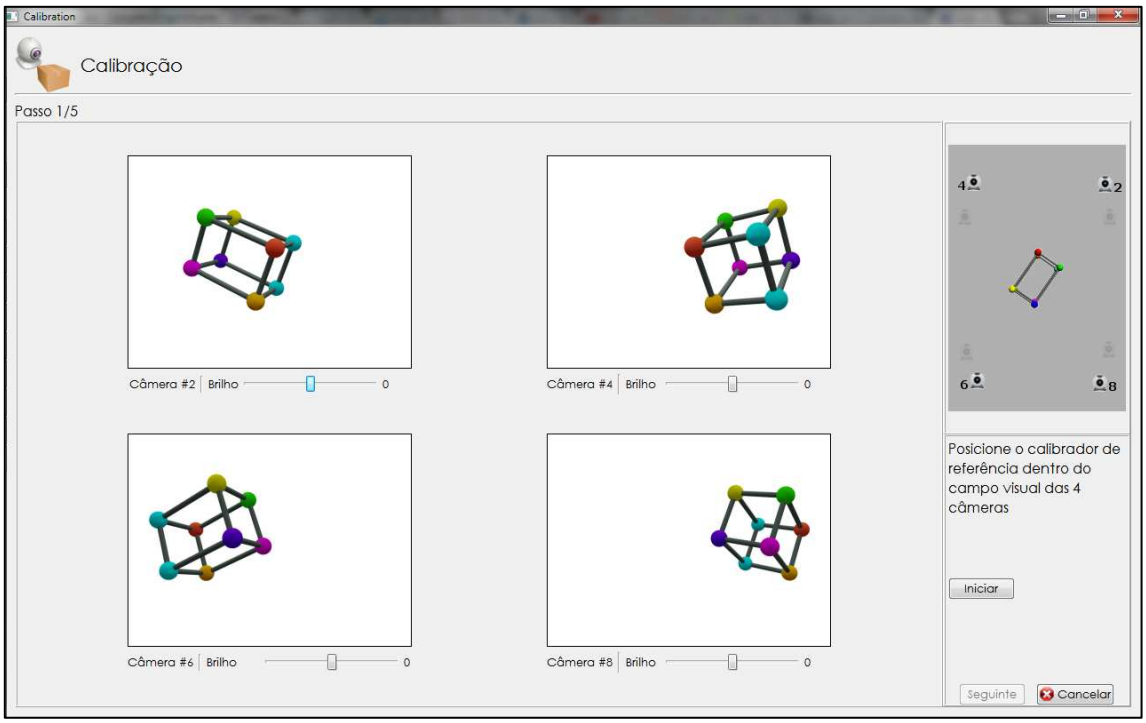

Figura 5.28: Tela de calibração global.

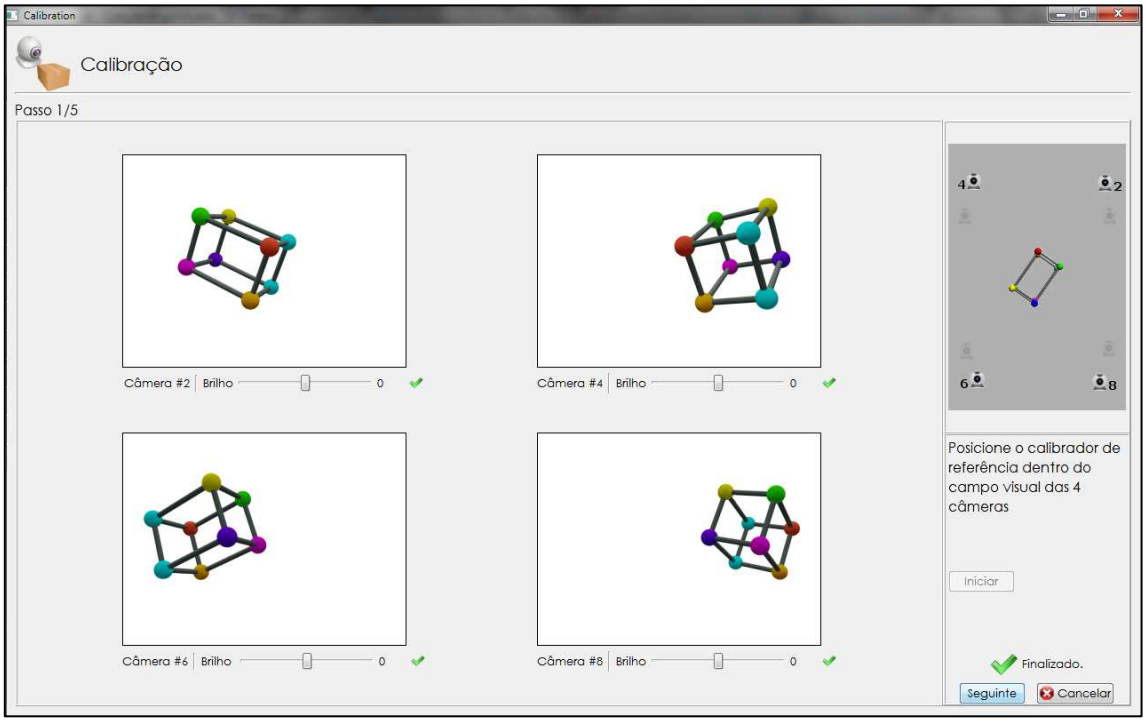

Figura 5.29: Tela de calibração global terminada.

Depois do premir em “Seguinte” é apresentada a tela de calibração local. Aqui o usuário deve posicionar a caixa na frente das câmeras um e dois, como é apresentada na figura superior direita da tela. Após o usuário premir em “Iniciar” o processo de calibração dessas duas câmeras começa e é analisada cada imagem. Se o processo de calibração termina corretamente é apresentada a tela da Figura 5.30 e o botão de “Seguinte” é habilitado.

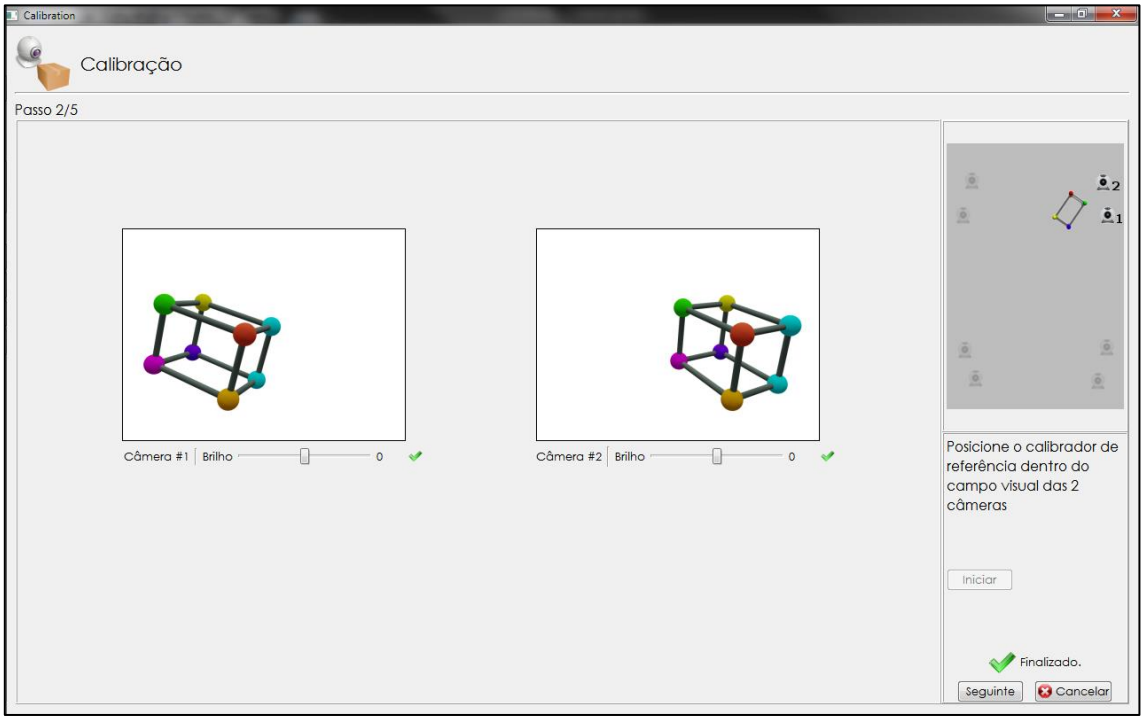

Figura 5.30: Tela de calibração local terminada.

O processo continua com as câmeras 3 e 4, 5 e 6, 7 e 8. Quando todas as câmeras estejam calibradas corretamente é gerado o arquivo ***camera_cali.xml*** com a informação obtida no processo. Finalmente a tela de calibração é fechada e de novo é apresentada a tela inicial, Figura 5.21, onde o usuário pode começar o processo de reconstrução e obtenção dos ângulos.

5.3 Reconstrução

O processo de reconstrução e obtenção de ângulos começa quando o usuário prime o botão de “Medir os ângulos” na tela inicial, Figura 5.21. Na Figura 5.31 é apresentada a tela de medida dos ângulos. Na primeira parte são iniciadas as câmeras 5, 6, 7 e 8. Tenta-se reconstruir e obter o vetor normal da roda quatro, ou seja, a roda frente às câmeras sete e oito. Se a obtenção do vetor é correta são obtidas as imagens da câmera cinco e seis, tenta-se reconstruir e obter o vetor normal da roda três. Se a roda três é reconstruída corretamente são desligadas as câmeras 5, 6, 7 e 8 e são ligadas as câmeras 1, 2, 3 e 4. Tenta-se reconstruir as rodas um e dois, se todo o processo é realizado corretamente são apresentados para o usuário os ângulos obtidos como mostra a Figura 5.32

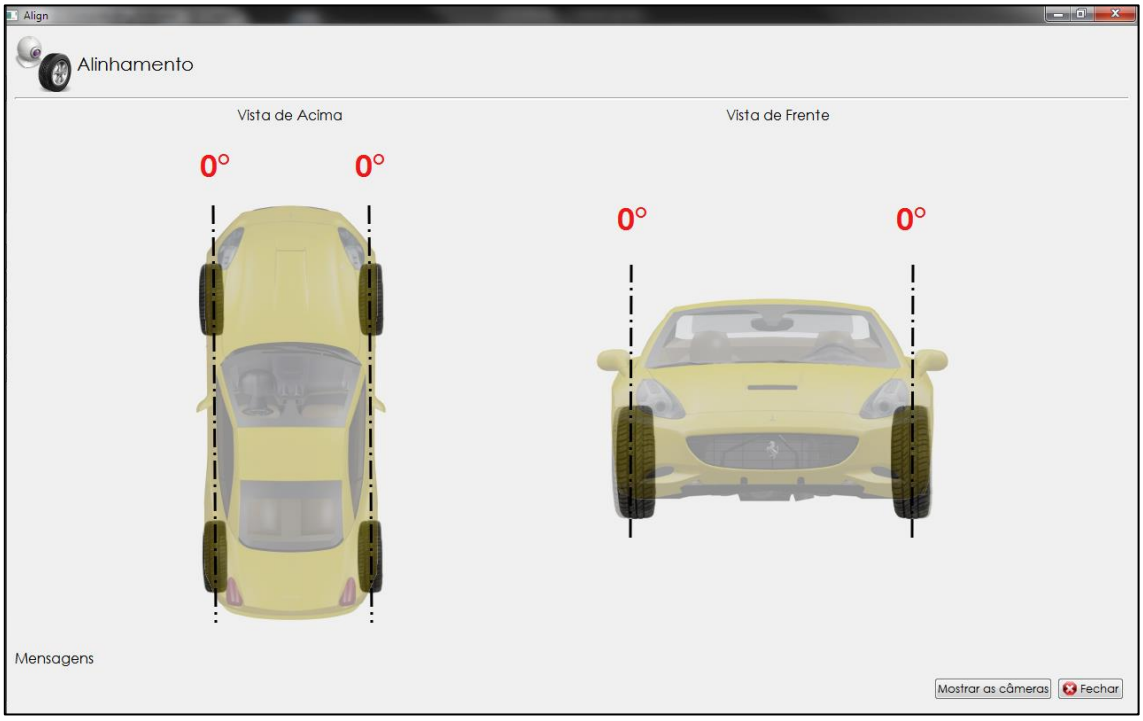

Figura 5.31: Tela de reconstrução e obtenção dos ângulos.

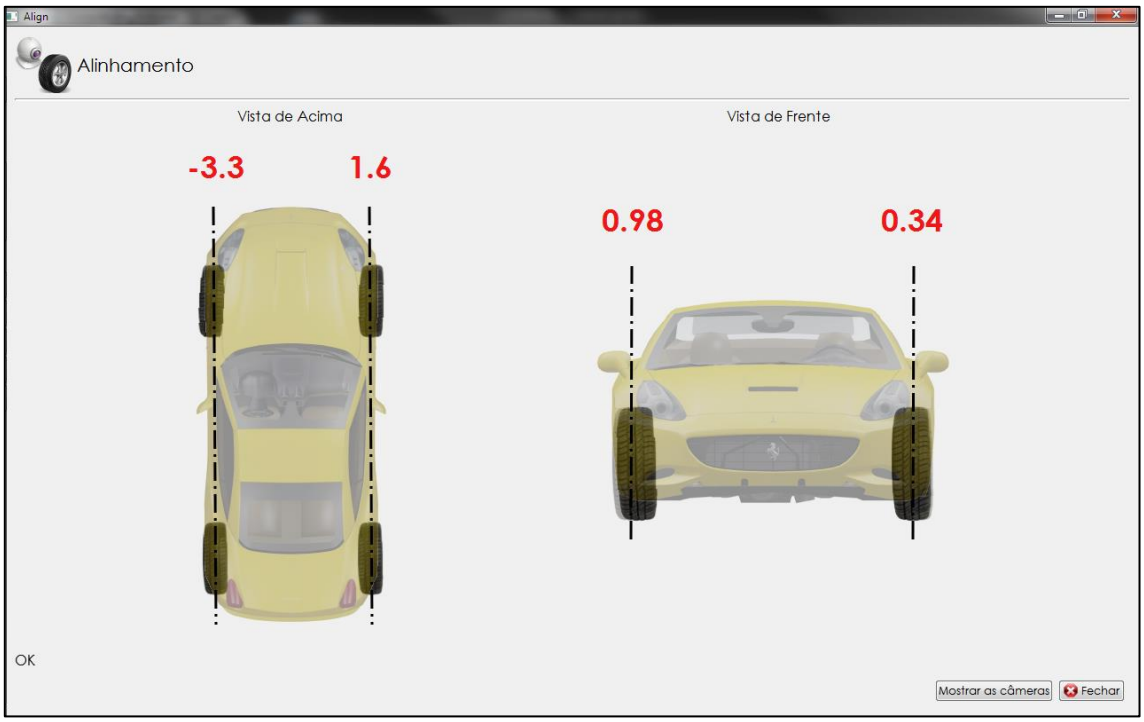

Figura 5.32: Tela apresentando os ângulos.

A tela de reconstrução e obtenção dos ângulos possui um botão de “Mostra as câmeras”, ele permite abrir uma tela onde são apresentadas todas as câmeras ligadas ao computador. O usuário pode escolher entre a câmera seguinte e a anterior usando as setas azuis ao lado da imagem. Nesta tela também é informado ao usuário a resolução da câmera, para facilitar o posicionamento das rodas do veiculo antes de se iniciar o processo de reconstrução e obtenção dos ângulos.

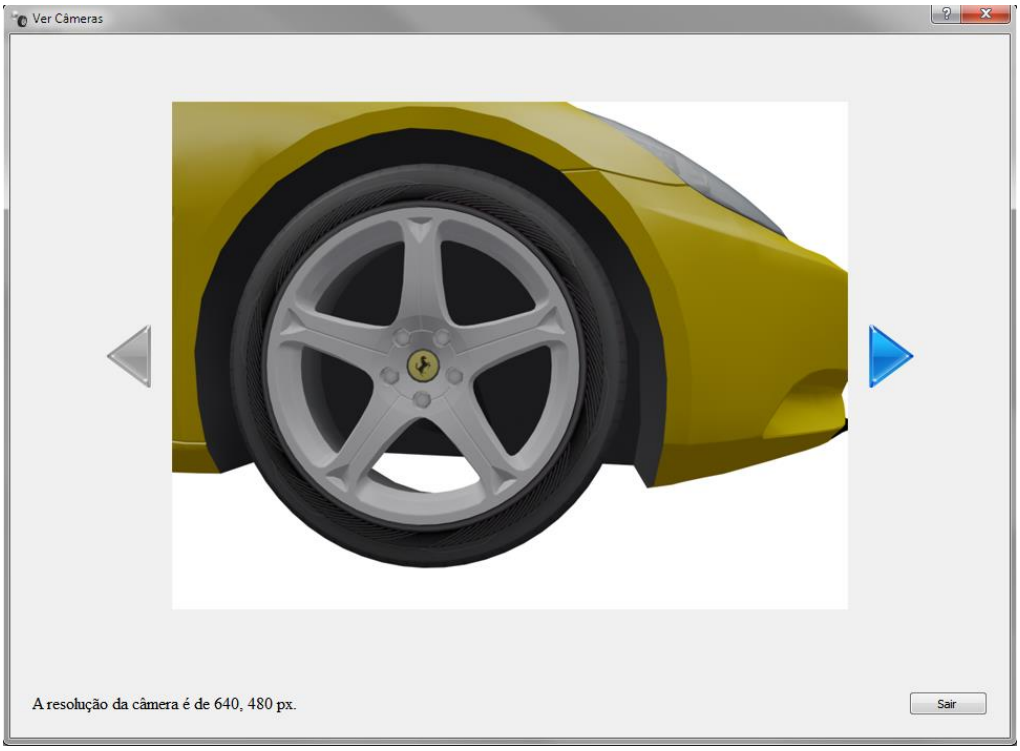

Figura 5.33: Tela para ver todas as câmeras.

Durante o processo de calibração e de reconstrução é possível mudar alguns valores importantes no processamento de imagens assim como apresentar imagens de cada etapa da obtenção dos parâmetros de calibração e da reconstrução da roda em tempo de execução. Para mudar os valores e obter informação adicional sobre tais processos foi realizada uma tela de parâmetros apresentada na Figura 5.34. Nesta tela é possível mudar limites nos algoritmos de procura de bordas, modificar as dimensões da caixa utilizada, a resolução das câmeras, número de pontos a serem utilizadas na reconstrução da roda entre outros. Todos esses valores são mudados sem necessidade de reiniciar a aplicação e são salvos no arquivo ***settings.xml***

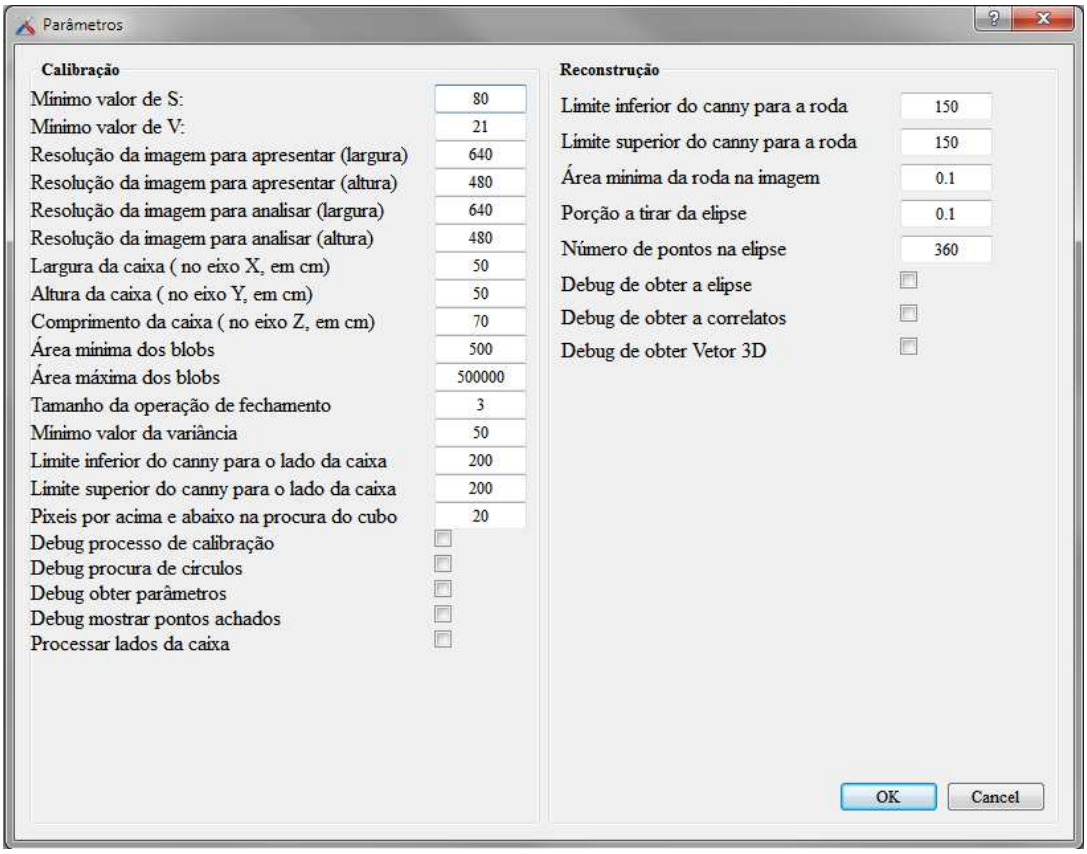

Figura 5.34: Tela parâmetros.

Neste capítulo apresentou-se assim o *software* desenvolvido, bem como os passos que o processo envolve e as diferentes opções de configuração que o usuário pode modificar. No capítulo seguinte será apresentada a arquitetura do *software,* onde serão definidos alguns conceitos de programação e apresentada a organização geral da aplicação.

6 ARQUITETURA DO SOFTWARE DE ALINHAMENTO

Primeiramente define-se o que é arquitetura de um *software.* Mesmo não se tenha uma definição única deste conceito é possível dizer que a arquitetura de *software* de um sistema de computação é o conjunto de estruturas necessárias para raciocinar sobre o sistema, que compreendem elementos de software, as relações entre eles e as propriedades de ambos (CLEMENTS, BACHMANN, *et al.*, 2010). No momento de se desenvolver um sistema ou *software* é importante ter noção clara e detalhada da arquitetura que se vai utilizar. Na literatura existem muitos enfoques e diferentes arquiteturas que são adotadas dependendo do sistema computacional e de seus requerimentos, mas uma característica que quase todos compartilham é a modularização e decomposição.

Com o aumento do tamanho e da complexidade de sistemas de *software* se faz necessária a decomposição dos sistemas em componentes modulares que cumpram uma função especifica, delimitada e bem separada dos demais módulos da aplicação, mas que ao mesmo tempo sejam fáceis de integrar ao sistema completo e permitam uma troca de mensagens padronizada, possibilitando que vários programadores trabalhem simultaneamente no mesmo programa. (GARLAN e SHAW, 1994).

Para cumprir os requisitos acima apresentados, a implementação do sistema de medida dos ângulos descrito no capítulo anterior foi planejada tendo presentes dois conceitos muito utilizados na área de arquitetura de *software*. O primeiro conceito é a Programação Orientada a Objetos (POO), e o segundo conceito e o padrão *Model-View-Controller* (MVC). O programa desenvolvido faz uso destes conceitos desde o começo. Por tal motivo, a primeira parte deste capítulo é dedicada a detalhar esses conceitos.

Na seção 6.2 são apresentadas as livrarias e os pacotes utilizados no software, todos eles de código livre e uso gratuito. Na seção 6.3 descreve-se o programa, como ele está dividido e os procedimentos que o usuário tem que seguir para obter as medidas dos ângulos, assim como a organização das classes e como estas são divididas nas diferentes camadas.

6.1 POO e padrão MVC

A utilização de padrões no desenvolvimento de *software* é muito importante já que facilita o entendimento de como está planejado o programa e como é que ele funciona. Além disso, permite uma interpretação e manutenção do código muito mais fácil e rápida. São por essas razões que foram utilizados dois conceitos de arquitetura de *software* que permitem um nível de abstração muito alto e um padrão de organização que permite dividir os componentes do sistema em camadas.

6.1.1 POO

A Programação Orientada a Objetos ou POO é um paradigma de programação que forma parte da chamada engenharia de *software* baseada em componentes, cuja ideia é produzir componentes reusáveis gerando a possibilidade ao programador de escolher quais componentes utilizar (PAGE-JONES, 1999). Neste paradigma os objetos do mundo real são modelados a partir de suas características (atributos) e suas funcionalidades (métodos). Assim por exemplo, no caso do projeto, uma câmera pode ser modelada a partir de suas características como resolução, posição no espaço, distância focal, etc. Como funcionalidades se podem listar: iniciar a câmera, deter a câmera, obter a imagem, etc.

Na engenharia de *software* é comum representar graficamente as classes a partir do chamado diagrama de UML. O *Unified Modeling Language* (UML) é uma linguagem visual para modelagem de *software* (CLEMENTS, BACHMANN, *et al.*, 2010). Nesta linguagem as classes são representadas em caixas com três partes bem definidas. A Figura 6. apresenta um exemplo de diagrama UML para a classe Câmera. O diagrama se divide em três partes, a primeira parte é o nome da classe, a segunda parte são os atributos da classe, e a ultima parte são os métodos. No digrama da classe é possível ver do que tipo é cada atributo assim como os valores retornados pelos métodos e que parâmetros eles recebem.

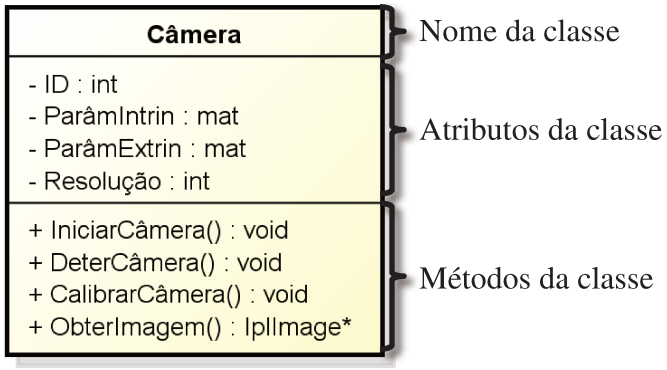

Figura 6.1: Representação gráfica da classe Câmera usando UML.

Na Figura 6.2 apresenta a declaração da classe Câmera na linguagem C++.

```
class Camera
{
private:
    int ID;
    mat ParamIntrin;
    mat ParamExtrin;
    int Resolucao;
public:
    Camera();
    void IniciarCamera();
    void DeterCamera();
    void CalibrarCamera();
    IplImage* ObterImagem();
};

#endif // CAMERA_H
```

Figura 6.2: Declaração da classe Câmera em C++.

A POO tem importantes conceitos que a diferença de outros paradigmas de programação e que são usados no presente trabalho. Os conceitos mais importantes dentro da POO são a herança, o encapsulamento e o polimorfismo.

A herança permite a reutilização de código quando um grupo de classes com caraterísticas similares utiliza os atributos e os métodos de uma classe superior (super-classe). Assim, por

exemplo, a classe círculo e a classe triângulo podem herdar o método acharArea() da classe FiguraGeometrica.

O encapsulamento é usado para realizar a separação de elementos internos e externos da classe. Isto impede que objetos externos à classe modifiquem seus estados ou atributos.

O polimorfismo é a capacidade de ter métodos com o mesmo nome pero com parâmetros diferentes. Isto possibilita ter o mesmo nome para métodos que operam seus parâmetros de forma diferente e obtém resultados também diferentes.

Umas das propriedades mais importantes dentro da POO é a possibilidade de relacionar as classes. Assim por exemplo a classe SistemaStereo tem dois objetos do tipo Câmera, é dizer, tem duas instancias da classe Câmera que representam cada uma das câmeras presentes num sistema de visão estéreo.

A POO tem muitas mais características que não estão contempladas no projeto do *software* desenvolvido neste trabalho, mas que permitem uma grande facilidade na hora de planejar e implementar grandes projetos de *software*.

6.1.2 Padrão MVC

O padrão *Model View Controller* (MVC) tem como proposito essencial fazer a ponte entre o modelo mental humano e o modelo digital que existe no computador (REENSKAUG, 2010). Para lograr esse proposito a aplicação é desacoplada em três camadas que incrementam a flexibilidade e seu reuso. O modelo (*Model*) são os dados da aplicação, nesta camada são definidas as classes que manipulam os dados e arquivos do sistema. O controlador (*Controller*) são as classes que determinam o fluxo da aplicação e aqui são processadas as ações do usuário. Finalmente na camada da vista (*View*) são contidas as classes que apresentam a informação ao usuário, ou seja, são os elementos da interface gráfica do usuário (GUI) (GAMMA, HELM, *et al.*, 1994).

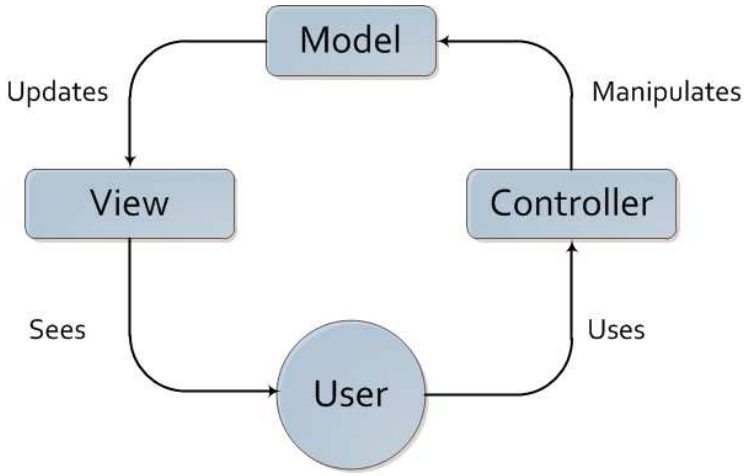

Figura 6.3: Componentes do padrão MVC

A Figura 6.3 apresenta os componentes e as iterações entre eles. No primeiro lugar o usuário gera ações que são processadas pelo controlador, o controlador manipula os dados no modelo, a camada do modelo atualiza os dados e os apresenta ao usuário através da camada de vista.

O *software* apresentado neste trabalho faz uso da POO e do padrão MVC para lograr um código mais simples, claro e fácil de atualizar e manter.

6.2 Pacotes e bibliotecas utilizadas

Nesta seção apresentam-se os pacotes e bibliotecas usadas assim como o *framework* onde foi desenvolvido o *software*.

Um dos requisitos iniciais do programava foi que todas as ferramentas utilizadas no seu desenvolvimento fossem de código livre e gratuitas. Assim foi feita uma pesquisa para determinar quais eram as melhores ferramentas em termos de facilidade de uso e documentação tendo em mente que fossem de código livre.

O *software* foi desenvolvido usando a linguagem de programação C++ no entorno Qt versão 4.8 (QT, 2012). Qt é um *framework* de desenvolvimento multiplataforma para C++ de uso livre. O que significa que o programa pode ser usando nas seguintes plataformas sem ter que mudar o código fonte:

- Microsoft Windows
- Microsoft Windows CE, Windows Mobile
- Symbian
- OS X (Apple OS X; suportando Cocoa)
- X11: X Window System (GNU/Linux, FreeBSD, HP-UX, Solaris, AIX, etc.)
- Embedded Linux
- Maemo, MeeGo
- Wayland

Para e aquisição e o processamento de imagens foi utilizada a biblioteca OpenCv versão 2.3.1. OpenCv (*Open Source Computer Vision*) é uma biblioteca com interfase para C++, C, Python e Java. Suporta Windows, Linux, Mac OS, iOS e Android. Foi projetada para ter eficiência computacional com um grande foco em aplicações em tempo real (OPENCV, 2012).

Para as operações matemáticas de álgebra lineal foi utilizado o pacote Armadillo versão 3.6.0. Ele é um pacote para C++ que permite operações entre matrizes e vetores assim como calculo de inversas, SVD, valores e vetores próprios etc. Foi escolhida porque é de código livre e têm uma sintaxe muito próxima a MatLab, além ser muito fácil de usar e ter uma muito boa documentação. (ARMADILLO, 2012).

Finalmente a documentação completa do código foi realizada utilizando a ferramenta Doxygen. Doxygen é um sistema de documentação para C++, C, Java, C# e PHP entre outros. Ela gera documentação em HTML, para ser visualizada no browser, em PDF, em RTF etc. É de uso muito difundido e muito bem documentado (DOXYGEN, 2012).

Figura 6.4: Bibliotecas utilizadas no *software* desenvolvido

No desenvolvimento do *software* também foi utilizado um pacote de modelado UML chamado Astah. Ele permite fazer a representação UML das classes e suas relações assim como mapas e mentais e organização dos pacotes (ASTAH, 2012).

6.3 Organização do *software*

O *software* proposto no presente trabalho está dividido em três partes. Cada parte é desenvolvida como um programa independente, assim possui-se um executável para o processo de configuração, outro para o processo de calibração, e outro para o processo de medidas dos ângulos. Para lançar cada executável foi desenvolvido um programa principal que atua como menu, apresentado na Figura 5.21.

O *software* está completamente orientado a objetos, e cada parte compartilha classes de uso comum. A Figura 6.5 apresenta a organização das classes comuns.

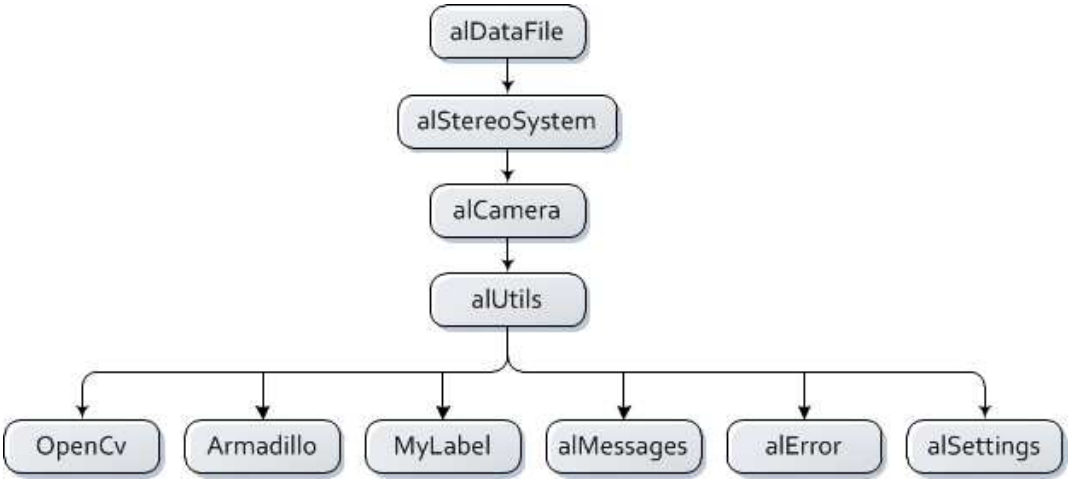

Figura 6.5: Classes comuns no sistema

6.3.1 Classes comuns

As classes comuns contem as funções e propriedades que são usadas nos três programas. As classes das bibliotecas de *OpenCv* e *Armadillo* são a base do programa. A classe *MyLabel* herda os métodos e propriedades da classe *QLabel* do *framework* Qt. E permite definir um novo *QLabel* que possui a possibilidade de obter as coordenadas onde o usuário clicou no processo de calibração de cores. A classe *alMessages* possui todas as mensagens que serão apresentadas ao usuário durante o processo de calibração e medidas dos ângulos. A classe *alError* possui só um campo estático do tipo *QString* que armazena qualquer mensagem de erro gerado no *software.* Finalmente a classe *alSettings* armazena todos os valores estabelecidos pelo usuário e que determinam o comportamento dos algoritmos de processamento de imagens, Figura 5.34.

A classe *alUtils* inclui todas as classes anteriormente apresentadas e contem funções e métodos de uso comum em todas as outras classes. Esta classe possui funções para a visualização de imagens, conversão de sistema de cores, cálculo de equações de retas, cálculo de regressões entre outras.

A classe *alCamera* possui as propriedades e métodos que modelam uma câmera real. Entre suas propriedades destacam-se as matrizes de parâmetros intrínsecos e extrínsecos, a resolução das imagens obtidas, seu número dentro do sistema de alinhamento entre outros. Entre seus métodos destacam-se a obtenção dos parâmetros do sistema estéreo, o método de calibração, a obtenção das imagens, a inicialização e detenção da câmera etc.

A classe *alStereoSystem* possui dois objetos do tipo *alCamera* e modela um sistema estéreo. Dentro de suas propriedades mais importantes estão o vetor de translação e a matriz de rotação de uma câmera com respeito à outra. Ela possui os métodos utilizados para a reconstrução da roda: obter as elipses, obter pontos correlatos e reconstrução 3-D.

Finalmente a classe *alDatFiles* é a classe que lê e escreve os arquivos gerados durante todo o programa. Ela possui as funções que permitem verificar se um arquivo é gerado ou lido corretamente. É importante destacar que está é a única classe que tem acesso aos arquivos no disco.

O conjunto de classes anteriormente descritas são comuns para as outras três classes que controlam cada uma das etapas do processo. Nas seguintes seções são descritas essas classes.

6.3.2 Classe alConfig

A primeira etapa do processo é a configuração do sistema. Aqui são selecionadas as câmeras que serão usadas no processo e são assignadas suas posições dentro do sistema. A classe *alConfig* possui as propriedades e métodos que permitem completar esta tarefa. Na Figura 6.6 é apresentado o diagrama UML da classe *alConfig.* Aqui o método *alSaveCamerasInfo()* invoca ao método da classe *alDataFiles* para salvar a informação da configuração realizada pelo usuário.

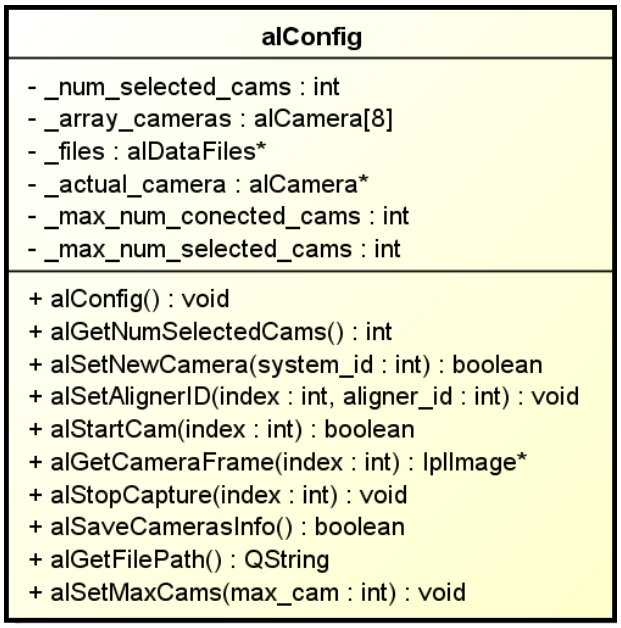

Figura 6.6: Diagrama UML da classe *alConfig*

6.3.3 Classe alCal

O processo de calibração é realizado utilizando os métodos e as propriedades desta classe. Primeiramente são obtidos os valores das cores no sistema HSV. Depois é verificado o funcionamento das câmeras, é estabelecida a resolução de cada câmera e são obtidas suas imagens para posteriormente começar o processo de calibração chamando ao método *alCalibrate()* na classe *alCamera*. A Figura 6.7 apresenta o diagrama UML da classe *alCali*

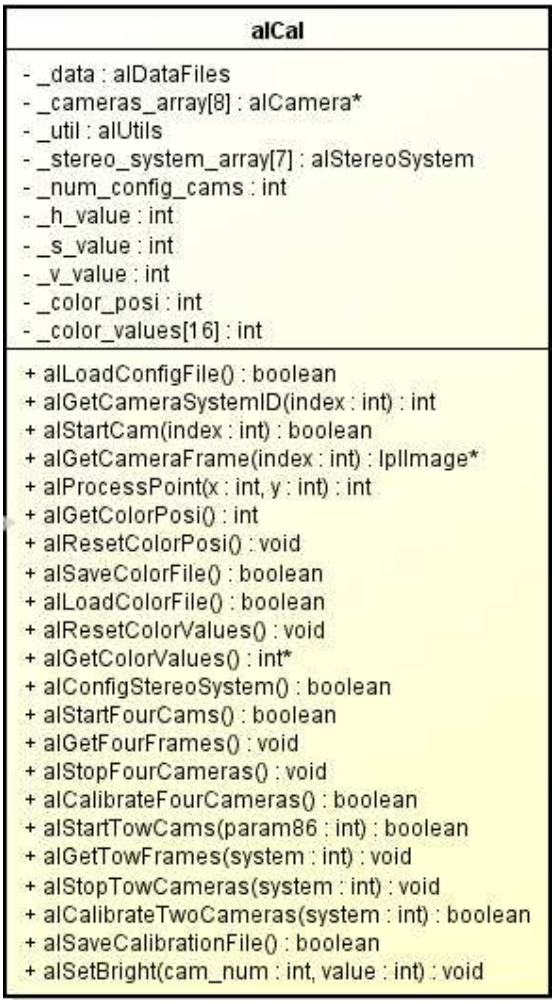

Figura 6.7: Diagrama UML da classe *alCali*

6.3.4 Classe alRecons

Esta classe possui os métodos para a obtenção dos ângulos de *camber* e *toe* a partir da reconstrução das rodas e da aplicação do algoritmo de PCA. O método *alGetAngles(...)* inicializa as câmeras, obtém as imagens e as processa para obter a reconstrução 3-D de cada roda. Depois é aplicado o algoritmo de PCA para obter o vetor normal ao plano gerado pela roda e, após o processamento vetorial, obter os ângulos de alinhamento procurados. A Figura 6.8 apresenta o diagrama UML da classe *alRecons.*

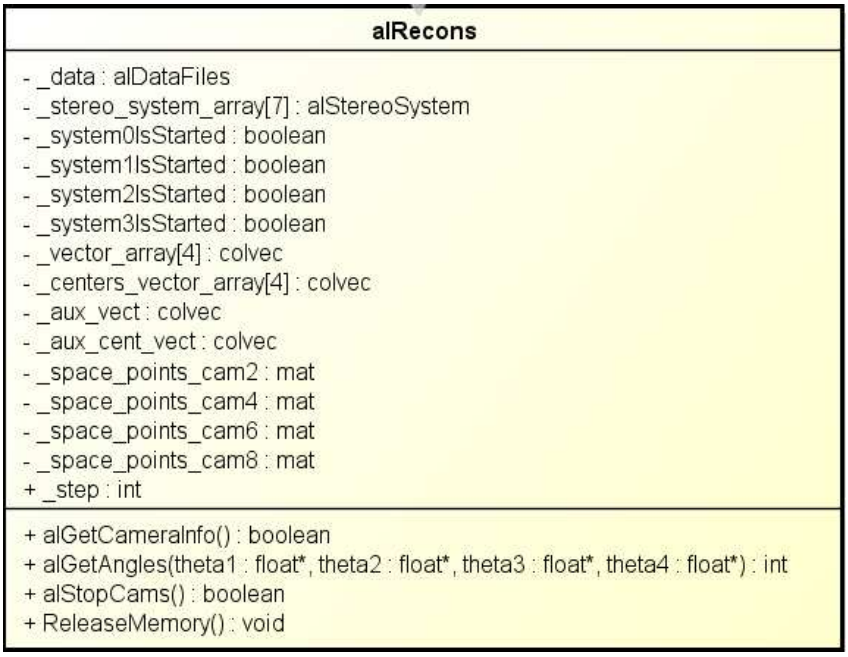

Figura 6.8: Diagrama UML da classe *alRecons*

6.3.5 MVC no *software*

O padrão MVC foi utilizado no desenvolvimento do software para organizar, separar e reutilizar o código de forma mais eficiente, permitido realizar tarefas como atualização e manutenção do código de formas mais rápida e simples.

As classes utilizadas foram separadas nos três grupos do padrão MVC. A Figura 6.9 apresenta as classes organizadas nas três camadas do padrão. As classes *Configuration, Calibration* e *Align* declaram as três telas apresentadas ao usuário. Elas compõem a camada de *View* no padrão MVC. Elas respondem às ações do usuário chamando os métodos da camada do *controller*. A camada do *controller* está composta pelas classes *alConfig*, *alCal* a *alRecons*. As calsses da camada do *controller* determinam o fluxo do programa e as chamadas dos algoritmos de processamento de imagens. A terceira camada, *model,* é composta por uma única classe de

aceso a dados. A classe *alDataFiles* é a única que salva e lê os arquivos gerados durante o todo o processo.

Esta forma de organizar o programa permite, por exemplo, modificar os nomes dos arquivos ou mudar a interface do usuário sem necessidade de mudar muito código, devido a que a logica do programa permanece igual.

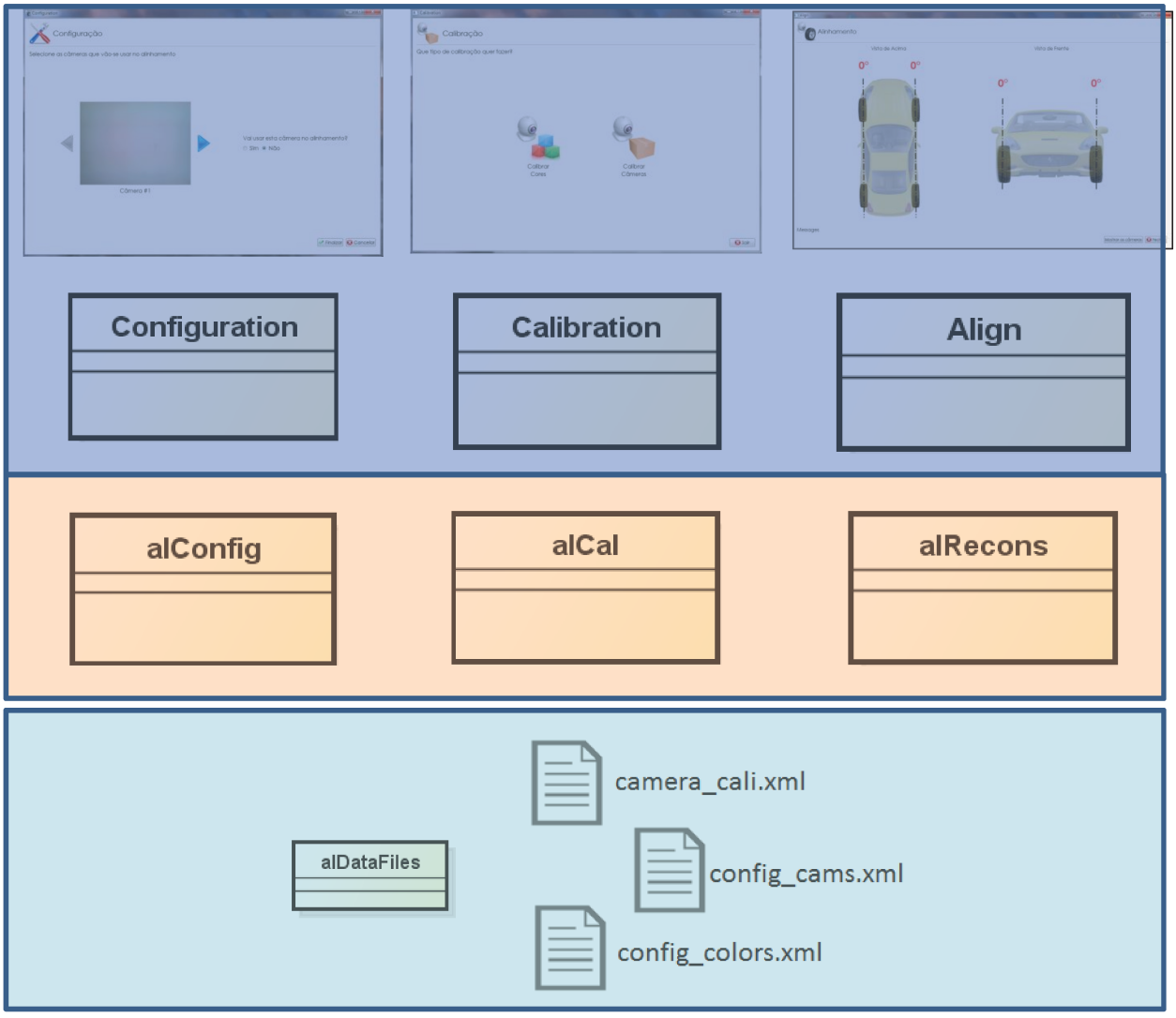

Figura 6.9: Padrão MVC no *software* de medida de ângulos, camada *view* em azul, camada *controller* em vermelho e camada *model* em verde.

7 RESULTADOS

Neste capitulo serão apresentados resultados de utilização do *software* desenvolvido usando imagens virtuais e reais.

7.1 Resultados utilizando imagens virtuais.

O *software* foi testado seguindo o mesmo procedimento descrito no Capitulo 5. Para obter as imagens virtuais elaborou-se um modelo virtual tridimensional do sistema para testes, Figura 7.. As imagens são obtidas a partir de câmeras virtuais com as mesmas características da câmera real: resolução de 1280x720 pixels com um ângulo de visão (FOV) de 75°.

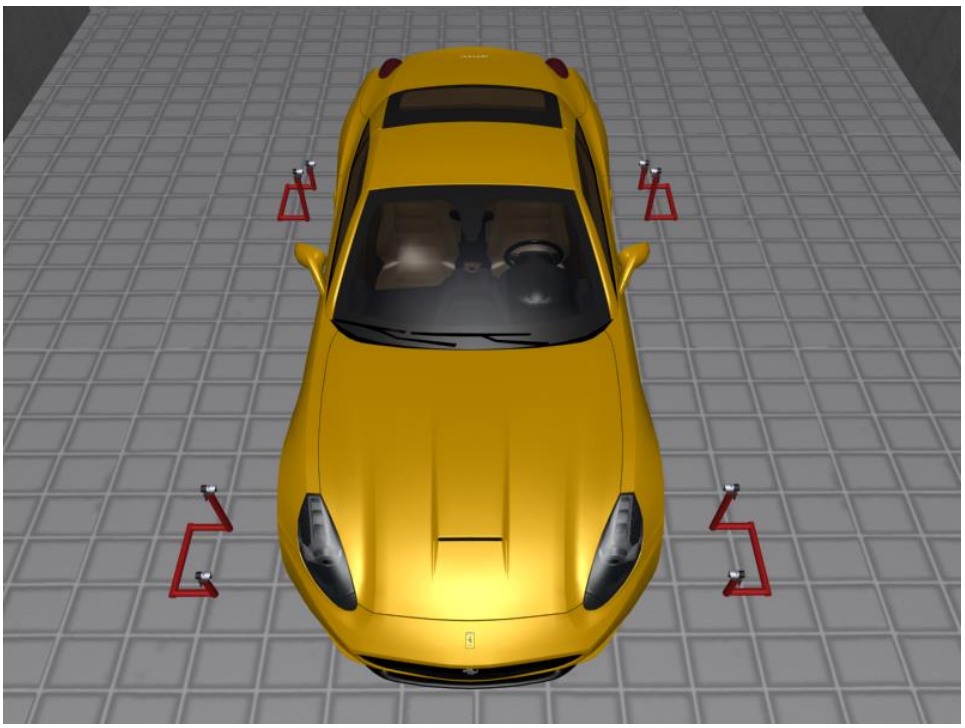

Figura 7.1: Entorno virtual 3-D para testes do programa

O procedimento de calibração é realizado utilizando duas caixas padrão vazada cujas características são descritas na Tabela 7.1, a caixa maior é utilizada para a calibração global. Esta caixa necessita ser visualizada pelas câmeras de referencia que estão distantes cerca de 2,0m da caixa de calibração global. A caixa menor é utilizada para a calibração local. As dimensões das caixas de calibração são definidas baseadas no critério de que as mesmas devem ser visíveis pelas câmeras posicionadas no ambiente de medição, ocupando a maior área possível do campo de imagem de cada câmera.

Tabela 7.1: Características das caixas virtuais.

Caixa calibração global		Caixa calibração local	
Dimensão	Valor (cm)	Dimensão	Valor (cm)
Largura	65	Largura	17
Altura	75	Altura	17
Profundidade	85	Profundidade	27
Rádios das esferas	7.5	Rádios das esferas	2.5
Radio dos lados	2.5	Radio dos lados	1.25

Foram feitos testes do algoritmo de calibração apresentado neste trabalho para comparar sua precisão frente ao algoritmo de calibração presente no *toolbox* de MatLab. Na Figura 7.2 é apresentado o modelo da caixa vazada gerada em 3dsMax. O teste consistiu em calibrar 10 vesses a câmera usando o algoritmo proposto neste trabalho e comparar os valores da matriz dos parâmetros intrínsecos com os valores obtidos da calibração, da mesma câmera, usando um padrão plano, Figura 7.3, e o algoritmo de calibração presente no *toolbox* de MatLab. Foram utilizadas imagens de 800x600 *px* com a caixa e o tabuleiro em diferentes posições.

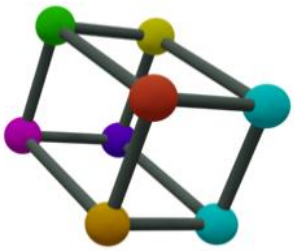

Figura 7.2: Caixa vazada usada nos teste de calibração

Figura 7.3: Tabuleiro usando nos testes do *toolbox* de MatLab.

Os resultados do teste são resumidos na Tabela 7.2. Dos dados na tabela pode-se concluir que os parâmetros da calibração intrínseca com os dois métodos deram um máximo erro de ± 0.032 para os valores meios. Também pode-se concluir que o algoritmo apresenta uma boa estabilidade com desvios máximo de 0,0021.

Tabela 7.2: Comparação numérica dos algoritmos de calibração

	OpenCv				*MatLab*			
	Câmera 1		Câmera 2		Câmera 1		Câmera 2	
Exp#.	Parâmetros Intrínsecos (10^3)		Parâmetros Intrínsecos (10^3)		Parâmetros Intrínsecos (10^3)		Parâmetros Intrínsecos (10^3)	
	Sx	u	Sx	u	Sx	u	Sx	u
	Sy	v	Sy	v	Sy	v	Sy	v
Ex1.	1,1677	0,3997	1,1677	0,3998	1,2037	0,4133	1,1613	0,3989
	1,1676	0,2982	1,1676	0,2984	1,1941	0,2784	1,1621	0,2733
Exp2.	1,1697	0,3966	1,1697	0,3963	1,1815	0,4062	1,1936	0,3811
	1,1698	0,2979	1,1698	0,2977	1,1722	0,2701	1,1936	0,2760
Exp3.	1,1672	0,4003	1,1671	0,4004	1,1967	0,41717	1,2130	0,3719
	1,1669	0,2992	1,1668	0,2993	1,1883	0,29177	1,2116	0,2702
Exp4.	1,1661	0,4005	1,1662	0,4004	1,2063	0,3873	1,1841	0,4025
	1,1661	0,2997	1,1662	0,2996	1,2016	0,2707	1,1840	0,2663
Exp5.	1,1638	0,3986	1,1639	0,3983	1,2051	0,4166	1,1755	0,3730
	1,1651	0,3027	1,1654	0,3026	1,1997	0,2754	1,1785	0,2701
Exp6.	1,1687	0,3984	1,1688	0,3984	1,2042	0,4059	1,2177	0,3624
	1,1690	0,2988	1,1691	0,2988	1,1965	0,2608	1,2210	0,2672
Exp7.	1,1634	0,4004	1,1634	0,4004	1,2147	0,3904	1,1724	0,3877
	1,1634	0,3017	1,1634	0,3017	1,2075	0,2831	1,1705	0,2887
Exp8.	1,1696	0,3985	1,1695	0,3984	1,2032	0,4079	1,1802	0,3715

	1,1699	0,2978	1,1698	0,2978	1,1948	0,2868	1,1852	0,2756
Exp9.	1,1682	0,3989	1,1684	0,3987	1,1768	0,4048	1,1759	0,3748
	1,1681	0,2989	1,1683	0,2989	1,1689	0,2747	1,1710	0,2672
Exp10.	1,1664	0,3991	1,1662	0,3992	1,1989	0,4198	1,1927	0,3673
	1,1659	0,2998	1,1657	0,2999	1,1903	0,2755	1,1955	0,2591
OpenCv					*MatLab*			
	Câmera 1		Câmera 2		Câmera 1		Câmera 2	
	Parâmetros Intrínsecos (10^3)		Parâmetros Intrínsecos (10^3)		Parâmetros Intrínsecos (10^3)		Parâmetros Intrínsecos (10^3)	
Media	1,1671	0,3991	1,1671	0,3990	1,1991	0,4076	1,1857	0,3770
	1,1672	0,2995	1,1672	0,2995	1,1914	0,2760	1,1865	0,2706
Desvio	0,0021	0,0013	0,0021	0,0014	0,0122	0,0116	0,0190	0,0147
	0,0020	0,0016	0,0020	0,0016	0,0130	0,0081	0,0192	0,0100

A Figura 7.4 apresenta os resultados do *software* para um ângulo de toe de -20° para a roda dois e um ângulo de camber de 20° para a roda um.

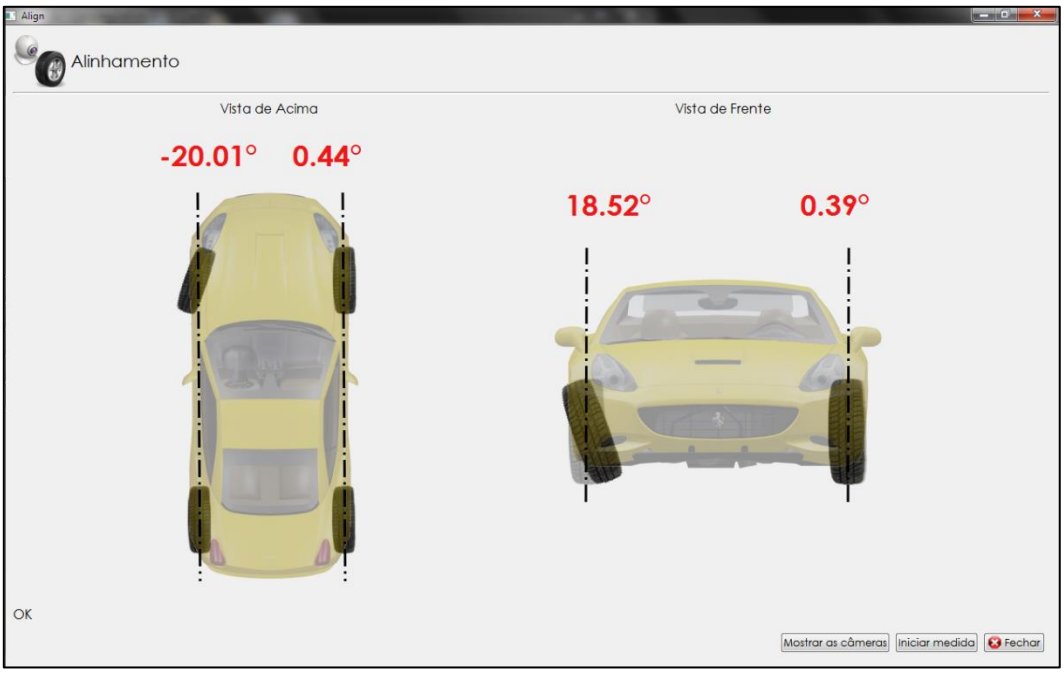

Figura 7.4: Apresentação dos resultados no *software*.

Na Tabela 7.3 apresentam-se os resultados obtidos pelo *software* para um ângulo de 20° tanto para *toe* quanto para *camber* nas duas rodas.

Tabela 7.3: Ângulos medidos com imagens virtuais de 1280x720 pixels.

Dados introduzidos				Dados medidos			
Roda 1		Roda 2		Roda 1		Roda 2	
Toe	*Camber*	*Toe*	*Camber*	*Toe*	*Camber*	*Toe*	*Camber*
0°	0°	0°	0°	0.86°	-1.07°	-0.31°	0.12°
20°	0°	0°	0°	19.21°	-0.74°	-0.31°	0.12°
-20°	0°	0°	0°	-18.34°	-1.42°	-0.31°	0.12°
0°	20°	0°	0°	0.45°	18.52°	-0.31°	0.12°
0°	-20°	0°	0°	1.03°	-20.81°	-0.31°	0.12°
0°	0°	20°	0°	0.86°	-1.07°	19.04°	-0.18°
0°	0°	-20°	0°	0.86°	-1.07°	-20.01°	0.39°
0°	0°	0°	20°	0.87°	-1.07°	-0.42°	19.78°
0°	0°	0°	-20°	0.85°	-1.08°	-0.62°	-19.64°
0°	0°	-20°	20°	0.87°	-1.07°	-20.09°	20.89°

7.2 Resultados utilizando imagens reais

Para testar o programa foram fabricadas as estruturas que suportam as câmeras, o cubo de calibração grande, o cubo de calibração pequeno e foi utilizada uma camionete. Os testes foram realizados na faculdade de Engenharia Mecânica da UNICAMP. A Figura 7.5 apresenta a montagem para os testes.

Figura 7.5: Montagem para os testes com imagens reais

A caixa de calibração global tem as características apresentadas na Tabela 7.4 e é apresentada na Figura 7.6. A caixa de calibração local tem as características apresentadas na Tabela 7.5 e é apresentada na Figura 7.7.

Tabela 7.4: Características da caixa de calibração global

Caixa de calibração global	
Dimensão	Valor (cm)
Largura	65
Altura	75
Profundidade	85
Rádios das esferas	7.5
Radio dos lados	2.5

Figura 7.6: Caixa de calibração global.

Tabela 7.5: Características da caixa de calibração local

Caixa de calibração local	
Dimensão	Valor (cm)
Largura	17
Altura	17
Profundidade	27
Rádios das esferas	2.5
Radio dos lados	1.25

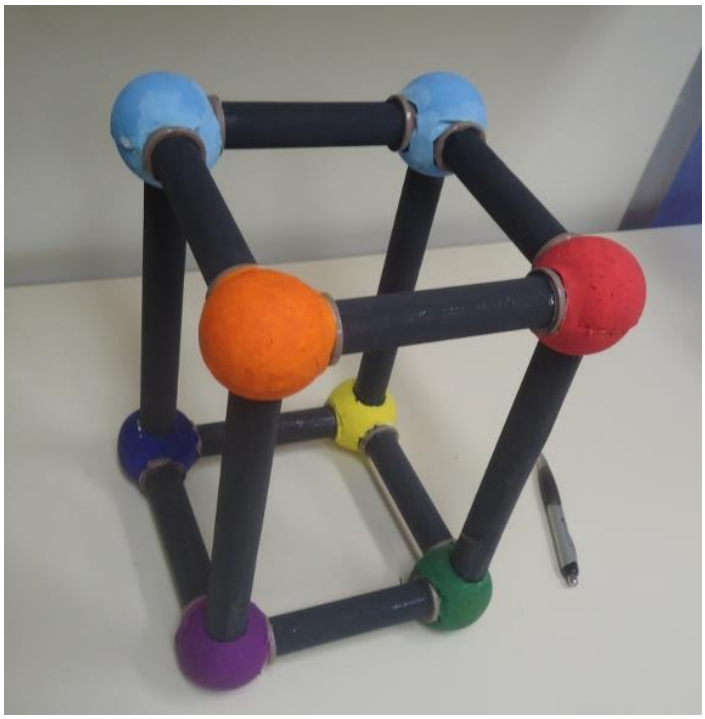

Figura 7.7: Caixa de calibração local

Após completar a montagem das estruturas com as câmeras executou-se o *software*. Na tela inicial, apresentada na Figura 7.8, selecionou-se o botão “Configurar”. Aqui é executado o programa para configurar as câmeras e estabelecer a posição das mesmas no sistema de alinhamento. Na tela apresentada na Figura 7.9 foram selecionadas as câmeras que foram utilizadas no teste. Na Figura 7.10 é apresentada a tela onde foram numeras as câmeras de acordo com a foto do carro na direita.

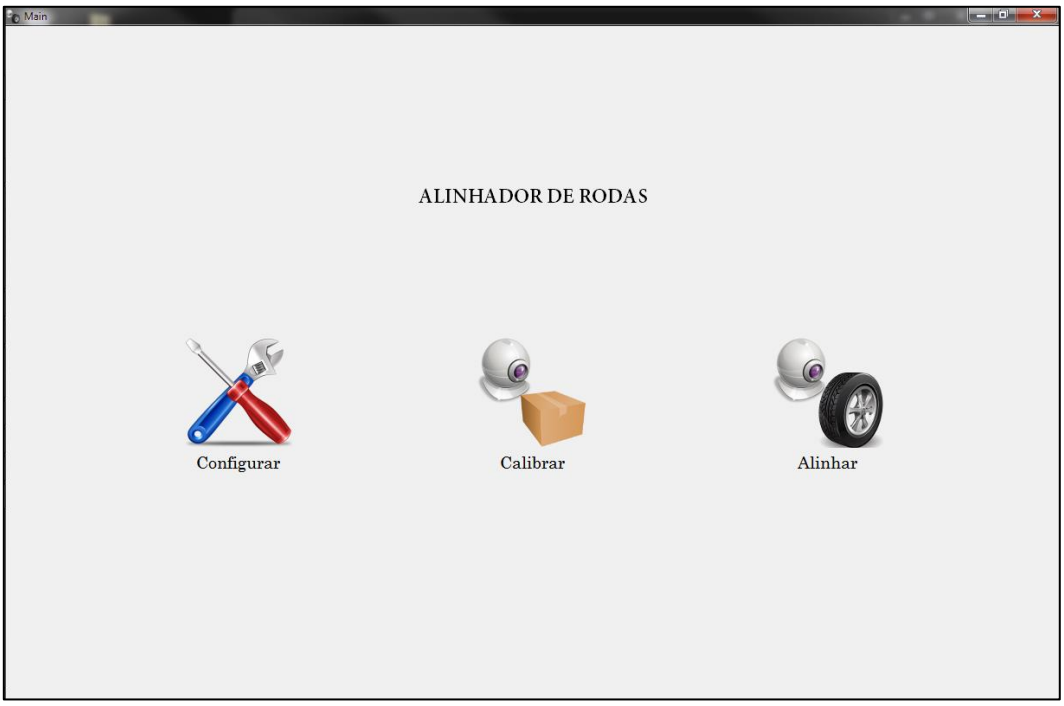

Figura 7.8: Tela inicial do programa.

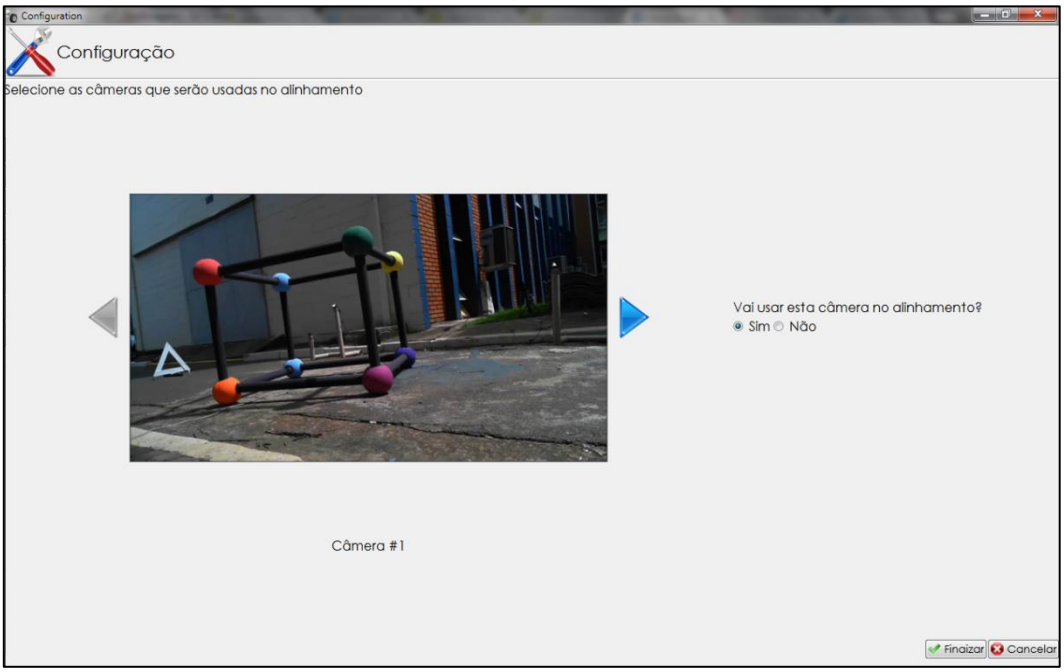

Figura 7.9: Configuração das câmeras.

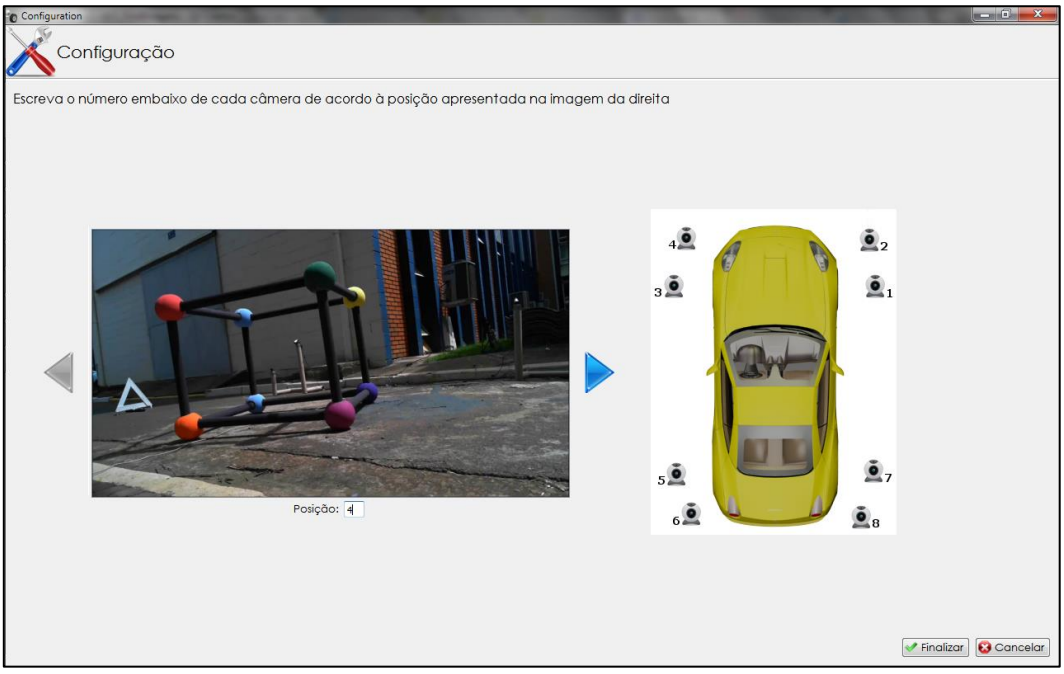

Figura 7.10: Numeração das câmeras.

Terminado o processo de configuração foi gerado um arquivo chamado ***config_cams.xml*** com a informação inserida. Depois foi realizada a calibração de cores. O primeiro passo é estabelecer os valores H das cores nas esferas. Foram calibradas todas as cores utilizando a caixa de calibração global. Na Figura 7.11 é apresentada a calibração feita para a cor Violeta. Na Figura 7.12 é apresentada a tela de verificação da calibração de cores, aqui foram selecionados os círculos na imagem na ordem apresentada na tabela da direita.

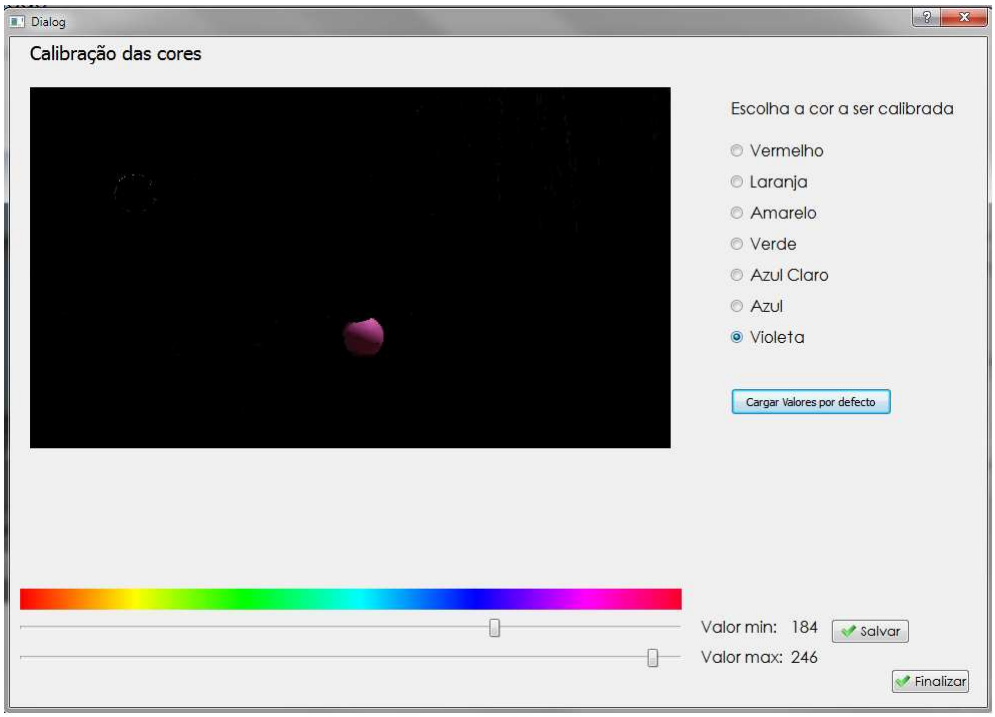

Figura 7.11: Calibração das cores.

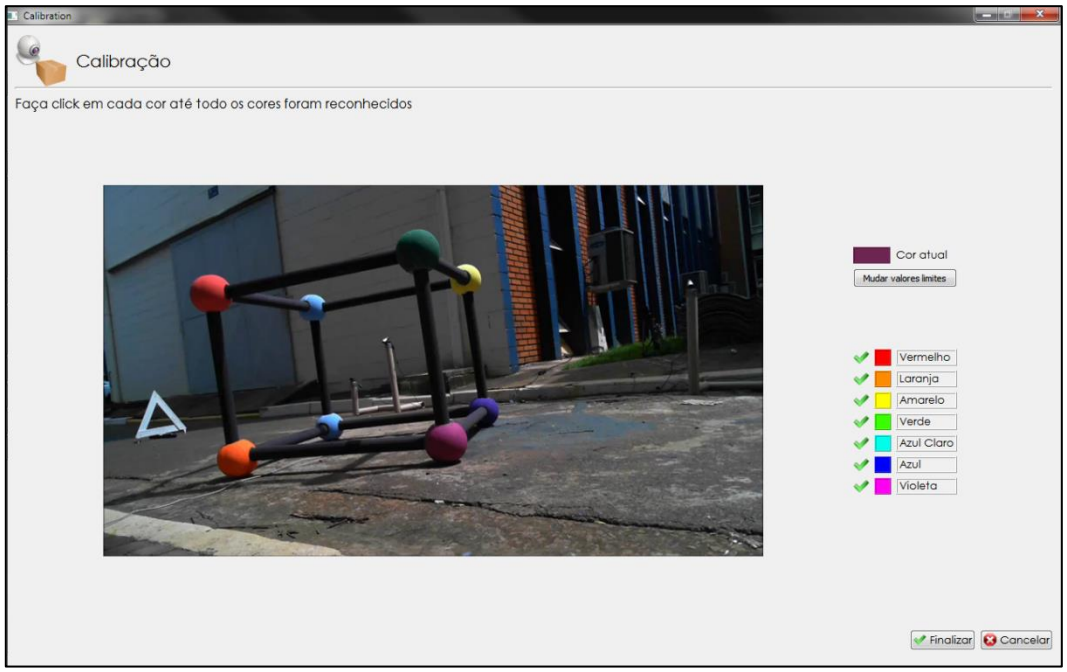

Figura 7.12: Verificação da calibração de cores

Após calibrar as cores foi realizada a calibração das câmeras. A Figura 7.13 apresenta a tela de calibração global com as imagens na caixa maior. O programa possui a opção de apresentar para o usuário as imagens geradas durante os processos de calibração e reconstrução. Após posicionar a caixa no centro do sistema de forma que seja visualizada pelas quatro câmeras de referência, inicia-se o processo de calibração. A Figura 7.14 apresenta a primeira parte do processo de calibração, aqui são reconhecidas as oito esferas usando só a informação das cores. A Figura 7.15 apresenta o segundo passo na calibração global, a obtenção da posição do centro das esferas usando os lados da caixa, na imagem procura-se o lado que une a esfera verde e vermelha. Na Figura 7.16 em cor verde e vermelho são apresentadas as duas retas obtidas do lado da caixa, em azul esta desenhada a reta media que se intersectará com a reta obtida do lado que une a esfera vermelha com a ciam superior ou com a reta do lado que une a esfera vermelha com a laranja. A interseção das duas retas de lados diferentes proporciona uma localização mais exata do centro da esfera.

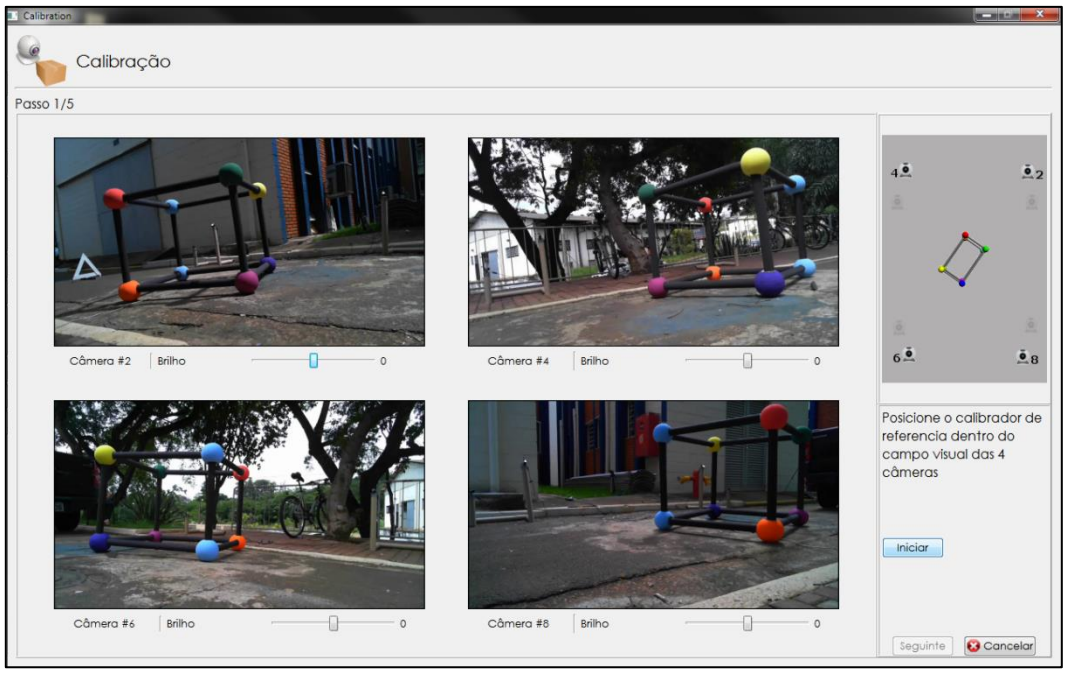

Figura 7.13: Inicio da calibração global.

Figura 7.14: Identificação das esferas.

Figura 7.15: Identificação do lado da caixa.

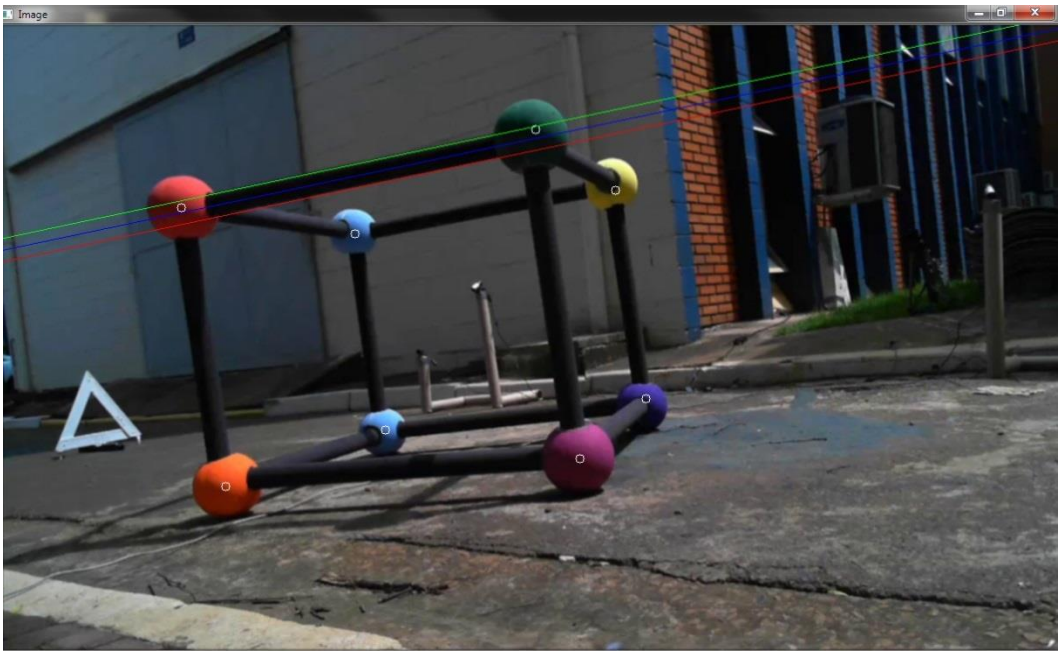

Figura 7.16: Retas obtidas ao processar o lado da caixa que une a esfera verde com a esfera vermelha.

Figura 7.17: Retas obtidas ao processar o lado da caixa que une a esfera laranja com a esfera vermelha.

A Figura 7.18 apresenta, em azul, o centro da esfera verde obtida só usando a informação da cor, e em vermelho o centro obtido usando a interseção de duas retas geradas a partir da informação dos lados da caixa. Pode observar que usando a informação dos lados da caixa pode-se obter uma posição mais precisa do centro da esfera.

Figura 7.18: O circulo azul é o centro da esfera verde obtida usando só a informação da cor e em vermelho usando a informação dos lados da caixa.

A Figura 7.19 apresenta a tela de calibração global finalizada corretamente.

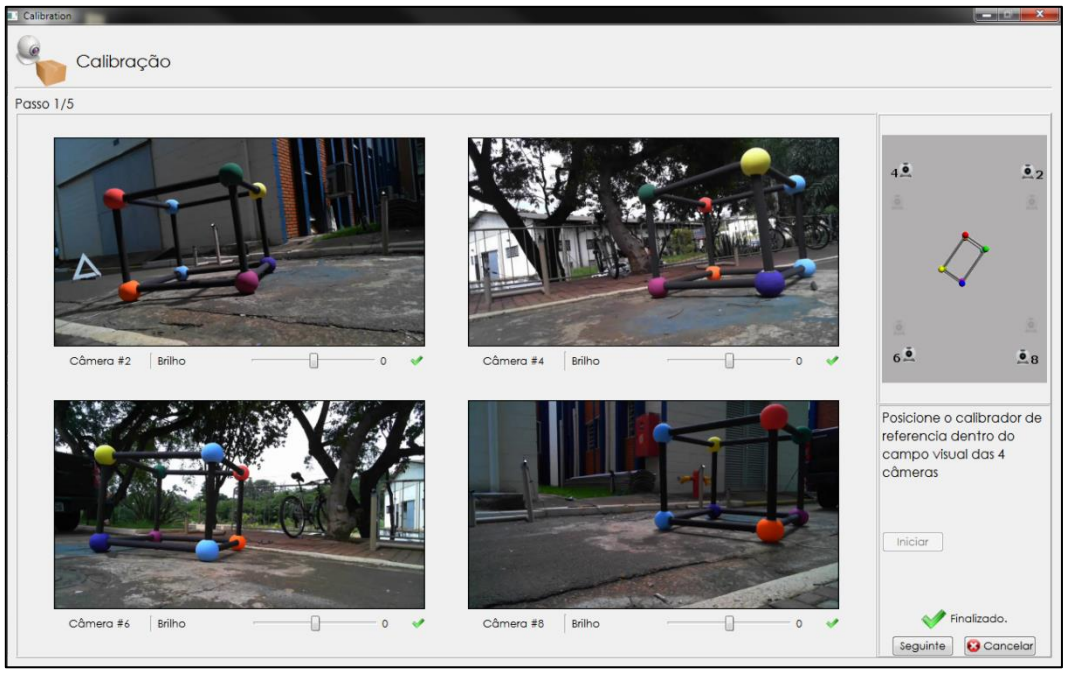

Figura 7.19: Processo de calibração global finalizado corretamente.

Finalizado o processo de calibração global é possível continuar com o a calibração local. A Figura 7.20 apresenta o posicionamento da caixa de calibração local na frente das câmeras um e dois. A Figura 7.21 apresenta a tela de calibração local finalizada corretamente.

Figura 7.20: Posicionamento da caixa de calibração local.

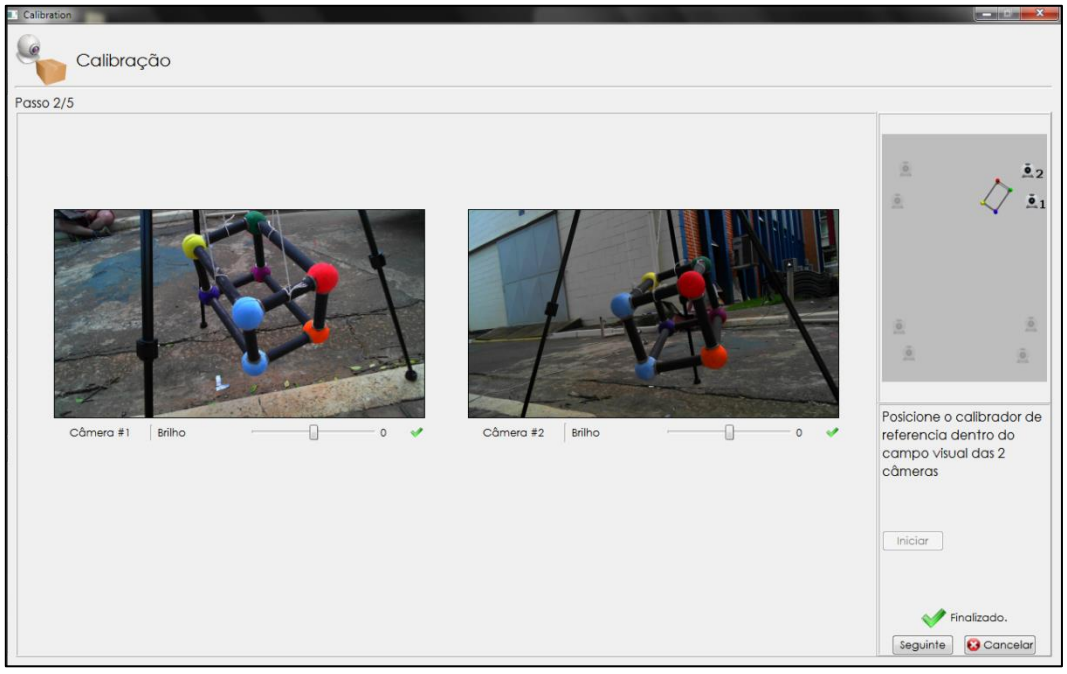

Figura 7.21: Processo de calibração local finalizado corretamente para as câmeras um e dois.

O processo de calibração local é repetido para cada sistema estéreo, a caixa foi movimentada à frente de cada câmera e foram realizadas as quatro calibrações locais. Após o processo de calibração foi terminado pode-se proceder ao processo de obtenção dos ângulos.

A obtenção dos ângulos começa com o posicionamento do carro na frente das câmeras. São realizadas três medidas de alinhamento das rodas dianteiras de um veículo, cuja condição de alinhamento é desconhecida . A primeira medida é feita com as rodas esterçadas ligeiramente para a esquerda, (figura 7.31). Na segunda medida as rodas são esterçadas ligeiramente para a direita (fig. 7.32). Na terceira medida as rodas são esterçadas fortemente para a esquerda (fig. 7.33). O *software* obtém inicialmente os vetores normais às rodas traseiras direita e esquerda. Caso esses vetores sejam calculados corretamente, o *software* define então o vetor de movimentação do veículo e prossegue para o processamento da informação das duas rodas dianteiras.

A Figura 7.22 apresenta o contorno da roda obtida pelo *software* na imagem da câmera número sete. O processo também é realizado para a câmera oito, são obtidas as equações das elipses nas duas imagens. Após obter as elipses é eliminada uma porção superior e inferior de cada uma delas e são obtidos os pontos correlatos como foi apresentado na seção 4.4. A Figura 7.23 apresenta os pontos correlatos obtidos para as imagens das câmeras sete e oito.

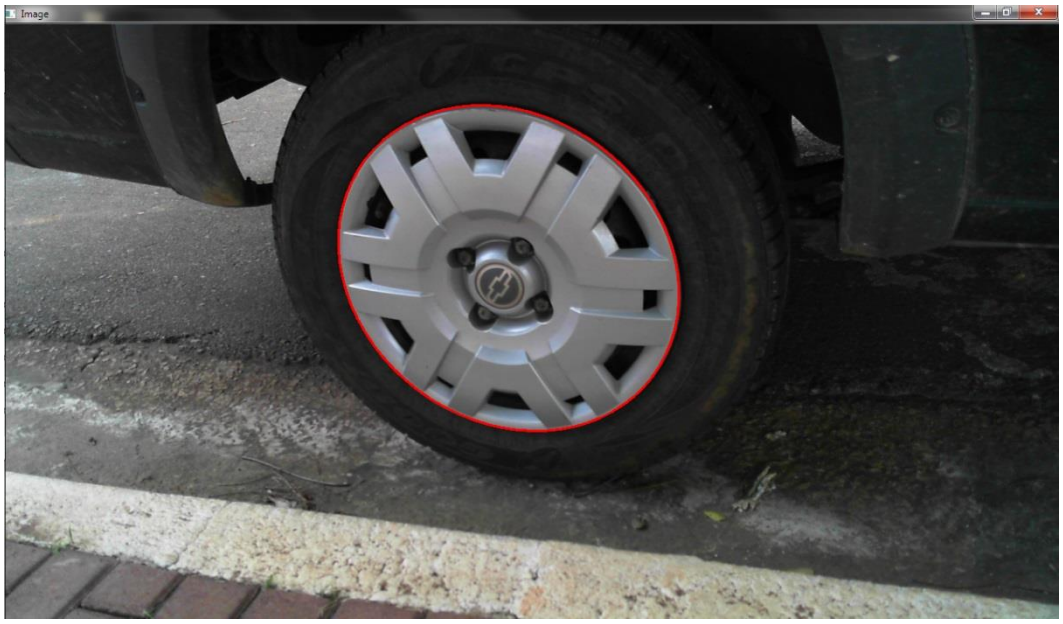

Figura 7.22: Contorno da roda traseira direta obtido pelo *software*.

Figura 7.23: Pontos correlatos na roda traseira direita.

O processo é repetido para as quatro rodas e finalmente é processada a informação dos vetores para obter os ângulos procurados.

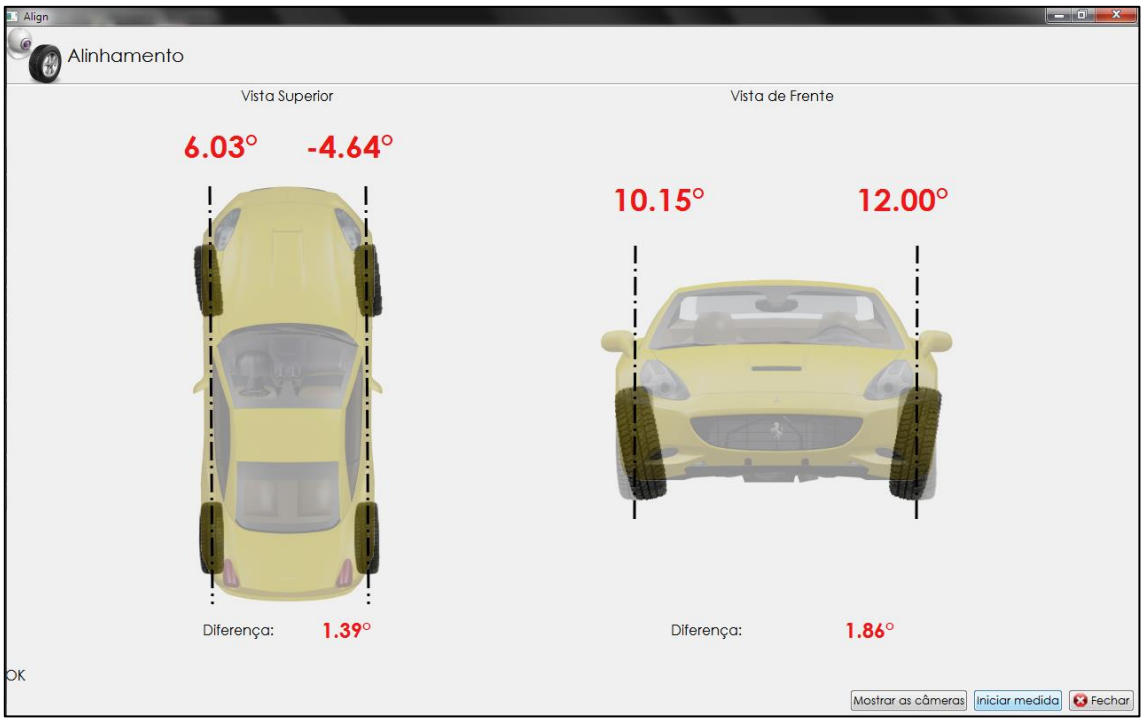

Figura 7.24: Ângulos medidos pelo *software* quando a direção esta centrada.

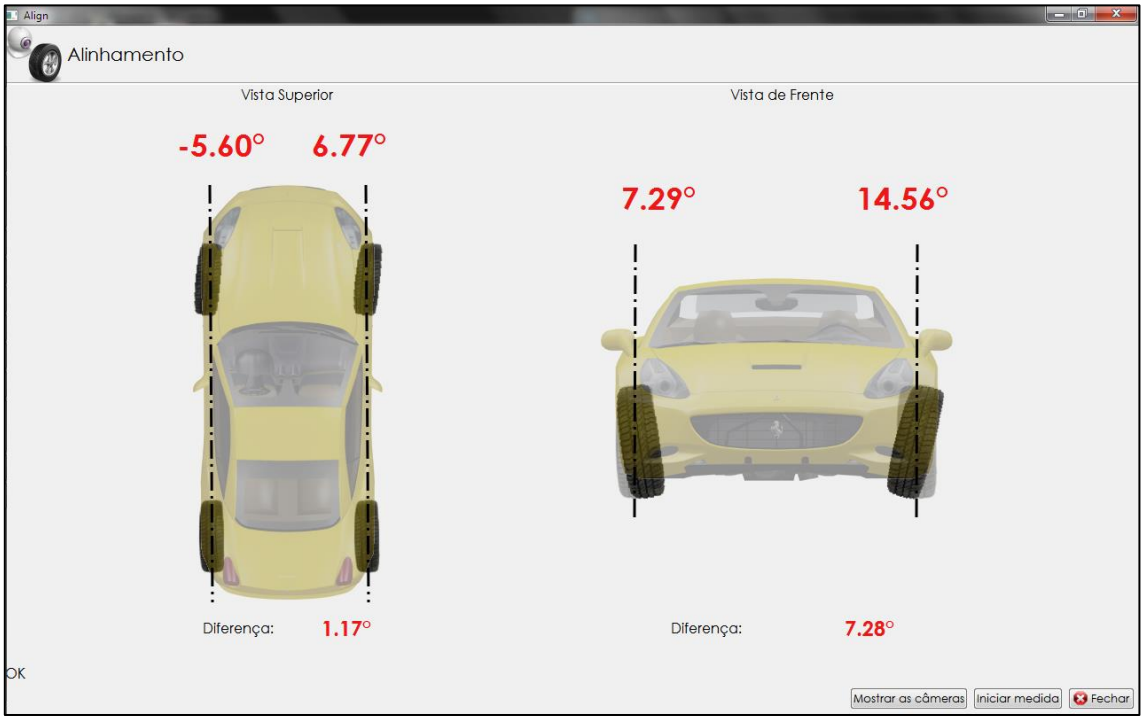

Figura 7.25: Ângulos medidos quando a direção esta virada para a direita.

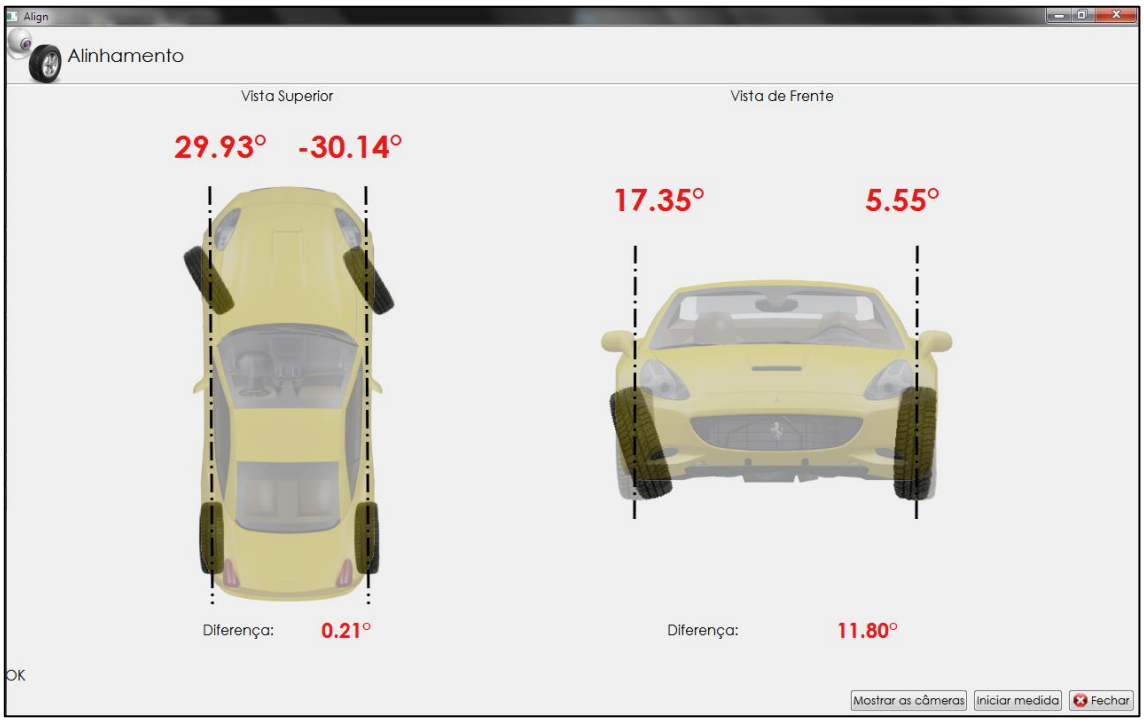

Figura 7.26: Ângulos medidos quando a direção esta virada para a esquerda.

Um resumo das medidas realizadas é apresentado na Tabela 7.6. As medidas de convergência/divergência, especificamente para os pequenos ângulos de esterçamento das rodas (medidas 1 e 2), revela um ângulo relativo médio de divergência de 1,25 graus, compatível com o esperado em veículos de tração dianteira. A medida de *camber* das rodas é em média 8.72 graus para a roda direita e 13,28 graus para a roda esquerda, o que pode indicar a necessidade de correção. A grande variação do ângulo de *camber* da roda esquerda, na condição de maior esterçamento das rodas, indica uma angulação de *caster* que também necessita ser corrigida.

Tabela 7.6: Resumo das medidas

Especif. Medida	CONV/DIV (graus)	CONV/DIV TOTAL (graus)	*CAMBER* (graus)
Med.1 - DD	-4,64	1.39	10,15
Med.1 - DE	6,03		12,00
Med.2 - DD	6,77	1.17	7,29
Med.2 - DE	-5,60		14.56
Med.3 - DD	-30,14	-0.21	17,35
Med.3 - DE	29,93		5,55

7.3 Documentação do *software*

A documentação de um software é umas das partes mais criticas e mais importantes já que permite um fácil entendimento e atualização do código por parte de futuros programadores. Neste projeto foi utilizada a ferramenta de *software* livre Doxigen (DOXYGEN, 2012) para a criação de uma documentação padronizada. Esta ferramenta permite, a traves de palavras chaves, a descrição de métodos, parâmetros, funções, variáveis, propriedades etc. A Figura 7.27 apresenta a documentação de uma função que calcula os parâmetros de uma reta a partir de dois pontos sobre ela.

```
/*! \brief Calculates the line equation
    \param[in] x1,y1 x and y coordinates of the first point
    \param[in] x2,y2 x and y coordinates of the second point
    \param[out] m The line slope
    \param[out] b The y-intercept of the line.
    \return True if was successful and false if the line is vertical and the slope is infinity
*/
bool EqRec(float x1, float y1, float x2, float y2, float* m, float* b);
```

Figura 7.27: Documentação em código usando DOxigen.

A ferramenta DOxigen permite gerar uma página HTML, entre outros formatos, com a documentação do código, apresentando-a de forma organizada e de fácil interpretação. A Figura 7.28 apresenta a documentação para o usuário final da função apresentada na Figura 7.27

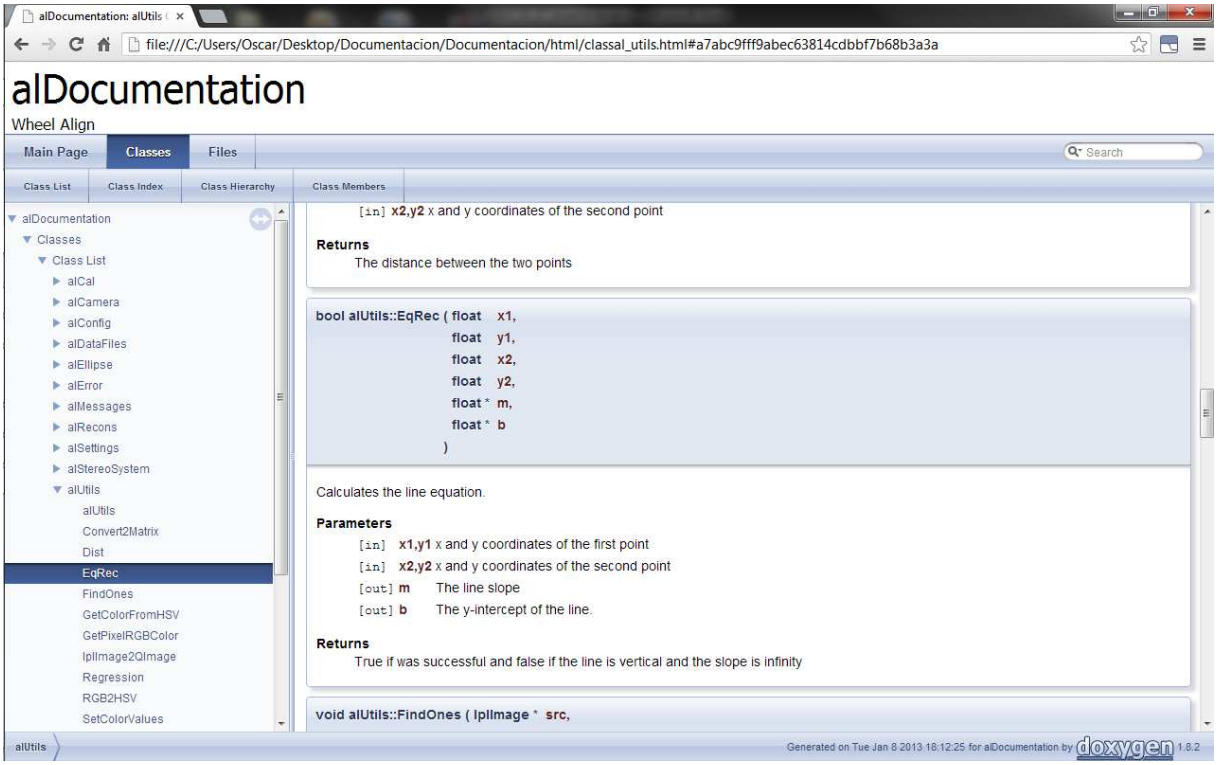

Figura 7.28: Página HTML com a documentação da função EqRec.

A Figura 7.29 apresenta um exemplo da documentação para a classe *alRecons*. Na página HTML são listados e descritos todos os métodos e propriedades da classe assim como são detalhados todos os parâmetros de cada método ou função.

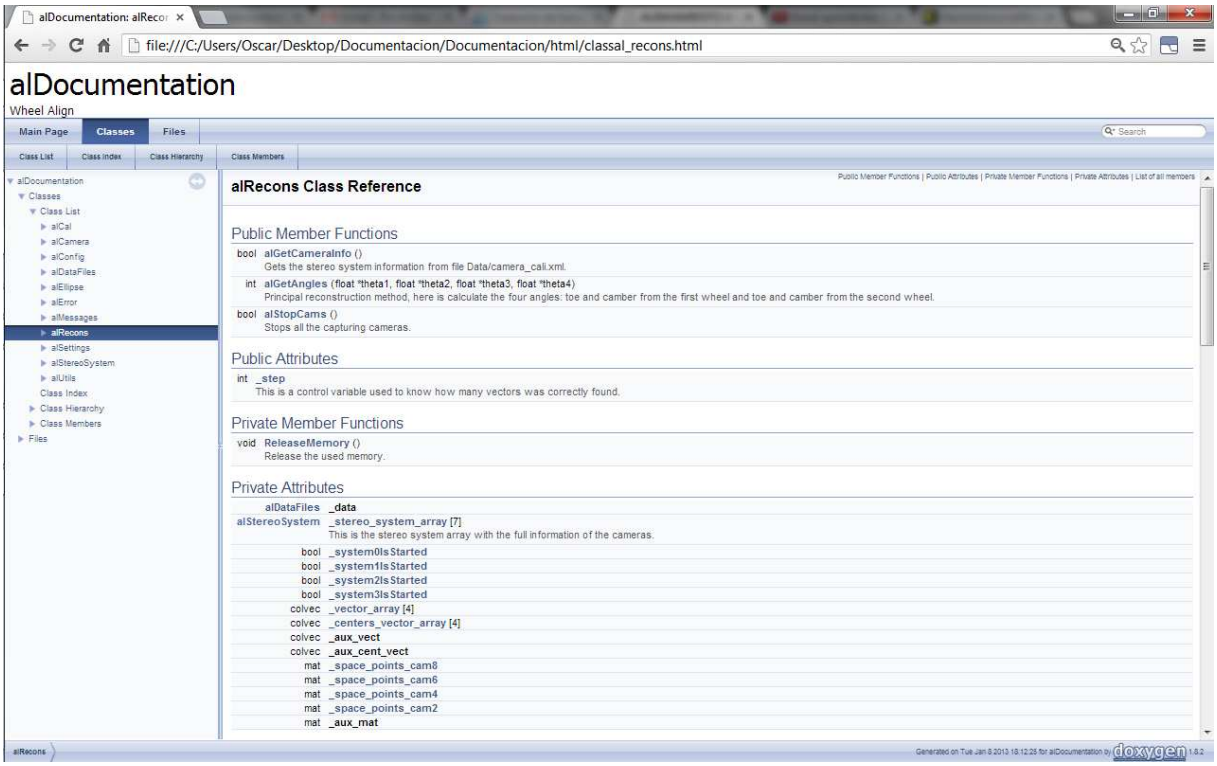

Figura 7.29: Exemplo de documentação da classe *alRecons*

Também foram realizados comentários no código que facilitam a compreensão e o seguimento dos diferentes algoritmos de processamento de imagens.

Como parte da documentação foi gerado um diagrama UML do sistema. Este diagrama permite ver as dependências entre as classes assim como conhecer o nome e o tipo das propriedades, o nome, os parâmetros e o valor retornado por cada método na classe. O diagrama foi gerado usando o *software* de modelado UML chamado Astah, (ASTAH, 2012).

Figura 7.30: Diagrama UML do sistema

8 CONSIDERAÇÕES FINAIS

8.1 Conclusões

O *software* desenvolvido para a medição de ângulos de alinhamento utilizou técnicas de visão computacional assim como de arquitetura de software para gerar um programa estável, confiável e de fácil manutenção e atualização.

Os dados obtidos nos diferentes testes e simulações demostram um bom funcionamento dos algoritmos e do programa, embora a precisão requerida por este tipo de dispositivos ainda não esteja alcançada.

Os testes de calibração mostraram que os algoritmos de detecção de círculos e dos lados da caixa ainda podem ser melhorados adicionando mais etapas de verificação para evitar falsos positivos na segmentação da caixa de calibração.

O programa apresenta uma interface gráfica muito simples e com instruções passo a passo na tela, que facilita sua operação sem demandar um demorado processo de capacitação prévia do usuário.

Foram utilizados padrões de arquitetura de *software* e foi gerada uma documentação completa do programa para facilitar futuras modificações nos diferentes algoritmos.

8.2 Proposta de trabalhos futuros

As sugestões de continuação desse trabalho são apresentadas a seguir:

- Melhora dos algoritmos de segmentação e calibração para fazê-los mais eficientes em imagens de maior resolução.

- Construir um sistema que utilize um maior número de câmeras com a finalidade de aumentar a precisão das medidas.

- Realizar o teste do programa num ambiente real e comparar as medidas feitas usando diferentes equipamentos comerciais.

- Testar o programa com câmeras de maior resolução do que 1280x720 px para tentar alcançar a precisão requerida para estes sistemas.

- Implementar técnicas de paralelização utilizando GPUs para diminuir o tempo de processamento da imagem já que este tempo aumenta conforme aumenta a resolução das imagens utilizadas.

- Explorar padrões de calibração diferentes às caixas vazadas apresentadas nesse trabalho pode-se utilizar, por exemplo, padrões desenhados nas caras de uma caixa comum ou utilizar uma caixa vazada com *leds* nos cantos em vez das esferas coloridas. Procurando uma melhoria na precisão do sistema.

REFERÊNCIAS

3DSMAX. Autodesk 3ds Max. **Modeling, animation, and rendering software**. Disponivel em: <http://usa.autodesk.com/3ds-max/>. Acesso em: 1 nov. 2012.

ABADPOUR, A. Color Image Processing Using Principal Component Analysis. **Sharif University of Technology**, 2005.

AHMED, ; HEMAYED, ; FARAG, A. Neurocalibration: a neural network that can tell camera calibration parameters. **IEEE International Conference on Computer Vision**, p. 463–468, 1999.

ALMEIDA, R. Z. H.; MARTINS, F. P. R.; FLEURY, A. T. Computer Vision Method to Estimate the Coverage of an Aluminum Alloy Plate Submitted to Peen Forming Process. **19th Congress of Mechanical Engineering**, 2007.

ARMADILLO. Site principal de Armadillo. **Armadillo C++ linear algebra library**, 2012. Disponivel em: <http://arma.sourceforge.net/>. Acesso em: Dezembro 2012.

ASTAH. Site principal de Astah., 2012. Disponivel em: <http://astah.net/#>. Acesso em: Dezembre 2012.

BATISTA, J. Explicit monoplane camera calibration. **Technical report, Institute of Systems and Robotics, University of Coimbra, Portugal.**, 1996.

BATISTA, J.; ARAÚJO, H.; DE ALMEIDA, A. T. Interatice Multi-Step Explicit Camera Calibration. **Proc. Sixth International Conference on Computer Vision ICCV98**, p. 709–714, 1998.

BIANCHI, R. A. C. Um Sistema de Medição de Volume por Visão Computacional. **V Simposio Brasileiro de Automação Inteligente**, 2007.

BOSCH. **Alinhador 3D da BOSCH**. Disponivel em: <http://www.bosch.com.br/br/equiteste/produtos/direcao/A3D.asp>. Acesso em: 26 Outubro 2012.

CANNY, J. A Computational Approach To Edge Detection. **IEEE Trans. Pattern Analysis and Machine Intelligence**, p. 679–698, 1986.

CHANG, F.; CHEN, C.-J.; LU, C.-J. A Linear-Time Component-Labeling Algorithm Using Contour Tracing Technique. **Computer Vision and Image Understanding**, v. 93, p. 206-220, 2004.

CLEMENTS, P. et al. **Documenting Software Architectures:** Views and Beyond. 2. ed. Boston: Pearson Education, Inc, 2010.

DOXYGEN. Doxygen. **Doxygen Manual**, 2012. Disponivel em: <http://www.stack.nl/~dimitri/doxygen/>. Acesso em: Dezembro 2012.

EBERLIN. EBERLIN. **Máquinas e equipamentos automotivos**, 2008. Disponivel em: <http://www.eberlin.com.br/produtos_detalhes.php?recordID=74>. Acesso em: Janeiro 2013.

FAUGERAS, O.; QUAN, L.; STRUM, P. Self-Calibration of a 1d Projective Camera and Its Application to The Self-Calibration of a 2d Projective Camera. **IEEE Transactios on Pattern Analysis and Machine Intelligence**, v. 10, p. 1179–1185, 2000.

FENIXCAR. Fenixcar. **Equipamentos Automotivos**. Disponivel em: <http://www.equipamentosautomotivos.com.br/shop/category/52/alinhadores-de-direÇÃo>. Acesso em: 26 Outubro 2012.

FORSYTH, D.; PONCE, J. **Computer Vision:** A Modern Approach. [S.l.]: Prentice Hall, 2002.

GAMMA, E. et al. **Design Patterns:** Elements of Reusable Object-Oriented Software. Portland, Oregon: Addison-Wesley, 1994.

GARLAN, D.; SHAW, M. **An Introduction to Software Architecture**. Pittsburgh: World Scientific Publishing Company, 1994.

GMBH, B. **Measuring Method and Measuring Unit for Determining the Spatial Position of a Wheel Rim As Well As a Wheel Alignment Measuring System**. 7,860,295, 2005.

GONZALEZ, R. E.; SCHWARTZ, W. R.; PEDRINI, H. Dimensionality Reduction Through PCA over SIFT and SURF Descriptors, 2012.

HAGERMAN, J. **Camber, Caster and Toe:** What Do They Mean?, 2011. Disponivel em: <http://www.ozebiz.com.au/racetech/theory/align.html>. Acesso em: 1 nov. 2012.

HUYNH, D. **Self-calibration and the fundamental matrix**. Western. 2011.

JANUARY, D. B. **Steering Geometry and Caster Measurement**. Missouri. 2007.

JURJO, D. L. B. R. et al. Vibration Analysis of Slender Columns Under Self-Weight Using Digital Image Processing Techniques. **Euromech Colloquium 483 (Geometrically Non-linear Vibrations of Structures)**, Portugal, 2007.

KINECT. **Microsoft Kinect for Windows**, 2010. Disponivel em: <http://www.microsoft.com/en-us/kinectforwindows/>. Acesso em: 1 nov. 2012.

KURKA, P. R. G. et al. Automatic estimation of camera parameters from a solid calibration box. **Journal of the Brazilian Society of Mechanical Sciences and Engineering**, 2012.

KURKA, P. R. G.; MINGOTO, C. R. **Método e Uso Para Conferência De Alinhamento De Suspensão**. PI1105356-9, 2011.

KURKA, P. R. G.; MINGOTO, C. R. Guidelines to Specify a Camera for Computer Vision Measurement of Vehicle Suspension Alignment Angles Based on the Desired Sensibility. **Symposium on Photonics and Optoelectronics (SOPO)** , Shanghai, p. 1-3 , Maio 2012.

MANJUNATH, B. S. Color and texture descriptors. **IEEE Transactions on Circuits and Systems for Video Technology**, p. 703- 715, 2001.

MCLAUGHLIN, R. A. Randomized Hough Transform: Improved Ellipse Detection With Comparison. **Pattern Recognition Letters**, v. 19, p. 3-4, 1998.

MINGOTO, C. R. **Aplicações de Visão Computacional Para Sistema de Medição de Alinhamento de Suspensão Veicular**. UNICAMP. CAMPINAS, São Paulo. 2012.

MINGOTO, C. R. **Tese de doutorado: Método de medição de alinhamento de suspensão veicular não intrusivo baseado em visão computacional.** UNICAMP. CAMPINAS, São Paulo, p. 121. 2012.

MUDROVÁ, M.; PROCHÁZKA, A. Principal Component Analysis in Image Processing. **Institute of Chemical Technology, Prague**, 2005.

OPENCV. Open Source Computer Vision. **OpenCv**, 2012. Disponivel em: <http://opencv.org/>. Acesso em: Dezembro 2012.

OPENGL. OpenGL. **Site oficial de OpenGL**, 2012. Disponivel em: <http://www.opengl.org/>. Acesso em: Dezembro 2012.

PAGE-JONES, M. **Fundamentals of Object-Oriented Design in UML**. [S.l.]: Dorset House Publishing, 1999.

QT. Digia. **Site da Qt**, 2012. Disponivel em: <http://qt.digia.com/>. Acesso em: Dezembro 2012.

REENSKAUG, T. M. H. MVC XEROX PARC 1978-79. **Site de Trygve Reenskaug criador do MVC**, 2010. Disponivel em: <http://heim.ifi.uio.no/~trygver/themes/mvc/mvc-index.html>. Acesso em: dez. 2012.

ROJAS, O.; KURKA, P. **Algoritmo de Procura de Círculos em uma imagem RGB**. Proceedings of the SPS 2012. Campinas, BR: [s.n.]. 2012.

SHAPIRO, L.; STOCKMAN, G. **Linda SHAPIRO and Gorge STOCKMAN**. 1. ed. [S.l.]: Prentice Hall, 2001.

SNAP-ON DO BRASIL. Alinhadores SNAPON. **Sun Equipamentos**, 2004. Disponivel em: <http://www.snaponsun.com.br/alinhadores.htm>. Acesso em: out. 2012.

SOUZA, A. A. D. S.; GONÇALVES, L. G. Mapeamento de Ambientes em Grade de Ocupação Probabilista a Partir de Visão Estéreo. **Simpósio Brasileiro de Automação Inteligente**, São João del-Rei - MG - Brasi, v. X, p. 259-265, Setembro 2011.

SURAL, S. Segmentation and histogram generation using the HSV color space for image retrieval. **International Conference on Image Processing.**, p. II-589- II-592, 2002.

TRUCCO, E.; VERRI, A. **INTRODUCTORY TECHNIQUES FOR 3-D COMPUTER VISION**. [S.l.]: Prentice Hall, 1998.

TSAI, R. Y. A Versatile Camera Calibration Techniaue for High-Accuracy 3D Machine Vision Metrology Using Off-the-shelf TV Cameras and Lenses. **IEEE JOURNAL OF ROBOTICS AND AUTOMATION**, p. 323-344, 1987.

VISUALINER. **Snap-on do Brasil**. Disponivel em: <http://www.snaponsun.com.br/alinhadores.htm>. Acesso em: 26 Outubro 2012.

XIE, Y.; JI, Q. A New Efficient Ellipse Detection Method. **Proc. 16th International Conference on Pattern Recognition.**, v. 2, p. 957 - 960, 2002.

XPERIA. Sony. **Site da Sony Mobile Brasil**, 2012. Disponivel em: <http://www.sonymobile.com/br/products/phones/xperia-s/specifications/>. Acesso em: 10 nov. 2012.

ZHANG, Z. A Flexible New Technique for Camera Calibration. **IEEE Transactios on Pattern Analysis and Machine Intelligence**, p. 1330–1334, 2000.