Escola Politécnica da Universidade de São Paulo

Departamento de Engenharia Mecânica

PMC 581 – Projeto Mecânico II

Projeto e implementação de um sistema de visão estéreo computacional - Tracking

Relatório Final

Prof. Coordenador: Eduardo L. L. Cabral

Alunos: Marcelo Previato Simões

Rogério Shimizu L. Santos

São Paulo 2001

SUMÁRIO

1 INTRODUÇÃO

O presente trabalho consiste em desenvolver um sistema de visão estéreo computacional que simule a visão humana sendo aplicado em um sistema simples de rastreamento de objetos (*tracking*). O projeto depois de implementado poderá ser utilizado em metrologia, sistemas de perseguição de objetos mais sofisticados e até mesmo em *real-time*, monitoramento de locais de trabalho, no estudo de algoritmos de inteligência artificial, etc.

A visão estéreo computacional apresenta diversos modelos de configuração podendo ser utilizadas duas ou três câmeras que adquirem imagens que, com tratamento por software, possibilitam a determinação do ponto focalizado. No presente projeto, a configuração será adaptada de acordo com os parâmetros da visão humana de um adulto, sendo utilizadas duas câmeras do tipo *webcam*, correspondendo aos olhos humanos. O espaçamento entre elas será dado pelo espaçamento médio dos olhos (cerca de 10cm). O sistema de câmeras deverá se movimentar de forma a atender os três graus de liberdade da visão humana: dois graus referentes à rotação de cada globo ocular em torno do eixo vertical, percorrendo o espaço na horizontal; o outro grau é acoplado às duas câmeras, correspondendo ao movimento de levantar e abaixar os olhos, percorrendo o espaço na vertical.

Os três acionamentos serão utilizados conjuntamente com o intuito de que as câmeras apontem para um mesmo ponto. A idéia inicial era que esse ponto fosse definido pelo usuário, porém para tornar o aparato mais útil, o posicionamento será fornecido pelo software para que as câmeras acompanhem o centro de um objeto, configurando assim um sistema de rastreamento. Para realizar essas tarefas serão desenvolvidos alguns softwares amigáveis para o usuário poder realizar o processo. Será também utilizado o software comercial de captura de imagens da webcam. Através de um software também será realizado o acionamento dos motores através de uma interface entre o microcomputador (interface paralela) e os motores, sendo necessário o uso de um *hardware* eletrônico para isso.

Apresentou-se até o momento os requisitos do projeto do ponto de vista da movimentação e de parâmetros obrigatórios na configuração do sistema. A partir dessas premissas, buscou-se levantar algumas possibilidades para cada subsistema. Essas possibilidades foram avaliadas, determinando o conjunto que melhor atende aos requisitos tanto funcionais quanto econômicos.

A partir da solução adotada foi construído o mecanismo cinemático do sistema de visão estéreo. Algumas configurações e materiais adotados inicialmente foram alterados para serem utilizados os componentes que se encontravam à disposição.

Concomitantemente foi desenvolvido o controlador que acionaria os motores de passo através de sinais de controle enviados pela interface paralela de um microcomputador. Esse controlador foi desenvolvido usando a associação dos circuitos integrados L297-L298.

No que se refere aos softwares, como o sistema não é em tempo real, cada módulo pôde ser desenvolvido separadamente: processamento das imagens obtidas, cálculo da nova posição a partir dos dados do sistema e movimentação dos motores. O software de aquisição de imagens utilizado é comercial e acompanha a webcam.

Com isso busca-se que a partir de duas imagens provenientes de cada umas das câmeras possa ser realizada a perseguição de um simples objeto, no caso uma pequena lâmpada.

O presente relatório apresenta duas partes distintas, uma que se concentra no estudo teórico com levantamento de soluções e estudo de viabilidade dessas soluções, o que corresponde às atividades realizadas no primeiro semestre de projeto. A outra parte se concentra na implementação real dos subsistemas (atividades do segundo semestre) projetados na primeira parte. Poderá ser observado que diversas idéias foram alteradas no decorrer do projeto para poder atender aos objetivos do projeto.

2 MOTIVAÇÃO – SISTEMA DE VISÃO ESTÉREO

O sistema de visão humana permite obter informações como distâncias de objetos, profundidades, cores, textura, etc. O início da visão computacional se baseou em tentar adquirir com equipamentos, a mesma sensação da visão humana, não obtendo apenas imagens planas, mas com elas também conseguindo a percepção de distância e profundidade.

O sistema humano, como o de outros animais, é composto de dois olhos que apresentam uma certa distância entre eles, a linha que une os centros de cada olho é chamada de *linha de base (base line)*. Os raios de luz que saem do objeto, após passarem pelo sistema de lentes presentes no globo ocular, acabam por formar na retina uma imagem. As imagens formadas, uma em cada retina, são diferentes pois o objeto está sendo observado por ângulos diferentes.

As duas imagens são processadas no cérebro humano que nos fornecem uma imagem final com características tridimensionais (3D) e com profundidade (ver esquema da figura 1). Isso não seria obtido com apenas uma imagem, porém a mente humana, com mecanismos de adaptação, consegue mesmo com apenas uma imagem obter certas características 3D, porém há certa perda na percepção de profundidade. Fato percebido em pessoas com apenas um olho.

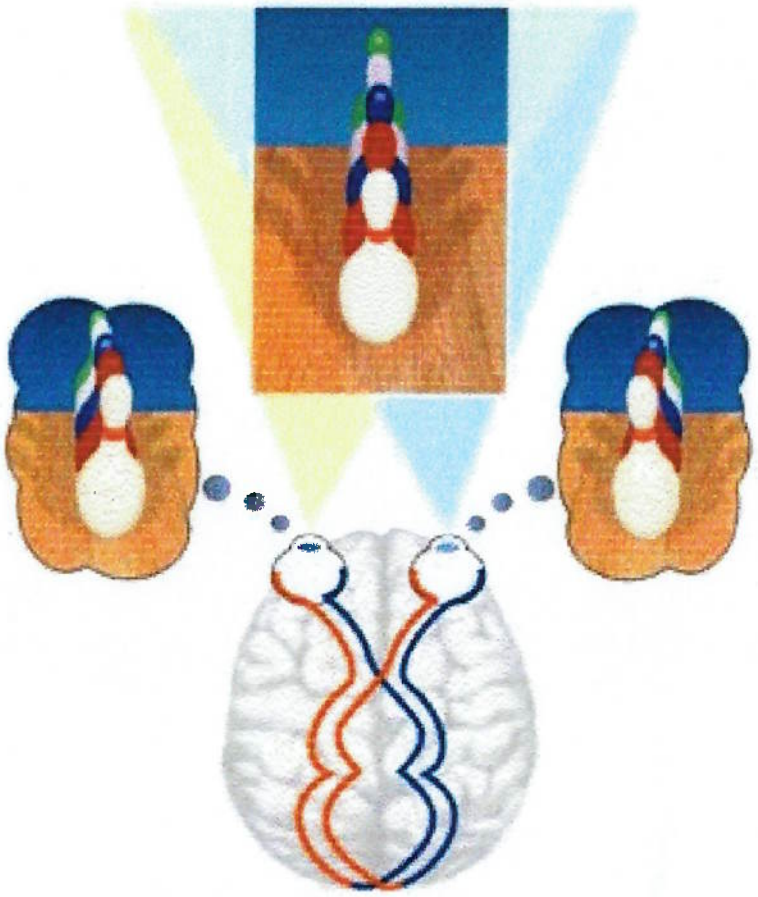

Figura 1: Princípio da Visão Estéreo Humana

Os sistemas artificiais de visão funcionam pelo mesmo princípio, sendo as câmeras o equivalente aos olhos humanos. Os raios de luz passam pela lente da câmera e as imagens são formadas ou num filme fotográfico ou num sistema de pontos que pode ser interfaceado com um microcomputador que armazenará essas imagens. A qualidade da imagem está diretamente relacionada com a precisão desse sistema de pontos. Porém ao contrário da mente humana, o sistema artificial não possui mecanismos adaptativos para obter com apenas uma imagem a idéia de profundidade e de todas as características tridimensionais. Um exemplo dessa visão projecional pode ser visto nas figuras abaixo:

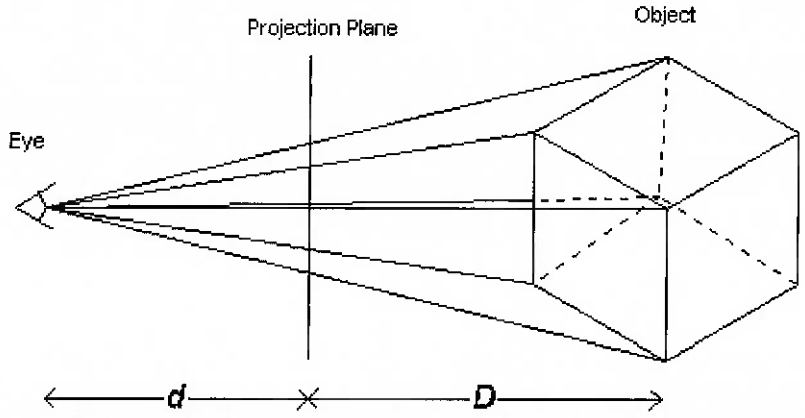

Figura 2 : visão com apenas um plano de projeção

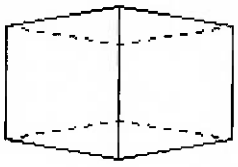

Figura 3: Imagem obtida com um único plano

Percebe-se que, devido à posição do plano projecional, a imagem obtida não tem noção de profundidade adequada. Para resolver isso, passaram a ser utilizadas duas ou mais câmeras. No momento em que são encontrados os pontos correspondentes às imagens, com algoritmos específicos, consegue-se obter uma imagem bem próxima da que seria vista pelo olho humano. Esse sistema é chamado de visão estéreo e o mesmo exemplo anterior seria visto da seguinte forma:

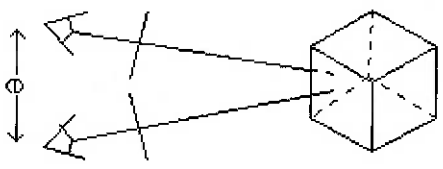

Figura 4 : Sistema com 2 planos de visão (estéreo)

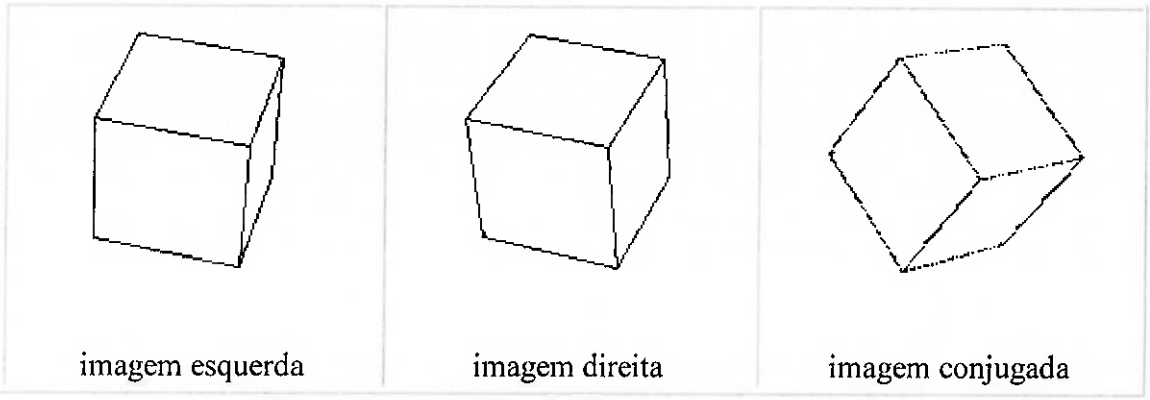

Figura 5: Imagens obtidas e sua conjugação

Percebe-se pela imagem conjugada neste exemplo e no exemplo da figura 1 que a imagem resultante expressa melhor as características de profundidade do objeto. Podendo ser obtido artificialmente efeito semelhante ao obtido pelo olho humano.

Diversas pesquisas são realizadas nessa área, buscando algoritmos e configurações de sistema que consigam cada vez mais aproximar a visão artificial da real. São pesquisados meios de representar mais fielmente os graus de liberdade do olho humano e também o processo de conjugação de imagens. Na movimentação existe, por exemplo, o movimento de vergência do olho que no presente projeto não será considerado e é um movimento bem estudado atualmente. No processo de conjugação de imagens procuram-se obter imagens cada vez mais precisas e usar algoritmos mais eficientes que consigam um efeito computacional cada vez mais próximo do que ocorre no cérebro humano.

Por isso, o mecanismo a ser projetado e implementado pode ser utilizado em diversas pesquisas posteriores, tanto nas melhoras do sistema abordadas anteriormente, quanto para adaptação do sistema para aplicações práticas de como sistemas de metrologia, de rastreamento de pontos, de inteligência artificial.

A aplicação inicial em que será utilizado o sistema já nos fornece uma boa idéia da utilidade dos sistemas de visão computacional na engenharia e aplicações científicas.

3 ESTUDO DE VIABILIDADE - SUBSISTEMAS

O sistema estudado pode ser dividido em diversos subsistemas, de modo que se possa estudar e desenvolver o projeto mais facilmente. São eles:

- Sistema de captura de imagens
- Mecanismo cinemático
- Sistema de acionamento do mecanismo
- Sistema Computacional – softwares de captura de imagens, processamento de imagens, acionamento dos motores e posicionamento das câmeras.

No estudo de viabilidade apresentado, segue uma descrição de algumas alternativas, apresentando suas vantagens e desvantagens.

3.1 Sistema de captura de imagens

Para a captura de imagens há uma grande variedade de câmeras. As mais modernas apresentam tamanho bem reduzido, sendo mais adequadas à nossa aplicação. Esse tamanho reduzido é necessário para atender à especificação das distâncias entre olhos (linha de base aproximada de 10cm) e permitir a movimentação delas sem choques. Tamanho reduzido aliado ao peso reduzido fazem com que o torque necessário para acionamento seja bem menor, podendo ser utilizados motores menores e usar menor potência do acionamento destes, evitando também sistemas de redução o que poderia tornar o mecanismo com dimensões maiores e com custo maior de fabricação.

Foram pesquisados alguns tipos de câmeras comerciais disponíveis. As mais comuns são as câmeras de vigilância e as câmeras para Internet, mais conhecidas como *webcam*. As do primeiro tipo, apresentam tamanho (aproximadamente 40x40x10mm) e peso reduzidos (120 a 200 gramas). O preço é bem parecido com o das *webcam*, custando R$ 120 o modelo preto e branco e cerca de R$160 o modelo colorido. Uma desvantagem dessa câmera

é que sua saída é analógica, sendo necessária uma placa de aquisição de dados para coletar as imagens e provavelmente o desenvolvimento de um *software* para processá-las.

Por isso, adotou-se no projeto o modelo de *webcam* que também apresenta tamanho (diâmetro aproximado de 50mm) e pesos reduzidos (cerca de 150 gramas) o que permite um torque menor de acionamento. O preço aproximado de R$ 150 é similar aos das câmeras anteriores. A maior vantagem realmente se encontra no fato desse tipo de câmera apresentar saída digital, sendo conectada diretamente ao microcomputador sem necessidade de uma placa de aquisição.

Até o momento, encontra-se disponível uma Webcam da marca Logitech QuickCam Express (ver figura 6) que é conectada ao microcomputador via conexão USB (*Universal Serial Bus)* e apresenta um software próprio para captura de imagens que foi utilizado para a aquisição de imagens do projeto.

Figura 6: WebCam utilizada

A conexão USB é uma conexão moderna que vem sendo muito utilizada porque é possível conectar diversos periféricos em cascata (máximo valor teórico de 127). Os ramos onde se apresenta mais eficiente são de: imagens digitais, multimídia e telecomunicações via microcomputador.

Diversos componentes como: scanners, impressoras, zipdrives, teclados, telefones e câmeras digitais apresentam esta conexão. O conector (figura 7) apresenta 4 pinos sendo 2 para dados e 2 para alimentação que

provém do próprio computador não sendo necessária uma fonte externa. A conexão é feita através de uma porta específica (figura 8) do microcomputador.

Figura 7: Conector USB

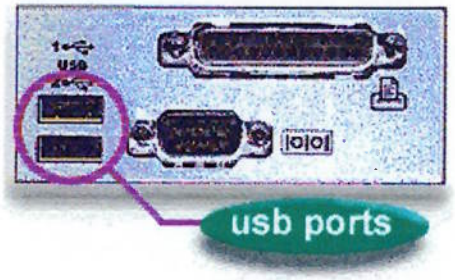

Figura 8: Porta USB no computador

Principalmente no aspecto da aquisição de imagens, outra vantagem da WebCam com USB é que este tipo de conexão é cerca de 10 vezes mais veloz que a serial tradicional (padrão RS-232), alcançando as velocidades de 12Mb/s. Com isso pode-se obter um número de imagens em intervalos de tempo menor o que é interessante em aplicações onde as velocidades são maiores.

3.2 Projeto do sistema cinemático de movimentação das câmeras

Iniciou-se o projeto estudando o modo de movimentação das câmeras no sistema. Para que simulem a visão humana, elas devem ser capazes de realizar movimento de rotação na horizontal e na vertical, mas sempre focalizando o mesmo ponto no espaço.

Para realizar tal movimento, buscaram-se algumas alternativas que atendessem a tais requisitos. Como alternativa mais simples e viável, optou-se

por utilizar um eixo principal, que movimentaria as duas câmeras simultaneamente, simulando assim o movimento de rotação do globo ocular ao redor do eixo horizontal.

Sobre esse eixo, devem ser fixadas duas outras câmeras, que simulam cada olho separadamente. Os dois olhos de uma pessoa possuem movimentos de rotação no eixo vertical independentes, mas estão sempre focalizando o mesmo ponto do espaço. O sistema então deve retratar exatamente isso, mas para garantir o foco no mesmo ponto, utilizaremos controle através de software.

Assim, o sistema final possui três graus de liberdade no espaço. O movimento de rotação no eixo horizontal e o movimento de rotação de cada câmera no eixo vertical. Posteriormente, de acordo com as cargas e o mecanismo de acionamento adotado, haverá um dimensionamento do sistema. A partir do tamanho dos motores e eixos pode-se ter uma idéia mais precisa de onde serão posicionados tais componentes.

A idéia inicial do mecanismo a ser construído pode ser visto na figura abaixo:

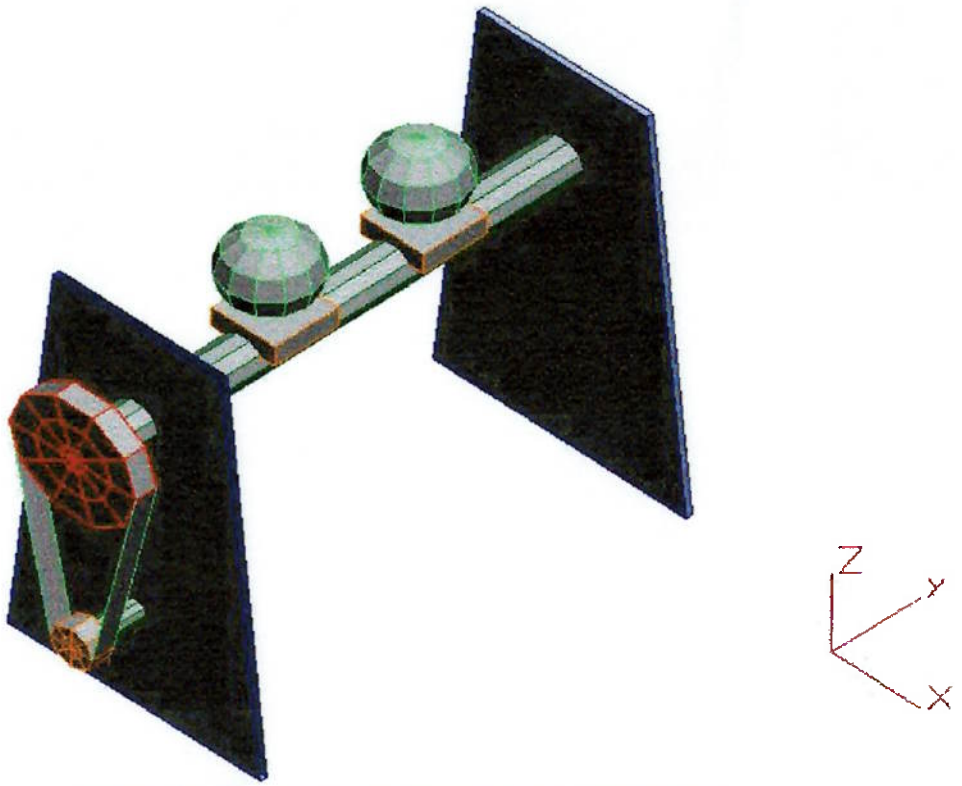

Figura 9: Esquema do Mecanismo Cinemático

Como poderá ser visto posteriormente o equipamento final apresentou uma configuração diferente por razões construtivas e de disponibilidade de materiais.

Para a construção desse mecanismo pretendia-se utilizar pedaços de chapa de alumínio para construção do suporte do eixo que também foi feito de alumínio. Foram levantados os preços desses materiais e também dos rolamentos (ver item de Análise Econômica), o que pela quantidade mínima de venda de alumínio ($1m^2$) fez com que o material da base e dos suportes dos rolamentos fosse alterado para aço inoxidável. Obviamente o peso do equipamento final foi superior (mas ainda aceitável) ao que se tivesse sido construído de alumínio, porém com a adoção do aço inoxidável foi possível o uso de refugos que baratearam extremamente o custo de fabricação do equipamento.

3.3 Tipos de acionamento

Após estudar o mecanismo cinemático, iniciou-se o estudo dos tipos de acionamento que serão utilizados em cada caso: acionamento do eixo principal e o acionamento das câmeras. Deve-se escolher qual o melhor tipo de motor a ser utilizado no projeto em questão. Para isso, observamos as diversas possibilidades: motor de passo, motor DC, servo-motores.

Os *motores DC* não são muito bons para o caso. É difícil controlar a velocidade de rotação e os ângulos que foram rotacionados. Seria necessário um encoder rotativo para determinar com a precisão necessária do projeto, o que aumenta seus custos. Além disso, é mais difícil controlar a posição final do motor, já que quando é enviado o sinal para parar o motor, este ainda rotaciona mais um pouco, devido a sua corrente na armadura. Isso ocasiona uma baixa estabilidade do mecanismo. Além disso, deve-se notar que o eixo principal irá suportar cargas durante todo o tempo, e isso implica que o motor deverá realizar um torque contrário a estas cargas quando desejamos manter o eixo parado em uma determinada posição (*holding torque)*. Com o motor DC, essa é uma tarefa muito mais complexa de ser realizada do que com o uso do motor de passo.

O mesmo se aplica aos *servo-motores*, que precisam ser controlados em malha fechada. Para isso, seria preciso um encoder e de um controlador para comparar a posição atual com a posição designada, e assim reposicionar o motor. Tudo isto implica em um custo muito alto para um posicionamento tão simples.

Os *motores de passo* são as melhores alternativas. Pode-se numerar diversas vantagens que os motores de passo possuem sobre seus concorrentes:

- Controle em malha aberta: uma das maiores vantagens dos motores de passo é a possibilidade de determinar a posição angular deste apenas contando o número de passos que foram executados;
- Acurácia: os motores de passo possuem uma acurácia em sua posição muito alta, normalmente da ordem de 3-5% de um passo. Outra vantagem é que esse pequeno erro não é cumulativo de um passo para o outro. Essa acurácia depende principalmente da precisão mecânica das partes que compõem o motor de passo.
- Alto Torque: os motores de passo possuem alto torque a baixas rotações. No projeto em questão, onde as velocidades dos motores são mínimas, o torque produzido será bem alto.
- *Holding* Torque: ou o torque produzido pelo motor quando este está parado. Isso é importante no presente projeto, já que existe carga estática quando se deixa o sistema parado.
- Preço: os motores de passo possuem um preço bem acessível no mercado. Como será apresentado posteriormente no estudo da viabilidade econômica, estes podem ser encontrados sucateados por preços baixíssimos.

Definido o tipo de acionamento como sendo motores de passo, iniciou-se o estudo da conexão entre motores e os eixos.

3.4 Conexões entre eixos e acionamentos

Existem no caso 3 conexões entre eixos e acionamentos: a conexão entre o motor e o eixo principal e as duas conexões entre os motores e as câmeras. Estudaram-se cada caso separadamente, pensando nas diversas possibilidades de conexões entre eixos e motores.

No caso de conexão entre o motor e o eixo principal, decidiu-se que o acionamento poderia ser feito através de acionamento direto, polia, correia dentada ou engrenagens.

O *acionamento direto* não é uma boa alternativa. O torque para acionar o eixo principal pode não ser suficiente. É recomendável utilizar qualquer tipo de redução para aumentar o torque fornecido pelo motor. Com essa alternativa consegue-se diminuir as dimensões dos motores, o que torna o sistema mais compacto e econômico.

Utilizando polias, consegue-se a redução desejada. O problema principal das polias é a possibilidade de escorregamento. Como no projeto é muito importante saber o ângulo de rotação do eixo, e esse valor é determinado através da rotação dos motores, o escorregamento na conexão entre o motor e o eixo acarretaria dados incorretos no programa. A precisão no posicionamento nas câmeras é fato primordial.

A correia dentada soluciona o problema do escorregamento e da redução. O preço dela também é bem acessível e seu funcionamento é confiável.

A conexão utilizando engrenagens é muito confiável. O grande problema das engrenagens é o preço. Esse item inviabiliza essa alternativa, já que se tem outras possíveis soluções que são igualmente satisfatórias a um custo muito mais acessível.

Com os dados disponíveis, consideramos que o acionamento por correia dentada o mais indicado para o caso. Ele garante o torque necessário, o não escorregamento e possui um preço relativamente barato.

Após essa etapa, iniciou-se o estudo da conexão entre os motores de passo e as câmeras. Novamente, teve-se que estudar diversas possibilidades e escolher uma delas para ser utilizada no projeto.

Acionamento direto não apresenta muitos problemas neste caso. Como a única função do motor é mover uma câmera (no caso uma webcam) de no máximo 200 gramas e diâmetro de 50mm, o torque que ele precisa é mínimo. Além disso, a conexão direta facilita no cálculo do valor do ângulo de rotação da câmera, já que a rotação do motor é igual ao valor de rotação da câmera.

A conexão por polias possui o mesmo problema que o no caso da conexão ao eixo principal. O escorregamento invalidaria qualquer dado de rotação que o software indicasse, já que este possui capacidade apenas de controlar o motor, e não de observar o eixo.

A conexão por correia dentada é uma alternativa que não apresenta qualquer problema, mas sua utilização não é necessária, já que não observamos a necessidade de uma redução nesta conexão. Apenas acarretaria um maior custo sem grandes vantagens.

Desse modo, se o motor de passo for capaz de executar uma rotação suficiente pequena de acordo com as especificações do projeto, não é necessária qualquer redução e assim, o acionamento direto é a melhor alternativa a ser adotada.

3.5 Sistema Computacional – Software

Com relação ao sistema computacional, este será o responsável pela aquisição das imagens das câmeras, pelo processamento das imagens coletadas, pelo acionamento dos motores de passo e pelo correto posicionamento das câmeras na nova posição calculada.

Na parte de aquisição de imagens, é necessário o uso de um microcomputador com velocidade e memórias suficientes para adquirir e processar as imagens, além de estar disponível pelo menos uma porta de conexão USB (para aquisição de imagens) e uma porta paralela (para acionamento dos drivers dos motores). Esses requisitos são facilmente atendidos por qualquer microcomputador recente. O sistema operacional utilizado deve ser o Microsoft Windows 98 ou superior, pois são os sistemas que suportam bem a conexão via USB.

O software utilizado para a aquisição de imagens é o software comercial que acompanha a WebCam. Em certo momento do projeto foi pesquisado o funcionamento e os sinais da porta USB para saber se seria possível implementar um software próprio para a captura de ambas as imagens, já que o software comercial captura apenas uma das imagens. Porém como foi decidido realizar um sistema de *tracking* sem ser em tempo real, não haveria a necessidade da confecção de tal software, pois bastaria capturar ora a imagem de uma câmera ora a imagem de outra sem nenhum problema, principalmente pela porta USB apresentar tecnologia *plug-and-play,* ou seja, poderia ser efetuada a troca de câmeras que estivessem conectadas ao computador sem ter que reinicializar o equipamento. As duas imagens obtidas então seriam armazenadas para a etapa de processamento.

Os softwares a serem desenvolvidos devem apresentar uma interface amigável com o usuário para que este opere corretamente o equipamento. Como não será em tempo real os módulos que compõem o sistema computacional poderão ser desenvolvidos e testados separadamente e se houver tempo disponível realizar a integração destes em um único programa. Esse programa poderia então ser alterado de acordo com a sua aplicação em novas pesquisas sejam elas em sistemas de rastreamento mais sofisticados ou mesmo em outras aplicações como metrologia e inteligência artificial.

O software de processamento de imagens seria responsável em determinar a posição do ponto a ser acompanhado nas duas imagens obtidas, para facilitar essa tarefa será utilizado como objeto uma lâmpada, pois assim é possível ter uma boa discrepância entre os pontos de maior e menor luminosidade.

Através das características físicas dos equipamentos é possível realizar a correspondência entre os pontos reais e os pontos da imagem, com o intuito de se determinar a nova posição de apontamento das câmeras. A transformação de coordenadas para cálculo do ângulo de apontamento e conseqüentemente o cálculo do número e direção dos passos de cada motor serão também feitos por software. Para o acionamento dos motores serão enviados três sinais pela porta paralela para o controlador: sinal de pulsos de acordo com o número de passos, sinal de *half-step ou full-step* e sinal de sentido de rotação.

Percebe-se que se trata de um software que pode ser desenvolvido em linguagens como C ou Java, que apresentam funções ou classes para desenvolvimento de interface gráfica e de interfaceamento via porta paralela.

3.6 Viabilidade Econômica

Antes de iniciar-se a execução do projeto, deve-se verificar se este é viável economicamente. Para isso, foi feita uma pesquisa sobre o preço dos diversos componentes que serão utilizados no sistema cinemático, de controle e de aquisição de imagens.

Os preços dos componentes estão ilustrados abaixo:

Componente	Preço
Mecanismo de Atuação	
Motores de passo (1 motor de 7.5 graus/passo e 2 de 0.9 graus/passo)	R$ 45,00
Mecanismo Cinemático	
Chapa de alumínio 1m^2	R$ 72,00
2 Rolamentos NSK	R$ 18,00
Eixo de alumínio (não utilizado) Eixo de aço inoxidável (refugo)	R$ 22,00 R$ 8,00
Correia Dentada e polias	R$ 25,00
Outros (parafusos, porcas, buchas,etc)	R$ 7,00
Hardware de Controle dos Motores de Passo	
Componentes para 2 controladores	R$ 80,00
Sistema de Captura de Imagens	
2 WebCams	R$ 300,00
Total	**R$ 555,00**

Tabela 1: Análise Econômica Estimada

Pode-se verificar que a maior parte dos custos concentra-se no sistema de captura de imagens, ou seja, as duas WebCams. Por esse motivo,

utilizaremos apenas uma câmera para teste, a qual não será acoplada de modo definitivo no mecanismo, sendo acoplada cada momento em um dos motores do equipamento.. Com exceção deste item, o projeto é claramente viável, com um custo razoável.

4 EXECUÇÃO DO PROJETO

Terminado o estudo de viabilidade do projeto, com a solução adotada sendo considerada viável, iniciou-se a etapa de execução do projeto. Cada passo dessa etapa de execução do projeto é apresentado abaixo detalhadamente.

4.1 Motores de passo

Os motores de passo foram adquiridos por preços relativamente baixos em lojas do centro da cidade que vendem motores sucateados.

Conforme foi dito nos itens anteriores, os motores de passo para acionamento das câmeras não necessitavam apresentar grande torque e de preferência deveriam ter tamanhos reduzidos. A principal exigência era apresentar um passo bem reduzido para não ser necessário o uso de um sistema de redução.

Os motores adquiridos foram 2 da Sanyo Denki Co. LTD do modelo 103-4902-12T1 (*8 wire universal stepper motor)* que são utilizados para acionamento de *floppy disks.* Esse motor apresenta as seguintes características:

- **0.9 grau por passo (half-step 0.45)**
- 4.3 Ohms/espira
- diâmetro do eixo 0.188" - comprimento 0.75"
- Unipolar or bipolar

Foram procuradas mais informações em diversos *sites* além do da própria Sanyo Denki, porém não foram obtidas outras informações importantes como torque desenvolvido e torque de alimentação.

Para o motor que acionaria o eixo principal, os requisitos necessários seriam um maior torque de acionamento, não sendo necessário um passo tão reduzido já que seria utilizado um sistema de redução.

Sendo assim, o motor adquirido foi o modelo 28BB-H101-54 da Minebea Co.LTD. que apresentava as seguintes características:

- **7.5 graus por passo**
- 3.6 Ohms;espira
- Tensão de acionamento : 5V
- Unipolar ou bipolar

Para este motor também não foram encontradas outras informações. Sendo assim, seria necessário acionar os motores para determinar se estes conseguiriam atender às necessidades de torque e de tensão. Por isso, em seguida, procurou-se construir os controladores dos motores de passo.

4.2 Hardware de Controle

Decidiu-se inicialmente que seriam usados dois controladores, um para os dois motores menores, responsáveis pelo controle da câmera, com um multiplexador para determinar qual motor será acionado por vez, e o outro controlador para o motor de maior potência, para garantir que o controle de potência não cause a queima dos componentes.

Dentre as diferentes alternativas existentes para acionamento dos motores de passo, foi adotada a que utiliza a conexão L297-L298 (circuitos integrados da SGS Thomson Microelectronics) pois é bem simples e atende as necessidades do projeto. O esquema dessa conexão pode ser visto na figura abaixo:

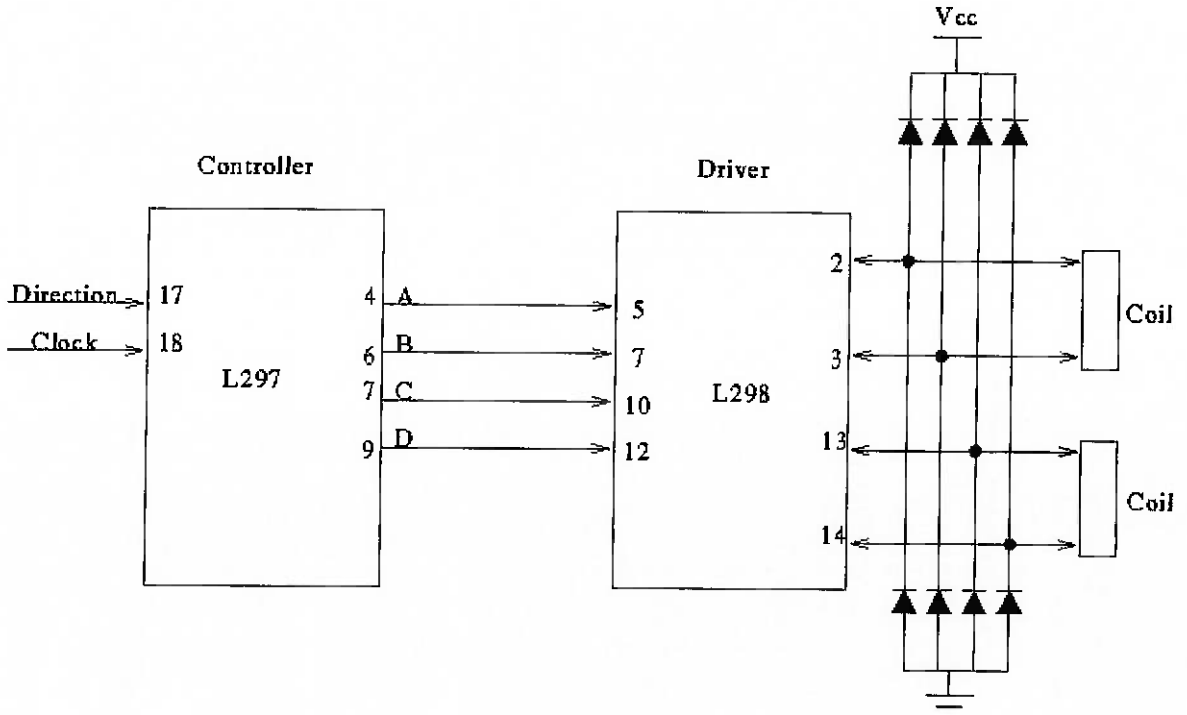

Figura 10 : Esquema do controlador com L297-L298

O L297 é o modulo de controle para acionamento dos motores de passo. Ele pode controlar motores com 4 fases (A,B,C,D) tanto para casos de acionamento unipolar quanto bipolar. É este circuito que fornece os sinais de pulso do motor, podendo ser selecionado através de sinais de controle as operações de *half* ou *full step* e também a direção do movimento. Para testar essas operações foram utilizadas duas chaves do tipo ON/OFF. Essas chaves foram retiradas posteriormente pois esses sinais agora são enviados pela porta paralela do microcomputador.

O L298 é o módulo de potência do controlador, pois trabalha com tensões e correntes maiores que o L297 para poder acionar o motor de passo. O circuito implementado com todas as conexões e com os valores de resistências e capacitores utilizados pode ser observado na figura 11.

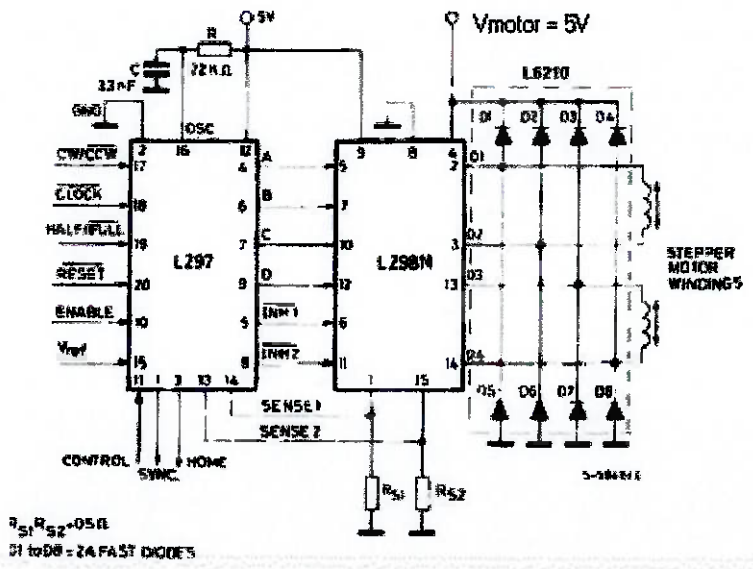

Figura 11: Esquema completo da ligação

Para testar os motores independentemente do software de controle, foi utilizado um CI 555 para se obter sinais de *clock* a partir de uma fonte de tensão contínua de 5V. A freqüência de pulsos é variada por um potenciômetro ligado ao 555.

Com o prosseguimento do projeto, o controle dos motores foi dado através de um microcomputador, ou seja, conectou-se a saída paralela do microcomputador aos controladores, retirando-se o 555 do circuito. Assim, o controle da velocidade dos motores foi feito através do software de posicionamento do microcomputador, mandando sinais pela paralela mais lentamente ou rapidamente.

Foi implementado o controlador e este testado para os motores de potência menor. Foi possível realizar a movimentação desses motores junto à câmera usando a voltagem normal de 5V (proveniente de uma fonte externa), variando-se a direção e os sistemas *half/full step* por software.

4.3 Mecanismo Cinemático

Para a implementação do mecanismo cinemático foram alterados detalhes construtivos do modelo original (apresentado no estudo de viabilidade)

devido aos materiais disponíveis. O preço elevado da placa de alumínio, mesmo para a quantidade mínima de venda, fez com que o material do eixo e da estrutura de sustentação desse fosse alterado para aço inoxidável. Esse material foi facilmente encontrado em pequenas quantidades e a um preço bem cômodo, o que reduziu consideravelmente o custo inicial estimado do projeto. Esse material também apresenta propriedades mecânicas que se adequam ao projeto e como o alumínio não sofre oxidação. O ponto negativo do uso desse material seria o peso final do sistema, porém este ficou com peso razoável capaz de ser transportado facilmente.

A base metálica de aço inoxidável tem dimensões de 300mm X 210mm com dobras de 15 mm em cada um dos quatro lados. Essas dobras foram realizadas para evitar que as arestas da placa causassem acidentes, além de servir como estética para o equipamento. Para o alinhamento do equipamento com a superfície em que será posicionado, foram colocadas quatro buchas comerciais de nylon de 20mm de altura e de 31mm de diâmetro que servem como "pés". Para a fixação das buchas na base foram realizados quatro furos de diâmetro ¼" na base, onde foram inseridos parafusos sextavados de ¼" que foram rosqueados a cada bucha.

O suporte do eixo, ao invés de ser feito com a placa de alumínio apresentada anteriormente, foi feito com o uso de duas hastes de aço inoxidável de diâmetro ½" com 160mm de comprimento. Em cada uma delas, 10mm de uma das extremidades foi torneado para a obtenção de rosca externa. Essa rosca externa junto com o uso de porcas é responsável pela fixação das hastes na base, para isso obviamente foi necessário a abertura de dois furos na base com diâmetro de ½". Percebe-se que a haste, após a sua colocação, apresentou a altura desejada de 150mm, que foi considerada como sendo adequada para o projeto, pois permite a boa movimentação das câmeras nos 360° de rotação (em relação ao eixo horizontal) sem tocar em nenhum dos demais componentes do sistema.

Para a escolha do diâmetro do eixo foi realizado inicialmente um pré-dimensionamento baseado nos esforços resultantes para a movimentação das câmeras no caso de torque crítico (maior comprimento de braço entre o centro de massa estimado das câmeras e o centro do eixo). Também foi verificada a deflexão máxima a ser obtida no eixo (Teorema de Castigliano) sendo que o

valor encontrado foi menor que 0,5mm o que é bem aceitável para este projeto. Dentre os materiais disponíveis para uso, o que melhor se adequou ao calculado foi o aço inoxidável com diâmetro de 3/8". Este eixo foi usinado em torno para a confecção de rebaixos de 1mm e 2 mm destinados ao assentamento dos rolamentos e da polia de transmissão respectivamente.

No que se refere aos rolamentos utilizados no projeto, estes deveriam ser blindados para evitar a entrada de poeira e outros detritos que poderiam prejudicar a vida útil e a precisão do rolamento, além de não ser necessária a lubrificação constante do rolamento. O tipo ideal de rolamento seria o de esferas de contato angular, pois este diminui as folgas do sistema e conseqüentemente aumenta a precisão, porém não há rolamentos desse tipo blindados, logo a escolha adotada foi o uso de rolamentos de uma carreira de esferas blindados. No catálogo do fabricante de rolamentos foi adotado o rolamento nº608 de 38mm x 16mm. Cada rolamento foi inserido dentro de um alojamento e este foi fixado à haste suporte através de uma conexão rosqueada. Outros furos foram abertos no alojamento do rolamento para possibilitar a retirada deste de dentro do alojamento em caso de danos.

Os motores de passo para acionamento das câmeras foram fixados (com parafusos) a pequenas chapas de aço inox que foram soldadas cada uma em um anel de aço inox que foi colocado no eixo principal. Esse anel apresenta um parafuso com porca de pressão que é responsável pela fixação desse anel no eixo. Caso seja necessária a mudança de posição entre as câmeras ou mesmo a retirada destas, basta soltar esses parafusos e reposicionar (ou retirar) os anéis sobre o eixo. Apesar de trabalhar-se com uma distância fixa, o projeto ganha flexibilidade com esta configuração.

Para acoplar as WebCam ao seus respectivos motores foram utilizadas buchas de nylon de 20mm de diâmetro com 19 mm de altura. Os ajustes foram forçados e com isso foi possível obter a relação de transmissão unitária conforme foi apresentado no estudo de viabilidade.

Para o acionamento do eixo principal, foi adotado o mecanismo por correia dentada. A polia menor apresenta diâmetro de 25mm e a polia maior diâmetro de 70mm assim foi possível realizar a redução necessária para aumentar o torque e diminuir o passo repassado ao eixo principal. Com esses diâmetros e com a posição das polias foi possível a escolha de uma correia

dentada comercial da Gates do tipo 231-C.B.11. após a correia dentada ser bem esticada pôde-se definir o posicionamento do motor de passo maior, que foi fixado à uma pequena chapa de aço inox através de parafusos e esta também foi fixada à base através de parafusos.

Após descrever resumidamente quais foram as etapas de usinagem dos componentes, pôde-se perceber que algumas etapas gastaram boa parte do tempo, como a usinagem das polias dentadas e do alojamento do rolamento. Percebe-se que o sistema apresenta a maior parte das conexões feitas através de parafusos, isso já prevendo a necessidade de troca dos componentes.

Abaixo é apresentado um esquema do mecanismo construído, vale perceber as diferenças e semelhanças entre o modelo final (figura 12) e o inicialmente planejado pelo estudo de viabilidade (figura 9).

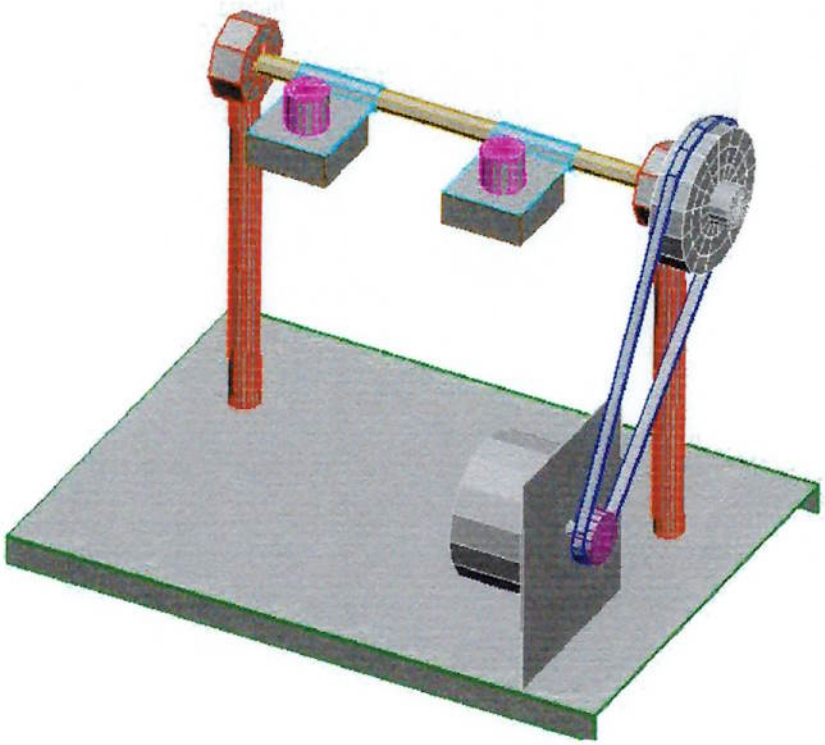

Figura 12: Esquema do mecanismo cinemático construído

4.4 Interface via Porta Paralela e Software de Acionamento

A interface paralela é uma interface muito utilizada em projetos principalmente devido a sua velocidade na transmissão de dados quando comparada com a transmissão serial. Isso porque a interface paralela transmite oito dados simultaneamente através de oito cabos (cada *bit* tem sua própria linha de transmissão), ao contrário da interface serial que se caracteriza pela transmissão de dados por uma única linha de transmissão. Por isso, um caracter de '*n' bits* transmitido via serial demora 'n' vezes mais tempo do que se fosse transmitido via paralelo.

Porém a utilização de transmissão de dados paralelamente apresenta-se limitada quanto à distância. Isso porque as transmissões paralelas por usarem sinais compatíveis com TTL (5V) apresentam uma diminuição da imunidade a ruídos, apresentando uma limitação da utilização de cabos longos, pois estes atuariam como antenas, produzindo erros nos dados recebidos. Outra desvantagem é o maior custo que se têm com cabos e conexões quando estão envolvidas maiores distâncias. Assim, o uso da transmissão paralela fica limitado a pequenas distâncias, cerca de 5m.

Vale ressaltar que mesmo sabendo que a porta paralela do computador é compatível com a tecnologia TTL, não se deve apenas ligar CIs TTL em sua saída pois no computador podem ser tanto das tecnologias TTL LS (*Low Power Schottky)* ou TTL HC (*High Speed CMOS)* que não possuem corrente suficiente para excitar CIs TTL. Para verificar se a amperagem é suficiente para excitar o CI pode-se fazer um teste com um LED e um resistor, caso esta seja muito baixa será necessária uma amplificação do sinal. Esses níveis de tensão e preocupação de amperagem se referem ao sinal de controle envido pela porta paralela, isso porque a tensão e a amperagem de alimentação do circuito deve ser feita com o uso de uma fonte externa de 5V, pois a porta paralela não tem capacidade de acionar tais ordens de grandeza de amperagem podendo sofrer danos caso seja requisitado correntes maiores por parte do circuito, e isso é bem provável já que os motores necessitam altos valores de corrente quando comparados aos valores fornecidos pela porta paralela.

Para realizar a transmissão é utilizado um cabo de conexão paralela. Esse cabo apresenta:

- 8 linhas de dados
- 4 linhas de controle
- 5 linhas de status

Os conectores podem ser do tipo Centronics, que apresenta 36 pinos de conexão, ou da IBM, atualmente o mais comum, que se utilizada de apenas 25 pinos. Abaixo segue a pinagem correspondente tanto para conector Centronics quanto conector IBM. Vale ressaltar a importância do conhecimento da direção dos pinos em *in ou out*, isso porque os 3 sinais de controla devem ser enviados por pinos de direção *out*.

Pinos (25)	Pinos (36)	Sinal Standard	Direção	Registrador
1	1	* Strobe	In / Out	Control
2	2	Dado 0	Out	Data
3	3	Dado 1	Out	Data
4	4	Dado 2	Out	Data
5	5	Dado 3	Out	Data
6	6	Dado 4	Out	Data
7	7	Dado 5	Out	Data
8	8	Dado 6	Out	Data
9	9	Dado 7	Out	Data
10	10	* Ack	In	Status
11	11	Busy	In	Status
12	12	Paper	In	Status
13	13	Select	In	Status
14	14	* Auto Linefeed	In / Out	Control
15	32	* Error	In	Status
16	31	* Inicializar	In / Out	Control
17	36	* Select- Printer	In / Out	Control
18-25	19-30	GROUND (terra)	GRN	----

(*) Sinais que são ACTIVE-LOW (ativos em nível baixo)

Tabela 2 : Pinagem e sinais para conexão paralela

Dos diferentes modos de operação das portas paralelas (SPP, ECP, EPP) será usado o método SPP, que apesar de não possuir grandes velocidades é o modo de operação mais simples, porém funciona muito bem, podendo ser bidirecional de acordo com a possibilidade desse tipo de

transmissão pela porta instalada. Abaixo é apresentado o *handshake* desse processo de transmissão.

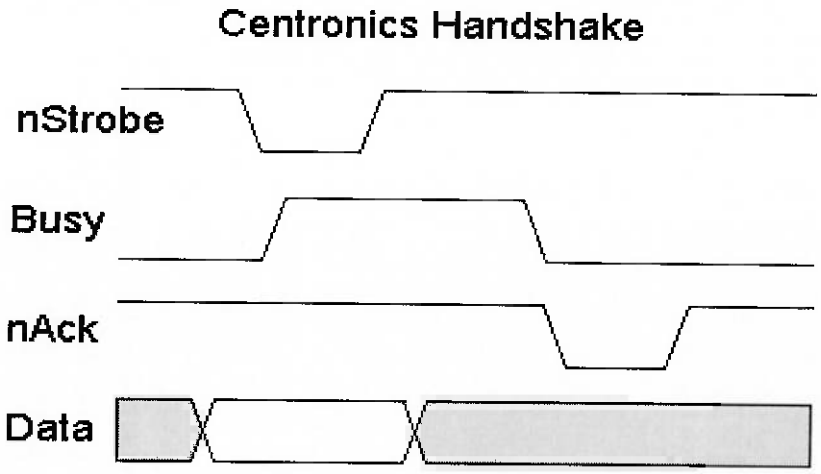

Figura 13 : Handshake Standard

1°) Os dados são enviados para a porta paralela através dos pinos 2 a 9
2°) Faz-se a verificação se o dispositivo de recebimento, por exemplo uma impressora, está livre. Se estiver o sinal "busy" (pino11) está em nível lógico '0'.
3°) O sinal de "Strobe" (pino 1) vai para nível lógico baixo (que simboliza que existem dados na via) , há uma espera em torno de 1 microssegundo, retornando o sinal a nível lógico '1'.
4°) Na borda de subida do "strobe" os dados são lidos, tornando a impressora ocupada (busy em '1')
5°) Para verificar se o dado foi aceito, utiliza-se um pulso de cerca de 5 microssegundos na linha de "ack". Essa verificação pode ser esquecida quando procura-se ganhar velocidade na transmissão.

Apesar dos sinais serem gerados por *hardware,* a leitura de dados e todo o controle (se há dados na linha, se está esta ocupada, se o dado foi recebido) é controlado por *software*. Através dos programas procura-se verificar cada passo para que não ocorra perdas na transmissão, principalmente quanto aos tempos de espera.

Para a implementação de programas se faz necessário o conhecimento da porta onde está conectado o dispositivo para se determinar quais registradores serão analisados.

♦ Endereços das Portas Paralelas

Os endereços-base normalmente utilizados para as portas paralelas são:

378H : LPT1

278H : LPT2

3BCH : (antiga, usada para video cards)

Identificando-se em qual porta está conectado o dispositivo, pode-se acessar através de *software* os registradores pelos seguintes endereços.

REGISTRADOR	ENDEREÇO
Data	Base + 0
Status	Base + 1
Control	Base + 2

Tabela 3 : endereços dos registradores

♦ Registradores

No modo standard de operação são utilizados apenas 3 registradores : o de dados, o de status e o de controle.

- Registrador de Dados

É um registrador usado no modo standard apenas para saída, se for lido o valor obtido corresponderá ao último byte enviado. Se a porta for bidirecional, pode-se ativar no registrador de controle esse tipo de operação, assim a porta também poderá servir para leitura. No caso do projeto é utilizado o modo bidirecional para efetuar o envio e o recebimento de dados.

D7-D0 : dados enviados

Sinal	D7	D6	D5	D4	D3	D2	D1	D0
Pino	9	8	7	6	5	4	3	2

Tabela 4 : Registrador de Dados

- Registrador de Status

É um registrador que verifica a condição atual da conexão. è usado apenas para leitura.

S7 – Busy : 1 – dispositivo livre 0 - dispositivo ocupado

S6 – Ack : 1 – transferência em progresso 0 – transferência completa

S5 – Paper : 1 – sem papel 0 – papel disponível

S4 – OFON: 1 – impressora ligada 0 – impressora desligada

S3 – Erro : 1 – sem erro 0 – erro ocorrido

S2, S1, S0 : Não utilizados

Sinal	NBUSY	NACK	PAPER	OFON	nERRO	X	X	X
Pino	11	10	12	13	15	----	----	----

Tabela 5 : Registrador de Status (leitura apenas)

- Registrador de Controle

Esse registrador pode ser utilizado tanto para entrada quanto para saída, servindo para a transferência de dados e demais operações envolvidas.

C7, C6, C5 : não utilizados

C4 - IRQ : interrupção 1 – habilitada 0 – desabilitada

C3 - DSL : seleção da impressora 1- selecionada 0 – não selecionada

C2 – INI : inicialização da impressora 1- operação normal 0 - inicializar

C1 – ALF: automatic linefeed 1- ALF pela impressora 0 – ALF pelo usuário

C0 – STROBE: 1 – transferência de dados 0 – sem transferência de dados

Sinal	X	X	X	IRQ	DSL	nINI	ALF	NSTR
Pino	----	----	----	----	17	16	14	1

Tabela 6: Registrador de Controle

Em anexo está o módulo de controle em linguagem C que será responsável pelo envio dos 3 sinais de controle pela porta paralela: pulsos de acionamento do L297 de acordo com o número de passos a serem dados, sinal de seleção entre *half step ou full step* e sinal de seleção de sentido de rotação. O uso desse módulo é restrito ao Windows 95/98/ME (pois há conflitos com o kernel do Windows 2000/NT) e ao modo de operação SPP, ou seja, porta paralela configurada como "normal" ou "standard". Os métodos EPP e EPC não eram objetivos do projeto e podem ser que funcionem ou não, logo o usuário deve estar atento ao se utilizar desse módulo.

4.5 Aquisição de Imagens

Conforme já foi dito, a aquisição de imagens é feita com o próprio software comercial que é fornecido junto a WebCam. Caso a simulação a ser feita fosse realizada em tempo real seria necessário que as imagens capturadas pelas câmeras fossem instantaneamente enviadas para o software de processamento.

Para realizar essa tarefa seria necessário que os softwares de aquisição e processamento fossem integrados para diminuir os tempos de envio de imagens, só que para realizar isso seria preciso desenvolver o software de aquisição. Essa tarefa seria baseada no estudo detalhado dos sinais captados pela USB e o conhecimento mais aprofundado de detalhes construtivos da câmera que acabam não sendo disponíveis ao usuário facilmente. Outra inviabilidade do software comercial é que este só capta uma imagem por vez.

Porém no nosso projeto estamos interessados em realizar o rastreamento sem ser em tempo real, com isso, as imagens podem ser adquiridas e armazenadas em arquivos do tipo bitmap ou jpeg (formatos salvos pela WebCam utilizada). Para isso, primeiro se conecta uma das câmeras ao programa, por exemplo, a da esquerda e depois de capturada a imagem, troca-se a câmera para capturar a outra imagem, nesse exemplo, a da direita. Esses arquivos são abertos em um software de processamento de imagens à parte, desenvolvido no projeto.

Para exemplificar esse processo foi realizado um teste com a obtenção de imagens de um ponto luminoso numa posição conhecida em relação à câmera. O ponto luminoso num ambiente escuro é um exemplo em que a identificação do objeto se torna bem simples no processamento de imagens, o que torna mais simples de ser rastreado.

As duas imagens coletadas através do software comercial são apresentadas abaixo, seu formato é bitmap de dimensões 160 x 120 pixels.

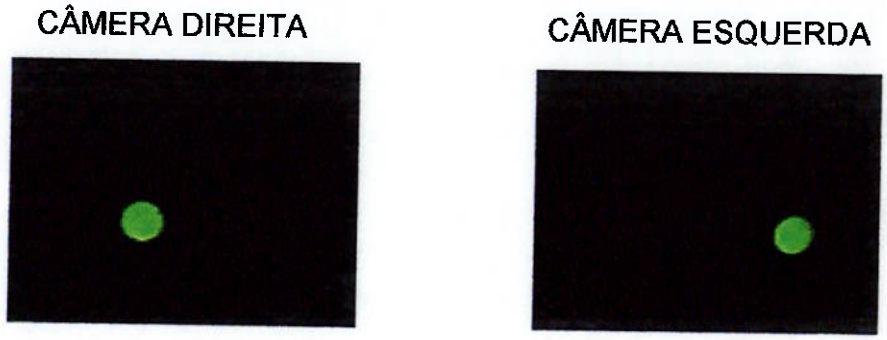

Figura 14: Imagens adquiridas

4.6 Processamento de Imagens

Na etapa de processamento de imagens, as duas figuras obtidas das câmeras são processadas com o intuito de encontrar o ponto imagem correspondente ao objeto real. No caso a ser analisado, o interesse é determinar o centro do ponto luminoso. Para isso foi desenvolvido um software usando uma linguagem de alto nível, o Director Lingo, que se baseia em eventos. Esse software tem seu código em anexo.

O software carrega as duas imagens e em seguida transforma as imagens coloridas em imagens preto e branco (binarização da imagem). Essa transformação se baseia em dois parâmetros: primeiro, os níveis RGB (Red Green Blue) de cada pixel da figura (variam de 0 a 255) e o segundo é o nível de ajuste que é escolhido pelo usuário podendo variar entre 0 e 750 (valores adotados no projeto).

O programa carrega os parâmetros RGB e os soma, esse resultado é comparado com o ajuste adotado pelo usuário, se estiver maior que o ajuste o

pixel se torna branco, se estiver abaixo o pixel se torna preto. As figuras resultantes são apresentadas na tela do programa que é bem amigável e de fácil utilização. A partir disso o programa calcula as componentes x e y do centro do conjunto de pontos brancos. O valor apresentado é em pixels sendo a referência (0,0) o centro da imagem.

Para as figuras adquiridas anteriormente esse efeito é bem fácil de ser obtido, pois já há uma grande discrepância entre os pontos iluminados e os pontos escuros. O resultado obtido é apresentado na figura abaixo:

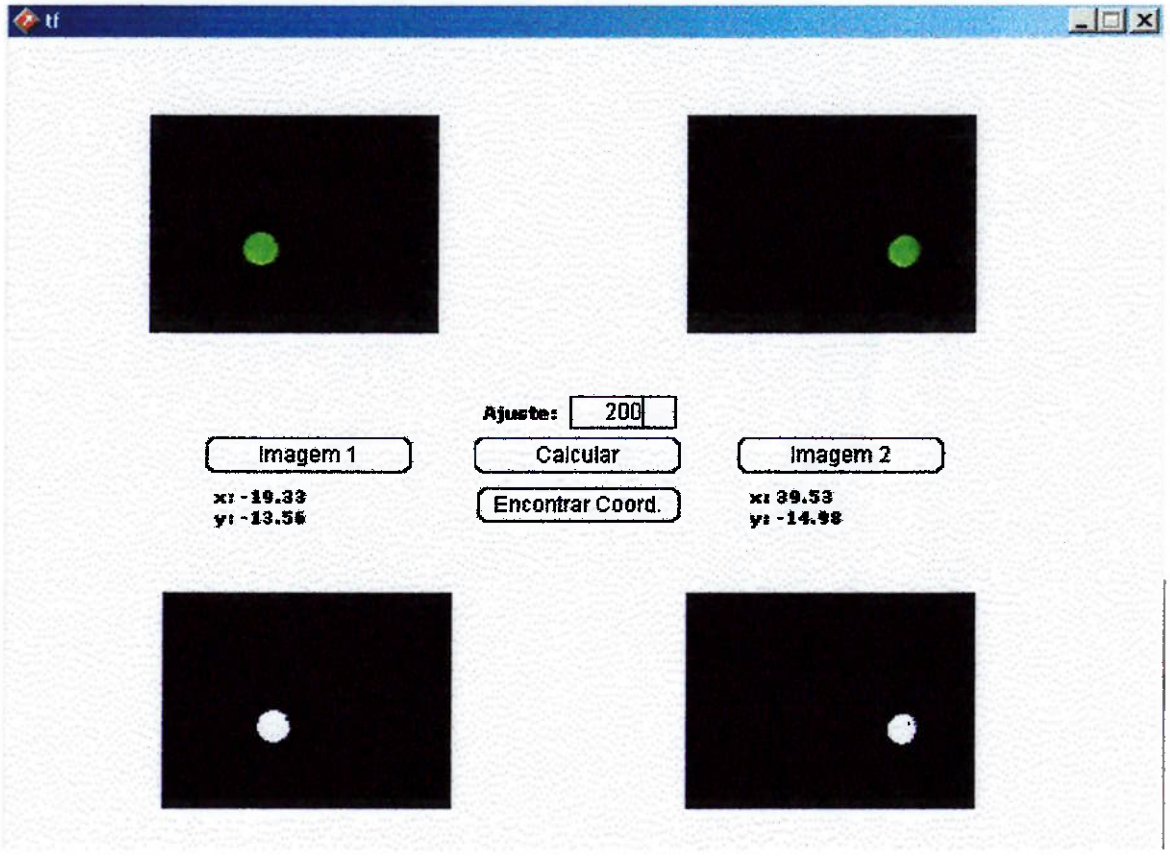

Figura 15: software de processamento desenvolvido

O ajuste de 200 permitiu uma boa imagem resultante, se este valor fosse bem baixo, apesar da discrepância entre o negro e o luminoso, o resultado seria diferente, pois mais pontos brancos apareceriam. Se o ajuste fosse maior, pontos luminosos seriam perdidos. Por isso essa flexibilização do ajuste é importante, porque depende muito do ambiente em que são capturadas as imagens.

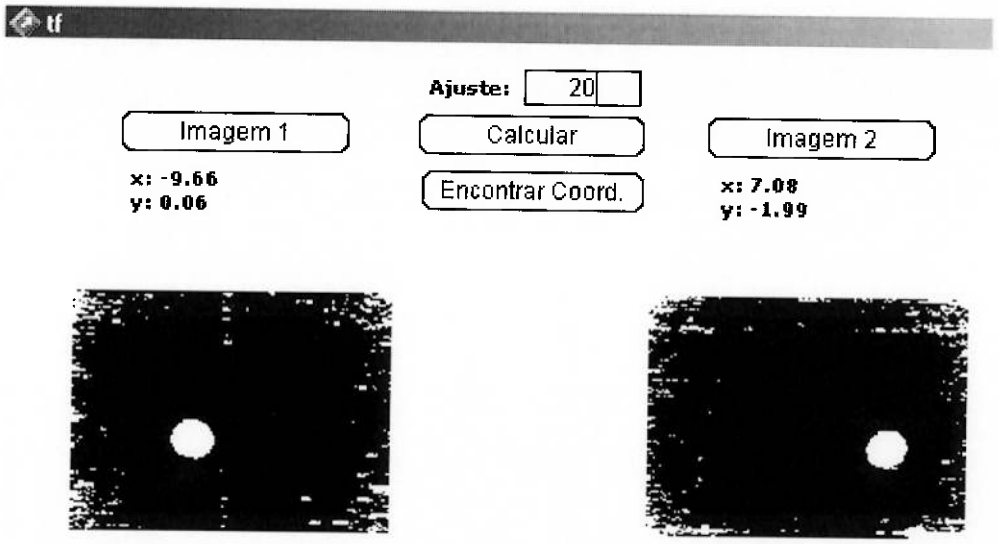

Figura 16: Efeito do ajuste menor

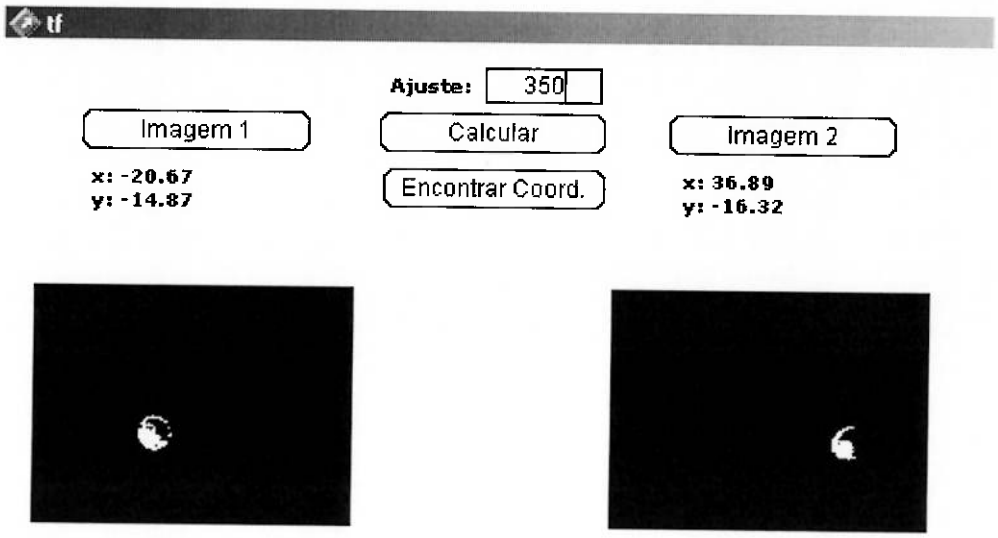

Figura 17: Efeito do ajuste maior

4.7 Triangularização e reposicionamento das câmeras

Para efetuar o reposicionamento das câmeras antes é necessário determinar o posicionamento real do objeto através das duas imagens obtidas, para isso é necessário realizar a triangularização que é representada pela figura abaixo:

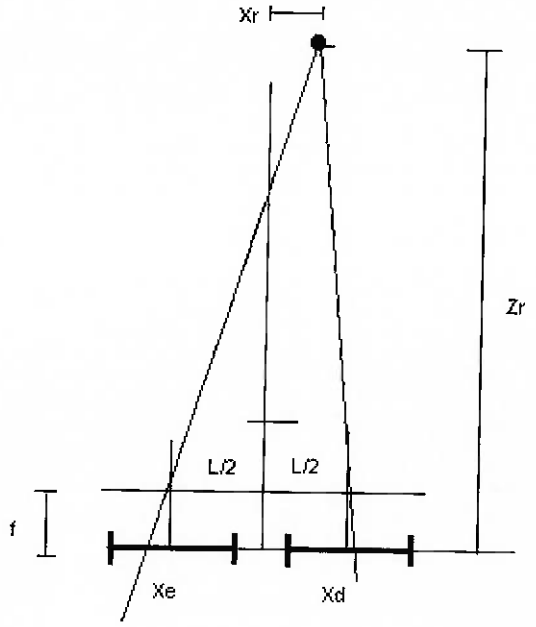

Figura 18: Triangularização

Por semelhança de triângulos obtém-se, ajustando-se alguns sinais em função dos valores fornecidos pelo software de processamento de imagens (nessa situação fornece Xe positivo e Xd negativo)

$$\frac{X_E}{f} = \frac{l/2 + X_R}{Z_R - f} \qquad \frac{-X_D}{f} = \frac{l/2 - X_R}{Z_R - f}$$

Realizando as multiplicações para cada equação:

$$\begin{cases} X_E \cdot (Z_R - f) = fl/2 + fX_R \\ -X_D \cdot (Z_R - f) = fl/2 - fX_R \end{cases} \quad \text{somando as equações :}$$

$$X_E Z_R - X_E f - X_D Z_R + X_D f = f \cdot l$$

A partir dessa equação obtém-se uma equação que determina o valor de Zr (distância real do objeto). O valor de $X_E - X_D$ é chamado de *disparidade*.

$$Z_R = \frac{f \cdot (l + X_E - X_D)}{X_E - X_D} \qquad (1)$$

Com esse valor obtido pode-se determinar a posição ao longo do eixo x do objeto real Xr utilizando-se uma das equações de semelhança de triângulos.

$$X_R = \frac{l}{2} + (Z_R - f) \cdot \frac{X_E}{f} \qquad (2)$$

Já os ângulos de rotação das câmeras são dados por:

$$tg\alpha_D = \frac{\frac{l}{2} - X_R}{Z_R}$$

$$tg\alpha_E = \frac{\frac{l}{2} + X_R}{Z_R}$$

Vale ressaltar que equações semelhantes podem ser determinadas para o outro plano de rotação (rotação ao longo do eixo horizontal).

Repare que a equação (1) apresenta valores conhecidos, com exceção de *f* e *fl*. Esses dois parâmetros são parâmetros lineares no sistema, o mesmo não acontece com o parâmetro *l*, por isso esses dois valores serão determinados a partir de uma aproximação para dois pontos conhecidos no espaço. Com isso será possível realizar a calibração do equipamento, pois está sendo realizada uma aproximação e diferentes fatores além da distância focal estão embutidos nos valores desses parâmetros como por exemplo os efeitos de distorção.

O teste foi realizado com as câmeras retas e apontando para um ponto luminoso a uma distância conhecida das câmeras. As imagens obtidas para o primeiro ponto já foram apresentadas nos tópicos anteriores. Os valores

conhecidos da posição do ponto luminoso na direção X e Z foram: $X_R = 20mm$ $Z_R = 380mm$

Pelo processamento das imagens pode-se obter os valores dos centros luminosos com relação ao eixo x:

(X) Imagem Esquerda = 39.53 e Imagem direita = -19.33

Para um segundo ponto de teste com coordenadas reais conhecidas de $X_R = -50mm$ $Z_R = 410mm$, obteve-se:

(X) Imagem Esquerda = -1.46 e Imagem direita = -54.79

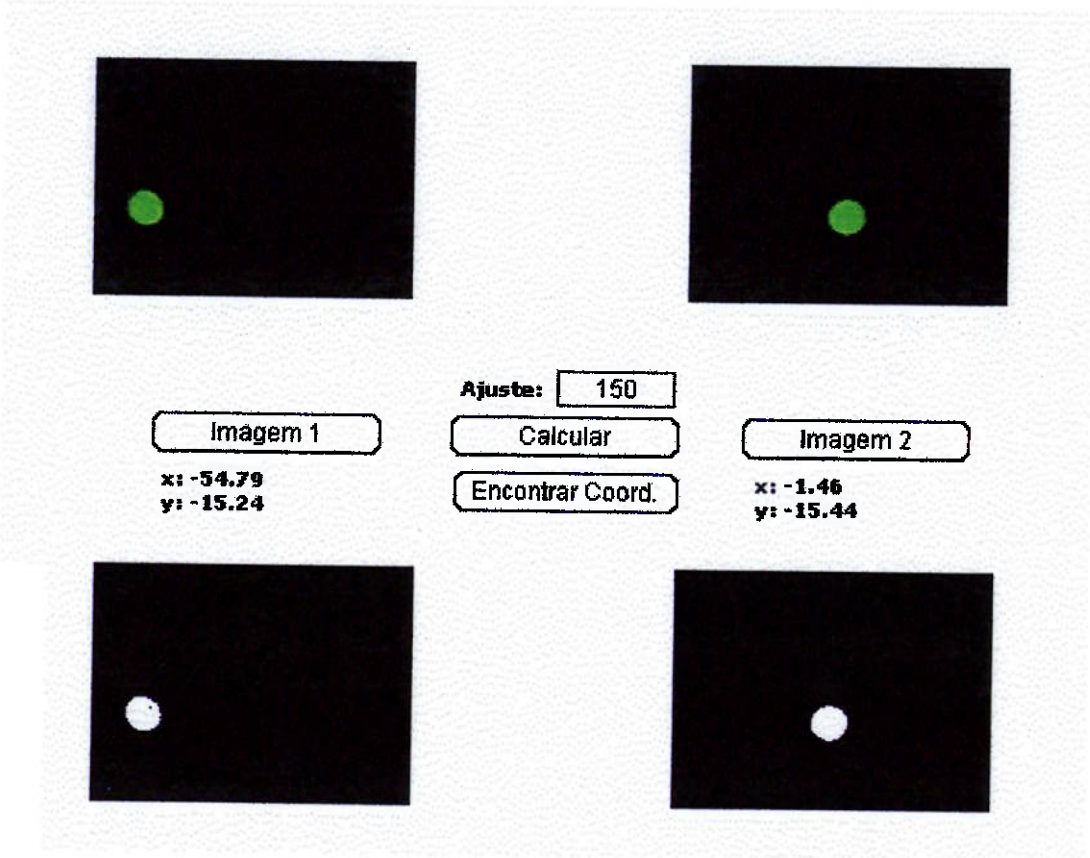

Figura 19: processamento do segundo ponto de teste

1° teste $X_E = 39.53$ $X_D = -19.33$ 2° teste $X_E = -1.46$ $X_D = -54.79$

Substituindo na equação (1) para ambos os testes :

$$(1°) \quad 380 = \frac{f \cdot (l + 58{,}86)}{58{,}86} \Rightarrow 58{,}86 \cdot f + f \cdot l = 22366{,}8 \qquad (3)$$

$$(2°) \quad 410 = \frac{f \cdot (l + 53{,}33)}{53{,}33} \Rightarrow 53{,}33 \cdot f + f \cdot l = 21865{,}3 \qquad (4)$$

Subtraindo as equações obtém-se os dois parâmetros procurados. Vale ressaltar que esses parâmetros têm embutidos o efeito do uso das imagens em pixels e as distâncias em mm.

$$5{,}53 \cdot f = 501{,}5 \Rightarrow \mathrm{f} = 90{,}7$$
$$\mathrm{f} \cdot \mathrm{l} = 22366{,}8 - 90{,}7 \cdot 58{,}86 = 17028{,}2$$

Com esses parâmetros e as equações (1) e (2) , além das equações dos ângulos de rotação poderia ser feito o posicionamento das câmeras. Porém foi observado que esses parâmetros variavam muito de acordo com os pontos de testes, o que tornavam os resultados inconsistentes. Por isso foi feita uma nova abordagem para o posicionamento nas câmeras, levando em conta, além da mudança de perspectiva, fatores como a indeterminação do eixo de coordenadas das câmeras e o efeito dos fatores de escala.

Essa abordagem é apresentada no próximo tópico.

4.8 Posicionamento por Matrizes Homogêneas

Inicialmente foi definido o sistema de coordenadas mundial ($O_W X_W Y_W Z_W$), fixo, cujo centro O_W se situa no ponto médio entre as câmeras e X_W é na sobre o eixo que suporta as câmeras. Como os pontos medidos

estarão à frente do equipamento definiu-se Zw com valores positivos para esses pontos.

Em seguida foram determinados sistemas de coordenadas nas câmeras. Na câmera da esquerda ($O_1X_1Y_1Z_1$) e na câmera da direita ($O_2X_2Y_2Z_2$). Esses sistemas são fixos à câmera e conseqüentemente se movimentam em relação ao sistema de coordenas mundiais. Essa representação pode ser vista na figura abaixo:

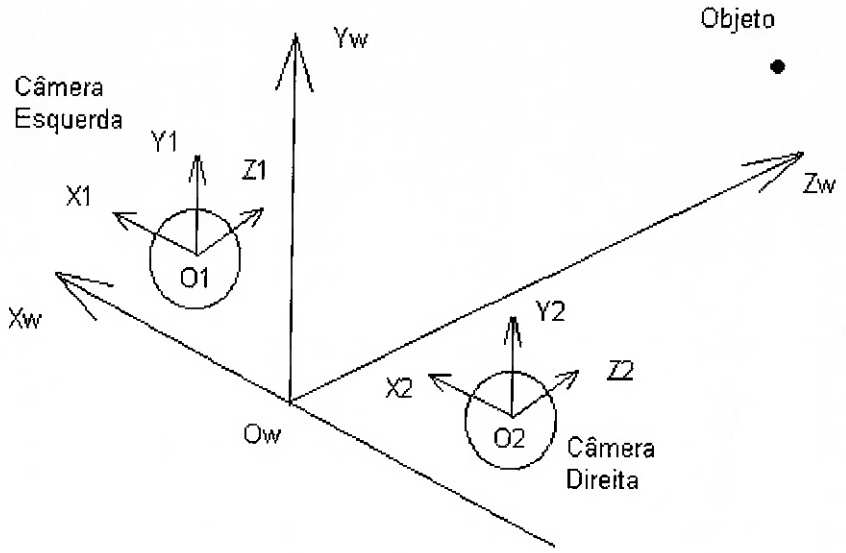

Figura 20: Sistemas de Coordenadas

Os graus de liberdade desses sistemas são três, porém apenas dois para cada câmera: rotação ao redor de Y com o movimento do motor de passo menor e rotação ao redor de X com o movimento do motor de passo maior, sendo que este grau de liberdade é acoplado nas duas câmeras, ou seja, a rotação é a mesma. Já as rotações em Z não são possíveis, estando Z_W, Z_1 e Z_2 sempre paralelos.

Apesar de no desenho apresentado as câmeras estarem alinhadas, será considerado que existe uma rotação α ao redor de X e uma rotação de β ao redor de Y. Com isso, as matrizes de rotação correspondentes á essas rotações são:

$$R_{X\alpha} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\operatorname{sen}\alpha \\ 0 & \operatorname{sen}\alpha & \cos\alpha \end{bmatrix} \qquad R_{Y\beta} = \begin{bmatrix} \cos\beta & 0 & \operatorname{sen}\beta \\ 0 & 1 & 0 \\ -\operatorname{sen}\beta & 0 & \cos\beta \end{bmatrix}$$

Compondo as duas rotações, sendo estas feitas em relação aos eixos originais, obtém-se:

$$R = R_{Y\beta} \cdot R_{X\alpha} = \begin{bmatrix} \cos\beta & \operatorname{sen}\alpha \cdot \operatorname{sen}\beta & \cos\alpha \cdot \operatorname{sen}\beta \\ 0 & \cos\alpha & -\operatorname{sen}\alpha \\ -\operatorname{sen}\beta & \operatorname{sen}\alpha \cdot \cos\beta & \cos\alpha \cdot \cos\beta \end{bmatrix}$$

No que se refere às translações, o centro de coordenadas das câmeras não tem suas posições conhecidas, por isso iremos usar os valores T_X, T_Y, T_Z, para as translações em X, Y e Z respectivamente. Compondo a matriz de rotação anterior com as translações apresentadas é obtida a matriz de transformação homogênea correspondente entre o sistema das câmeras e o sistema de coordenadas mundiais:

$$A_{câmeras}^{realW} = \begin{bmatrix} \cos\beta & \operatorname{sen}\alpha \cdot \operatorname{sen}\beta & \cos\alpha \cdot \operatorname{sen}\beta & T_X \\ 0 & \cos\alpha & -\operatorname{sen}\alpha & T_Y \\ -\operatorname{sen}\beta & \operatorname{sen}\alpha \cdot \cos\beta & \cos\alpha \cdot \cos\beta & T_Z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Além desses fatores, outro fator que deve ser considerado é o fator de escala. Isso porque a câmera apresenta certos fatores de redução ou ampliação da imagem além da óptica do processo. Com isso será definida uma matriz homogênea de escalabilidade cujos elementos são:

Sx : fator de escala na direção X

Sy : fator de escala na direção Y

Sz : fator de escala na direção Z

$$S = \begin{bmatrix} Sx & 0 & 0 & 0 \\ 0 & Sy & 0 & 0 \\ 0 & 0 & Sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Esses fatores são os que não haviam sido considerados anteriormente, pois só havia sido levada em conta a aproximação por semelhança de triângulos, ou seja, a mudança de perspectiva.

Essa mudança é representada também por uma matriz homogênea na forma abaixo, sendo o parâmetro *f* a distância focal da lente:

$$F = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1/f & 1 \end{bmatrix}$$

O efeito da distorção causado pela própria lente da câmera não será considerado, pois seu efeito é pequeno comparado aos demais.

Com isso, ao realizar a multiplicação dessas três matrizes homogêneas obtemos a matriz homogênea de transformação de coordenadas da câmera para o sistema de coordenadas do mundo real.

$$H_{câmera}^{realW} = A \cdot S \cdot F = \begin{bmatrix} S_X \cos\beta & S_Y \operatorname{sen}\alpha \cdot \operatorname{sen}\beta & S_Z \cos\alpha \cdot \operatorname{sen}\beta - T_X/f & T_X \\ 0 & S_Y \cos\alpha & -S_Z \operatorname{sen}\alpha - T_Y/f & T_Y \\ -S_X \operatorname{sen}\beta & S_Y \operatorname{sen}\alpha \cdot \cos\beta & S_Z \cos\alpha \cdot \cos\beta - T_Z/f & T_Z \\ 0 & 0 & -1/f & 1 \end{bmatrix}$$

Com o uso dessa matriz tem-se a correspondência entre as coordenadas do sistema da câmera e do mundo real.

$$\begin{bmatrix} x_{câmera} \cdot c \\ y_{câmera} \cdot c \\ z_{câmera} \cdot c \\ c \end{bmatrix} = H_{câmera}^{realW} \cdot \begin{bmatrix} x_W \\ y_W \\ z_W \\ 1 \end{bmatrix}$$

O resultado da quarta linha *c* deve multiplicar as demais linhas e através das coordenadas obtidas pelos pontos na imagem é possível obter os pontos nas coordenadas reais.

Como pode ser visto a matriz $H_{câmeras}^{realW}$ é composta de diversos parâmetros além dos ângulos de rotação. Para determinar esses parâmetros foi necessário realizar a calibração das câmeras.

Para isso foram utilizados alguns pontos de posições conhecidas no mundo real. Esses pontos foram captados com as câmeras com seus sistemas de coordenadas paralelos ao sistema de coordenadas mundiais, isto é, ângulos de rotação nulos. Claro que não é garantido esse paralelismo perfeito, mas para pequenos ângulos os valores de senos e co-senos são bem próximos dos valores de ângulos nulos de rotação.

Portanto, considerando os ângulos de rotação $\alpha = \beta = 0$, a matriz de transformação homogênea $H_{câmeras}^{realW}$ fica sendo:

$$H_{câmeral}^{realW} = A \cdot S \cdot F = \begin{bmatrix} S_X & 0 & -T_X/f & T_X \\ 0 & S_Y & -T_Y/f & T_Y \\ 0 & 0 & S_Z - T_Z/f & T_Z \\ 0 & 0 & -1/f & 1 \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} x_{câmera} \cdot c \\ y_{câmera} \cdot c \\ z_{câmera} \cdot c \\ c \end{bmatrix} = \begin{bmatrix} S_X & 0 & -T_X/f & T_X \\ 0 & S_Y & -T_Y/f & T_Y \\ 0 & 0 & S_Z - T_Z/f & T_Z \\ 0 & 0 & -1/f & 1 \end{bmatrix} \cdot \begin{bmatrix} x_W \\ y_W \\ z_W \\ 1 \end{bmatrix}$$

Com isso são obtidas as seguintes equações:

$$\begin{cases} x_{\text{câmera}} \cdot c = S_X \cdot x_W - {T_X}/{f} \cdot z_W + T_X \\ y_{\text{câmera}} \cdot c = S_Y \cdot y_W - {T_Y}/{f} \cdot z_W + T_Y \\ z_{\text{câmera}} \cdot c = \left(S_z - {T_Z}/{f} \right) \cdot z_W + T_Z \\ c = {-z_W}/{f} + 1 \end{cases}$$

A terceira equação não será utilizada, pois a imagem apresenta valores em x e y apenas, sendo que os parâmetros da terceira equação não fazem parte das demais equações, podendo então, ser desconsiderados. Substituindo o valor de *c* na primeira e na segunda equação:

$$\begin{cases} x_{\text{câmera}} - x_{\text{câmera}} \cdot {z_W}/{f} = S_X \cdot x_W - {T_X}/{f} \cdot z_W + T_X \\ y_{\text{câmera}} - y_{\text{câmera}} \cdot {z_W}/{f} = S_Y \cdot y_W - {T_Y}/{f} \cdot z_W + T_Y \end{cases}$$

$$\begin{cases} x_{\text{câmera}} \cdot f - S_X \cdot f \cdot x_W + T_X \cdot z_W - T_X \cdot f = x_{\text{câmera}} \cdot z_W \\ y_{\text{câmera}} \cdot f - S_Y \cdot f \cdot y_W + T_Y \cdot z_W - T_Y \cdot f = y_{\text{câmera}} \cdot z_W \end{cases}$$

Os parâmetros a serem determinados na calibração são $f, S_X f, T_X, T_X f$ com os valores das imagens no eixo x e $f, S_Y f, T_Y, T_Y f$ para os valores das imagens no eixo y. O valor de *f* são próximos e será utilizada a sua média.

Vale ressaltar que esses sete parâmetros são para cada uma das câmeras, sendo assim são necessários quatro pontos conhecidos no espaço e conseqüentemente oito imagens para determinar todos os parâmetros. Esses valores foram obtidos dentro do programa de posicionamento e orientação e usados para realizar essa tarefa.

No software de posicionamento, os resultados do software de processamento são inseridos, e são calculados os valores das coordenadas do objeto no mundo real. Esses valores são usados para o cálculo dos ângulos de rotação de cada motor e determinação dos sentidos de rotação. Dividindo os ângulos pelo passo do motor é obtido o número de passos aproximados para o posicionamento, já que o ângulo a ser rodado nem sempre é múltiplo do passo do motor. Esses sinais são repassados à porta paralela através de dois pinos de saída, no caso foram adotados o pino 2 para direção e o pino 3 para geração dos pulsos.

5 RESULTADOS

Foram realizados diversos testes para verificar o funcionamento dos softwares desenvolvidos. Para exemplificar os resultados obtidos serão apresentados resultados de 10 pontos diferentes dos usados na calibração. Para exemplificar o procedimento serão apresentadas as imagens correspondentes ao ponto 9 que foi o que apresentou melhores resultados.

As imagens capturadas são binarizadas e tem seus centros determinados, a partir disso, o software calcula valores aproximados das coordenadas do ponto no mundo real. A figura abaixo apresenta o resultado desse procedimento.

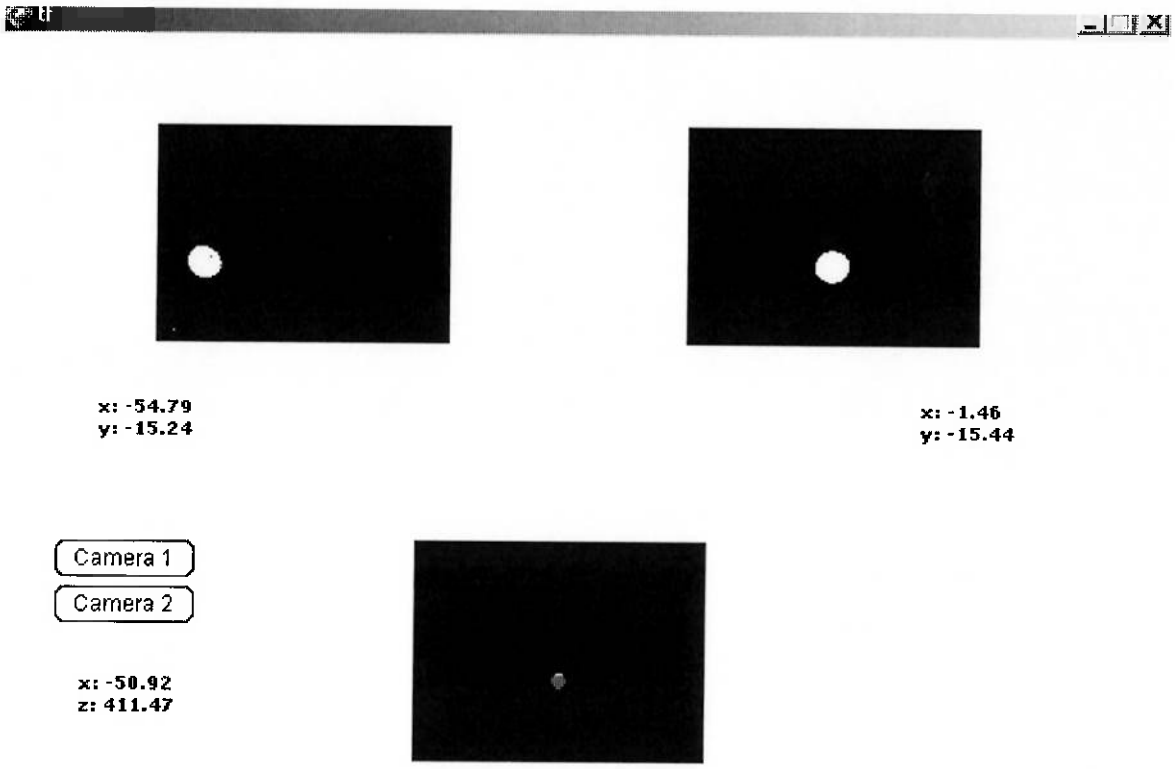

Figura 21: resultado do ponto 9

Os 10 pontos testados, incluindo o ponto 9, se encontram na tabela abaixo:

PONTO	Xreal mm	Zreal mm	Xcalculado	Zcalculado	Erro x %	Erro z %
1	20	360	16,54	377,20	17,3	4,7
2	50	360	45,60	370,97	8,8	3,0
3	25	260	23,09	271,07	7,6	4,3
4	40	460	37,85	452,53	5,4	1,6
5	5	410	3,53	426,96	29,4	4,1
6	-15	360	-18,30	373,00	22,0	3,6
7	-5	310	-6,20	326,91	24,0	5,5
8	-5	160	-6,33	168,12	26,6	5,1
9	-50	410	-50,92	411,97	1,8	0,5
10	-45	310	-46,45	317,75	2,5	2,5

Tabela 7: Comparação de resultados

Percebe-se que os erros no eixo x são maiores que os no eixo z em porcentagem, sendo que os primeiros se concentram nos pontos que apresentam baixos valores de x no mundo real. Já os erros em relação ao eixo z são mais uniformes, mas também são maiores para os pontos mais próximos do eixo de simetria do aparelho (x=0).

Porém esses erros, ao se calcular principalmente os valores de x, não afetam tanto o resultado final do posicionamento, pois a diferença em módulo dos valores de x não é muito elevada, não comprometendo tanto o posicionamento quanto pode parecer inicialmente.

Abaixo segue uma tabela comparativa dos passos reais e dos calculados pelo software de posicionamento e acionamento dos motores. Foram usadas as fórmulas dos ângulos e o valor de 0,45° para *half-step. (Lembrando que l=100mm distância entre as câmeras)*

$$tg\alpha_D = \frac{\frac{l}{2} - X_R}{Z_R} \qquad tg\alpha_E = \frac{\frac{l}{2} + X_R}{Z_R}$$

PONTO	Passos esq. real	Passos esq. calculado	Passos dir. real	Passos dir. calculado
1	24	*22*	11	*11*
2	35	*32*	0	*2*
3	36	*34*	12	*13*
4	25	*24*	3	*3*
5	17	*16*	14	*14*
6	12	*11*	23	*23*
7	18	*17*	22	*22*
8	35	*32*	42	*41*
9	0	*0*	30	*31*
10	2	*1*	38	*38*

Tabela 8 : Comparação dos passos reais e calculados

Percebe-se a pequena diferença entre os passos reais necessários e os passos calculados após todo o processo.

Com esses ângulos foram efetuadas as rotações, e após o posicionamento foram capturadas as novas imagens. É esperado que essas imagens estejam o mais centralizado possível, pois este é o objetivo do projeto, centralizar em um determinado ponto como se estivesse sendo seguido.

Após serem capturadas, essas imagens são novamente processadas pelo software de processamento de imagens, como pode ser observado na imagem abaixo que exemplifica o ponto 9.

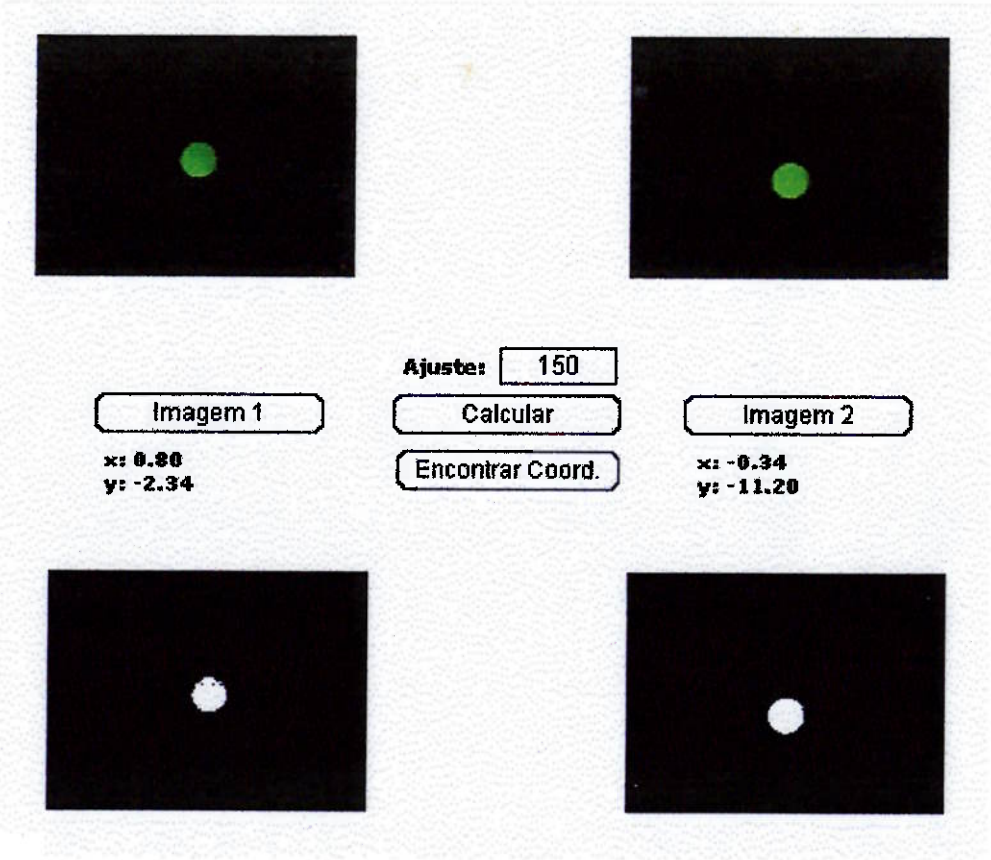

Figura 22: Coordenadas após focalização

Percebe-se que o erro após todo o processo ficou menor que 1 pixel em uma imagem de 160 pixels na direção x, o que é um resultado bem satisfatório. Vale ressaltar que a diferença de valores na coordenada y ocorreu pela movimentação acidental do aparelho nessa direção durante a tomada das imagens.

O resultado apresentado foi um dos melhores obtidos, já que para esse ponto foram obtidos os melhores resultados calculados em relação ao mundo real. Nos demais pontos esse erro em pixels é maior, como pode ser visto na tabela abaixo.

PONTO	Erro em pixels direita	Erro em pixels Esquerda
1	*2.40*	*1.78*
2	*5.65*	*3.21*
3	*3.11*	*5.30*
4	*4.35*	*3.59*
5	*2.11*	*3.78*
6	*3.26*	*3.82*
7	*1.89*	*2.48*
8	*3.22*	*4.46*
9	*0.80*	*0.34*
10	*4.67*	*3.31*

Tabela 9 : Erros em pixels

Observa-se que os erros não ultrapassam 6 pixels em nenhum dos 10 casos, claro que isso é uma resposta para essa amostragem, mas reflete um bom resultado do experimento, o que o torna válido para a aplicação de perseguição simples. Há muitas fontes de erros que interferem no processo, uma forma de melhorar os resultados seria trabalhar em cima delas, por exemplo:

- Imprecisões construtivas
- Imprecisões nos pontos usados na calibração
- Efeitos da distorção das lentes
- Falta de informações sobre os parâmetros das câmeras, sendo necessário a estimativa desses parâmetros.
- Iluminação do ambiente
- Elemento de teste: já que a pequena lâmpada verde utilizada se apresenta como uma mancha na imagem
- Efeito do ajuste: variando os ajustes, para as mesmas figuras se encontram valores diferentes de erros em pixels
- Passo limitado em 0.45º dos motores de passo

O posicionamento realizado e os resultados apresentados foram para a direção x. Na direção y (rotação do eixo horizontal) foi aplicado o mesmo equacionamento, sendo que os programas também calculam o valor de y nas imagens e no mundo real, bem como o número de passos necessários na movimentação. Só não foi possível apresentar os resultados, pois o motor maior que é responsável por esse movimento não funcionou, ou por problemas do próprio motor ou por falta de corrente fornecida pelo controlador, sendo necessário um controlador que suporte maior potência.

6 CONCLUSÕES

Conforme pôde ser observado no tópico de resultados, o projeto conseguiu atender bem às necessidades de um sistema simples de rastreamento não real-time. Os erros em pixels foram pequenos e não comprometeram o apontamento, haja visto o grande número de fontes de erro do sistema.

Porém esses erros devem ter sido os responsáveis pelo não funcionamento da triangularização simples (mudança de perspectiva). Pois é um método amplamente utilizado e que deveria apresentar bons resultados. Possivelmente o conhecimento dos parâmetros construtivos das câmeras como a distância focal e a resolução em pixels por mm (ou pixels por polegadas) da célula de captação da imagem ajudariam a encontrar resultados bem melhores, mas infelizmente essas informações não se encontram disponíveis.

Espera-se que o sistema implementado possa ser utilizado em pesquisas futuras na área de visão estéreo, principalmente no que se refere aos softwares, pois a parte mecânica e a eletrônica se encontram satisfatórias. Com relação ao sistema computacional este apresenta diversos módulos que foram desenvolvidos separadamente, já que o sistema de rastreamento não é em real-time. Para pesquisas futuras com o uso desse sistema, os softwares poderão ser integrados e modificados de acordo com as novas necessidades. Apesar da simplicidade, a atual aplicação fornece importantes informações a respeito dos tipos de processos envolvidos em um sistema de visão estéreo computacional.

7 BIBLIOGRAFIA

[1] Acarnley, P.P., *Stepping motors :a guide to modern theory and practice /P P Acarnley.,* 3rd Edition, Stevenage, 1992

[2] Crowder, R.M., *Electric drives and their controls,* Oxford Press, New York, 1995

[3] Kaminski, P.C., *Desenvolvendo produtos com planejamento, criatividade e qualidade,* Editora LTC, Rio de Janeiro, 2000

[4] Kenjo, T., *Stepping motors and their microprocessor controls,* Oxford Press, New York, 1984

[5] Shigley, J.E., *Cinemática dos Mecanismos,* Edusp, São Paulo, 1969

[6] Shigley, J.E., *Elementos de Máquinas*, Editora LTC, Rio de Janeiro, 1984

[7] Skarski, B., *Síntese cinemática dos mecanismos*, Editora da Unicamp, Campinas, 1980

[8] JRC Japan Radio Co., *Stepper Motors Basics*, Japan.

[9] Fu,K.S. , Gonzales,R.C., Lee, E.S., *Robotics: Control, Sensing, Vision and Inteligence,* McGraw-Hill, 1987

8 ANEXOS

Seguem anexados os códigos-fonte dos programas implementados.

8.1 Programa de Processamento de Imagem

Esse programa foi desenvolvido no Director Lingo, mas poderia ser desenvolvido usando linguagem C ou Java. Esse programa abre duas imagens e as binariza em função de um valor de ajuste. Cada pixel é analisado pelas suas componentes no sistema RGB (*red-green-blue)*.

Após a binarização da imagem o programa calcula as coordenadas (x,y) do centro da região branca em relação ao centro da imagem.

```
// Software de processamento de imagens

on mouseUp

 member(7).image = member(5).image
 member(8).image = member(6).image
 the floatPrecision = 2
 cut = integer(member("cut").text)
 if cut > 750 then
  alert("Ajuste deve estar entre 0 e 750")
  exit
 end if

 xTotal = 0
 yTotal = 0
 jTotal = 0
 iTotal = 0
 width = member(7).width-1
 height = member(7).height-1
```

```
  repeat with i = 0 to width
    repeat with j = 0 to height

      color = member(7).image.getPixel(i,j)
      media = color.red + color.green + color.blue
      if media < cut then
        member(7).image.setPixel(i,j,rgb(0,0,0))
        tipo = 0
      else
        member(7).image.setPixel(i,j,rgb(255,255,255))
        tipo = 1
      end if
      xTotal = xTotal + i*tipo
      yTotal = yTotal + j*tipo
      jTotal = jTotal+tipo
      iTotal = iTotal+tipo
    end repeat
  end repeat
  if iTotal > 0 and jTotal> 0 then
    member("t1").text =  "x"&&float(xTotal)/iTotal&RETURN&"y"&&
float(yTotal)/jTotal
  else
    member("t1").text =  "x 0"&RETURN&"y 0"
  end if

  width = member(8).width-1
  height = member(8).height-1
  xTotal = 0
  yTotal = 0
  jTotal = 0
  iTotal = 0

  repeat with i = 0 to width
    repeat with j = 0 to height
```

```
        color = member(8).image.getPixel(i,j)
        media = color.red + color.green + color.blue
        if media < cut then
          member(8).image.setPixel(i,j,rgb(0,0,0))
          tipo = 0
        else
          member(8).image.setPixel(i,j,rgb(255,255,255))
          tipo = 1
        end if
        xTotal = xTotal + i*tipo
        yTotal = yTotal + j*tipo
        jTotal = jTotal+tipo
        iTotal = iTotal+tipo

      end repeat
    end repeat
    if iTotal > 0 and jTotal> 0 then
      member("t2").text =  "x"&&float(xTotal)/iTotal&RETURN&"y"&&
  float(yTotal)/jTotal
    else
      member("t2").text =  "x 0"&RETURN&"y 0"
    end if
  end mouseUp
```

8.2 Programa de Posicionamento e Acionamento dos Motores

Esse programa realizado em C, obtém pelos processos já analisados as coordenadas reais através dos valores obtidos pelo software de processamento de imagens. Com esses valores o programa calcula os ângulos de rotação e o número de passos a serem dados e em que sentido de rotação.

A partir disso é usado o módulo de controle da porta paralela para envio dos sinais.

```
/*****************************************************************************************
***************
* Include Header Files
*****************************************************************************************
*************
*/
#include <dos.h>
#include <bios.h>
#include <stdio.h>
#include <conio.h>
#include <prn_io.h>
/*
Esse arquivo "prn_io.h" sera mostrado posteriormente no relatorio
*/
#include<graphics.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>
#include <math.h>

/*
*****************************************************************************************
*************
* Descrição: Rotinas de I/O da Porta Paralela
*
* As funções abaixo implementam um modulo em baixo nível em C para
interfaceamento com a
* porta convencional de um PC, habilitando 12 saídas de dados, e 5 entradas
* Nota sobre os endereçamentos das portas:
*
```

```
*  LPT1 = 0x0378 ou 0x03BC
*  LPT2 = 0x0278 ou 0x0378
*  LPT3 = 0x0278
*
* Esse modulo assume que a sa´da sera feita atrave´s de LPT1 em 0x0378.
* A porta de paralela tem 3 registradores de 8-bits:
*
*  Data Register (base + 0) ..... saídas
*
*   7 6 5 4 3 2 1 0
*   . . . . . . . * D0 ..... (pino 2), 1=High, 0=Low (true)
*   . . . . . . * . D1 ..... (pino 3), 1=High, 0=Low (true)
*   . . . . . * . . D2 ..... (pino 4), 1=High, 0=Low (true)
*   . . . . * . . . D3 ..... (pino 5), 1=High, 0=Low (true)
*   . . . * . . . . D4 ..... (pino 6), 1=High, 0=Low (true)
*   . . * . . . . . D5 ..... (pino 7), 1=High, 0=Low (true)
*   . * . . . . . . D6 ..... (pino 8), 1=High, 0=Low (true)
*   * . . . . . . . D7 ..... (pino 9), 1=High, 0=Low (true)
*
*  Status Register (base + 1) ..... entradas
*
*   7 6 5 4 3 2 1 0
*   . . . . . * * * Undefined
*   . . . . * . . . Error ..... (pino 15), high=1, low=0 (true)
*   . . . * . . . . Selected ..... (pino 13), high=1, low=0 (true)
*   . . * . . . . . No paper ..... (pino 12), high=1, low=0 (true)
*   . * . . . . . . Ack ..... (pino 10), high=1, low=0 (true)
*   * . . . . . . . Busy ..... (pino 11), high=0, low=1 (inverted)
*
*  Control Register (base + 2) ..... saídas
*
*   7 6 5 4 3 2 1 0
*   . . . . . . . * Strobe ..... (pino 1),  1=low, 0=high (inverted)
*   . . . . . . * . Auto Feed .... (pino 14), 1=low, 0=high (inverted)
```

```
*     . . . . . * . .  Initialize ... (pino 16), 1=high, 0=low (true)
*     . . . . * . . .  Select ..... (pino 17), 1=low, 0=high (inverted)
*     * * * * . . . .  sem uso
*
* Pinos 18-25 são terra.
*
* Esse modulo mantém shadow buffers para os registradores de data e
controle, para que um
* único bit possa ser modificado sem perturbar outros bits com o mesmo
registro.  Todas as
* entradas e saídas possuem uma uma função correspondente para ler ou
escrever,
* Além disso, a função PrnIO_Data() permite escrever o registro inteiro de uma
vez,
* permitindo comunicação com equipamentos que comunicam com vários bits.
*
* Esse módulo foi escrito para ser usado com Borland C/C++ v5.0.  Ele pode
ser usado com outros
* compiladores, mas você terá que mudar as funções import() e outport() para
funções equivalentes
* das bibliotecas destes compiladores.
*
**********************************************************************************
**********
*/

/*
**********************************************************************************
**********
* Constantes
**********************************************************************************
**********
*/
#define BASE_ADDRESS     0x0378                  // endereço da porta: LPT1
```

```
#define DATA_REGISTER    (BASE_ADDRESS+0)         // endereços dos
registradores
#define STATUS_REGISTER   (BASE_ADDRESS+1)
#define CONTROL_REGISTER  (BASE_ADDRESS+2)
#define F 235
#define L 100
#define PI 3.141592
#define PASSO1 0.45
#define PASSO2 2.5
#define TIME 50

/*
*****************************************************************************************
**************
* Dados
*****************************************************************************************
**************
*/
static unsigned char data_reg;
static unsigned char control_reg;

float pMotor1=90, pMotor2=90, pMotor3=0;
float pM1Now=90, pM2Now = 90, pM3Now = 0;
// Supoem que os motores estejam zerados.
float Xobj, Yobj, Zobj, Xe, Xd;


// ..... Funcoes .....


/*
*****************************************************************************************
**************
* PrnIO_Init()
```

```
 *
 * Descricao: Funcao inicial
 * Arguments  : none
 * Returns    : none
 ********************************************************************************************
 **************
 */
 void PrnIO_Init (void)
 {
  data_reg = control_reg = 0;                    // clear shadow registers
 }


 /*
 ********************************************************************************************
 **************
 * Descricao: Rotinas de entrada e saída.
 ********************************************************************************************
 **************
 */
 void PrnIO_D0 (char state)
 {
  if (state == 0)
        data_reg &= ~0x01;
  else
        data_reg |=  0x01;
  outport(DATA_REGISTER,data_reg);
}


void PrnIO_D1 (char state)
{
 if (state == 0)
       data_reg &= ~0x02;
```

```
    else
          data_reg |=  0x02;
    outport(DATA_REGISTER,data_reg);
   }


  void PrnIO_D2 (char state)
  {
    if (state == 0)
          data_reg &= ~0x04;
    else
          data_reg |=  0x04;
    outport(DATA_REGISTER,data_reg);
  }


  void PrnIO_D3 (char state)
  {
    if (state == 0)
          data_reg &= ~0x08;
    else
          data_reg |=  0x08;
    outport(DATA_REGISTER,data_reg);
  }


 void PrnIO_D4 (char state)
 {
   if (state == 0)
         data_reg &= ~0x10;
   else
         data_reg |=  0x10;
   outport(DATA_REGISTER,data_reg);
 }
```

```
 void PrnIO_D5 (char state)
 {
  if (state == 0)
        data_reg &= ~0x20;
  else
        data_reg |=  0x20;
  outport(DATA_REGISTER,data_reg);
 }


 void PrnIO_D6 (char state)
 {
  if (state == 0)
        data_reg &= ~0x40;
  else
        data_reg |=  0x40;
  outport(DATA_REGISTER,data_reg);
 }


 void PrnIO_D7 (char state)
 {
  if (state == 0)
        data_reg &= ~0x80;
  else
        data_reg |=  0x80;
  outport(DATA_REGISTER,data_reg);
 }


 void PrnIO_Data (unsigned char dataout)
 {
```

```
    data_reg = dataout;
    outport(DATA_REGISTER,data_reg);
  }


  char PrnIO_Error (void)
  {
   if (inport(STATUS_REGISTER) & 0x08)
         return 1;
   else
         return 0;
  }


 char PrnIO_Selected (void)
 {
  if (inport(STATUS_REGISTER) & 0x10)
        return 1;
  else
        return 0;
 }


char PrnIO_PaperOut (void)
{
 if (inport(STATUS_REGISTER) & 0x20)
       return 1;
 else
       return 0;
}


char PrnIO_Acknowledge (void)
{
```

```
    if (inport(STATUS_REGISTER) & 0x40)
            return 1;
    else
            return 0;
  }


  char PrnIO_Busy (void)
  {
   if (inport(STATUS_REGISTER) & 0x80)
            return 0;
   else
            return 1;
  }


  void PrnIO_DataStrobe (char state)
  {
   if (state == 0)
            control_reg |=  0x01;
   else
            control_reg &= ~0x01;
   outport(CONTROL_REGISTER,control_reg);
  }


 void PrnIO_Autofeed (char state)
 {
  if (state == 0)
          control_reg |=  0x02;
  else
          control_reg &= ~0x02;
  outport(CONTROL_REGISTER,control_reg);
 }
```

```
 void PrnIO_InitOut (char state)
 {
  if (state == 0)
        control_reg &= ~0x04;
  else
        control_reg |=  0x04;
  outport(CONTROL_REGISTER,control_reg);
 }


 void PrnIO_Select (char state)
 {
  if (state == 0)
        control_reg |=  0x08;
  else
        control_reg &= ~0x08;
  outport(CONTROL_REGISTER,control_reg);
 }


void start() {
        //textColor(15);
        clrscr();

        printf("\nPrograma de Controle dos motores atraves da Paralela\n");
}

int menu(void) {

        int choice;

        PrnIO_D0(1);
```

```
        //PrnIO_D1(1);

        do {
        printf("\nEscolha o metodo de apontamento:\n");
        printf("1 - Ajuste por 3 coordenadas.\n");
        printf("2 - Atraves de Xe e Xd.\n");
        choice = getche();
        } while ((choice != 49) && (choice != 50));

        if (choice == 49) {
                printf("\n\nEntre com o valor da coordenada x:");
                scanf("%f",&Xobj);
                printf("Entre com o valor da coordenada y:");
                scanf("%f",&Yobj);
                printf("Entre com o valor da coordenada z:");
                scanf("%f",&Zobj);
                return(1);
        }
        else {
                printf("\n\nEntre com o valor de Xe:");
                scanf("%f",&Xe);
                printf("Entre com o valor de Xd:");
                scanf("%f",&Xd);
                return(2);
        }
}

void setCoordinates(){

        Zobj = (F*L)/(Xe - Xd);
        printf("\n\n  Distancia do objeto: %.0f\n\n\n", Zobj);
        Xobj = (Zobj*(Xe+Xd))/(2*F);
        printf("\n\n  Distancia X: %.0f\n\n\n", Xobj);
}
```

```
void adjustCoordinates() {

        //aki vai o programa que determina quanto temos que rotacionar cad
        //motor para ele apontar para a coordenada especificada.
        //
   pMotor1 = pMotor1 - (atan2((L/2 + Xobj),Zobj))*(180/PI);
   pMotor2 = pMotor2 - (atan2((L/2 - Xobj),Zobj))*(180/PI);
   printf("Angulos: %.2f    %.2f", pMotor2, pMotor1);

}

void rotateMotor2(){

        int nPassos, dir, i;
        nPassos = ((atan2((L/2 + Xobj),Zobj))*(180/PI))/PASSO1;
        nPassos = abs(nPassos);
        printf("\n\nNPassos: %d\n", nPassos);
        if (pMotor2<=pM2Now) {
                dir = -1;
                PrnIO_D0(1);
        }
        else {
                dir = 1;
                PrnIO_D0(0);
        }


       for(i=0; i<nPassos; i++){

                pM2Now = pM2Now+PASSO1*dir;
                printf("\nPosicao atual do motor 2: %.2f", pM2Now);
                PrnIO_D1(1);
                delay(TIME);
```

```
            PrnIO_D1(0);
            delay(TIME);

        }
}

void rotateMotor1(){

        int nPassos, dir, i;
        nPassos = ((atan2((L/2 - Xobj),Zobj))*(180/PI))/PASSO1;
        nPassos = abs(nPassos);
        printf("\n\nNPassos: %d\n", nPassos);
        if (pMotor1<=pM1Now) {
            dir = -1;
            PrnIO_D0(0);
        }
        else {
            dir = 1;
            PrnIO_D0(1);
        }

        for(i=0; i<nPassos; i++){

            pM1Now = pM1Now+PASSO1*dir;
            printf("\nPosicao atual do motor 1: %.2f", pM1Now);
            PrnIO_D1(1);
            delay(TIME);
            PrnIO_D1(0);
            delay(TIME);

        }
}
```

```
void main(void){

        int choice;
        start();
        PrnIO_Init();
        choice = menu();
        if (choice == 2){
                setCoordinates();
        }
        adjustCoordinates();
        printf("\n\nPronto para rotacionar camera 1...");
        getch();
        rotateMotor1();
        printf("\n\nAguardando verba para camera 2...");
        getch();
        rotateMotor2();
//rotateMotor3();

printf("\n\n\nPress any key to continue...");
getch();


end:
}


    ********************************************************************************
**
```