SISTEMA DE VISÃO ESTÉREO EM TEMPO REAL

Juliano Tusi Amaral Laganá Pinto[1]; Vanderlei Cunha Parro[2]

[1] Aluno de Iniciação Científica da Escola de Engenharia Mauá (EEM/CEUN-IMT);
[2] Professor da Escola de Engenharia Mauá (EEM/CEUN-IMT).

Resumo. *Este trabalho apresenta o desenvolvimento de um sistema de visão computacional com reconhecimento de imagem em tempo real. Deseja-se reconhecer uma bola laranja de tênis de mesa no espaço prevendo sua trajetória futura. Todo sistema descrito foi prototipado no software MATLAB e é realizado em uma máquina executando algoritmos escritos na linguagem de programação C++.*

Introdução

A visão computacional é a ciência e tecnologia das máquinas que enxergam. Atualmente a visão computacional é aplicada em projetos como processamento de imagens no diagnóstico de doenças na Medicina, controle de qualidade de produtos na Indústria, detecção de alvos inimigos na área Militar e visão de veículos autônomos na Indústria Automobilística. Neste trabalho é demonstrado passo a passo o desenvolvimento de um sistema de visão capaz de reconhecer uma bola de tênis de mesa laranja no espaço de uma mesa de jogo.

Material e Métodos

Modelagem matemática do fenômeno

Para o modelo que será descrito a seguir adotou-se um sistema de coordenadas fixo na borda da mesa que as câmeras estarão, com o eixo 'z' orientado na direção contrária da visão das câmeras, como mostrado na figura 1.

Figura 1 - Representação do sistema de coordenadas fixo na mesa

Considerando a bola de tênis de mesa como um corpo rígido e desprezando o efeito do atrito com o ar, segue que a única força exercida sobre a bolinha é o seu peso (na direção contrária do versor y). Usando essa simplificação aplica-se a segunda lei de Newton, o que resulta em:

$$a\ t\ = \begin{matrix} 0 \\ -g \\ 0 \end{matrix} \qquad \text{(I)}$$

Integrando a equação I duas vezes em relação ao tempo e estipulando condições iniciais x_0, y_0, z_0, V_{x0}, V_{y0} e V_{z0} obtém-se a seguinte equação:

$$x\ t\ = \begin{matrix} V_{x0}.t + x_0 \\ -\frac{g.t^2}{2} + V_{y0}.t + y_0 \\ V_{z0}.t + z_0 \end{matrix} \qquad \text{(II)}$$

que define a posição da bola durante todos os instantes em que a única força aplicada nela é seu peso.

Durante o impacto da bola com a mesa a equação acima não descreve corretamente o seu comportamento. Durante tal impacto, existe outra força sendo aplicada ao objeto (a força de contato com a mesa) e por esse motivo é necessário modelar esse caso separadamente. Essa colisão foi modelada como uma colisão parcialmente elástica entre corpos rígidos (adicionando a simplificação de que a mesa também é um corpo rígido), portanto segue a seguinte equação (WITKOWSKI, Francisco Mauro. Física I. São Paulo, SP: EP, 1995. pt. 3.):

$$v_a' - v_b' \,.n = -\varepsilon.\ v_a - v_b \,.n \qquad \text{(III)}$$

onde n é o vetor normal ao impacto, ε é o coeficiente de restituição entre os dois corpos (nesse caso entre a bola de tênis de mesa e a mesa), v_a e v_b são respectivamente a velocidade da bola e da mesa antes do impacto e v_a' e v_b' são as velocidades dos mesmos corpos após o impacto. No caso estudado em especial podemos adicionar a suposição de que a mesa está em repouso, além disso, por causa do modo como foi definido o sistema de coordenadas (figura 1) e supondo que a bola é uma esfera perfeita, todas as colisões com a mesa terão como vetor n o próprio versor y. Substituindo essas particularidades na equação acima obtém-se:

$$\begin{matrix} V_x' \\ V_y' \\ V_z' \end{matrix} \begin{matrix} 0 \\ .\ 1 \\ 0 \end{matrix} = -\varepsilon. \begin{matrix} V_x \\ V_y \\ V_z \end{matrix} \begin{matrix} 0 \\ .\ 1 \\ 0 \end{matrix} \qquad \text{(IV)}$$

o que resulta em:

$$V_y' = -\varepsilon . V_y \qquad \text{(V)}$$

As equações (II) e (V) são suficientes para definir o comportamento da bola durante a situação de interesse.

Hardware

Para esse projeto foi desenvolvido inicialmente uma prototipagem completa em ambiente MATLAB®, onde criaram-se câmeras virtuais com posicionamento conhecido em relação a uma mesa de tênis de mesa. Também nesse ambiente modelou-se o fenômeno de lançamento de uma bola e foi possível realizar as análises necessárias de posicionamento e velocidade através das câmeras criadas.

Com o ambiente virtual desenvolvido foi possível definir trajetórias conhecidas para a bola e confronta-las com as estimativas fornecidas pelo sistema de visão computacional implementado nas câmeras virtuais, dessa forma levantou-se uma distribuição do erro de estimativa do sistema ao longo do volume de trabalho de interesse definido para o projeto.

Após os testes no ambiente virtual foi iniciado a implementação do sistema de visão computacional utilizado no projeto seguindo as seguintes etapas:

1- Definição das câmeras: Nessa etapa selecionou-se qual câmera atenderia aos requisitos propostos pelo projeto. Com base em uma análise de mercado escolheu-se as câmeras *Play Station Eye*® devido ao seu baixo custo, fácil disponibilidade para compra e alta taxa de aquisição de imagens (180 quadros por segundo).

2- Adequação das câmeras ao sistema desenvolvido: A câmera definida para ser utilizada nessa aplicação não possui interface com o computador e para corrigir esse imprevisto optou-se pela utilização de um *drive* que realizasse essa

comunicação entre computador e câmeras. No entanto, o único *drive* encontrado e selecionado para tal função não atendeu ao requisito de comunicar duas câmeras simultaneamente o que fez com que fosse acrescido ao escopo desse projeto o desenvolvimento dessa solução. Utilizou-se como base o software disponível capaz de realizar a comunicação com apenas uma câmera.

3- Implementação do sistema de visão computacional: Partindo do ambiente virtual optou-se pela transcrição e implementação dos algoritmos utilizados pela visão computacional para a linguagem C, com o auxílio da biblioteca *OpenCV*. A mudança se fez necessária visto que as câmeras embora se comunicassem com o computador, não eram reconhecidas pelo MATLAB®.

Para que o sistema de visão computacional funcione de maneira correta é de essencial importância ter um ambiente com iluminação controlada. Para que o ambiente apresentasse a luminosidade desejada em todo o volume de trabalho proposto (mesa oficial de tênis de mesa), foram utilizados seis pontos de luz, sendo:

- Dois refletores centrais fixados em cima da mesa de tênis de mesa com lâmpadas fluorescentes brancas de 85 *Watts* de potência, tensão 220 V e soquete E42.
- Quatro lâmpadas fluorescentes brancas, fixadas lateralmente a mesa, de 60 *Watts* de potência, tensão 110V e soquetes E27.

Desenvolvimento do ambiente de simulação

Foi desenvolvido um ambiente de simulação gráfico na plataforma Matlab com a utilização do produto Simulink 3D Animation. O Simulink 3D Animation possibilita que o usuário crie um modelo tridimensional da cena a ser simulada (no nosso caso a mesa de ping-pong e a bolinha) e, de acordo com alguma lógica pré-estipulada, atue em propriedades dos objetos no modelo tais como posição, orientação e tamanho. Utilizando as equações (I) e (2) podemos alterar a posição do modelo tridimensional da bolinha de modo que ela se comporte como em um jogo de tênis de mesa. Esse mesmo produto ainda possibilita o posicionamento de câmeras virtuais na cena, com resolução, distância focal, posição e orientação definidas pelo usuário e a visualização da imagem que uma câmera nessas condições criaria. A figura (2) mostra o modelo tridimensional criado quando visto por cada uma das câmeras.

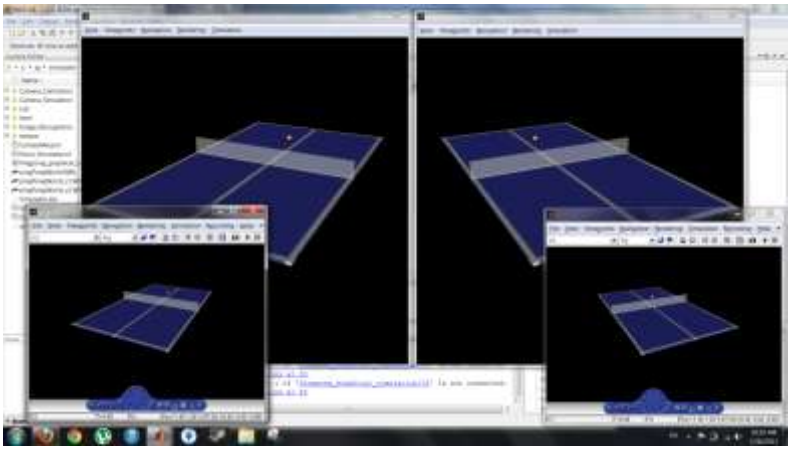

Figura 2 - Ambiente virtual desenvolvido no Simulink 3D Animation®

Reconhecimento de Imagem

O algoritmo desenvolvido não leva em conta todas as características que distinguem uma bolinha do resto da imagem, mas apenas a sua cor. Desse modo, diminui-se a precisão e a sua robustez, mas aumenta-se consideravelmente sua velocidade de execução já que o número de testes necessários para eleger uma região como bolinha é diminuido. Essa falta de

robustez não é um fator limitante nesse projeto, já que é possível condicionar o volume de interesse visto pelas câmeras de modo que nada tenha a mesma cor que a bola.

Filtro por cor

A primeira etapa do reconhecimento é fazer um filtro por cor (algoritmo que separa o que é da cor desejada do resto da imagem). Isso ocorre do seguinte modo: o software varre cada um dos pixels da imagem, um por vez, e verifica se a cor daquele pixel assemelha-se com a cor da bolinha (mais detalhes sobre essa etapa adiante). Ao mesmo tempo que ele faz isso ele vai criando uma nova imagem (com as mesmas dimensões da antiga) que nas posições onde ele encontrou pixels com cores parecidas com a da bola são gravados o valor '1' e em todos os outros '0'; essa abordagem é conhecida na literatura como segmentação por cor e o seu resultado quando aplicada ao nosso caso está exemplificado na figura ç.

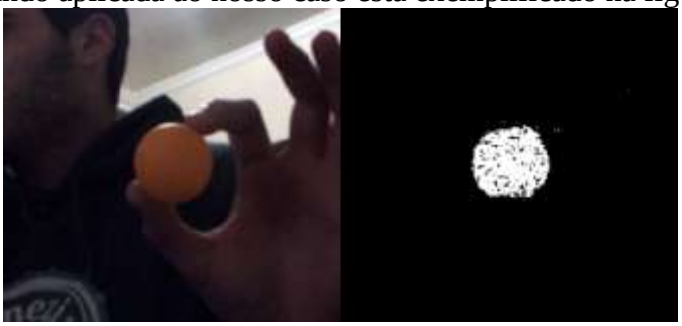

Figura 3 - Filtro por cor aplicado para reconhecer uma bola laranja na imagem. Note que os pixels de valor 1 são brancos e os pixels de valor zero são pretos

A explicação dada acima do filtro por cor está completa, mas é necessário adicionar uma observação: ele é aplicado em uma imagem no espaço de cores HSV, para que a informação de matiz esteja desacoplada das outras características da cor Para maiores informações sobre os espaços de cores RGB, HSV e outros, consultar (GONZALEZ, R.; WOODS, R. Digital Image Processing. 3. Ed. [S.l.]: Prentice Hall, 2007.).

Para que a primeira etapa descrita seja seguida (o sistema saiba quais as cores dos pixels que ele deve considerar como bolinha) é necessária uma prévia calibração do sistema. A abordagem utilizada foi a seguinte: a bola é colocada em diversas posições na mesa de ping-pong, sujeitas à condições de iluminação ligeiramente diferentes, enquanto as câmeras tiram fotos. Em cada uma dessa fotos o usuário utiliza uma interface gráfica para informar ao software a posicão do topo e da base da bolinha. O programa então junta todos os pixels em um vetor e calcula a média e o desvio padrão para cada uma das componentes de cor desses pixels (matiz, saturação e brilho). Com base nessas médias e desvios padrões são estabelecidos limites inferiores e limites superiores que se as componentes de cor de um pixel em questão respeitarem ele será considerado como bolinha. Estes limites são definidos pelo seguinte conjunto de equações:

$$\mu_H - t.\sigma_H \leq H \leq \mu_H + t.\sigma_H \quad \text{(VI)}$$

$$\mu_S - t.\sigma_S \leq S \leq \mu_S + t.\sigma_S \quad \text{(VII)}$$

$$\mu_V - t.\sigma_V \leq V \leq \mu_V + t.\sigma_V \quad \text{(VIII)}$$

onde H, S e V são os valores do pixel em questão; μ_H, σ_H, μ_S, σ_S, μ_V e σ_V são respectivamente a média e o desvio padrão das componentes H,S e V do pixels; e t é um número positivo que representa a tolerância escolhida para o algoritmo.

Cálculo do centro geométrico

Após a binarização da imagem (criação da segunda imagem cujos pixels só podem ter valor '0' ou '1') a próxima etapa é o cálculo do centro geométrico da imagem binária. O cálculo das coordenadas x e y do centro geométrico de um conjunto de n pixels com valor 1, nas posições (x_1, y_1), (x_2, y_2), (x_3, y_3)... (x_n, y_n) é dado pelas seguintes expressões:

$$CG_x = \frac{\sum_{i=1}^{n} x_i}{n} \quad \text{(IX)}$$

$$CG_y = \frac{\sum_{i=1}^{n} y_i}{n} \quad \text{(X)}$$

Após o centro da bola ser encontrado nas imagens vistas por cada uma das câmeras essas informações são enviada para o subsistema de estereoscopia, que, com base nelas, triangula a posição tridimensional que a bolinha estava no momento que as fotos foram tiradas.

Estereoscopia

Para cada uma das duas câmeras utilizadas no projeto existe uma matriz, chamada matriz de projeção, que relaciona as coordenadas de um ponto no espaço (definido em relação à um sistema de coordenadas fixado na câmera 1) com as coordenadas de sua projeção no plano da imagem da câmera, de acordo com a seguinte relação:

$$\begin{matrix} a_1 \\ b_1 \\ w_1 \end{matrix} = P_1 \begin{matrix} X \\ Y \\ Z \\ 1 \end{matrix} \quad \text{(XI)}$$

$$\begin{matrix} a_2 \\ b_2 \\ w_2 \end{matrix} = P_2 \begin{matrix} X \\ Y \\ Z \\ 1 \end{matrix} \quad \text{(XII)}$$

onde a_1, b_1, w_1 e a_2, b_2, w_2 são, respectivamente, as coordenadas homogêneas da projeção do ponto nos planos de imagem da câmera 1 e da câmera 2 em pixels ($\frac{a_i}{w_i}$ é a coordenada em x, $\frac{b_i}{w_i}$ é a coordenada em y); e X, Y e Z são as coordenadas tridimensionais da bola no sistema de coordenadas tridimensionais fixado na câmera 1. As convenções para os sistemas de coordenadas citados acima (sistema de coordenadas do plano da imagem e sistema de coordenadas tridimensionais fixado na câmera 1) estão ilustradas na figura (4). Para mais informações sobre matrizes de projeção, consulte (BRADSKI, G.; KAEHLER, A. Learning OpenCV: Computer Vision with the OpenCV Library. 1. Ed. [S.l.]: O'Reilly Media, Incorporated, 2008).

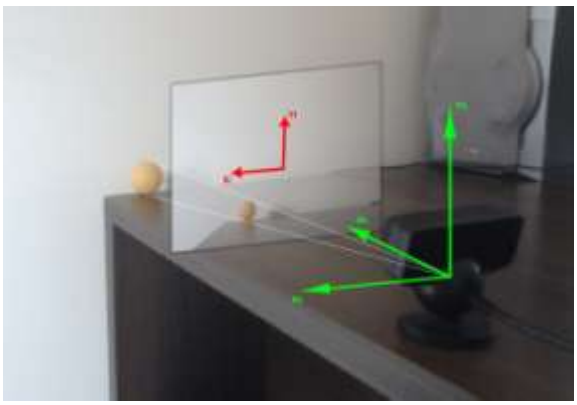

Figura 4 - A figura mostra a câmera e a imagem vista por ela. O sistema de coordenadas em verde é o global, aonde as coordenadas **X**, **Y** e **Z** são especificadas; já o sistema de

coordenadas em vermelho é o do plano de projeção da imagem, no qual os valores $\frac{\mathbf{a_i}}{\mathbf{w_i}}$ e $\frac{\mathbf{b_i}}{\mathbf{w_i}}$ são especificados. As matrizes $\mathbf{P_1}$ e $\mathbf{P_2}$ transformam pontos no sistema de coordenadas verde para o vermelho.

Solução do sistema

Procuramos realizar a reconstrução tridimensional da bola com base nas coordenadas de suas projeções no plano de imagem de cada câmera (estereoscopia). Nesse caso são conhecidos os valores $\frac{a_1}{w_1},\frac{b_1}{w_1}$ e $\frac{a_2}{w_2},\frac{b_2}{w_2}$ (as coordenadas da bola encontradas nas imagens da câmera 1 e da câmera 2) e as matrizes P_1 e P_2 (encontradas no processo de calibração explicado adiante) e queremos encontrar as coordenadas tridimensionais da bolinha.

Para resolver o sistema formado pelas equações M e N para X, Y e Z, procede-se da seguinte forma, de acordo com (HILLMAN, PETER; White Paper: Camera Calibration and Stereo Vision. Square Eyes Software, Edinburgo, 27 de julho de 2005):

Calcula-se as matrizes A e B

$$A = \begin{matrix} x_1.p_{31}^1 - p_{11}^1 & x_1.p_{32}^1 - p_{12}^1 & x_1.p_{33}^1 - p_{13}^1 \\ y_1.p_{31}^1 - p_{21}^1 & y_1.p_{32}^1 - p_{22}^1 & y_1.p_{33}^1 - p_{23}^1 \\ x_2.p_{31}^2 - p_{11}^2 & x_2.p_{32}^2 - p_{12}^2 & x_2.p_{33}^2 - p_{13}^2 \\ y_2.p_{31}^2 - p_{11}^2 & y_2.p_{32}^2 - p_{12}^2 & y_2.p_{33}^2 - p_{12}^2 \end{matrix} \quad \text{(XIII)}$$

$$B = \begin{matrix} p_{14}^1 - x_1.p_{34}^1 \\ p_{24}^1 - y_1.p_{34}^1 \\ p_{14}^2 - x_2.p_{34}^2 \\ p_{24}^2 - y_2.p_{34}^2 \end{matrix} \quad \text{(XIV)}$$

Com base nas matrizes XIII e XIV calcula-se o vetor W, composto pelas três coordenadas X, Y e Z.

$$W = A^T.A^{-1}.A^T.B \quad \text{(XV)}$$

Calibração das câmeras

Para encontrar os valores p_{km}^i citados anteriormente é necessária uma etapa de calibração. A abordagem mais comum é mostrar um objeto para as câmeras parecido com um tabuleiro de xadrez. Essa geometria é favorável pois é simples reconhecer os vértices dos quadrados com um software de reconhecimento de imagem e ao mesmo tempo também é simples especificar as coordenadas de cada uma das arestas em relação à um sistema de coordenadas fixo no tabuleiro. A figura (5) ilustra os dois sistemas de coordenadas. Levando em conta a complexidade aparente do problema, a robustez necessária que uma possível solução deveria apresentar e o tempo disponível de projeto, o grupo decidiu implementar uma solução pronta, ao invés de desenvolver uma solução original.

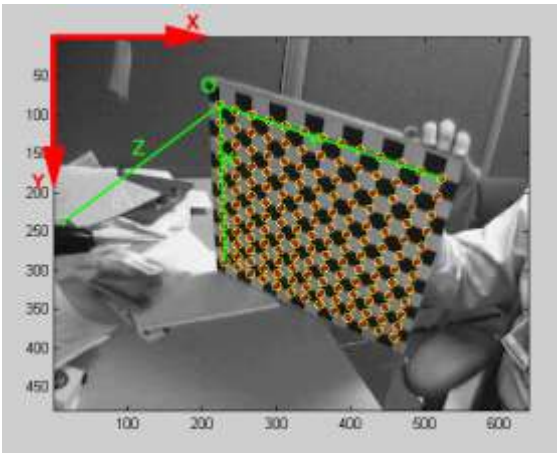

Figura 5 - Sistemas de coordenadas em um tabuleiro de xadrez: sistema verde é o fixado no objeto e o vermelho é o plano de projeção da imagem

Na implementação em Matlab (durante a prototipagem dos algoritmos) foi utilizada a toolbox open-source de calibração de câmeras desenvolvida no Instituto de Tecnologia da California, "*Camera Calibration Toolbox for Matlab"*. Essa toolbox consiste em um conjunto de códigos escritos para a plataforma Matlab com a finalidade de calibrar câmeras; para utilizá-la basta seguir o guia apresentado pelos criadores em seu website.

Após a prototipagem do sistema em Matlab os algoritmos foram traduzidos para a linguagem C++. Nessa etapa, ao invés de traduzir todos os códigos da toolbox mencionada acima, o grupo optou por utilizar um série de funções da biblioteca OpenCV para encontrar os vértices de um padrão de calibração mostrado para as câmeras e para, com base nisso, estimar os valores p_{km}^{i}.

Filtro de Kalman

Para melhorar as medidas obtidas de posição da bola de tênis de mesa e para estimar seu vetor velocidade foi utilizado um filtro de Kalman. O filtro de Kalman é um filtro recursivo que estima todos os estados (incluindo os não-observáveis) de um sistema dinâmico linear com base em uma série de medidas com ruído gaussiano (BRADSKI, G. KAEHLER, A.).

O funcionamento do filtro pode ser resumido às seguintes equações, divididas em duas etapas:

a etapa de predição:

$$x_{k|k-1} = F_k . x_{k-1|k-1} + B_k . u_k \quad \text{(XVI)}$$

$$P_{k|k-1} = F_k . P_{k-1|k-1} . F_k^T + Q_k \quad \text{(XVII)}$$

e a etapa de atualização:

$$y_k = z_k - H_k . x_{k|k-1} \quad \text{(XVIII)}$$

$$S_k = H_k . P_{k|k-1} . H_k^T + R_k \quad \text{(XIX)}$$

$$K_k = P_{k|k-1} . H_k^T . S_k^{-1} \quad \text{(XX)}$$

$$x_{k|k} = x_{k|k-1} + K_k . y_k \quad \text{(XXI)}$$

$$P_{k|k} = \left(I - K_k . H_k \right) . P_{k|k-1} \quad \text{(XXII)}$$

onde:
x : estimativa dos estados ; F_k : matriz de transição dos estados; B_k : modelo dinâmico linear das entradas de controle ; u_k : entradas de controle ; P : matriz de covariancia dos estados (uma medida da precisão da estimativa dos estados); Q_k : matriz de covariância do ruído do processo; y_k : resíduo das medições (diferença entre o estimado pelo modelo matemático e o que foi medido); z_k : vetor das medidas; H_k : modelo dinâmico linear de observação, que mapeia o espaço de estados real no espaço de estados observado; R_k : matriz da covariância do ruído de observação; S_k : resíduo da covariância; K_k : ganho de Kalman (ganho que maximiza a redução da matriz de covariância dos estados ao longo do tempo)

No caso do movimento da bola de tênis de mesa, o modelo dinâmico linear adotado foi:

$$\begin{matrix} x_{k+1} \\ y_{k+1} \\ z_{k+1} \\ V_{x_{k+1}} \\ V_{y_{k+1}} \\ V_{z_{k+1}} \end{matrix} = \underbrace{\begin{matrix} 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{matrix}}_{F} \cdot \begin{matrix} x_k \\ y_k \\ z_k \\ V_{x_k} \\ V_{y_k} \\ V_{z_k} \end{matrix} \quad \text{(XXIII)}$$

Portanto:

$$B_k = 0 \quad \text{(XXIV)}$$

E os estados medidos são as coordenadas tridimensionais da bola, disso segue que:

$$H_k = \begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{matrix} \quad \text{(XXV)}$$

As matrizes Q_k e R_k foram estimadas em:

$$Q_k = \sigma_1 . \begin{matrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{matrix} \quad \text{(XXVI)}$$

$$R_k = \sigma_2 . \begin{matrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{matrix} \quad \text{(XXVII)}$$

onde σ_1 e σ_2 foram constantes determinadas experimentalmente. Os valores que obtiveram resultados satisfatórios com o sistema de visão estéreo foram $\sigma_1 = 0.2$ e $\sigma_2 = 10$. A figura (6) ilustra o comportamento do filtro enquanto descreve-se uma elipse com a bola na frente das câmeras (início na parte inferior da elipse ilustrada, sentido de rotação anti-horário).

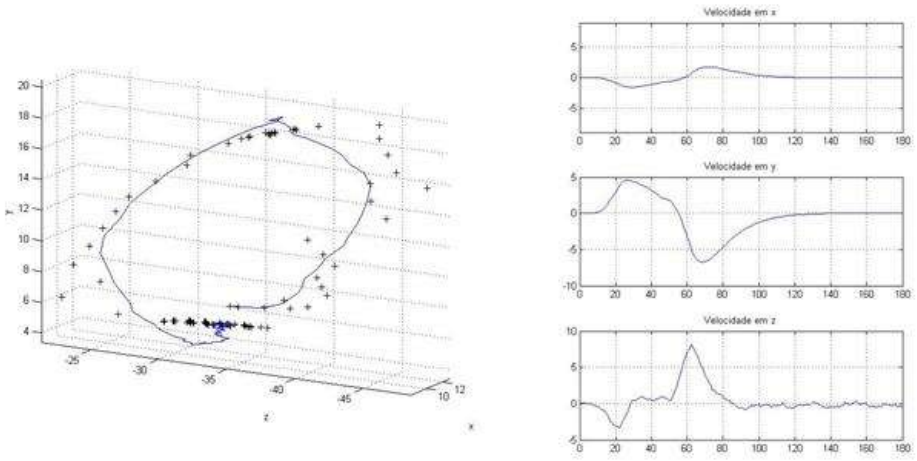

Figura 6 - Pontos assinalados com cruzes são as medidas do sistema estéreo, pontos unidos pela linha contínua são as estimativas do filtro de Kalman. À direita: estimativas de velocidade em cada uma das direções conforme a bola foi descrevendo a elipse na frente das câmeras.

Resultados e Discussão

A fim de concretizar os resultados deste trabalho, foram realizados testes com a visão computacional desenvolvida. Primeiramente, mapeou-se o plano da mesa com uma malha divida em quadrados de lado 10 centímetros criando pontos de referência para os testes. A seguir, calibraram-se as câmeras com o tabuleiro de xadrez conforme os procedimentos citados anteriormente.

Com o objetivo de homogeneizar ao máximo as condições de iluminação, foram colocadas seis lâmpadas brancas sobre a mesa de tênis de mesa. Ligou-se a iluminação e calibrou-se a visão do filtro por *treshold*.

Colocou-se a bola laranja em cada ponto traçado na malha sobre o plano da mesa com o objetivo de verificar no espaço onde a visão computacional reconhece o objeto laranja. Os testes foram realizados em três alturas em Y diferentes: 5, 20 e 40 centímetros. A seguir foram traçados gráficos com a nuvem de pontos de calibração entre as distâncias em X, Y e Z e a posição das câmeras. A figura (7) ilustra as nuvens de calibração.

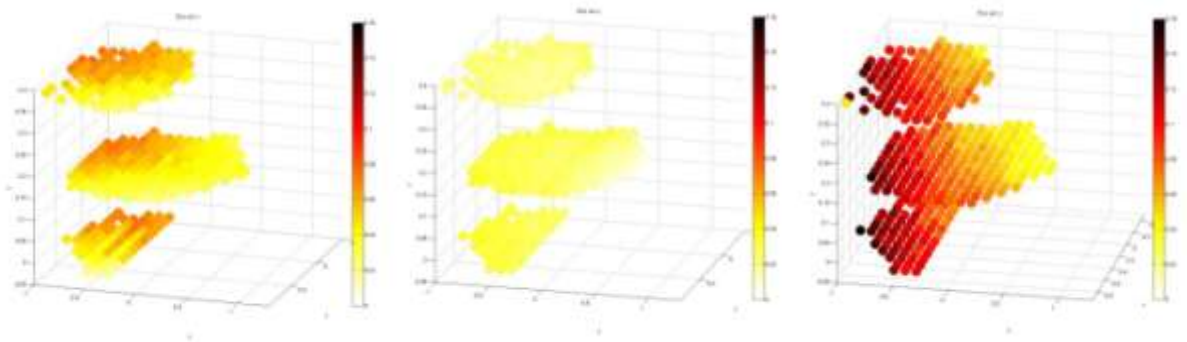

Figura 7 - Erros de posicionamento da visão computacional em respectivamente (a) eixo X (b) eixo Y (c) eixo (Z)

Conforme explicitado na figura (7), o maior erro de medida deu-se no eixo Z com valor de 14 centímetros.

Conclusões

Após o desenvolvimento e testes conclui-se que os objetivos iniciais foram alcançados. A visão computacional desenvolvida neste projeto pode reconhecer uma bola de tênis de mesa em um espaço considerável sobre o plano da mesa. Quanto ao erro de medida do sistema, os resultados foram satisfatórios. Verificou-se que quanto mais distante o objeto está das câmeras maior é o erro, conforme esperado.

Referências Bibliográficas

BOUGET, J.; Camera Calibration Toolbox for Matlab, 10 de Outubro de 2013, http://www.vision.caltech.edu/bouguetj/calib_doc/, acessado em Agosto de 2012.

BRADSKI, G.; KAEHLER, A. Learning OpenCV: Computer Vision with the OpenCV Library. 1. Ed. [S.l.]: O'Reilly Media, Incorporated, 2008

GONZALEZ, R.; WOODS, R. Digital Image Processing. 3. Ed. [S.l.]: Prentice Hall, 2007.

WITKOWSKI, Francisco Mauro. Física I. São Paulo, SP: EP, 1995. pt. 3.