4
Implementação e Resultados

A dissertação teve dois produtos como resultado da sua implementação. O primeiro foi um programa para calibração das câmeras e o segundo foi um *plugin* para o dispositivo óptico no ViRAL. Além desses dois produtos, foi desenvolvido um *plugin* de cena do ViRAL, a fim de se utilizar o dispositivo óptico para controle de objetos em um grafo de cena.

As ferramentas utilizadas para o desenvolvimento foram o ViRAL, o OpenCV [13], o OpenSceneGraph e a biblioteca Qt da Trolltech [28].

4.1.
Implementação

A Figura 27 ilustra a janela principal do programa de calibração. Nessa interface é possível selecionar qual câmera será calibrada. No caso do nosso programa utilizamos a imagem obtida através de um equipamento que combina o sinal de quatro câmeras, sincroniza-os e envia um único sinal para a placa de captura do computador. Dessa forma, é possível recuperar as imagens das quatro câmeras de uma só vez.

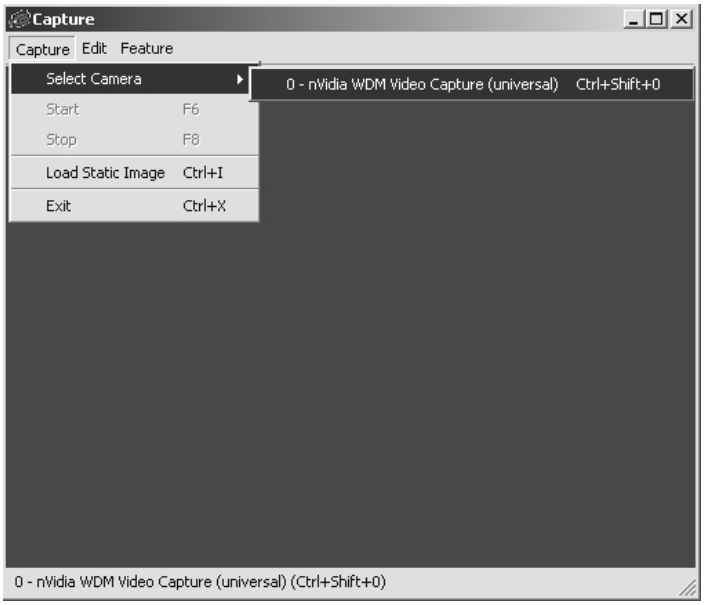


Figura 27: Janela principal do programa de calibração.

Depois de escolhida a câmera e os filtros necessários aplicados, um padrão (Figura 28i) é utilizado para calibração. A (Figura 28ii) ilustra a interface de aplicação de filtros na imagem.

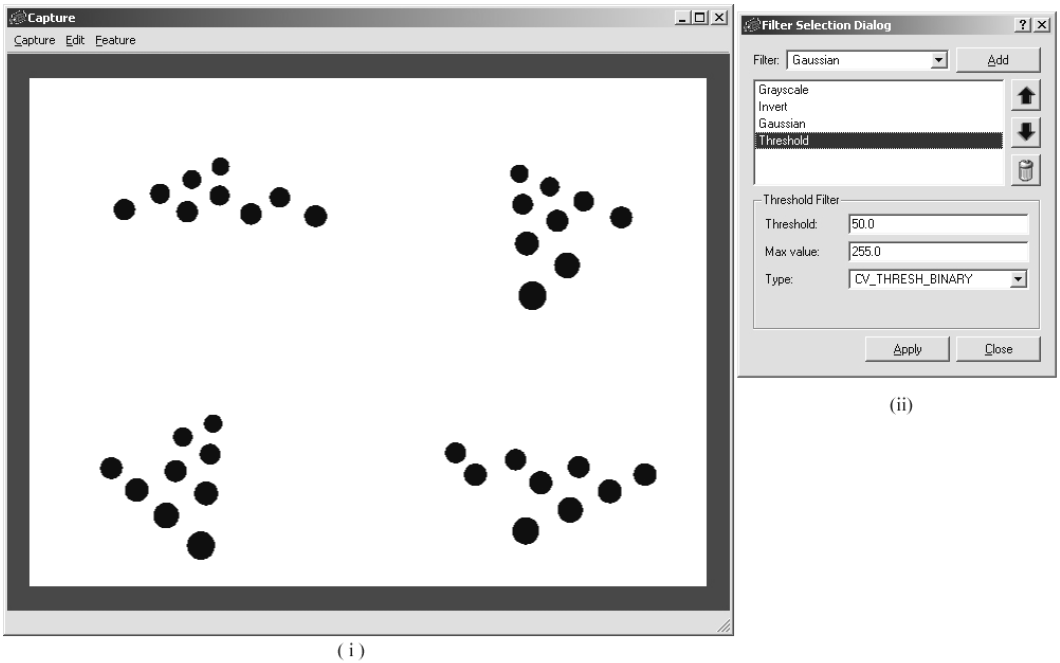


Figura 28: (i) Padrão de calibração; (ii) interface de aplicação de filtros.

O sistema não identifica automaticamente as esferas, ou seja, ele não associa a esfera da imagem à sua correspondente no mundo real. Para isso, o usuário deve manualmente identificá-las, como visto na Figura 29.

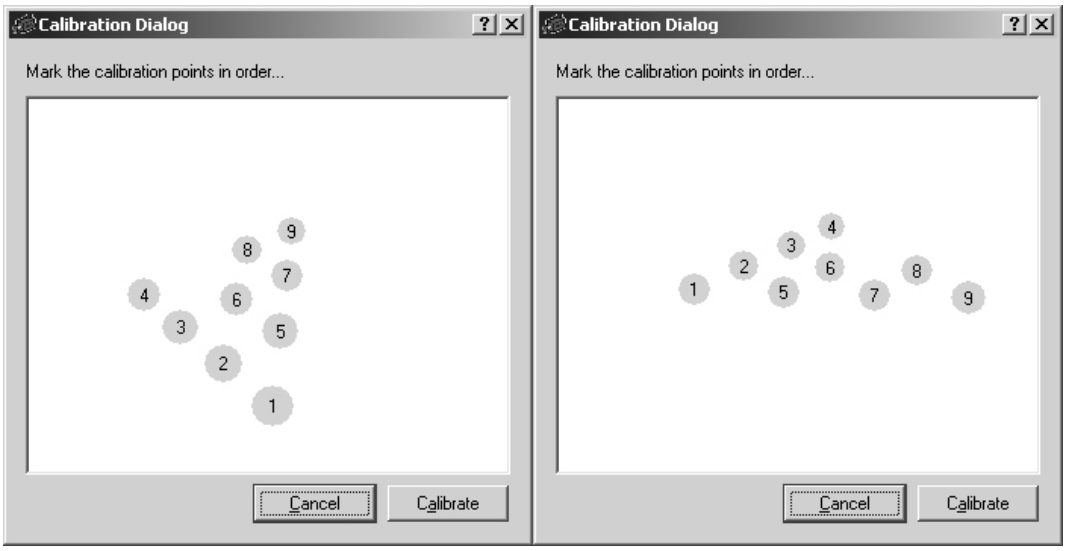
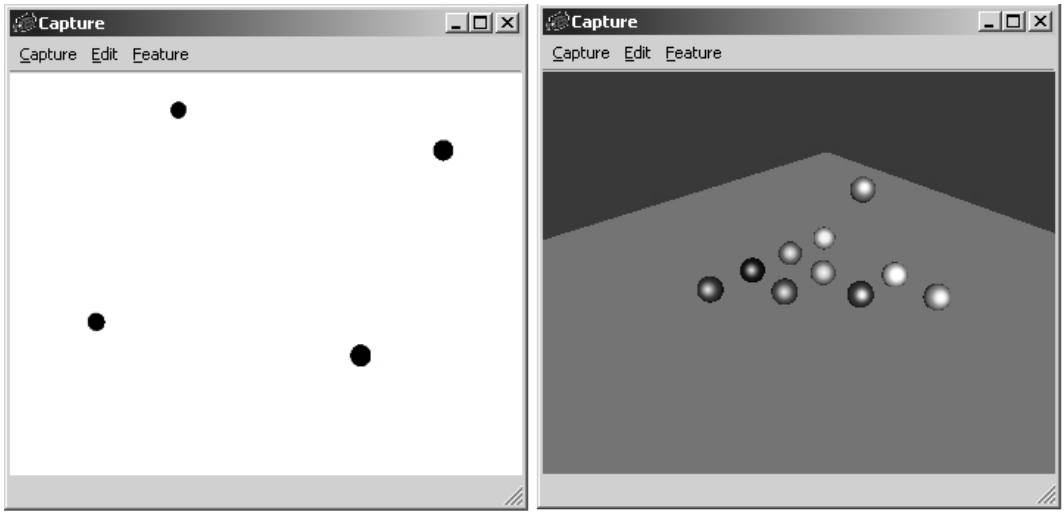


Figura 29: Interface de calibração: identificação das esferas vistas por duas câmeras distintas.

Depois das câmeras estarem calibradas, o padrão de calibração é removido do alcance de visão das câmeras e a esfera retroreflexiva começa a ser rastreada.

A Figura 30i ilustra a imagem dessa esfera e a Figura 30ii apresenta uma esfera virtual posicionada em um modelo tridimensional utilizando a posição obtida através do algoritmo proposto. A esfera em questão está destacada das demais.



(i) (ii)

Figura 30: (i) Imagem capturada da esfera sendo rastreada; (ii) Esfera rastreada inserida em um modelo virtual.

Para fins de testes iniciais, um modelo foi criado em um software de modelagem e, ao invés de capturar a imagem através de quatro câmeras reais, foi utilizada uma imagem combinada de quatro câmeras virtuais gerada pelo próprio programa de modelagem. Com isso, foi possível medir a precisão do algoritmo proposto.

No mundo virtual, a esfera foi posicionada nas coordenadas (12, 8, 6) cm. Através do algoritmo proposto, as posições obtidas utilizando diversas combinações de câmera foram:

Câmeras	Coordenadas (cm)	Erro (distância euclidiana)
1, 2, 3 e 4	(12,010698; 7,939839; 5,974222)	0,06632 cm
1, 2 e 3	(11,994664; 7,933684; 5,982123)	0,06889 cm
1, 2 e 4	(12,009713; 7,951878; 5,984271)	0,051551 cm
1 e 2	(11,985385; 7,967478; 6,007030)	0,036341 cm
2 e 4	(12,014355; 7,925166; 5,998602)	0,076211 cm
3 e 4	(12,036443; 7,949826; 5,953181)	0,077702 cm

Tabela 8: Erros obtidos utilizando câmeras virtuais.

O algoritmo de calibração foi utilizado, posteriormente, no mundo real e a altura e a distância focal das câmeras estão listadas na Tabela 9.

	Altura (cm)	Distância Focal (cm)
Câmera 1	177,6157	251,237
Câmera 2	182,062	235,780
Câmera 3	161,914	228,041
Câmera 4	229,727	309,485

Tabela 9: Altura e distância focal obtidas pela calibração das câmeras.

As câmeras estão localizadas a aproximadamente 260cm do solo. A diferença entre a altura real e a altura medida é compensada, pelo método de Tsai, na distância focal da câmera.

4.1.1. Integração com o ViRAL – *Plugin* de Dispositivo

O dispositivo óptico foi integrado ao ViRAL através de um *plugin* de dispositivo. A Figura 31 mostra a sua interface de configuração.

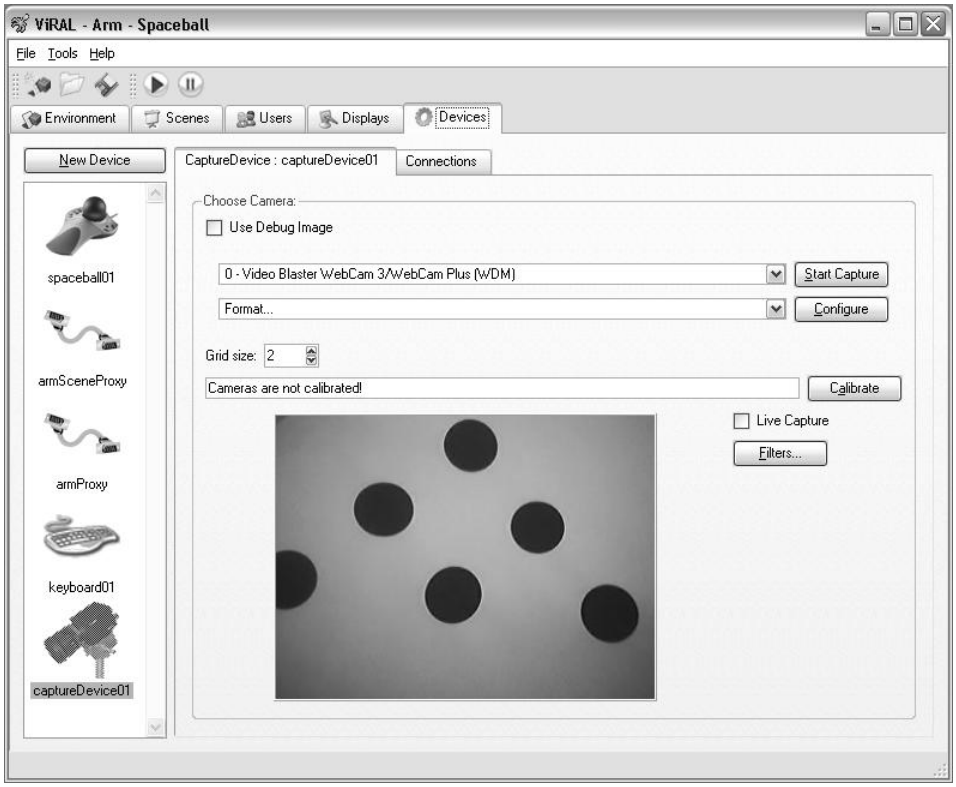


Figura 31: Interface de configuração do *plugin* do dispositivo óptico.

Esse dispositivo exporta dois eventos (*signals*) para o ViRAL: *translate* e *positionChanged* (Tabela 10). O primeiro envia a diferença de movimentação entre dois quadros e o segundo envia a posição absoluta do dispositivo.

```
class CaptureDevice : public vral::Device
{
signals:
  void translate( const Vector3 &translation );
  void positionChanged( const Vector3 &position );

public:
  virtual void dispatchChanges()
  {
    static Vector3 lastTrans;
    Tracker tracker;
    if( tracker.track( x, y, z ) )
    {
       vral::Vector3 translation( x, y, z );
       Vector3 diff = ( translation - lastTrans );
       lastTrans = translation;

       emit translate( diff );
       emit positionChanged( translation );
    }
  }
};
```

Tabela 10: Classe *CaptureDevice*.

A palavra chave *emit* é uma macro do Qt para informar que um sinal (*signal*) está sendo enviado. Os eventos desse dispositivo podem ser conectados aos eventos do *plugin* de cena descrito a seguir, através da interface de conexão de eventos do ViRAL.

O comando *track* da classe *Tracker* é responsável pela execução do algoritmo proposto neste trabalho. A Tabela 11 apresenta o seu pseudo-código:

```
Tracker::track()
{
  Para cada câmera faça
    Determinar a posição P da câmera;
    Encontrar a esfera na imagem da câmera;
    Transformar as coordenadas na imagem em coordenadas C
           do mundo baseado nos dados de calibração;
    Armazenar o segmento de reta PC;
  Para cada par de segmentos de retas
    Armazenar o ponto mais próximo entres eles;
  Calcular e retornar a médias M dos pontos mais próximos;
}
```

Tabela 11: Pseudo-código do método *track* da classe *Tracker*.

A criação de um dispositivo no ViRAL foi bastante simples. A primeira causa disso foi o Qt, pois o mesmo fornece um ambiente integrado de

desenvolvimento que facilita a criação de interface gráfica. Outro ganho do Qt foi quanto à criação de eventos, bastando apenas que fosse definida a palavra-chave *signal* no meio do código para que ele identificasse os eventos exportados.

Além disso, o sistema de *plugins* do ViRAL permitiu que a programação do dispositivo fosse completamente desacoplada do seu núcleo.

4.1.2. *Plugin* de Cena

O *plugin* de cena desenvolvido para o ViRAL é um exemplo simples onde um dispositivo (um *mouse* 3D) controla a posição e a orientação do usuário e o dispositivo óptico movimenta um objeto na cena.

O *plugin* de cena exporta para o ViRAL uma classe de cena *TrackScene* e um objeto de cena *TrackableObject*. A Tabela 12 e a Tabela 13 ilustram os principais métodos implementados nessas classes. Com a utilização do grafo de cena *OpenSceneGraph* foi possível alterar a posição da esfera modificando um objeto *Transformação*.

```
class TrackScene : public Scene
{
public:
  void initializeContext()
  {
    sceneView->setSceneData( mRootNode );
  }
  void preDrawImplementation( const ScenePreDrawingData &data )
  {
    sceneView->update();
  }
  void drawImplementation( const SceneDrawingData &data )
  {
      sceneView->.setProjMatrix( data.getProjection().get() );
      sceneView->setViewMatrix( data.getModelview().get() );

      sceneView->cull();
      sceneView->draw();
  }
private:
  osgUtil::SceneView *mSceneView;
  osg::Node *mRootNode;
}
```

Tabela 12: Classe *TrackScene* de um *plugin* de cena.

```
class TrackableObject : public vral::SceneObject
{
  public slots:
    void translate( const Vector3 &translation )
    {
      osg::Matrix matrix = mTransform->getMatrix();
      matrix = matrix * osg::Matrix::translate(
                                  translation.getX(),
                                 -translation.getZ(),
                                  translation.getY() );
      mTransform->setMatrix( matrix );
    }

    void setPosition( const Vector3 &translation )
    {
      osg::Matrix matrix = mTransform->getMatrix();
      matrix.setTrans( translation.getX,
                      -translation.getZ(),
                       translation.getY() );
      mTransform->setMatrix( matrix );
    }

  private:
    osg::MatrixTransform *mTransform;
};
```

Tabela 13: Classes *TrackableObject* de um *plugin* de cena.

A classe *TrackScene* contém o objeto *SceneView* do *OpenSceneGraph*. Esse objeto é responsável por configurar as matrizes de projeção e chamar os métodos *update/cull/draw*. O objeto *mRootNode* é o nó raiz da cena.

A classe *TrackableObject* contém uma objeto transformação do *OpenSceneGraph*, *mTransform*, que posiciona o objeto sendo rastreado corretamente na cena. A palavra-chave *slots* é um macro do Qt que identifica os eventos de entrada desse objeto. O evento *translate*, como o nome já diz, translada o objeto e o método *setPosition* atribui a ele uma posição absoluta.

A Figura 32, finalmente, ilustra 3 etapas da movimentação de uma esfera com o dispositivo óptico.

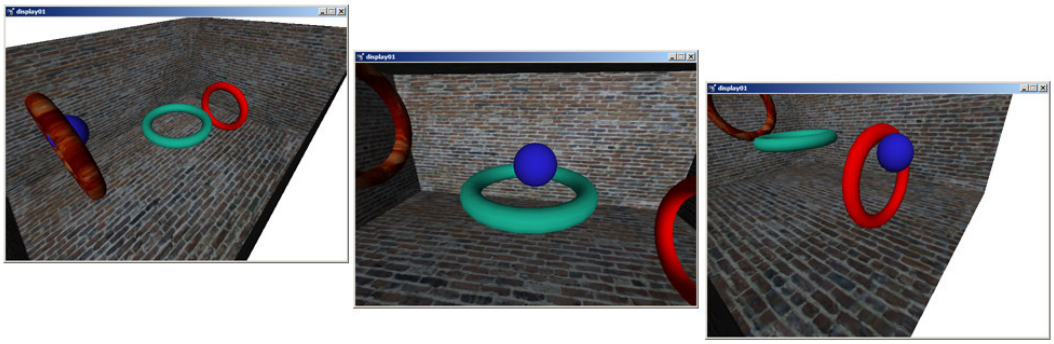


Figura 32: Execução do *plugin* de cena no ViRAL.

4.2. Resultados

A configuração do local de testes está ilustrado na Figura 33. As quatro câmeras foram calibradas e dez testes foram efetuados.

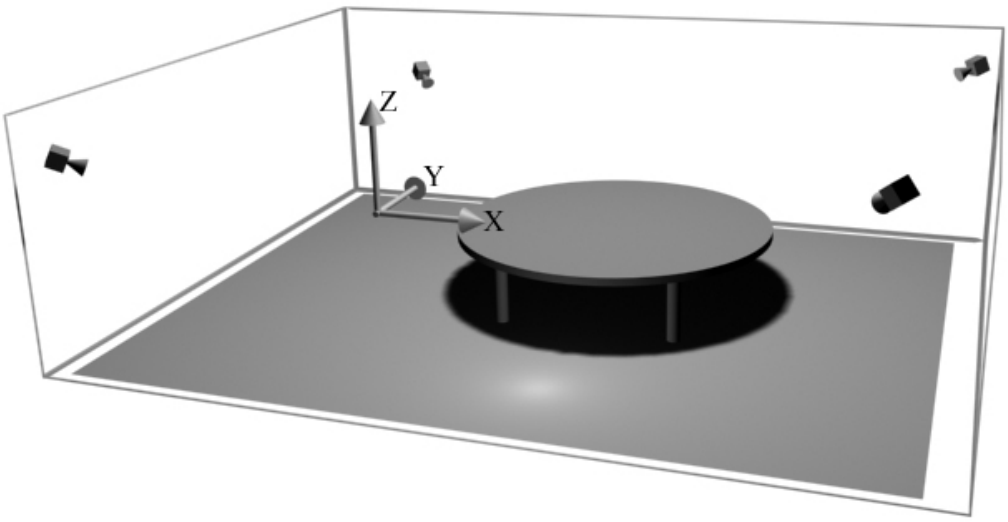


Figura 33: Modelo esquemático do local de testes.

O primeiro teste foi feito com a esfera parada a uma distância de aproximadamente 12cm sobre a mesa, utilizando diversas combinações de câmeras e sem utilização de filtro de Kalman. A calibração prévia das câmeras foi feita com 10 pontos. A posição da esfera foi medida e pode ser vista no gráfico da Figura 34.

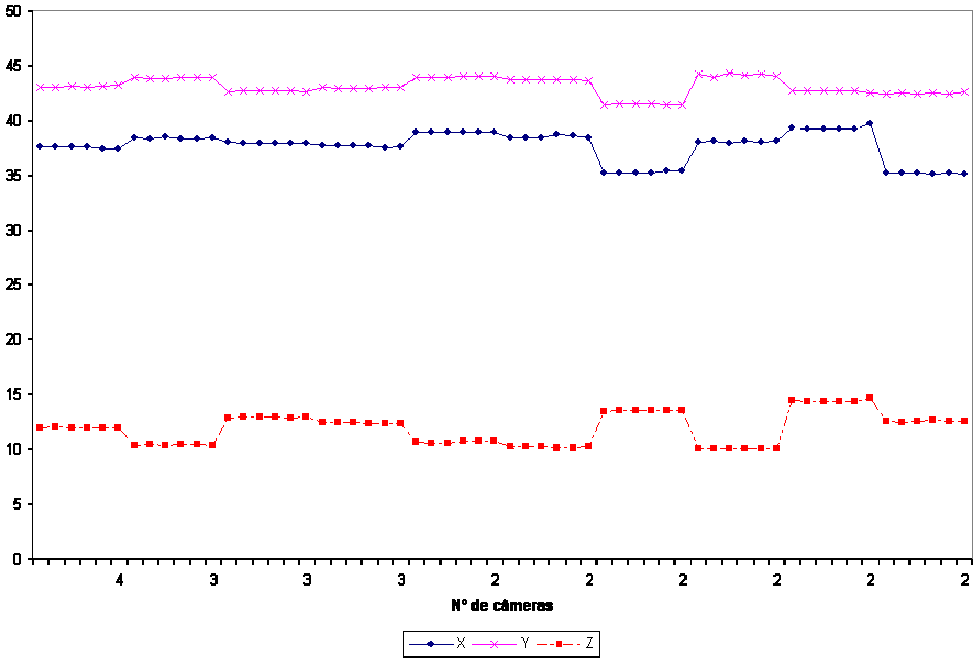


Figura 34: Teste 1 - Esfera em Z = *12* e câmeras calibradas com 10 pontos.

Podemos notar pelo gráfico que quando a configuração de câmeras é mantida, o movimento fica bastante estável. Porém, quando a configuração se modifica, ou seja, uma câmera deixa de localizar o objeto, existe um salto na sua posição. Isso se deve ao fato do fraco resultado da calibração obtida pelo método Tsai.

Uma homografia [26] foi utilizada para se obter mais pontos para calibração das câmeras. No segundo teste, foram usados 121 pontos para calibração, ao invés dos 10 pontos anteriores, e o resultado obtido foi bem melhor (Figura 35).

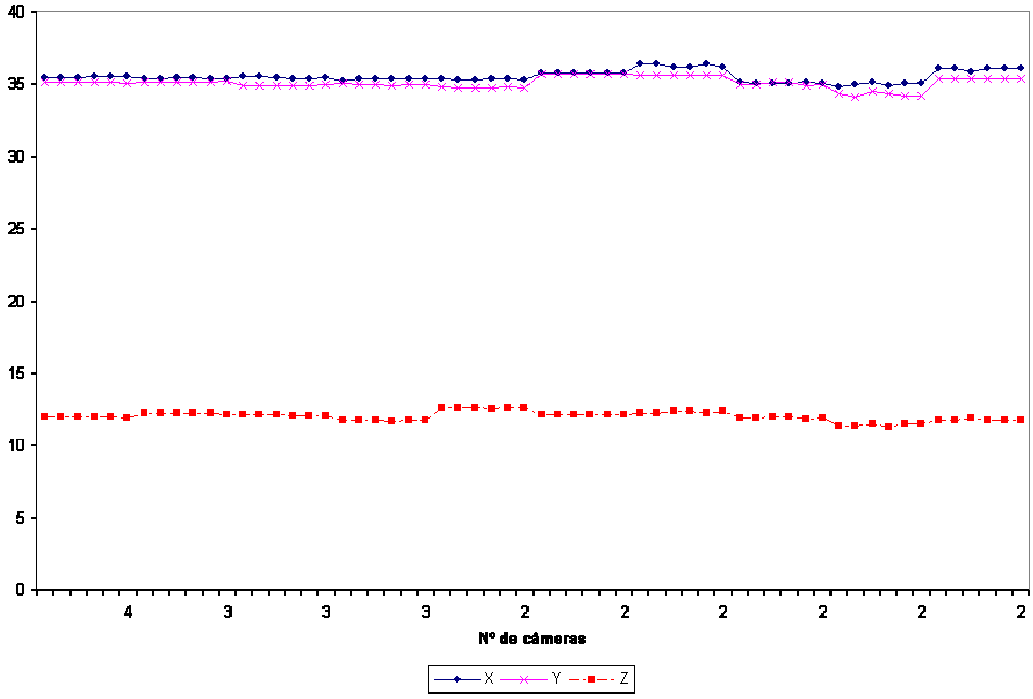


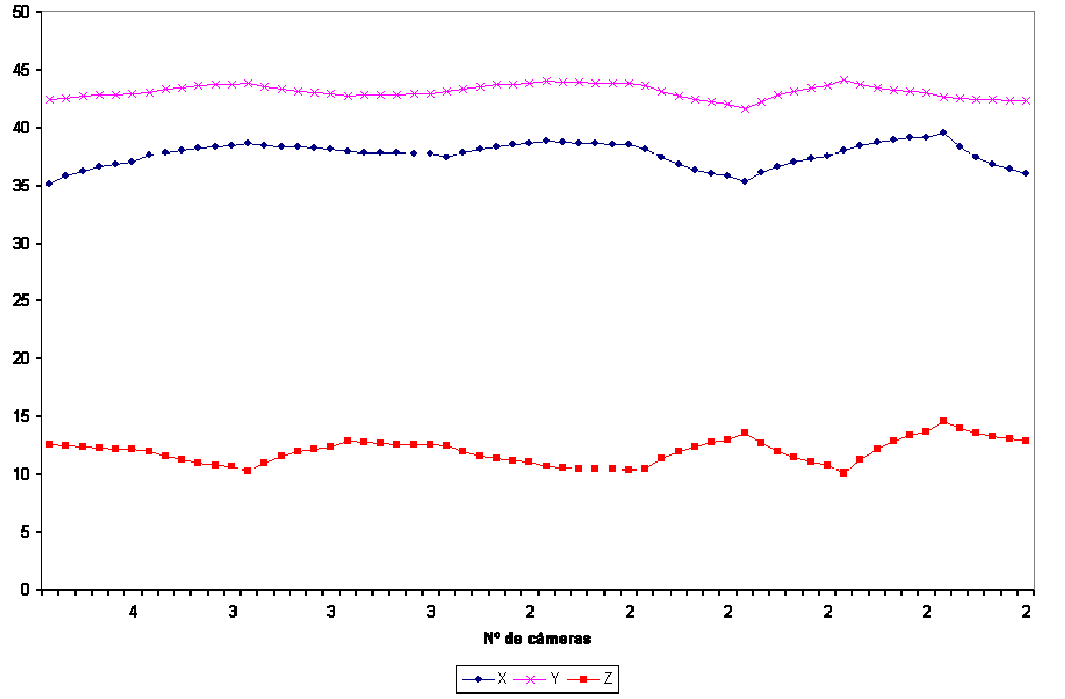
Figura 35: Teste 2 - Esfera em *Z = 12* e câmeras calibradas com 121 pontos.

As mesmas configurações foram utilizadas para o terceiro e quarto testes, porém o filtro de Kalman foi habilitado. Os valores utilizados nas matrizes de covariância *Q*, do processo, e *R* da medida foram:

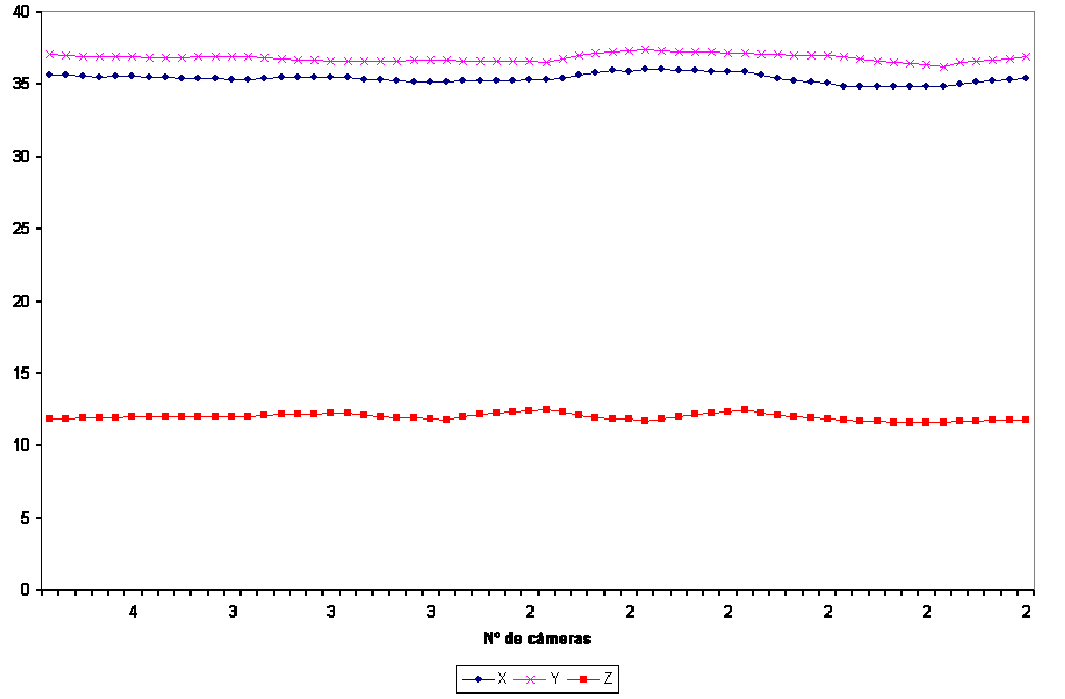
$$Q = \begin{bmatrix} 0{,}01 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0{,}01 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0{,}01 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0{,}1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0{,}1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0{,}1 \end{bmatrix} \text{ e } R = \begin{bmatrix} 0{,}1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0{,}1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0{,}1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0{,}1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0{,}1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0{,}1 \end{bmatrix},$$

onde os primeiros três elementos da diagonal estão relacionados à posição e os três últimos à velocidade. Com esses valores, o filtro tende a suavizar o

movimento quando a posição medida se desvia da sua rota normal, porém mantendo a mesma velocidade da medição. Os resultados destes testes podem ser vistos na Figura 36.



a) Teste 3 - Câmeras calibradas com 10 pontos.



b) Teste 4 - Câmeras calibradas com 121 pontos.

Figura 36: Esfera em *Z = 12* e filtro de Kalman habilitado.

Apesar de ainda existirem saltos quando se modifica a configuração de câmeras, a transição entre uma posição e outra é mais suave, tornando essa diferença mais aceitável. Com 121 pontos de calibração, as variações são praticamente imperceptíveis.

A Tabela 14 apresenta a posição média e o desvio padrão das coordenadas para os testes 1, 2, 3 e 4.

		X	Y	Z
Teste 1	Média	37,64939	43,07366	11,89308
	σ	1,342049	0,791235	1,453631
Teste 2	Média	35,49744	35,06238	11,97544
	σ	0,399178	0,401567	0,338104
Teste 3	Média	37,63999	43,07322	11,91327
	σ	1,022192	0,58011	1,047162
Teste 4	Média	35,35106	36,78472	11,92138
	σ	0,319098	0,259508	0,223056

Tabela 14: Posição média e desvio padrão dos testes 1, 2, 3, 4.

Pelo resultado apresentado, podemos observar uma diferença de cerca de 85% no desvio padrão quando comparamos a cota *Z* do teste 4 em relação ao teste 1. Isso mostra uma fraqueza do método Tsai quando poucos pontos são utilizados.

Os quinto e sexto testes foram feitos repetindo-se os procedimentos dos testes 1 e 2, porém a distância da esfera ao plano *Z* da mesa foi aumentada para 55cm. O resultado do quinto teste está na Figura 37 e o do sexto na Figura 38.

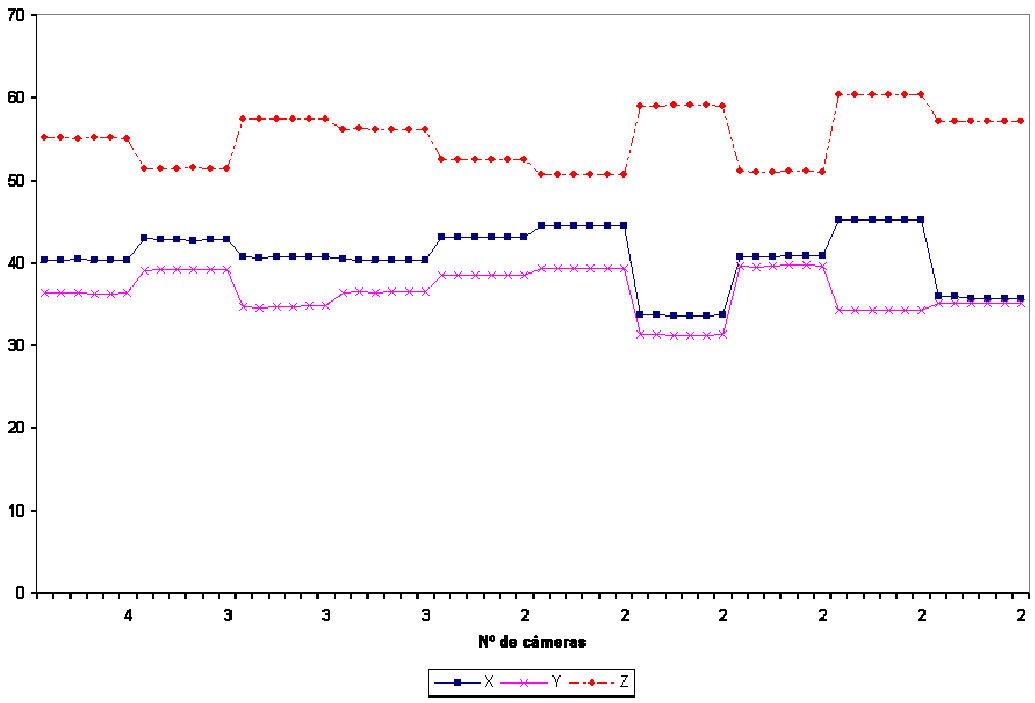


Figura 37: Teste 5 - Esfera em Z = *55* e câmeras calibradas com 10 pontos.

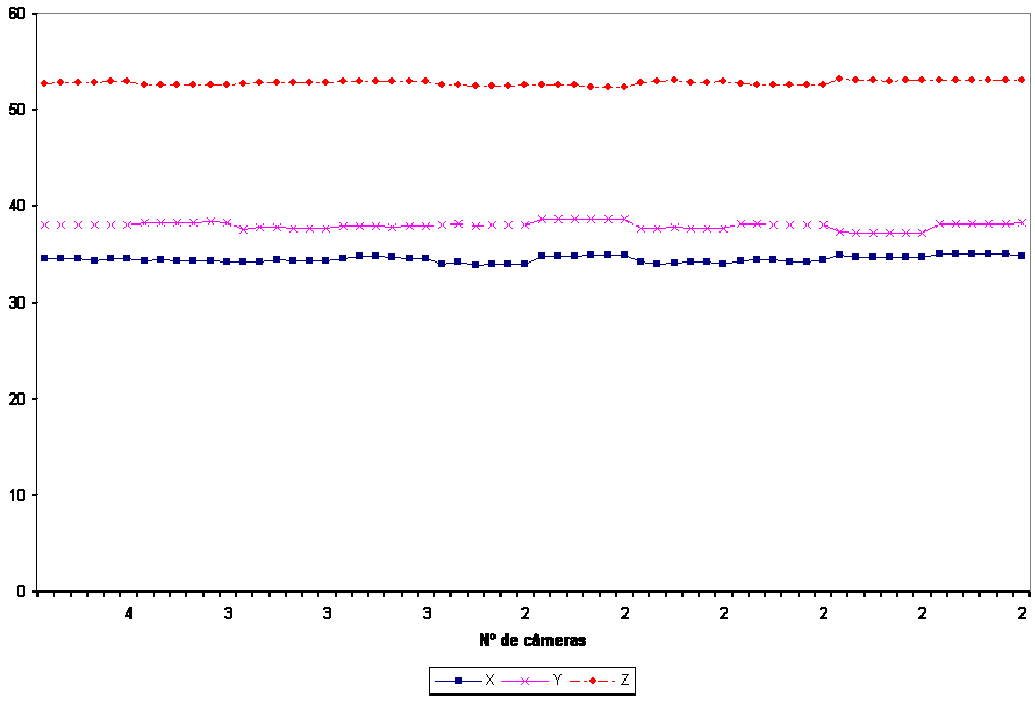
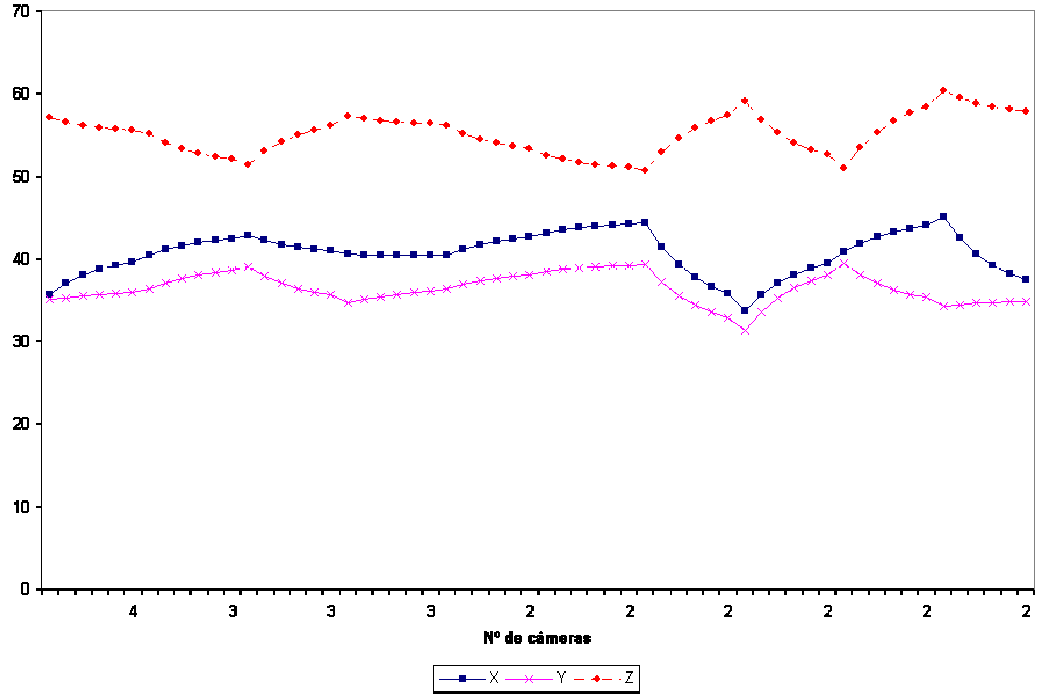


Figura 38: Teste 6 - Esfera em Z = *55* e câmeras calibradas com 121 pontos.

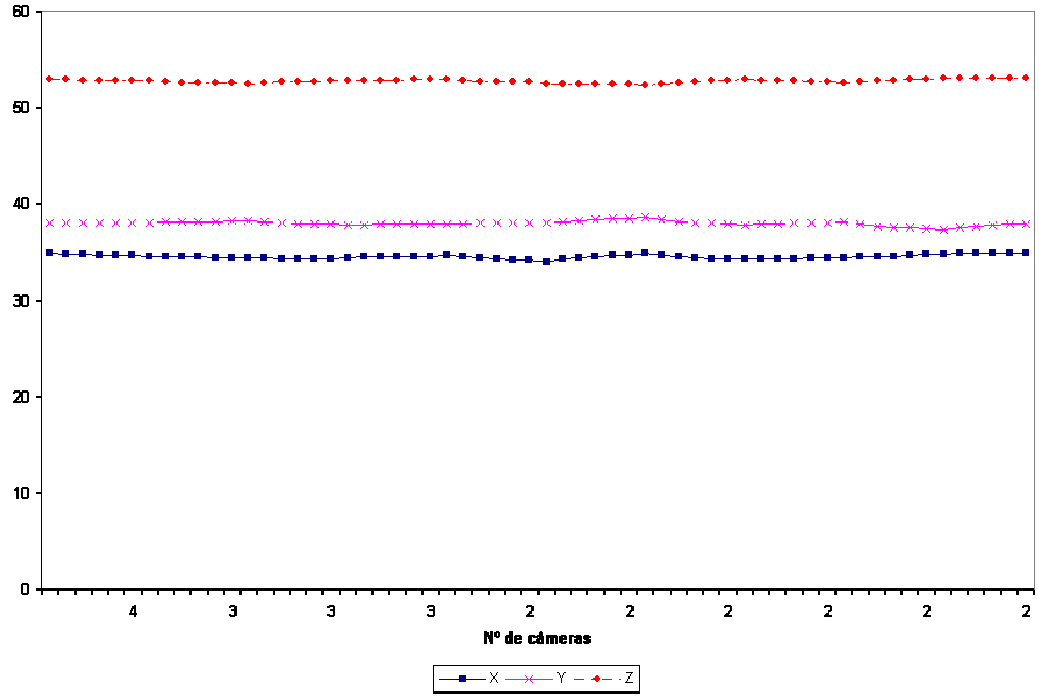
O comportamento foi bastante similar ao resultado anterior, porém, pelo teste 5, notamos que, quanto mais a esfera se afasta do plano de calibração, maior o erro quando se reduz o número de câmeras rastreando-a. Isso se deve ao fato do algoritmo de calibração ser um algoritmo coplanar. O erro pode ser reduzido

aumentando o número de pontos de calibração (verificado no resultado do teste 6) e através da utilização de um algoritmo tridimensional de calibração.

Os testes sete e oito foram realizados com a esfera à 55cm da mesa, porém com filtro de Kalman habilitado. Os resultados são vistos na Figura 39.



b) Teste 7 – Câmeras calibradas com 10 pontos.



b) Teste 8 – Câmeras calibradas com 121 pontos.

Figura 39: Esfera em Z = *55* e filtro de Kalman habilitado.

Da mesma forma, o filtro de Kalman suavizou o movimento quando a configuração de câmeras foi modificada. A Tabela 15 apresenta a posição média e o desvio padrão das coordenadas para os testes 5, 6, 7 e 8.

		X	Y	Z
Teste 5	Média	40,66364	36,43447	55,04712
	σ	3,487908	2,587444	3,358542
Teste 6	Média	34,4313	37,95047	52,74441
	σ	0,330229	0,377904	0,215029
Teste 7	Média	40,66211	36,4109	55,03846
	σ	2,534221	1,779225	2,411372
Teste 8	Média	34,49527	37,96208	52,71731
	σ	0,216536	0,241335	0,181227

Tabela 15: Posição média e desvio padrão dos testes 5, 6, 7, 8.

Notadamente, em todos os testes, o desvio padrão foi muito menor quando as câmeras foram calibradas com 121 pontos. A melhoria do desvio padrão da cota *Z* do teste 8 em relação ao teste 5 foi de 94,6%.

Os dois últimos testes foram feitos utilizando as quatro câmeras e movimentando a esfera na direção -*X* e depois *Y*, como ilustra a Figura 40.

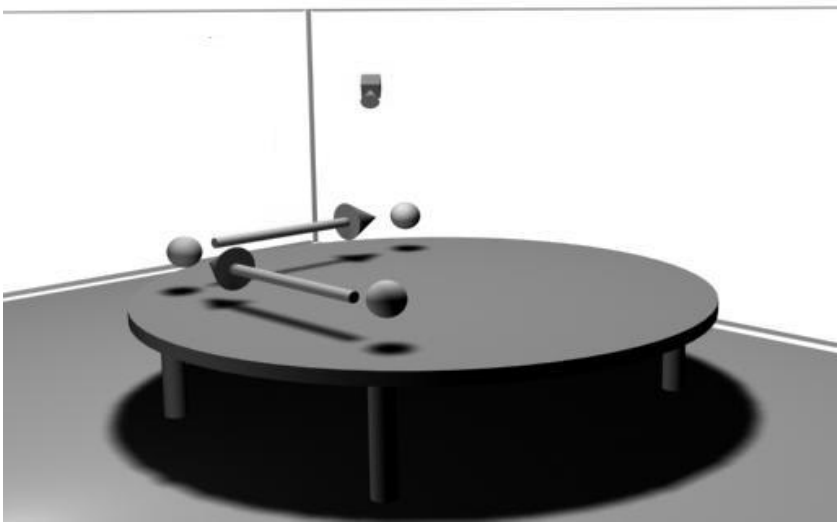


Figura 40: Esquemático do movimento da esfera.

O resultado do penúltimo deles, sem filtro de Kalman, pode ser visto na Figura 41 e o último, com filtro de Kalman habilitado, na Figura 42.

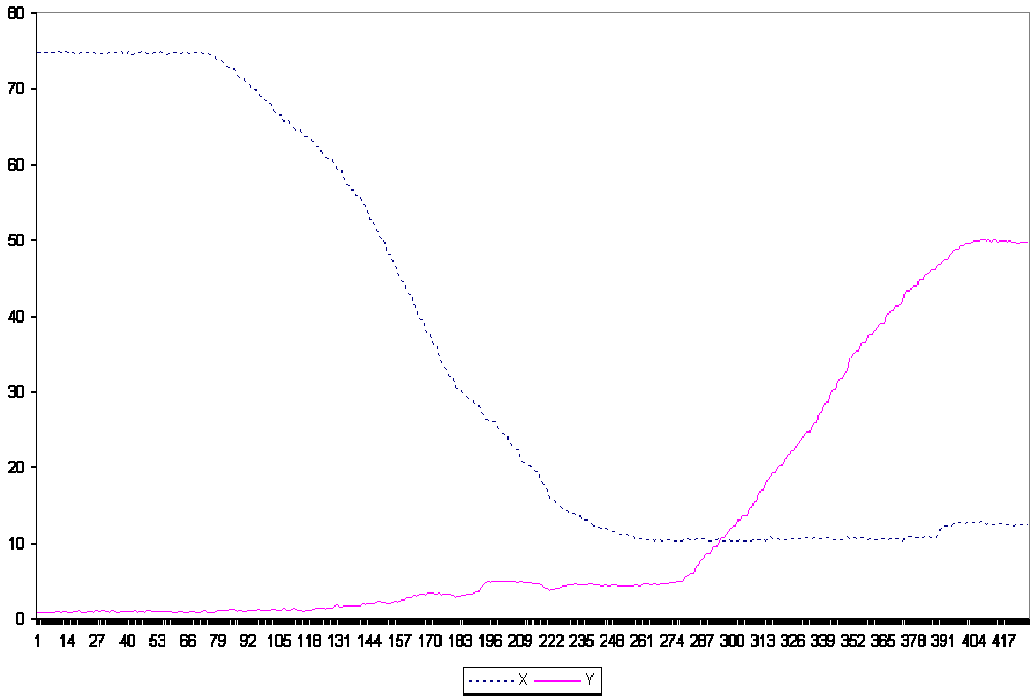


Figura 41: Esfera se movimentando em *X* e depois *Y*.

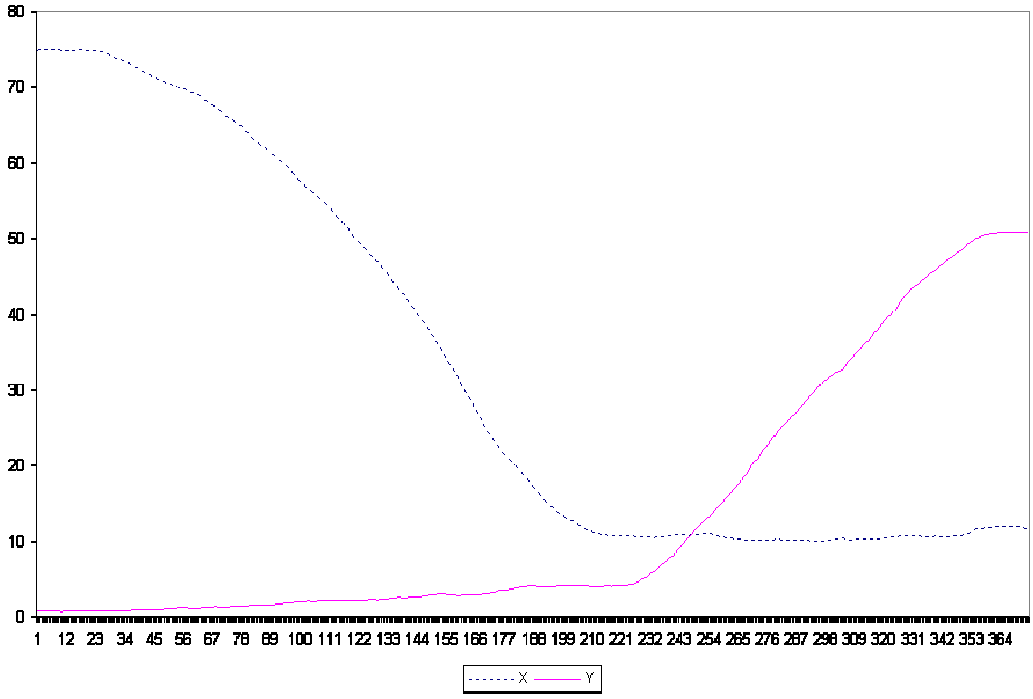


Figura 42: Esfera se movimentando em *X* e depois *Y* com filtro de Kalman habilitado.

Os gráficos ilustram o movimento esperado sendo que, quando o filtro de Kalman estava habilitado, o movimento descreveu uma trajetória mais suave.