Instituto de Matemática e Estatística
Universidade de São Paulo

Bacharelado em Ciências da Computação

Renderização de pessoas em movimento a partir de vídeos

Ricardo Juliano Mesquita Silva Oda

Supervisor: Prof. Dr. Carlos Hitoshi Morimoto

São Paulo - SP

Segundo semestre de 2011

Resumo

Tendo em mãos múltiplos vídeos gravados de pontos de vista distintos de uma mesma cena, é possível, através de técnicas de renderização, gerar novas visualizações da mesma.

Essa técnica permite uma reconstrução realística de uma cena previamente gravada, e também a movimentação no tempo e espaço nela, simulando uma câmera virtual.

Contudo a renderização de pessoas em movimento da cena apresenta vários problemas e desafios, este trabalho tem por objetivo investigar técnicas de modelamento e renderização de pessoas que apresentem resultados realistas.

Sumário

Resumo **1**

1 Introdução **3**

1.1 Objetivo 3

1.2 TV3D e sua motivação 3

1.3 Organização do texto 4

2 Parte Objetiva **6**

2.1 O que é renderização? 6

2.2 Sobre o TV3D 6

2.2.1 Método 8

2.2.2 Segmentação e Rastreamento 9

2.3 O problema 12

2.4 Solução proposta 13

2.4.1 O filtro 13

2.4.2 Condensation Filter 16

2.5 Implementação 19

2.6 Resultados 19

2.7 Conclusão 22

3 Parte Subjetiva **23**

3.1 Experiências 23

3.2 Desafios e dificuldades 23
3.3 Disciplinas relevantes 24
3.4 Trabalhos futuros 24

Referências **25**

Parte 1

Introdução

Foi no primeiro semestre de 2011 que meu interesse na área de processamento gráfico cresceu, enquanto eu cursava a matéria de Introdução à Computação Gráfica ministrada pelo Professor Carlos Hitoshi.

Nesse mesmo período, estava em busca do supervisor e tema para meu trabalho de conclusão de curso, e foi quando procurei o Prof. Hitoshi na tentativa de encontrar um assunto que me chamasse a atenção. Após algumas conversas, conheci um pouco do estudo do Jeferson sobre visadas livres[1][2] e o projeto TV3D Interativa[4]. Interessado, aceitei uma proposta de trabalhar nesse projeto.

1.1 Objetivo

O principal objetivo do trabalho foi estudar a renderização de objetos móveis em uma cena. Mais especificamente, foi uma tentativa de melhorar a renderização das pessoas em movimento nas cenas geradas pelo projeto TV3D Interativa.

1.2 TV3D e sua motivação

O projeto TV3D[4] é um sistema de vídeo interativo, feito utilizando o método desenvolvido pelo Jeferson[2]. É voltado para o aumento da interatividade com vídeos, que atualmente é quase inexistente em cenários reais.

A partir de imagens de vídeos de câmeras reais e um pouco de pré-processamento, o sistema permite o usuário controlar uma câmera virtual na cena, pondendo mudar seu ponto de vista ou navegar pela espaço tridimensional.

Assim, o projeto tem grande potencial tanto na área de entreterimento, como na área de segurança voltada para vigilância, além de outras. Pois a simulação da câmera virtual que pode ser controlada, introduz um novo tipo de interativadade em vídeos de cenários reais.

Ele está envolvido em três áreas da computação: processamento de imagens, visão computacional e computação gráfica. Processamento de imagens e visão computacional são utilizados nos frames dos vídeos de entrada para extrair informações da cena, como por exemplo a localização das pessoas. Em seguida são utilizadas técnicas de renderização de computação gráfica para gerar as novas visualizações da cena, utilizando a informação obtida no primeiro passo.

Se separarmos as imagens como um subconjunto diferente de dados em geral, temos a relação entre essas áreas de atuação representada na Figura 1.1.

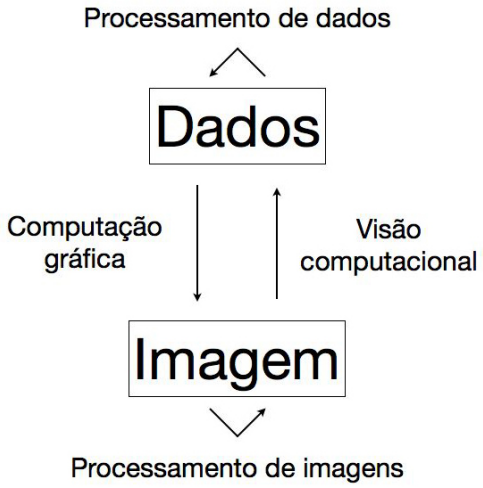


Figura 1.1: Relação entre as áreas de processamenot gráfico

1.3 Organização do texto

A monografia se dividirá em duas partes além desta introdução, uma objetiva e uma subjetiva.

Começando pela parte objetiva, contextualizará o assunto sobre a geração de novas visadas a patir vídeos esparsos, citando o trabalho do Jeferson. A seguir haverá a explicação dos conceitos teóricos utilizados no trabalho, e seus papéis na parte prática do desenvolvimento. Por fim, a conclusão possuirá a análise dos resultados obtidos.

A parte subjetiva relatará sobre as experiências vividas no processo de desenvolvimento

do trabalho. Também mostrará o papel do Bacharelado de Ciências da Computação em sua realização.

Parte 2

Parte Objetiva

Nessa parte do texto, serão apresentados os conceitos teóricos do trabalho, além da parte prática de desenvolvimento.

2.1 O que é renderização?

Começando pelo básico, temos essa pergunta: "O que é renderização?". E a resposta é simples, a renderização pode ser definida por qualquer processo que a partir de um modelo gere uma imagem.

Assim qualquer mudança no modelo também influência na imagem renderizada. Tanto que, a solução proposta por este trabalho tenta melhorar as informações dos modelos a fim de aperfeiçoar a renderização final das pessoas, e não o mexe com processo de renderização em si.

2.2 Sobre o TV3D

O projeto TV3D[4] e sua implementação também conhecida como FVV (Free Video View) é um sistema de vídeo interativo, voltado para o aumento da interatividade com vídeos, que atualmente é quase inexistente em cenários reais. É feito utilizando o método desenvolvido pelo Jeferson[2], que utiliza técnicas de processamento de imagens, visão computacional e computação gráfica para recriar cenários a partir de imagens de vídeos. A Figura 2.1 ilustra um resultado do projeto.

Utilizando a informação proveniente de múltiplas câmeras esparças fixas, o projeto permite a reconstrução da cena filmada e a simulação de uma câmera virtual, a qual pode ser

controlada pelo usuário e daí surge a interatividade com os vídeos.

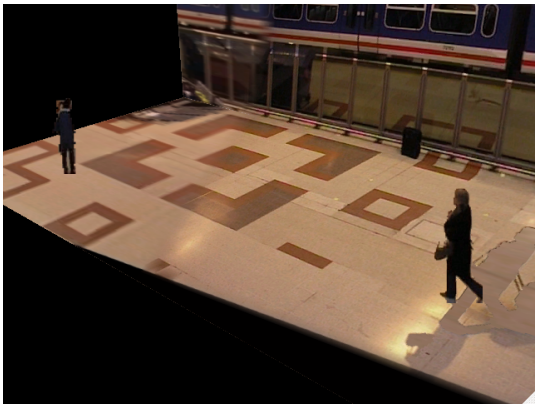
Existem várias formas de interação com vídeos, a proposta do projeto TV3D Interativa é a de uma visada livre. *"Uma visada é definida pela posição e orientação da câmera em relação à cena e também pode conter outros parâmetros"*[3]. Sendo livre a câmera pode obter a posição e ângulo que melhor satisfaça o usuário. Esse tipo de interativade com vídeos é comum em ambientes virtuais, principalmente de jogos tridimensionais, contudo é algo pouco utilizado em cenários reais.

A criação de novas visadas pode ser utilizada junto à sistemas de vigilância com vídeo em lugares públicos; em programas de TV; ou qualquer outro sistema que possua um monitoramento feito com vídeos. Assim, tem um grande potencial tanto na área de entreterimento, como na área de segurança voltada para vigilância, além de outras. Pois a simulação da câmera virtual que pode ser controlada, cria um novo tipo de interativadade com os vídeos.

(a) Imagem da câmera 0

(b) Imagem da câmera 1

(c) Imagem da câmera 2



(d) Nova vizualição com um ponto de vista arbitrário

Figura 2.1: Exemplo de um resultado da TV3D Interativa

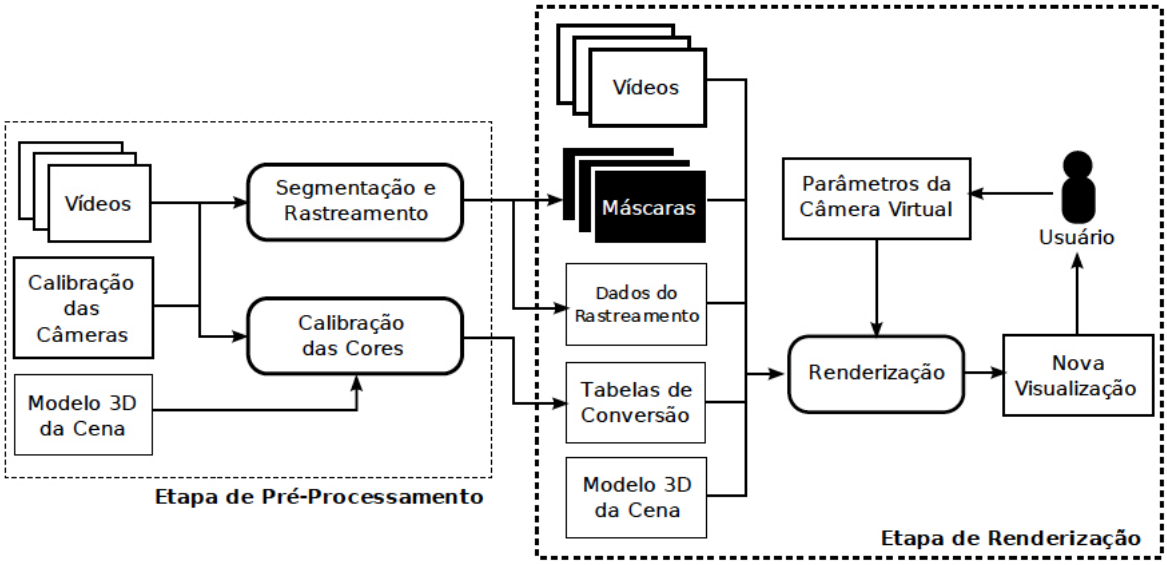


Figura 2.2: Diagrama de blocos do método, da referência [2]

2.2.1 Método

O método utilizado para geração das novas visualizações a partir de múltiplos vídeos é dividido em duas etapas. O diagrama da Figura 2.2 mostra a relação entre elas e da uma idéia geral do sistema.

Na primeira etapa é feito um pré-processamento da entrada com o objetivo de gerar os dados necessários para criação de novas visualizações da cena. E são necessários alguns requisitos iniciais: um modelo simples da cena e um ajuste das câmeras.

É necessário a criação de um modelo 3D simples da cena, de preferência com texturas reais obitdas das camêras para aumentar o realismo das novas vizualizações que são geradas em cima desse modelo.

As câmeras devem ser calibradas e sincronizadas. Ou seja, devemos saber as posições em que elas estão e a direção em que apontam, além de seus vídeos estarem sincronizados. Isso possibilita a combinação das imagens em um instante do tempo para geração das novas visualizações.

O próximo passo do sistema é o mais importante para este trabalho. A segmentação e o rastreamento dos objetos móveis no vídeos.

A segmentação permite diferenciar os objetos do fundo da cena, sendo muito importante para a renderização das pessoas nas novas visualizações. É a segmentação que extrai a imagem das pessoas se movendo nos vídeos que são utilizadas para gerar a imagem final

delas na câmera virtual. O rastreamento tem como objetivo localizar e rotular as pessoas nos vídeos, obtendo as posições reais das pessoas na cena e os diferenciando ao longo do tempo. É usado para posicionar os modelos dos objetos móveis no modelo 3D da cena.

O último passo do pré-processamento calibra as cores dos vídeos para diminuir a diferença de cor e brilho entre eles. É pertinente para a junção dos vídeos de diferentes câmeras que podem ter propriedades distintas.

A segunda etapa consiste da renderização, onde os dados obtidos na primeira etapa são utilizados para gerar as novas visualizações da cena em qualquer ponto de vista.

O processo é iterativo e ocorre a cada instante do vídeo. O modelo 3D simples da cena é renderizado a partir do ponto de vista escolhido pelo usuário. As imagens das câmeras do instante são tratadas: suas cores são corrigidas e os objetos móveis são removidos. As imagens de melhor ângulo de visão com relação ao ponto de vista escolhido pelo usuário são combinadas e projetadas sobre o modelo da cena. Assim temos uma imagem realistica do fundo da cena vista pelo ponto de vista solicitado.

A seguir cada objeto móvel é renderizado sobre um modelo na forma de painél texturizado utilizando a imagens segmentadas das câmeras com ponto de vista mais próximo ao escolhido pelo usuário. E esses painéis são colocados na cena nas posições obtidas pelo rastreamento. Aqui são onde existem os problemas que são investigados por este trabalho.

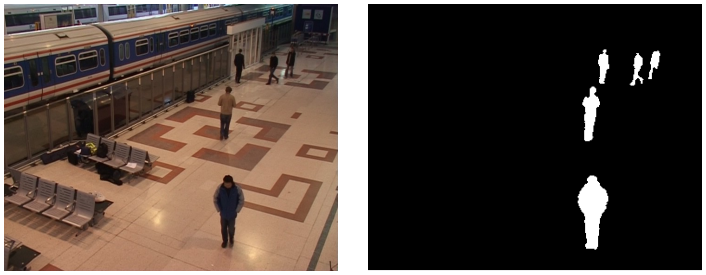
Vamos nos aprofundar um pouco mais na geração dos dados que influenciam na renderização das pessoas.

2.2.2 Segmentação e Rastreamento

A segmentação e o rastreamento permitem, respectivamente, separar os objetos móveis do fundo da cena e associar a cada um deles um rótulo e uma posição. A separação é nessária pois o método do Jeferson utiliza técnicas distintas para renderizar em momentos diferentes o fundo da cena e os objetos móveis. E as posições obtidas são essenciais para a disposição dos objetos no modelo 3D da cena.

A segmentação é feita usando uma técnica de subtração de fundo para gerar máscaras de segmentação. A subtração de fundo é um método de processamento de imagens que tem como objetivo isolar os objetos do fundo da imagem. O resultado pode ser visto na Figura 2.3.

Cada pixel de uma imagem é representado por um conjunto de canais de cores, por exemplo: R (vermelho), G (verde) e B (azul). Sendo cada canal um inteiro que pode ser utilizado em cálculos arbitrários. A idéia básica do método é subtrair uma imagem de uma



(a) Imagem da câmera (b) Máscara de segmentação

Figura 2.3: Exemplo do resultado de uma subtração de fundo

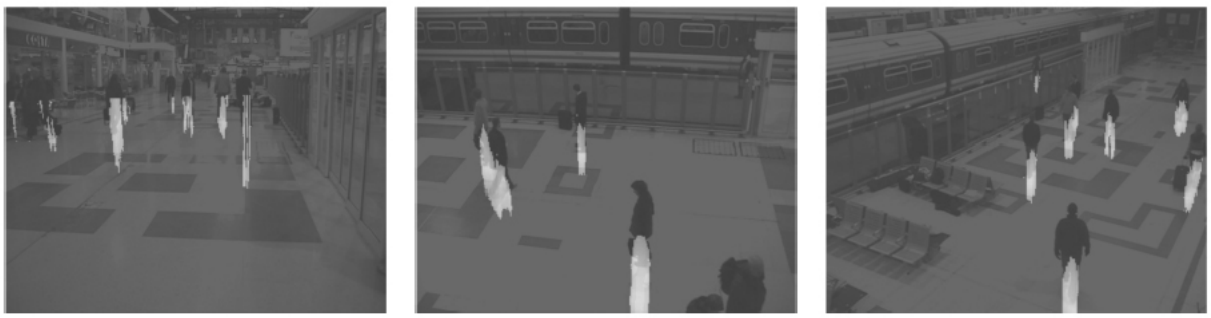
câmera no instante t da imagem do fundo dessa câmera. Assim haverão áreas com pouca diferença que serão consideradas como fundo, outras com uma diferença as quais podem ser objetos, esse limiar da diferença é um parâmetro do método.

Para isso é necessário treinar a imagem do fundo de cada câmera. O método do Thiago[5][6], outro ex-orientando do Hitoshi, utiliza misturadas de gaussianas para modelar os pixels de fundo.

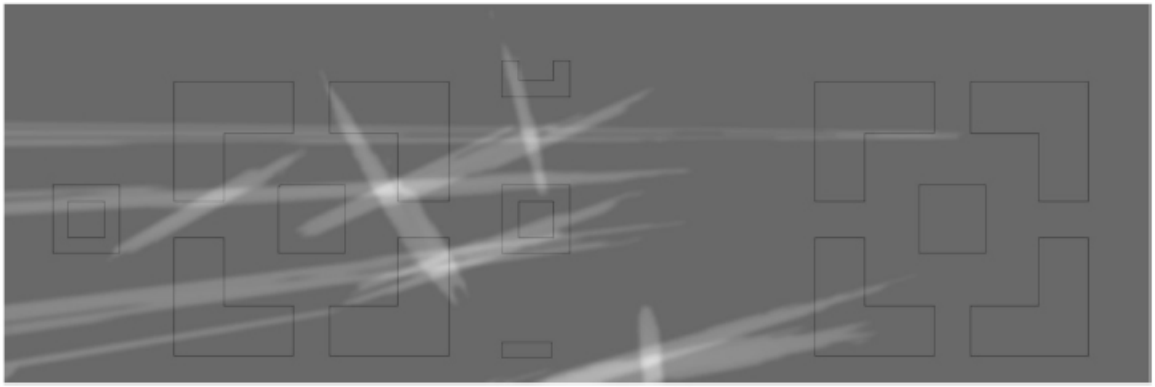
Utilizando esse metodo é possível conseguir a máscara de segmentação de cada instante t. As máscaras são imagens binárias (com somente duas cores: preto e branco) onde há a separação das imagens em dois planos, o plano do fundo (preto) e um primeiro plano com os objetos do instante (branco). Essas máscaras são usadas para caracterizar os objetos nas imagens originais das câmeras.

O rastreamento foi feito utilizando restrições homográficas com integração de suporte descritas nos trabalhos[1][5][6]. Utilizando as máscaras de segmentação e homografias calculadas a partir da calibração das câmeras, o método estima as posições dos objetos no plano do chão da cena. As homografias são transformações lineares que levam um ponto de um plano para outro, no caso são utilizadas as transformações que levam um ponto de um plano da câmera para o plano do chão do modelo da cena. Com as máscaras de segmentação é calculado o suporte de cada pixel x_i foreground (do plano dos objetos). O suporte $S(x_i)$ é dado pela massa de pixel foreground acima de x_i. Com o acumulado dos suportes de todas as câmeras projetadas no plano do chão são estimadas as posições das pessoas nos pontos de máximo. Como mostra a figura 2.4.

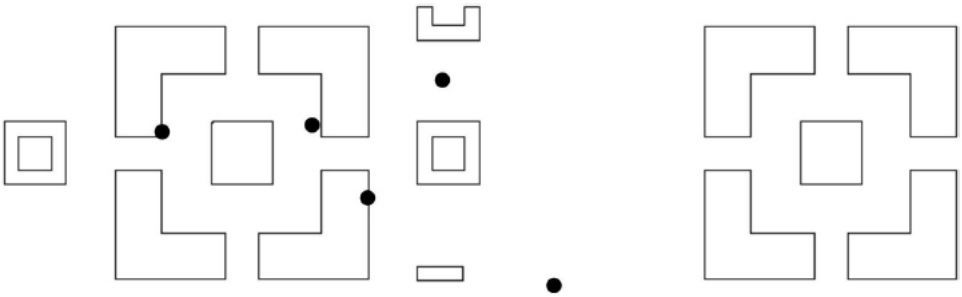
Os dados também são rastreados através do Kalman Filter para a rotulação de cada objeto. Ele tenta seguir cada pessoa ao longo do tempo para saber que em instantes diferentes dois dados de rastreamento são da mesma pessoa.



(a) Suporte em cada câmera



(b) Suporte acumulado das câmeras



(c) Estimação das posições das pessoas

Figura 2.4: Integração do suporte e estimação das posições, das referências [2][5]

2.3 O problema

Apesar dessas técnicas avançadas de segmentação e rastreamento, artefatos são presentes na renderização final das pessoas na implementação TV3D. Artefatos são erros na renderização. Logo existem falhas na construção do modelo das pessoas e/ou no processo de gerar a imagem delas a partir desse modelo.

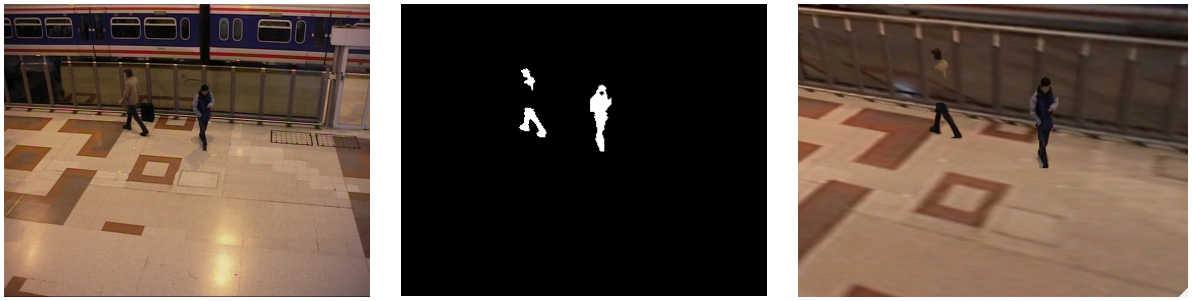
A renderização das pessoas no projeto TV3D é influenciada por ambas etapas do método, tanto no pré-processamento quanto na etapa de renderização. Os modelos de cada objeto são painéis texturizados gerados a partir dos dados obtidos no pré-processamento, no caso a segmentação das imagens destaca as pessoas em cada vídeo. E o posicionamento dos painéis no modelo da cena é feito a partir dos dados obtidos no rastreamento.

O objetivo deste trabalho é melhorar essa renderização. O código-fonte do projeto fonte disponibilizado para efetuar este trabalho foi somente o da etapa de renderização que recebe como entrada os dados do pré-processamento. Além do código-fonte também utilizamos entradas já pré-processadas. Junto aos vídeos de dois cenários, temos: as máscaras de segmentação, os dados de rastreamento, as tabelas de conversão de cores, o modelo simples 3D de cada cena e a calibração da camêra com suas homografias já calculadas.

Analisando tanto o código fonte quanto os resultados que a implementação provê, foram descobertos algumas causas dos erros na renderização de pessoas. E os problemas estudados foram na segmentação e rastreamento, ou seja, em dados essenciais para criação do modelo das pessoas.

Falhas na segmentação dos vídeos causam erros nas imagens finais, pois são elas que se responsabilizam pela definição e caracterização das pessoas nas imagens dos vídeos. O modelo da pessoa é definido por um painel texturizado com a imagem de uma câmera após aplicada a máscaras de segmentação, como se recortássemos a imagem da câmera somente no pixels foregrounds (brancos) de sua máscara de segmentação. Então em um instante do tempo se parte da pessoa não for destacada na máscara de segmentação, essa parte não aparecerá no modelo final nesse instante, como na Figura 2.5. Essas falhas na segmentação ocorrem principalmente devido à pequena diferença entre os pixels do fundo com os pixels da pessoal durante o processo de subtração de fundo.

Inconsistências nos dados de rastreamento afetam no posicionamento dos modelos dos objetos no modelo da cena, bem como a localização das pessoas nas imagens das câmeras para a aplicação das máscaras e texturização dos painéis. Foram encontrados erros de falsos positivos/negativos, posicionamento e rotulação. Erros de posicionamento são comuns pois estamos trabalhando com medidas e estimações. Os falsos negativos são definidos pela falta de dados de rastreamento de uma pessoa em algum instante, e os falsos positivos por dados a



(a) Imagem da câmera (b) Máscara de segmentação (c) Nova visualização

Figura 2.5: Exemplo de uma máscara de segmentação com falhas, gerando erros na renderização das pessoas. Grande parte do corpo da pessoa a esquerda não é renderizada

mais do que os reais como na Figura 2.6. Falsos positivos/negativos levam à renderização de pessoas a mais ou a menos na cena virtual, e ocorrem por erros nas estimações. A rotulação das pessoas tem erros devido aos falsos positivos e negativos que afetam o rastreamento, por exemplo, ao seguir uma pessoa para saber que o dados em diferentes instantes são da mesma pessoa ocorre um falso negativo, ou seja ela desaparece por um frame e reaparece no outro, mas nesse outro frame ela já está com outro rótulo pois é um "novo" elemento em cena.

2.4 Solução proposta

Com somente o acesso à parte de renderização da implementação, a solução proposta foi corrigir os dados de rastreamento através de um filtro. Assim alteramos os dados de entrada do processo de redenrização e melhoramos o resultado final sem alterar o sistema legado e sem a necessidade de recriar o pré-processamento inteiro. Além disso, foi proposto suavizar os dados de rastreamento para diminuir o ruído do posicionamento das pessoas ao longo do tempo. E re-gerar as máscaras de segmentação com o intuito de melhorar o resultado final.

2.4.1 O filtro

O filtro recebe de entrada os dados de rastreamento gerados pelo pré-processamento e gera novos dados de rastreamento. A Figura 2.7 mostra onde ele se localiza dentro do sistema.

O principal objetivo do filtro foi remover os falsos positivos/negativos dos dados de rastreamento, e assim, melhoraro também a rotulação das pessoas. E indiretamente aprimorar a renderização das pessoas nas imagens finais.

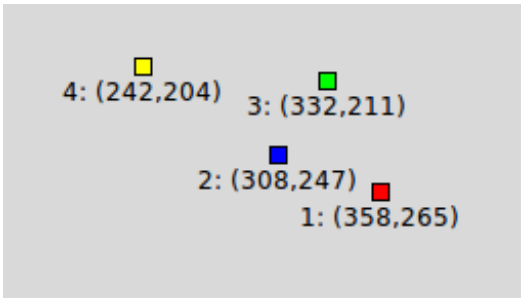
A idéia do filtro é fazer um novo rastreamento em cima dos dados de rastreamento obti-

(a) Imagem da câmera 0

(b) Imagem da câmera 1

(c) Imagem da câmera 2

(d) Imagem da câmera 3



(e) Dados de rastreamento

Figura 2.6: Exemplo de falso positivo: no vídeo há três pessoas mas nas informações de rastreamento obtidas existem quatro

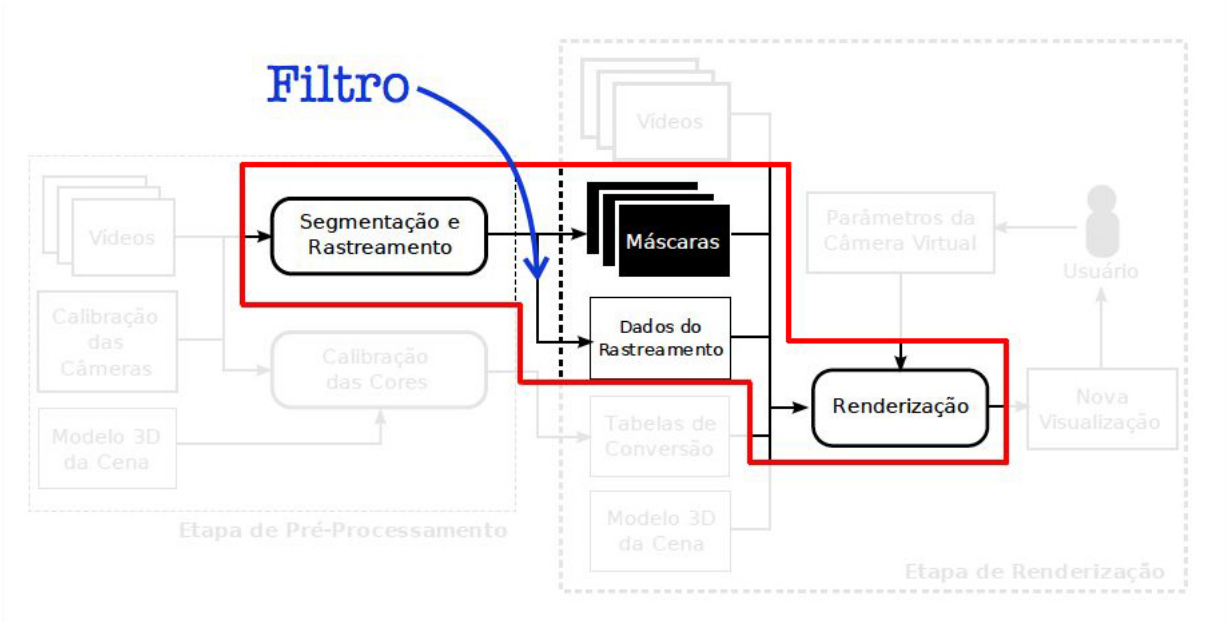


Figura 2.7: Localização do filtro no sistema

dos no pré-processamento, e ao longo do processo corrigir suas falhas. Se uma pessoa que estava presente em uma região ao longo de vários instantes sumir repentinamente mas reaparecer depois de um curto período, preenchemos essa lacuna com posições arbitrárias que façam sentido. Se houverem ruídos que não representam nenhuma pessoa, ou seja dados que apareçam por curtos períodos de tempo, descartamos esses dados. E assim tentamos resolver os problemas de falsos positivos/negativos, como esse processo requer um rastreamento também geramos novos rótulos para os objetos em movimento. Além de gerarmos novas estimações para o posicionamento das pessoas.

Uma idéia simplificada do algoritmo é a seguinte:

```
Filter(data, param):
 1. id := 0
 2. result := {}
 3. while(true):
 4.     obj := valid_obj(data, param)
 5.     if obj == NULL:
 6.         then break
 7.     tracked := follow_and_correct(data, result, obj, id, param)
 8.     remove(data, tracked, param)
 9.     id := id + 1
10. return result
```

O algoritmo tem como entrada e saída dados de rastreamento. Esses dados contêm o a definição do intervalo processado e as posições no plano do chão dos objetos rotulados em cada instante do tempo. Ele também precisa de algumas parâmetros:

start_frame:	Começo do intervalo que será filtrado, deve ser condizente com o intervalo dos dados de entrada.
end_frame:	Fim do intervalo que será filtrado, deve ser condizente com o intervalo dos dados de entrada.
v_const:	A velocidade média das pessoas em relação aos dados de entrada.
d_threshold:	Um limiar de distância, usado para delimitar o deslocamento máximo de uma pessoa considerando erros, é usado junto de sua velocidade média.
t_threshold:	Um limiar de tempo em frames de um falso negativo.
l_threshold:	Um limiar de tempo em frames de um falso positivo.

Essas constantes são utilizadas nas funções do algoritmo. Ele começa com *valid_obj* utilizando *d_threshold* e *t_threshold* que devolve um objeto que está relativamente isolado e tem uma consistência temporal para iniciar o rastreamento. Com um objeto para rastrear *follow_and_correct* o segue ao longo do tempo corrigindo falhas de falsos negativos, e gera novos

dados de rastramento com essas lacunas preenchidas, note que o rótulo inicial de cada objeto não é considerado aqui e novos rótulos são gerados, assim podemos começar seguindo um objeto, preencher lacunas e continuar seguindo um objeto com um identificador diferente de acordo com os dados de entrada. Para isso são necessários todos os parâmetros para verificar a consistência temporal e espacial dos objetos. A função *remove* diminui o conjunto de entrada, retirando os dados rastreados na iteração.

A idéia é remover os falsos negativos no rastreamento de cada objeto, e descartar objetos invalidos do conjunto inicial, que serão considerados falsos positivos. Além disso serão gerados novos rótulos que serão robustos em relação aos erros corrigidos anteriormente.

2.4.2 Condensation Filter

Mas como seguir as pessoas ao longo do tempo? Em um ambiente dinâmico, rastrear objetos que podem se movimentar aleatóriamente é desafiador. Filtros de partículas foram desenvolvidos com o intuíto de tratar esse tipo de dados mesmo que possuam ruído e erros. Eles são interessantes devido à flexibidade de representar distribuições além da Gaussiana.

Um filtro de partículas que ficou famoso na comunidade de visão computacional e inteligência artificial foi o Condensation Filter[7] (Conditional Density Propagation Filter), que utilizamos neste trabalho. É usado para prevêr o próximo estado de um sistema dado o histórico de estados anteriores do mesmo. Chama-se filtro de partículas por ser um método probabilístico e funcionar através de amostragens, um conjunto de amostras que podem ser chamadas de partículas são geradas a cada iteração e servem para fazer uma aproximação da função densidade de probabilidade do estado do sistema. Esses estados devem ser ponderados para a estimação do algoritmo ser ser corrigida e aprimorada a cada iteração.

Para utilizarmos o Condensation Filter é necessário modelar os estados e criar a uma função de ponderação das amostras. Os estados foram modelados da seguinte forma:

$$E = \begin{bmatrix} S_x \\ S_y \\ V_x \\ V_y \end{bmatrix} \tag{2.1}$$

Um estado E de um objeto guarda as posições (S_x, S_y) e sua velocidade do (V_x, V_y). Também é necessário modelar a transformação de um estado, mas isso é simples. Queremos que no próximo estado ele esteja na posição $S_k = S_{k-1} + V * \partial t$ e assim temos a matriz de

transformação:

$$A = \begin{bmatrix} 1 & 0 & \partial t & 0 \\ 0 & 1 & 0 & \partial t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

Mas $\partial t = 1$ pois as transformações ocorrem frame a frame. Agora falta a função de ponderação. O inverso da distância euclidiana entre as posições da amostra z e da medição e foi utilizado como peso para cada amostra:

$$P_z = \frac{1}{dist(e, z)} \tag{2.3}$$

O condensation funciona em três etapas importantes:

- predição
- medição
- atualização

Na primeira uma predição do estado atual é feita, a amostra com maior confiança é escolhida do conjunto de amostras. Na segunda há a medição do estado atual, através da entrada do filtro. E na terceira há a ponderação de cada amostra de acordo com a função criada. Assim podemos melhorar a idéia do algoritmo do filtro.

```
Filter(data, param):
 1. id := 0
 2. result := {}
 3. while(true):
 4.     obj := valid_obj(data, param)
 5.     if obj == NULL:
 6.         then break
 7.     cond := Condensation()
 8.     tracked := follow_and_correct(data, result, obj, id, param, cond)
 9.     remove(data, tracked, param)
10.     id := id + 1
11. return result
```

```
follow_and_correct(result, obj, id, param, cond):
 1. init_cond(cond, obj)
 2. tracked := {obj}   // elementos rastreados
 3. predicted := {obj} // predições
 4. fails := {}        // falhas
 5. found := 1
 6.
 7. // rastreamento para frente no tempo
 8. for t := obj.frame + 1 until param.end_frame, do:
 9.     // predição
10.     if (found):
11.         then z := predict(cond)
12.     // medição
13.     x := nearest(z, data, param)
14.     if (x != NULL):
15          found := true
16.         then tracked := tracked + {x}
17.              predicted := predicted + {z}
18.              // preenchimento dos buracos (falsos negativos)
19.              if (fails.size() > 0):
20.                  then predicted := predicted + fails
21.                       fails := {}
22.
23.              // atualização e ponderação
24.              update(cond, x)
25.         else
26               found = false
27.              if (fails.size() > t_threshold):
28.                  then break // muitos falsos negativos seguidos
29.                  else nf := nf + 1
30.                       fails := fails + {new_obj(z, t)}
31.
32. fails := {}
33. nf := 0
34. // rastreamento no outro sentido do tempo
35. for t := obj.frame - 1 until param.end_start, do:
36.     (...) // análogo
```

```
37.
38.  if (predicted.size() > l_threshold):
39.      then result := result + predicted
40.  return tracked
```

Uma inicialização *init_cond* é necessária onde o modelo do estado é definido e posição inicial do objeto é armazenada. Depois disso seguimos o objeto ao longo do tempo, fazendo a predição de sua posição e uma busca nos dados de entrada para checar a existência uma medida próxima à predição. Caso uma medida não seja encontrada, seguem duas possibilidades: encontramos um falso negativo, ou o objeto saiu de cena. Então marcamos um buraco nessa ponto ou paramos de rastrear o objeto. A constante *t_threshold* define o limiar para parar ou não. Por outro lado caso a medida seja encontrada, devemos guarda a predição que será nossa nova estimação, guardar o dado rastreado para ser removido da entrada, e corrigir as falhas gerando novos dados de rastramento nos buracos encontrados.

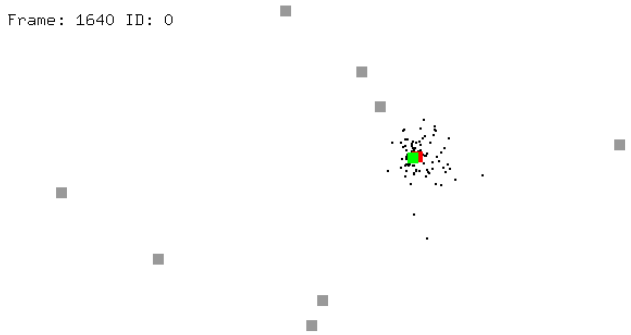
O rastreamento é feito nos dois sentidos do tempo pois o objeto encontrado no início do algoritmo pode se tonar válido somente no meio do tempo da cena. E ao fim do rastreamento se os dados tiverem uma vida muito curta, ou seja, um objeto foi rastreado por um período muito curto ele é descartado pois provavelmente é um falso positivo. A constante *l_threshold* define o limiar para o tempo de vida mínimo de um objeto.

2.5 Implementação

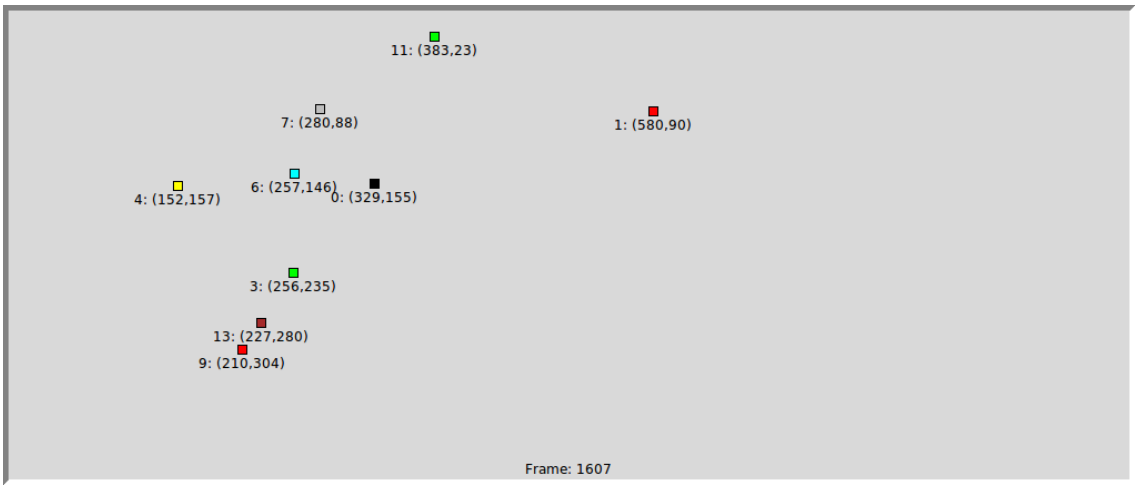
A implementação foi feita em C++ que é linguagem em que o projeto legado foi feito, sendo que a implentação do Condensation é do OpenCV, uma famosa biblioteca de processamento de imagens e visão computacional. Também foi implementada uma versão do algoritmo com animação, para depurá-lo visualmente. Além do filtro foram implementadas duas outras animações em python para análise dos dados de rastreamento, ambas mostram a movimentação das pessoas no plano do chão contudo uma delas mostra animação de dois dados ao mesmo tempo para comparação. A Figura 2.8 mostra a execução dessas animações.

2.6 Resultados

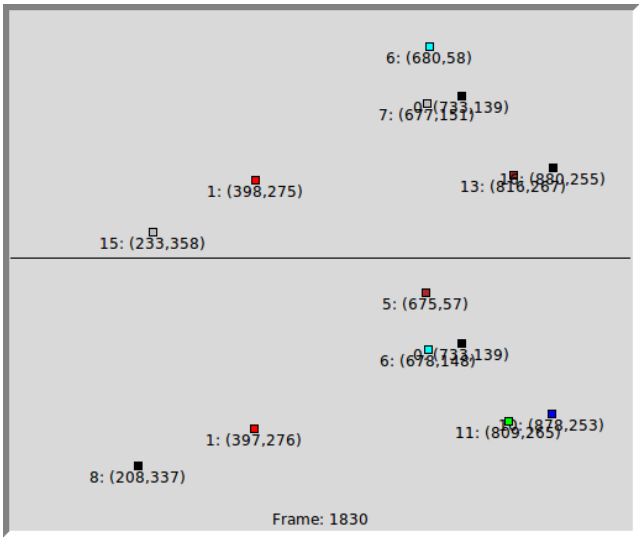
Os resultados foram feitos analisando a aplicação do filtro nos dados de rastreamento gerados com o pré-processamento dos vídeos do PETS 2006[8].



(a) Filtro de partículas com animação



(b) Animação dos dados de rastreamento em python



(c) Animação comparativa dos dados de rastreamento

Figura 2.8: Animações implementadas

O PETS (Performance Evaluation of Tracking and Surveillance) é um workshop anual internacional promovido pela IEEE, onde são apresentados projetos e artigos com relação à rastreamento e monitoramento de ambientes, com foco em tecnologias de visão computacional.

Em 2006, foram disponibilizados vídeos de estações de trens públicas da inglaterra. São imagens quatro câmeras na estação de trem, contudo uma delas possui uma campo de visão ruim para o projeto e suas imagens foram descartadas para o projeto TV3D Interativa.

É dificil medir a qualidade dos dados resultantes, sem possuir os dados reais da cena. Assim foram feitas algumas medidas comparativas:

Experimento	Original	0	1	2	3
Dist max	991.069	999.987	993.998	999.987	993.998
Dist min	3.60555	1	2.23607	1	2.23607
X max	1007	1009	1013	1009	1013
X min	-4	-7	-5	-7	-5
Y max	387	386	387	387	388
Y min	0	-6	-3	-6	-3
N max	10	10	10	10	10
N min	2	0	0	0	0
N mean	5.07383	4.95638	4.94094	4.94161	4.90134
# frames	1490	1490	1490	1490	1490
V max	50.2195	100.319	104.235	100.319	104.235
V min	0	0	0	0	0
V médio	4.50529	3.57064	3.57736	3.61673	3.60151
# v medidos	7516	7348	7316	7325	7259
# rótulos	44	28	32	27	29

Os experimentos tiveram as seguintes entradas:

Experimento	0	1	2	3
v_const	10	10	10	10
d_threshold	50	40	50	40
t_threshold	20	20	20	20
l_threshold	20	20	30	30

2.7 Conclusão

Visualmente não houve uma melhora significativa na renderização das pessoas. É difícil comparar dados de rastreamento sem ter um *ground truth*, ou seja sem saber os dados das posições reais dos objetos. Assim boa parte das análises durante o desenvolvimento foram feitas em cima de simulações providas pelas animações implementadas. Isso teve um efeito colateral: a falta de observação nas renderizações finais. Assim o resultado do trabalho acabou não se focando na renderização final, e sim nos dados de rastreamento em si. Contudo notamos a redução no número de rótulos observados ao aplicar o filtro nos frames 1510 a 2999 dos dados do PETS. Isso prova uma redução nos falsos positivos/negativos, e uma correção na rotulação dos objetos.

Parte 3

Parte Subjetiva

3.1 Experiências

Apesar das dificuldades no desenvolvimento do trabalho, foi bem interessante. O fato de trabalhar junto à um supervisor (um orientador na minha opinião), foi uma experiência única neste trabalho. Não fiz inciação ciêntifica durante o curso, assim não tinha tido esse tipo de prática.

3.2 Desafios e dificuldades

Fiquei muito tempo tentando fazer o código legado funcionar, desde descobrir quais bibliotecas eu precisava para compilar, até como executar o sistema.E depois disso ainda demorei bastante para entendê-lo, pelo menos a parte que era interessante para efetuar este trabalho. Só então foi possível investigar os problemas no sistema e estudar soluções para eles.

Tive dificuldades em lidar com a pesquisa pela qual não estou acostumado. Normalmente em uma matéria você aplica nos trabalhos aquilo que estuda em sala aula ou algum assunto direcionado pelo professor, porém neste trabalho de formatura tive que estudar várias tópicos novos em escopo abrangente procurando algo que eu não conhecia para resolver o problema proposto.

Também não tenho certeza se o sistema em que trabalhei era a versão mais nova do TV3D, pois os resultados presentes na monografia do Jeferson parecem melhores do que os gerados com o código legado em que trabalhei. Mas só percebi isso na analise dos resultados deste trabalho.

3.3 Disciplinas relevantes

As disciplinas que considero mais relevantes para realização deste trablho são:

1. **MAC0420 Introdução a Computação Gráfica** foi a matéria que me motivou à saber mais sobre processamento gráfico. Além disso, ela é essencial para entender conceitos como: cena, câmeras, renderização, etc.

2. **MAC0417 Visão e Processamento de Imagens** foi importante para entender conceitos de processamento de imanges como por exemplo a técnica de subtração de fundo. E também técnicas de visão computacional para extração de informações a partir de imagens.

3. **MAC0122 Princípios de Desenvolvimento de Algoritmos** é importante para qualquer área da computação, e foi útil no desenvolvimento do filtro.

4. **MAT0139 Álgebra Linear para Computação** foi muito importante para entender transformações lineares. Mas só percebi após cursar outras matérias como a MAC0420 ou MAC0300.

3.4 Trabalhos futuros

Com certeza é necessário gerar novas máscaras de segmentação para o projeto para melhorar a qualidade das imagens finais.

Uma idéia bem interessante da parte do Hitoshi seria fazer um método iterativo e realimentável de melhoria na geração de máscaras de segmentação utilizando os dados do rastreamento, e melhoria nos dados de rastreamento utilizando informações das máscaras. Mas o método deveria ser melhor estudado e aprimorado.

Referências

[1] da Silva, J. R. ; Santos, T. T., and Morimoto, C. H., *Real Time Novel View Scene Rendering From Multiple Sparse Videos.* In: Proc. of the XII Brazilian Symposium on Virtual and Augmented Reality - SVR 2010. Natal, RN, Brazil, 2010, v.1., p. 184-193. [Best paper award] `http://www.ime.usp.br/~hitoshi/tv3d/silvaSVR2010.pdf`

[2] da Silva, J. R., *Renderização interativa de câmeras virtuais a partir da integação de múltiplas câmeras câmeras esparsas por meio de homografias e decomposições planares da cena* MSc thesis, Universidade de São Paulo, 2010.

[3] da Silva, J. R., and Morimoto, C. H., Material do mini curso 7 apresentada no JAI 2010 `http://www.ime.usp.br/~hitoshi/jai2010/jai2010C7.pdf`

[4] da Silva, J. R. ; Santos, T. T., and Morimoto, C. H. PROJETO: TV 3D Interativa `http://www.ime.usp.br/~hitoshi/tv3d/`

[5] Santos, T. T. and Morimoto, C. H., *Detecção e rastreamento de múltiplos objetos em condição de oclusão severa por meio de integração de suporte sob restrição homográfica* PhD thesis, Universidade de São Paulo, 2009.

[6] Santos, T. T. and Morimoto, C. H., *Multiple camera people detection and tracking using support integration*, Pattern Recognition Letters, Volume 32, Issue 1, 1 January 2011, Pages 47-55

[7] Michael Isard, and Andrew Blake, CONDENSATION – conditional density propagation for visual tracking, Int. J. Computer Vision, 29, 1, 5–28, (1998)

[8] PETS 2006 - Workshop on Performance Evaluation of Tracking and Surveillance `http://www.cvg.rdg.ac.uk/PETS2006/`