



UNIVERSIDADE DO ESTADO DO  
RIO DE JANEIRO

IPRJ  
Instituto Politécnico  
Universidade do Estado do Rio de Janeiro

---

INSTITUTO POLITÉCNICO  
GRADUAÇÃO EM ENGENHARIA  
DE COMPUTAÇÃO

Pedro Felipe Pena Barata

Aplicações práticas em técnicas de reconstrução 3D

Nova Friburgo

2017



UNIVERSIDADE DO ESTADO DO  
RIO DE JANEIRO



---

INSTITUTO POLITÉCNICO  
GRADUAÇÃO EM ENGENHARIA  
DE COMPUTAÇÃO

Pedro Felipe Pena Barata

**Aplicações práticas em técnicas de reconstrução 3D**

Trabalho de Conclusão de Curso apresentado, como requisito parcial para obtenção do título de Graduado em Engenharia de Computação, ao Departamento de Modelagem Computacional do Instituto Politécnico, da Universidade do Estado do Rio de Janeiro.

Orientador: Prof. Dr. Ricardo Fabbri

Nova Friburgo

2017

Pedro Felipe Pena Barata

## **Aplicações práticas em técnicas de reconstrução 3D**

Trabalho de Conclusão de Curso apresentado, como requisito parcial para obtenção do título de Graduado em Engenharia de Computação, ao Departamento de Modelagem Computacional do Instituto Politécnico, da Universidade do Estado do Rio de Janeiro.

Aprovada em 30 de 09 de 2017.

Banca Examinadora:

---

Prof. Dr. Ricardo Fabbri (Orientador)  
Departamento de Modelagem Computacional – UERJ

---

Prof. Dr. Edirlei Soares  
Departamento de Modelagem Computacional – UERJ

---

Prof. Dr. Roberto Pinheiro  
Departamento de Modelagem Computacional – UERJ

Nova Friburgo  
2017

## **AGRADECIMENTOS**



## RESUMO

BARATA, Pedro Felipe Pena. *Aplicações práticas em técnicas de reconstrução 3D*. 2017. 41 f. Trabalho de Conclusão de Curso (Graduação em Engenharia de Computação) – Departamento de Modelagem Computacional, Instituto Politécnico, Universidade do Estado do Rio de Janeiro, Nova Friburgo, 2017.

A partir dos anos 2000, a área de reconstrução 3D vem sido amplamente explorada. No início, sensores de alcance, tanto aéreos quanto terrestres, eram empregados em diferentes aplicações, devido à facilidade de manuseio e ao baixo custo. Porém, constantes melhorias na tecnologia, sobretudo, nos *hardwares* e *softwares* no âmbito da reconstrução, fizeram com que hoje, quase duas décadas depois, novas técnicas surgissem.

Muitos cientistas que utilizavam a fotogrametria converteram seus esforços na área dos sensores à laser. Pois além de executarem uma reconstrução mais rápida, possuem uma altíssima acurácia, compensando seu alto custo inicial. Isto dificultou e desacelerou o processo de descoberta de novos algoritmos e métodos na área da fotogrametria.

Hoje em dia, graças à esse avanço, a fotogrametria, aliada a novos algoritmos, como o *Structure of Motion (SfM)*, pontos em comum e de combinação de imagens, por exemplo, consegue competir com scanners à laser e sensores de alcance.

ABORDAR O SFM, SIFT E CMVS (POUCO) !!! ?????

Com uma combinação de algoritmos, com o *SfM*, junto com o SIFT () e o CMVS (), é possível gerar uma reconstrução satisfatória apenas utilizando uma câmera de um *smartphone*.

O trabalho foi estruturado da seguinte maneira: previamente apresentam-se os objetivos do projeto, destacando suas funcionalidades e metas, a seguir divide-se em capítulos; O Capítulo 1, que introduz o funcionamento de cada algoritmo e técnica empregada, apresentando e debatendo, comparativamente pontos à favor e contra; O Capítulo 2 é dedicado à ferramenta gráfica utilizada para a obtenção dos resultados (VisualSfM). Finalmente, apresentamos os resultados e conclusões do trabalho, bem como sugestões para implementações e trabalhos futuros.

ABORDAR OS "TODO'S" DO HANGOUTS

Palavras-chave: Reconstrução densa. Nuvem de pontos. SfM. Triangulação.

## ABSTRACT

BARATA, Pedro Felipe Pena. . 2017. [41](#) f. Trabalho de Conclusão de Curso (Graduação em Engenharia de Computação) – Departamento de Modelagem Computacional, Instituto Politécnico, Universidade do Estado do Rio de Janeiro, Nova Friburgo, 2017.

Keywords:

## LISTA DE FIGURAS

Figura 1 - Kinects de primeira geração (a) consistindo de câmeras e projetores infra-vermelho (b) e de segunda geração, consistindo de tecnologia ToF (c). Ambos os kinects são largamente utilizados para escaneamento em tempo real, formando a base de scanners manuais (d), porém nem sempre são úteis para preservação detalhada de patrimônio. Um dos objetivos deste projeto é explorar os limites desta tecnologia. . . . .	12
Figura 2 - A reconstrução usando-se Kinect (de primeira ou segunda geração) usando software atual de super-resolução (c) fornece precisão similar a um sistema estéreo de média resolução, inferior um sistema a laser de alta qualidade (d) porém de baixo custo e muito mais versátil devido ao sistema de aquisição manual e a software amplamente utilizado e desenvolvido(????). . . . .	13
Figura 3 - Protótipo do scanner a laser de triangulação. O objeto a ser escaneado é uma réplica em tamanho real de um sarcófago egípcio (a). O scanner foi reconfigurado para escanear objetos maiores, pois a escultura possui 517 centímetros (b), o da cabeça também sofreu uma reconfiguração, este scanner gira em 90 graus, que faz o laser rotacionar, da posição horizontal para a vertical e também roda em torno da cabeça como um todo (c). Para a reconstrução, o primeiro passo foi alinhar cerca de 100 scans em diversas posicoes, após isso, utilizado um alinhamento automatico em pares dos scans, utilizando um algoritmo modificado de iteracoes de pontos próximos (ICP - <i>iterated-closest-points</i> ). Após isso, faz-se um processo de relaxação global a fim de minimizar erros de alinhamento por toda a estátua. Depois de alinhados, usa-se o algoritmo de profundidade volumétrica de processamento de imagens (VRIP - <i>volumetric range image processing</i> - do Brian Curless) (d). . . . .	13
Figura 4 - Algumas esculturas do Jardim do Nêgo . . . . .	14
Figura 5 - A reconstrução usando-se apenas imagens, sem controle de aquisição, como em um vídeo de um smartphone filmado em torno do objeto, fornece uma nuvem de pontos, que pode ser densificada (????????????), ou atribuída de curvas (?????????), de forma a preservar a resolução em áreas de alto conteúdo informativo. Tais representações estão sendo atualmente unificadas na pesquisa da área. Este projeto propõe explorar os limites da reconstrução 3D usando-se apenas imagens, no contexto de preservação de patrimônio. . . . .	14

Figura 6 - Exemplo de reconstrução à laser utilizando o método baseado em volume. . . . .	17
Figura 7 - É aplicado um filtro gaussiano na imagem original (a), com $\sigma = 1$ , tendo como resultado a imagem (b). Um outro filtro gaussiano é usado, porém, neste caso, o $\sigma = 2$ (c). Após isso, subtrai-se (b) de (c), obtendo o filtro DoG (d). . . . .	20
Figura 8 - Exemplo de funcionamento de detecção de espaço-escala extrema . . . . .	21
Figura 9 - Exemplo do resultado obtido do histograma orientado . . . . .	23
Figura 10 - Exemplo de um descritor de pontos-chaves, com uma matriz 2x2 e uma região 8x8 . . . . .	23
Figura 11 - Uma triangulação utilizando um ponto qualquer, $X_j$ . Onde cada câmera $C_1, C_2, C_3$ possui um <i>feature</i> correspondente a cada uma delas, respectivamente, $X_{1j}, X_{2j}, X_{3j}$ . . . . .	24
Figura 12 - Erro proveniente da reprojeção, onde os pontos $x$ e $x'$ estão mais próximos das medidas reais da imagem. . . . .	25
Figura 13 - Funcionamento do MVE. Começando com múltiplas imagens, técnicas SfM são empregadas para reconstruir os parâmetros das câmeras e os conjuntos de pontos esparsos. Mapas de profundidade são computados para cada imagem usando o MVS. Finalmente, uma malha colorida é extraída da união de todos os mapas de profundidade usando um algoritmo de aproximações de reconstruções de superfícies (FSSR). . . . .	27
Figura 14 - Interface gráfica (UMVE) . . . . .	29
Figura 15 - Reconstrução baseada em nuvem de pontos gerada a partir da reconstrução SfM. . . . .	29
Figura 16 - Uma imagem de entrada (à esquerda) e sua correspondência em mapas de profundidade (à direita), onde a parte roxa significa que não foi encontrada nenhuma mapa naquela região. . . . .	30
Figura 17 - A figura (a) é um exemplo onde a imagem possui dados na extensão EXIF (destacado em azul). Ao passo que a figura (b) é um frame de um vídeo, que não possui os dados das câmeras (destacado em azul). . . . .	32
Figura 18 - Final da reconstrução via UMVE, percebe-se que alguns pontos não foram considerados, tendo como resultado uma "nuvem de pontos" mais densa, basicamente. . . . .	33
Figura 19 - Processos dentro do comando <i>sfmrecon</i> , onde (a) estão sendo detectadas as <i>features</i> do conjunto de imagens. Em (b) está computado o <i>pairwise matching</i> e em (c) está no processo do <i>Bundle Adjustment</i> , usando condições-padrão para as câmeras. . . . .	34
Figura 20 - Término do comando <i>sfmrecon</i> , onde demorou cerca de 1 minuto e meio (75509 milisegundos). . . . .	35

Figura 21 - Término do comando <i>dmrecon</i> , onde demorou cerca de 4 horas (14502576 milisegundos) . . . . .	35
Figura 22 - Execução dos comandos <i>scene2pet</i> , nos níveis -F0, -F1, -F2 e -F3. . . . .	36
Figura 23 - Progressão do comando <i>fssrecon</i> , onde possui o ETA – <i>Estimated Time of Arrival</i> . . . . .	36
Figura 24 - Malha com ruídos proveniente do comando <i>fssrecon</i> . . . . .	37
Figura 25 - Resultado final, após a remoção dos ruídos da malha. . . . .	37

# SUMÁRIO

	<b>INTRODUÇÃO</b>	11
0.1	<b>Introdução e Justificativa</b>	11
0.1.1	<u>O Jardim do Nêgo, Nova Friburgo</u>	14
0.2	<b>Objetivos</b>	15
0.3	<b>Organização deste manuscrito</b>	16
0.4	<b>Laser</b>	16
1	<b>PONTOS DE INTERESSE</b>	19
1.1	<b>SIFT – <i>Scale Invariant Feature Transform</i></b>	19
1.1.1	<u>Detecção de espaço-escala extremos</u>	19
1.1.2	<u>Localização de pontos-chaves</u>	21
1.1.3	<u>Atribuição de orientação</u>	22
1.1.4	<u>Descriptor de pontos-chaves</u>	23
1.1.5	<u>Combinação de pontos-chaves</u>	24
1.2	<b>Triangulação – <i>Full pairwise image matching</i></b>	24
2	<b>RECONSTRUÇÃO DENSA</b>	26
2.1	<b>Introdução</b>	26
2.2	<b>HPMVS</b>	26
2.2.1	<u>falar sobre o hpmvs...</u>	26
2.3	<b>MVE</b>	26
2.3.1	<u>Guia de reconstrução com o MVE</u>	27
3	<b>KINECT</b>	38
4	<b>VISUALSFM</b>	39
5	<b>EXPERIMENTOS</b>	40
	<b>CONCLUSÃO</b>	41

# INTRODUÇÃO

## 0.1 Introdução e Justificativa

A reconstrução 3D de cenas gerais a partir de múltiplos pontos de vista usando-se câmeras convencionais, sem aquisição controlada, é um dos grandes objetivos de pesquisa em visão computacional, ambicioso até mesmo para os dias de hoje. Aplicações incluem a reconstrução de modelos 3D para uso em videogames (??), filmes (??), arqueologia, arquitetura, modelagem 3D urbana (*e.g.*, Google Streetview); técnicas de *match-moving* em cinematografia para fusão de conteúdo virtual e filmagem real (??), a organização de uma coleção de fotografias com relação a uma cena (*e.g.*, o sistema *Phototourism* (??) e a funcionalidade *Look Around* do Google Panoramio e Street View), manipulação robótica, e a metrologia a partir de câmeras na indústria automobilística e metal-mecânica.

Os desafios estão ligados às escolhas de grande escala de representações adequadas e de técnicas que possam modelar simultaneamente com materiais drásticamente diferentes (*e.g.*, não-Lambertianos), modelos geométricos (*e.g.*, variedades curvilíneas gerais, descontinuidades, texturas, deformações, em escalas diferentes), tipos de regiões (com ou sem textura), condições de iluminação variadas, sombras, fortes diferenças de perspectivas, desbalanceamento devido a excesso de detalhes em partes menos importantes, número arbitrário de objetos e câmeras não-calibradas.

Mesmo que um sistema completo esteja fora do alcance da tecnologia atual, um progresso significativo tem sido atingido nos últimos anos. Por um lado, uma tecnologia operacional tem evoluído, mais recentemente para sistemas de grande escala (????), a partir do desenvolvimento da detecção robusta de *features* (??), o *fitting* robusto e seleção de correspondências baseados em RANSAC, e o desenvolvimento de métodos de geometria projetiva para calibrar duas ou três imagens e progressivamente adicionar imagens e extrair estrutura 3D dessas *features* na forma de nuvens de pontos. Com o código fonte do sistema Bundler (????) liberado por Noah Snavely, e sua subsequente incorporação ao sistema VisualSfM (??), é possível utilizar este sistema para a reconstrução de patrimônio.

No paradigma usando-se apenas imagens convencionais – denominado **reconstrução estéreo multiocular passiva** – a posição das câmeras são estimadas a partir apenas de imagens, usando pontos de interesse, em seguida uma nuvem de pontos é reconstruída ?. As câmeras podem então ser utilizadas para obter modelos mais detalhados de reconstrução, como algoritmos de densificação (??) e interpolação (??) da nuvem de pontos, bem como demais algoritmos densos de visão estéreo multi-perspectiva/multiocular, como os do grupo de Michel Goesele (??), também com código disponível. Tais algoritmos, no entanto, têm problemas, em particular a reconstrução suaviza partes bem-

delineadas do objeto, e pode conter buracos em áreas homogêneas. Pode-se, portanto, utilizar a reconstrução 3D de curvas do pesquisador proponente (?????????) para auxiliar na reconstrução mais bem-delinada nesses casos problemáticos, bem como para ajudar no problema de escalabilidade quando a reconstrução 3D se torna muito grande. Um segundo paradigma, denominado **reconstrução estéreo multiocular ativa**, tem se tornado viável devido à indústria de videogames, e consiste na utilização de sistemas que alteram o funcionamento de câmeras convencionais, típicamente usando-se projetores infra-vermelho, laser ou câmeras ToF (time of flight), como no caso dos dispositivos Kinect, figura 1.

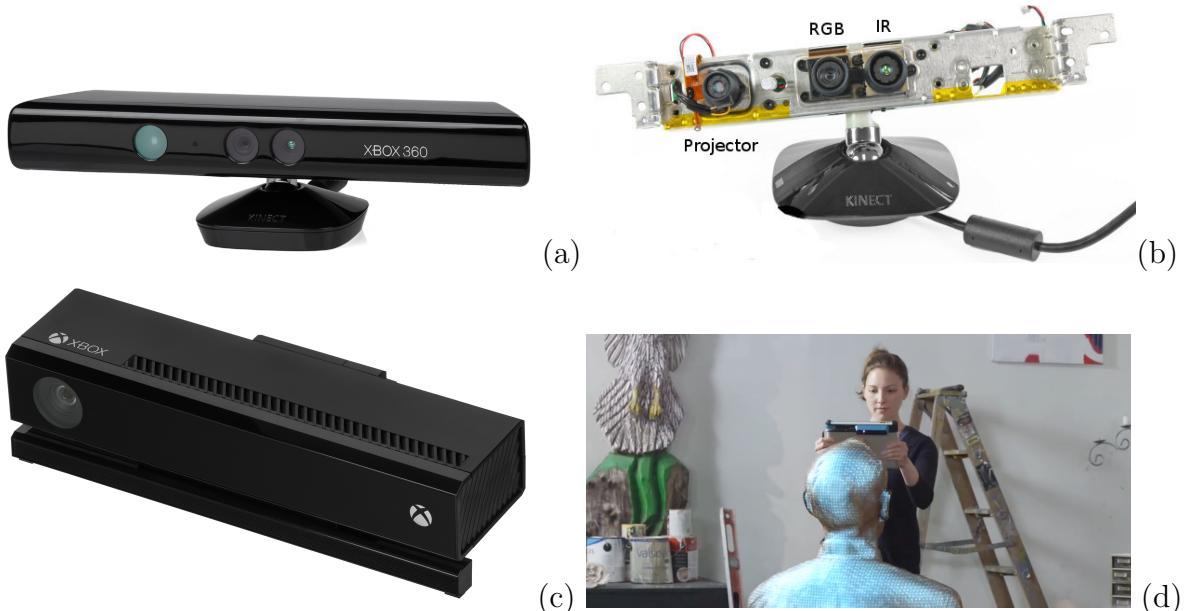


Figura 1 - Kinects de primeira geração (a) consistindo de câmeras e projetores infra-vermelho (b) e de segunda geração, consistindo de tecnologia ToF (c). Ambos os kinects são largamente utilizados para escaneamento em tempo real, formando a base de scanners manuais (d), porém nem sempre são úteis para preservação detalhada de patrimônio. Um dos objetivos deste projeto é explorar os limites desta tecnologia.

A preservação de patrimônio tem sido realizada tradicionalmente com scanners dedicados de alto custo, como no projeto David 3. O projeto teve início em 1992 e tem como objetivo a utilização de scanners a laser de profundidade (*rangefinder scanners*), aliado com algoritmos que combinam diferentes profundidades e cores da imagem, para realizar uma digitalização da parte externa e da superfície de forma acurada da estátua de David (porém, esse método pode ser utilizado em diferentes objetos no mundo real, como partes de máquinas, artefatos culturais e na indústria de video games, por exemplo). Para as partes mais detalhadas, foi utilizado um scanner de menor escala que faz uma pequena triangulação com laser de profundidade.

Seria de grande interesse explorar os dois paradigmas supracitados para avaliar as possibilidades disponíveis no estado da arte de reconstrução 3D para o escaneamento de

baixo custo para a preservação de Patrimônio. O que se pode atingir com apenas uma filmagem de esculturas realizada por um smartphone, sem calibração prévia e *in situ*, ou seja, sem ambiente controlado? Como esta reconstrução se compara nos dias de hoje com a reconstrução realizada por um scanner padrão baseado em Kinect?

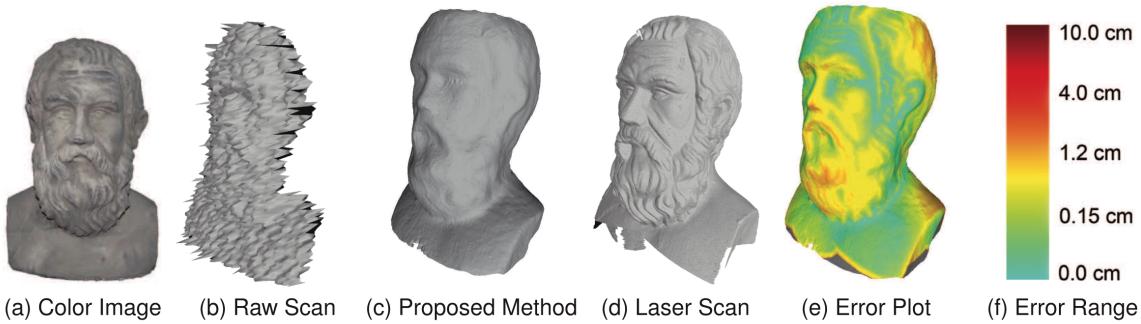


Figura 2 - A reconstrução usando-se Kinect (de primeira ou segunda geração) usando software atual de super-resolução (c) fornece precisão similar a um sistema estéreo de média resolução, inferior um sistema a laser de alta qualidade (d) porém de baixo custo e muito mais versátil devido ao sistema de aquisição manual e a software amplamente utilizado(????).

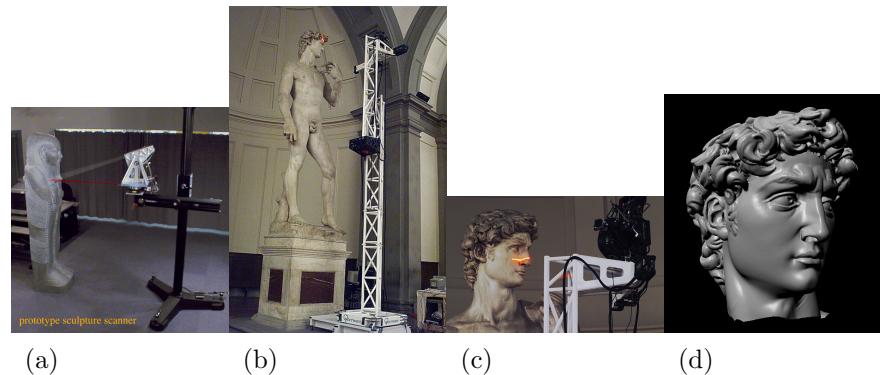


Figura 3 - Protótipo do scanner a laser de triangulação. O objeto a ser escaneado é uma réplica em tamanho real de um sarcófago egípcio (a). O scanner foi reconfigurado para escanear objetos maiores, pois a escultura possui 517 centímetros (b), o da cabeça também sofreu uma reconfiguração, este scanner gira em 90 graus, que faz o laser rotacionar, da posição horizontal para a vertical e também roda em torno da cabeça como um todo (c). Para a reconstrução, o primeiro passo foi alinhar cerca de 100 scans em diversas posicoes, após isso, utilizado um alinhamento automatico em pares dos scans, utilizando um algoritmo modificado de iteracoes de pontos próximos (ICP - *iterated-closest-points*). Após isso, faz-se um processo de relaxação global a fim de minimizar erros de alinhamento por toda a estátua. Depois de alinhados, usa-se o algoritmo de profundidade volumétrica de processamento de imagens (VRIP - *volumetric range image processing* - do Brian Curless) (d).

### 0.1.1 O Jardim do Nêgo, Nova Friburgo

No caso de Nova Friburgo, há a necessidade redobrada de preservação do patrimônio, em especial devido às chuvas e deslizamentos inerentes à região. O Jardim do Nêgo consiste em grandes esculturas em encostas, cobertas por um tapete de vegetação, as quais desfrutam de grande reconhecimento regional e internacional (??), figura 4.



Figura 4 - Algumas esculturas do Jardim do Nêgo

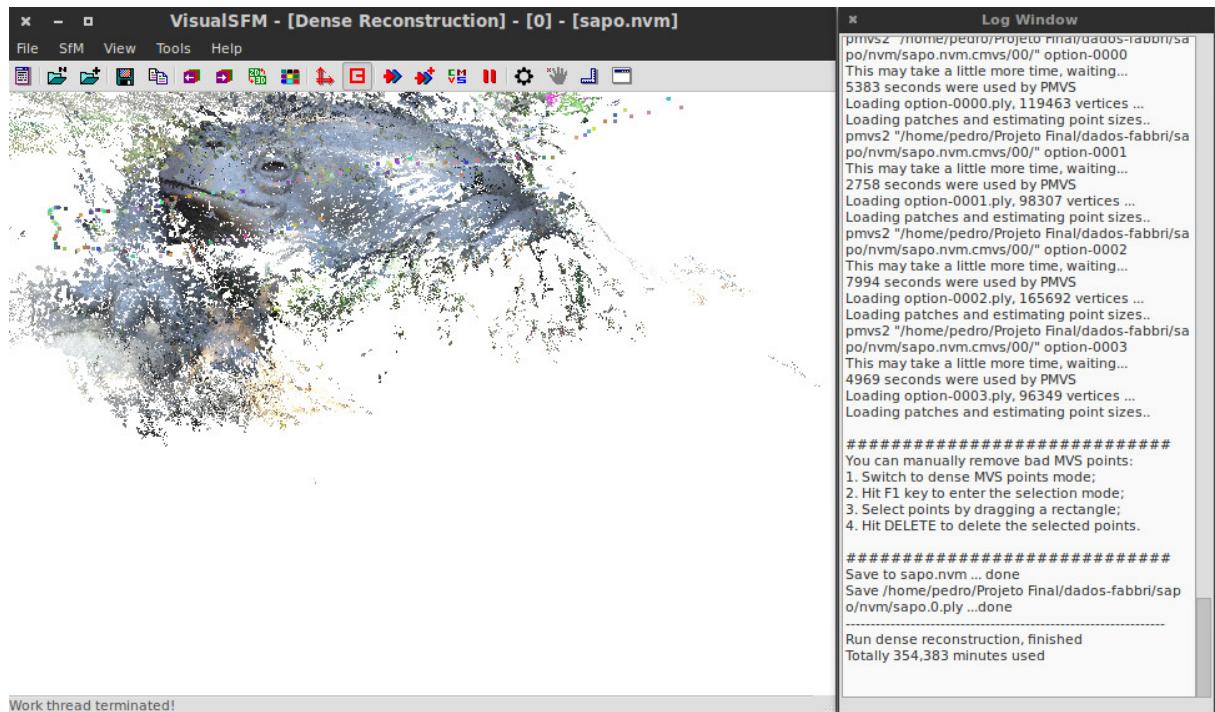


Figura 5 - A reconstrução usando-se apenas imagens, sem controle de aquisição, como em um vídeo de um smartphone filmado em torno do objeto, fornece uma nuvem de pontos, que pode ser densificada (??????????), ou atribuída de curvas (??????????), de forma a preservar a resolução em áreas de alto conteúdo informativo. Tais representações estão sendo atualmente unificadas na pesquisa da área. Este projeto propõe explorar os limites da reconstrução 3D usando-se apenas imagens, no contexto de preservação de patrimônio.

Idealizado e criado por Geraldo Simplicio (Nêgo), artista cearense que mora no

local a mais de 30 anos, ganhou notoriedade por suas esculturas de barro, com traços singulares e técnicas únicas. Hoje, trabalha para reconstruir o Jardim da tragédia de 2011 na região serrana, onde algumas estruturas foram destruídas. Portanto, com o consentimento do Nêgo, surgiu a motivação desta pesquisa: além de explorar métodos de reconstrução, também tem o objetivo de eternizar um patrimônio que é reconhecido no mundo todo.

A preservação das esculturas do Jardim do Nego se torna um desafio à pesquisa em reconstrução 3D, pois apresentam curvas bem delineadas, que são representadas de maneira suavizada e empobrecida por métodos convencionais. Algumas esculturas apresentam pouca textura, quase sem nenhum padrão de textura/musgo. Seria de grande interesse avaliar o potencial de técnicas atuais de reconstrução 3D geral sem controle de aquisição, as quais têm seu código fonte disponível na internet.

## 0.2 Objetivos

O presente projeto pretende fazer com que o aluno ganhe experiência com técnicas modernas de reconstrução 3D fotogramétrica, no contexto de uma aplicação bem-definida de preservação de patrimônio. A entrada do sistema deverá ser um conjunto de vídeos realizados por câmeras de baixo custo, ou um conjunto de escaneamentos realizados por scanners à mão de baixo custo baseados em Kinect.

O objetivo concreto do aluno será explorar as tecnologias supracitadas para desenvolver um esquema de escaneamento usando software aberto, câmeras e scanners de baixo custo, representando o estado da arte em reconstrução 3D sem restrições de aquisição. Perguntas fundamentais a serem respondidas são: que nível de detalhe, facilidade e precisão se pode obter usando-se apenas imagens e software aberto? É possível utilizar scanners de baixo custo baseados em Kinect com melhorias significativas em termos de qualidade, conveniência ou tempo de processamento? Quais são as restrições desses sistemas? Seria útil na prática uma reconstrução de curvas para auxiliar na reconstrução de nuvem de pontos e de superfícies densas? Onde o estado da arte deve ser avançado de forma a permitir uma solução mais conveniente e completa para a preservação de patrimônio?

O principal objetivo em termos de pesquisa científica será comparar as diferentes abordagens do estado da arte disponíveis para reconstrução 3D e explicitar suas limitações práticas. O aluno deverá, com o entendimento das abordagens, desenvolver um esquema de aquisição de esculturas que permita ampliar os detalhes ou ajudar a resolver os problemas dos métodos. Com a experiência obtida, o aluno estará pronto para desenvolver pesquisa futura na área de reconstrução 3D, com conhecimento de causa para avaliar direções de pesquisa de efetivo e alto impacto na prática.

### 0.3 Organização deste manuscrito

O trabalho foi estruturado da seguinte maneira: previamente introduzimos os métodos baseados em pontos de interesse no Capítulo 1, destacando suas funcionalidades. No Capítulo 2 discutimos e aprofundamos o funcionamento de cada algoritmo de reconstrução densa empregados, apresentando e debatendo, comparativamente, pontos à favor e contra; O Capítulo 3 é apresentada a técnica de reconstrução utilizada nos *Kinects*, da Microsoft; Com isso, temos o Capítulo 4, que é dedicado à ferramenta gráfica utilizada para a obtenção dos resultados (VisualSfM) dos algoritmos de reconstrução densa utilizados. Finalmente, apresentamos os resultados e conclusões do trabalho, bem como sugestões para implementações e trabalhos futuros.

### 0.4 Laser

A técnica de reconstrução baseada em laser é conhecida desde o século passado, pois oferecem uma alta qualidade geométrica de dados, os resultados são em tempo real e requer pouco tempo de captura de dados.

Existem alguns tipos de reconstruções empregando lasers, baseados em volume (ressonância, tomografia, por exemplo 6 ou em superfície (estereoscopia, *time of flight*, luz estruturada) . Neste caso, abordaremos o projeto de escaneamento da escultura de Michelangelo, David, que utiliza escaners baseados em superfícies, mais especificamente, utilizando luz estruturada.

Este projeto tem como motivação avançar a tecnologia de digitalização 3D e criar um acervo digital sobre alguns dos principais artefatos culturais. Foi utilizada uma tecnologia chamada de *rangefinder*, com uma equipe de mais de 30 professores, funcionários e estudantes da Universidade de Stanford desenvolveram algoritmos para combinar imagens de múltiplas gamas e cores, passaram os anos de 1998 e 1999 na Itália escaneando esculpturas de Michelangelo.

O projeto não teve mais nenhum avanço desde o verão de 2004, por falta de financiamento. Como resultado, modelos de alta qualidade só existem do David na resolução de 1,0 mm (56 milhões de triângulos) e São Mateus a 0,25 mm (372 milhões de triângulos). Um modelo também existe para o Atlas em 0,25 mm (aproximadamente 500 milhões de triângulos), mas contém erros de alinhamento. Após 6 anos de trabalho estudantil remunerado e voluntário, existem modelos para cada um dos 1.186 fragmentos. Esses modelos, que totalizam quase 8 bilhões de polígonos, se encontram no próprio site da Universidade de Stanford. Também foram disponibilizadas algumas métricas sobre este projeto 1.

Porém, devido ao seu alto custo com equipamentos, com *softwares* e sem falar na necessidade de estações robustas para armazenamento dos dados e para escaneamento de



Figura 6 - Exemplo de reconstrução à laser utilizando o método baseado em volume.

parimônios, outras técnicas foram emergindo com o passar dos anos, como a fotogrametria, que será abordada posteriormente.

Tabela 1 - Algumas métricas do projeto

Números de objetos escaneados	10 estátuas + 2 edificações + 1,163 fragmentos de mapa
Menor e maior objetos escaneados	1 polegada (fragmentos de mapa) e 23 pés (David)
Resolução espacial dos dados	0.29mm para geometria, 0.125mm para cor
Complexidade do maior conjunto de dados	2 bilhões de polígonos + 7,000 imagens (David)
Tamanho do maior conjunto de dados	32 <i>gigabytes</i> (David)
Quantia total de dados capturados	250 <i>gigabytes</i>
Tamanho do maior escaner	24 pés de altura, 1800 libras de peso
Peso total do equipamento levado para a Itália	4 toneladas
Número de pessoas envolvidas	32 (sem incluir subcontratantes e colaboradores)
Tempo médio para escaneamento	1 semana (exceto o David, que levou 1 mês)
Tempo total de escaneamento	5,000 horas de trabalho
Total de tempo para processamento de dados	4,000 horas de trabalho (até agora)
Custo do projeto	\$2,000,000

## 1 PONTOS DE INTERESSE

A reconstrução 3D pelo método da estrutura do movimento, ou *Structure from Motion (SfM)*. Tem como base a utilização de pontos de interesse (*features*), que são pontos ou áreas em comum entre as imagens usadas na reconstrução. Para encontrar estes pontos, diferentes algoritmos são empregados.

### 1.1 SIFT – *Scale Invariant Feature Transform*

Primeiramente, utiliza-se o SIFT (algoritmo de detecção de pontos de interesse, invariante à escala e à transformações, como rotação, translação e iluminação da imagem, por exemplo). O algoritmo pode ser dividido em cinco etapas, das quais:

- Detecção de espaço-escala extremos – *Scale-space Extrema Detection*
- Localização de pontos-chaves – *Keypoint Localization*
- Atribuição de orientação – *Orientation Assignment*
- Descritor de pontos-chaves – *Keypoint Descriptor*
- Combinação de pontos-chaves – *Keypoint Matching*

#### 1.1.1 Detecção de espaço-escala extremos

Em casos com cantos pequenos, a detecção funciona bem. Porém, raramente utilizaremos a mesma janela para detectar pontos-chaves em imagens com diferentes escalas, pois utilizamos imagens grandes e, consequentemente, cantos grandes. Para isso, precisamos de janelas grandes também.

Para resolver este problema, o filtro de escala-espacó é usado: o Laplaciano de Gaussiano (*Laplacian of Gaussian* – LoG). O LoG atua como um detector de partículas em diferentes tamanhos  $\sigma$ . (Onde  $\sigma$  é o parâmetro de escala). Por exemplo, o núcleo gaussiano com  $\sigma$  baixo, tem como resposta um alto valor para um canto pequeno. Enquanto um núcleo gaussiano com alto  $\sigma$ , se encaixa bem para um canto maior. Com esta lógica, podemos encontrar um máximo local através da escala e o espaço, o que nos fornece uma lista de  $(x, y\sigma)$ , o que significa que existe um ponto-chave em potencial, com o par  $(x, y)$  na escala  $\sigma$ .

Porém, como o LoG é um pouco custoso, computacionalmente. O SIFT utiliza um algoritmo aproximado do LoG, o DoG (Diferença de Gaussianos – *Difference of Gaussians*). O DoG é a diferença de um filtro Gaussiano de uma imagem, com dois valores diferentes de escala  $\sigma$ .

Uma aplicação prática do filtro DoG é a sequência de imagens a seguir:

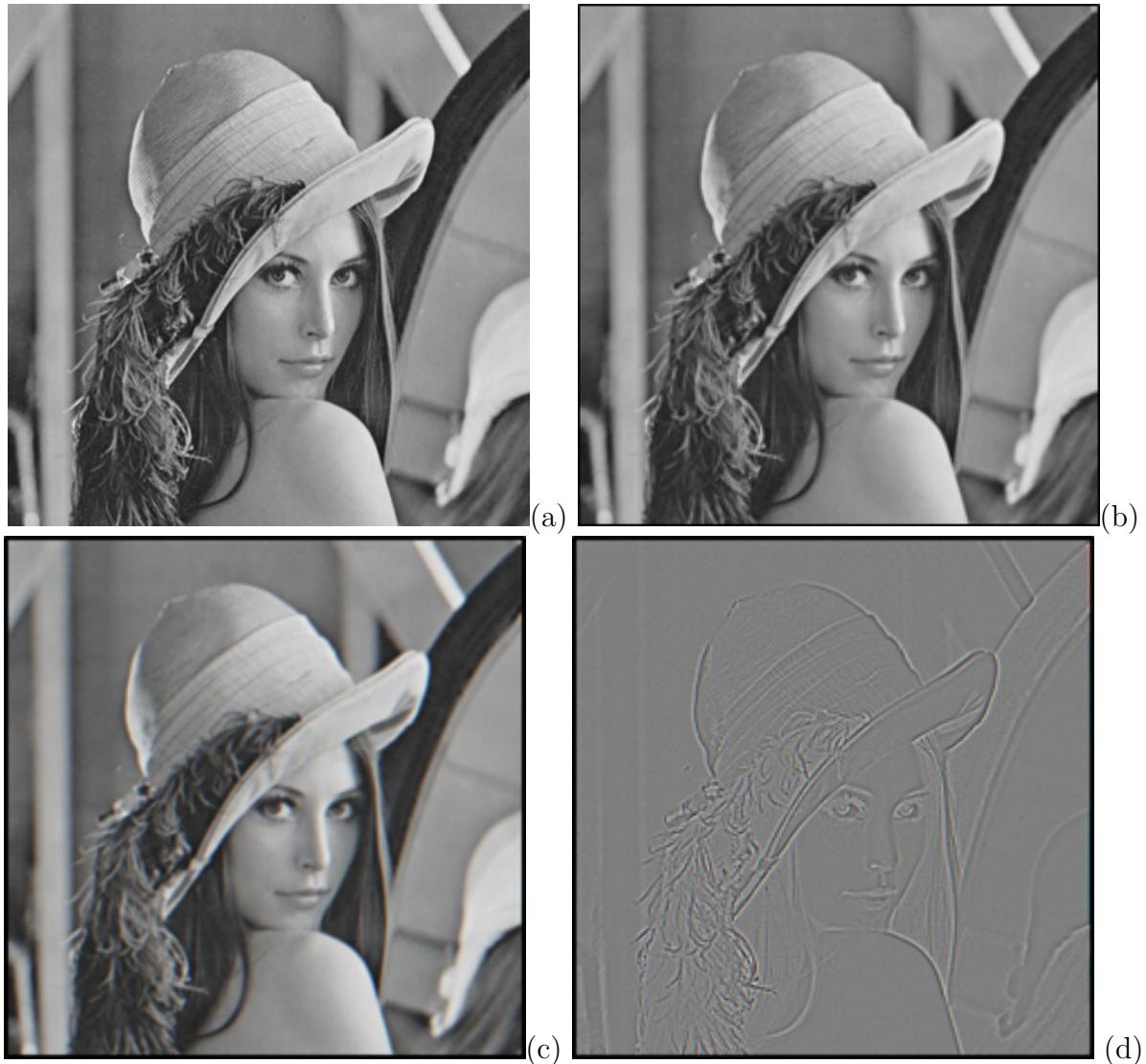


Figura 7 - É aplicado um filtro gaussiano na imagem original (a), com  $\sigma = 1$ , tendo como resultado a imagem (b). Um outro filtro gaussiano é usado, porém, neste caso, o  $\sigma = 2$  (c). Após isso, subtrai-se (b) de (c), obtendo o filtro DoG (d).

Uma vez que o DoG é aplicado, as imagens são utilizadas com o espaço e escala extremos. Por exemplo, na imagem 8, um pixel é comparado com seus 8 vizinhos, assim como comparado com os 9 pixels na próxima escala e os 9 pixels na escala anterior. Se esse pixel é um local extremo, ele é um ponto-chave em potencial. Isto é, este ponto-chave é melhor representado nesta escala.

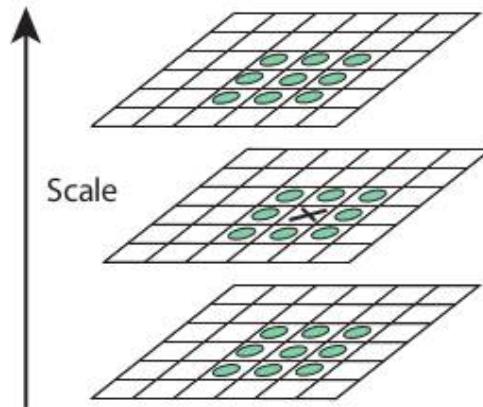


Figura 8 - Exemplo de funcionamento de detecção de espaço-escala extrema

No caso, as funções ficariam da seguinte forma:

$$g_\sigma(x) = \frac{1}{2\pi\sigma^2} e^{-\frac{1}{2} \frac{x^T x}{\sigma^2}} \quad (1)$$

$$I_\sigma = g_\sigma * I, \sigma \geq 0 \quad (2)$$

$$\nabla^2 g_\sigma(x) \quad (3)$$

$$DoG_\sigma(o, s) = I_\sigma(o, s + 1) - I_\sigma(o, s) \quad (4)$$

Onde 1 é a função padrão do operador gaussiano (núcleo), a equação 3 é o operador LoG, 4 é o operador DoG e  $\nabla^2$  é o operador Laplaciano.

### 1.1.2 Localização de pontos-chaves

A localização dos pontos extremos pode cair em um extremo local e não global. Logo, após a utilização do DoG e com os pontos-chaves em potencial localizados, eles precisam ser refinados para melhorar o resultado. Para isso, são utilizadas Séries de Taylor na escala e no espaço, e, se a intensidade nesse extremo é menor que o valor limite, este é rejeitado. Os *frames* do SIFT (pontos-chaves) são extraídos baseados nos

extremos locais (picos) a partir do DoG. Numericamente, extremos locais são elementos que possuem um menor (ou maior) valor em uma vizinhança em um espaço 3x3x3 (em escala e espaço). Depois de extraídos, estes pontos são interpolados quadraticamente (este passo é muito importante, especialmente nas escalas de menor resolução, para ter uma localização precisa do ponto-chave na resolução completa). Finalmente, eles são filtrados para eliminar respostas de baixo contraste ou respostas próximas as bordas.

Picos que são pequenos, na maior parte das vezes são gerados a partir de ruídos e necessitam ser descartados também. Isso é feito com uma comparação de valor absoluto do DoG no pico com o valor do pico limite e é descartado caso este valor é menor que o limite.

Para eliminar respostas em bordas, normalmente os picos mais rasos ou horizontais, são gerados por bordas e não possuem características estáveis, portanto estes picos precisam ser removidos. Para isso, dado um pico  $(x, y, \sigma)$ , o algoritmo avalia a matriz Hessiana  $(x, y)$  do DoG na escala  $\sigma$ . Então é computado um valor para esta equação (5):

$$v = \frac{(T_r D(x, y, \sigma))^2}{\text{Det } D(x, y, \sigma)} \quad (5)$$

Onde,  $T_r$  é o traço, ou seja,  $T_r(H) = D_{xx} + D_{yy}$  e a matriz  $D$  é do tipo

$$D = \begin{bmatrix} \frac{\partial^2 \text{DoG}}{\partial x^2} & \frac{\partial^2 \text{DoG}}{\partial x \partial y} \\ \frac{\partial^2 \text{DoG}}{\partial x \partial y} & \frac{\partial^2 \text{DoG}}{\partial y^2} \end{bmatrix}$$

No caso,  $v$  possui um valor mínimo (igual a 4) quando os autovalores da Jacobiana são iguais (pico curvado) e aumentam à medida que um dos autovalores aumenta e os outros permanecem baixos. Os picos são retidos se  $v < \frac{(t_e+1)(t_e+1)}{t_e}$ , onde  $t_e$  é o limite da borda.

### 1.1.3 Atribuição de orientação

Agora, uma orientação é atribuída a cada ponto-chave para obter a invariância à rotação da imagem. Uma vizinhança é obtida, dependente da escala, do gradiente da magnitude 6 e da direção 7 (usando diferenças finitas), ao redor da localização do ponto-chave.

$$m(x, y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2} \quad (6)$$

$$\theta = \tan^{-1} \left( \frac{(L(x, y + 1) - L(x, y - 1))}{(L(x + 1, y) - L(x - 1, y))} \right) \quad (7)$$

Então, um histograma de 36 orientações (*bins*) cobrindo 360 graus é criado. Onde ele é ponderado pelo gradiente da magnitude e por uma janela Gaussiana circular onde  $\sigma$  vale 1.5 em relação à escala do ponto-chave 9.

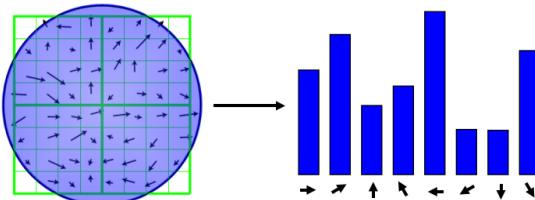


Figura 9 - Exemplo do resultado obtido do histograma orientado

O ponto mais do histograma é obtido e qualquer pico acima de 80% é considerado no cálculo da orientação. Pontos-chaves são criados com a mesma localização e escala, mas em diferentes direções, o que contribui para a estabilidade da correspondência.

#### 1.1.4 Descriptor de pontos-chaves

Com os pontos-chaves criados a partir do histograma orientado, cria-se agora o descriptor de pontos-chaves.

Uma vizinhança 16x16 ao redor do ponto-chave é escolhida e esta mesma vizinhança é dividida em 16 sub-blocos 4x4. Para cada bloco, um histograma orientado com 8 *bin* é criado. Logo, temos 128 valores válidos de *bin*. Esses valores são representados em forma de vetor para expressar o descriptor de pontos-chaves 10.

Além disso, são tomadas algumas medidas para deixar o descriptor mais robusto, como, por exemplo, invariante à luminosidade, rotação, etc.

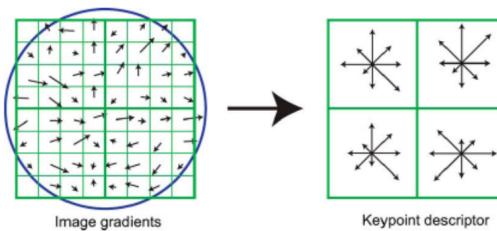


Figura 10 - Exemplo de um descriptor de pontos-chaves, com uma matriz 2x2 e uma região 8x8

### 1.1.5 Combinação de pontos-chaves

Pontos-chaves entre duas imagens são combinados a partir da identificação da vizinhança mais próxima. Mas, em alguns casos, a segunda combinação mais próxima pode ser parecida com a primeira. Isso se dá por ruídos presentes nas imagens ou algo assim.

Nesse caso, a razão da distância mais próxima para a segunda distância mais próxima é utilizada. Se essa razão for maior que 0.8, essa combinação é descartada. Esse método elimina cerca de 90% de combinações falsas, enquanto descarta apenas cerca de 5% de combinações corretas.

## 1.2 Triangulação – *Full pairwise image matching*

Com os pontos de interesse (*features*) extraídos, podemos agora fazer a triangulação entre os pontos das imagens.

A triangulação nada mais é que uma estimativa de um ponto em 3 dimensões, dado pelo menos duas câmeras conhecidas, onde, cada câmera com a projeção do *feature* correspondente àquele ponto 3D 11.

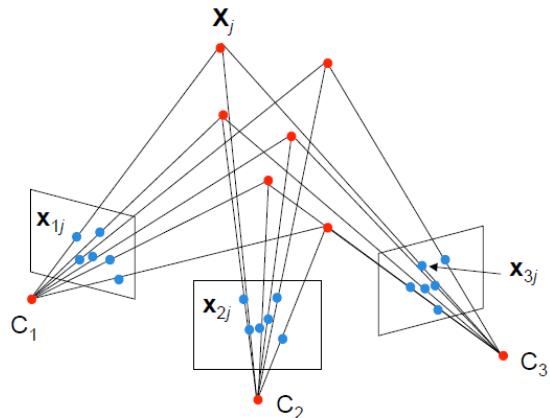


Figura 11 - Uma triangulação utilizando um ponto qualquer,  $X_j$ . Onde cada câmera  $C_1, C_2, C_3$  possui um *feature* correspondente a cada uma delas, respectivamente,  $X_{1j}, X_{2j}, X_{3j}$ .

Infelizmente, não é tão simples assim. Existem muitos fatores que contribuem para aumentar a dificuldade da triangulação: ruídos, posição das câmeras, o feixe das projeções não se encontram no mesmo ponto 3D, não se tem informação da projeções nas câmeras, etc. Entretanto, existem diversos algoritmos para resolução de cada um dos problemas enfrentados.

Como o foco deste manuscrito está no *software* de reconstrução VisualSfM, atentamos apenas ao algoritmo utilizado pelo mesmo. Neste caso, ele utiliza um algoritmo

chamado *Bundle Adjustment*.

Este algoritmo é utilizado para minimizar o erro geométrico proveniente da reprojeção da triangulação 12. Dentro dele, existe a técnica de resolução de problemas não-lineares chamada Levenberg-Marquardt.

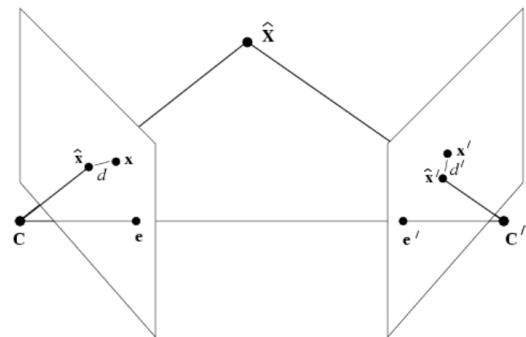


Figura 12 - Erro proveniente da reprojeção, onde os pontos  $x$  e  $x'$  estão mais próximos das medidas reais da imagem.

## 2 RECONSTRUÇÃO DENSA

### 2.1 Introdução

### 2.2 HPMVS

#### 2.2.1 falar sobre o hpmvs...

### 2.3 MVE

Um dos algoritmos utilizados para a técnica de reconstrução densa é o MVE – *Multi-View Environment*, feito por Simon Fuhrmann, Fabian Langguth e Michael Goesele. Este algoritmo utiliza fotos e produz uma malha triangular superficial como resultado. Diferente das reconstruções baseadas nas geometria das imagens, o MVE é focado na reconstrução multi-escala, um quesito importante na reconstrução de esculturas e acervo cultural. Portanto, com esta técnica é possível reconstruir grandes volumes de dados, contendo regiões detalhadas em alta resolução, em comparação com o resto da cena. O sistema ainda possui uma interface gráfica para o uma reconstrução baseada no SfM, amigável ao usuário (UMVE), onde permite a visualização e inspeção das imagens, mapas de profundidade e renderizar cenas e malhas 3D.

Sua base de operação é basicamente 13:

#### 1. Estrutura da formação – *Structure-from-Motion* (SfM)

- Reconstrói os parâmetros da câmera (posição e orientação) e seus dados de calibração (distância focal e distorção radial), encontrando correspondências esparsas mas estáveis entre as imagens. (Já foi abordado em outra seção deste manuscrito).

#### 2. Múltiplas visões estéreo – *Multi-View Stereo* (MVS)

- Utiliza a posição estimada das câmeras, encontrando as correspondências visuais nas imagens. Estas correspondências são trianguladas, produzindo a informação 3D, e, consequentemente a reconstrução 3D densa.

#### 3. Reconstrução de superfícies – Surface Reconstruction

- Tem como entrada uma densa nuvem de pontos, ou mapas de profundidade individuais. Produz uma malha superficial globalmente consistente.

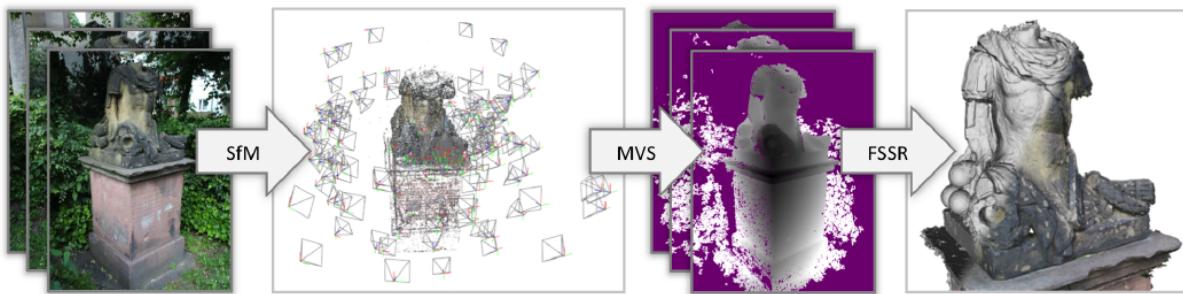


Figura 13 - Funcionamento do MVE. Começando com múltiplas imagens, técnicas SfM são empregadas para reconstruir os parâmetros das câmeras e os conjuntos de pontos esparsos. Mapas de profundidade são computados para cada imagem usando o MVS. Finalmente, uma malha colorida é extraída da união de todos os mapas de profundidade usando um algoritmo de aproximações de reconstruções de superfícies (FSSR).

Como não existem muitas opções para algoritmos de SfM, o MVE permite a utilização de *softwares* externos como o *Bundler* ou o próprio *VisualSfM*.

Uma vez com o passo do SfM feito, partimos para o MVS. Com os parâmetros de câmera conhecidos, a reconstrução densa geométrica é feita. Existem diversos algoritmos para a reconstrução densa, o MVE no caso, utiliza um algoritmo próprio, feito por um de seus criadores, Michael Goesele (*Multi-View Stereo for Community Photo Collections approach*), que reconstrói um mapa de profundidade para cada foto.

Embora abordagens baseadas em mapeamentos de profundidade produzirem uma grande quantidade de redundância, (isso se dá por causa das inúmeras fotos que são sobrepostas e possuírem partes similares da mesma cena), este algoritmo é altamente escalável para grandes cenas, pois apenas um pequeno conjunto de fotos vizinhas é necessário para a reconstrução. Outra vantagem da utilização dos mapas de profundidade como representação intermediária é que a geometria é parametrizada em seu domínio natural, e os dados por foto (como a cor, por exemplo) estão diretamente acessíveis nas imagens.

A redundância excessiva nos mapas de profundidade pode ser pesado. Não com relação ao armazenamento, mas na questão do processamento computacional exigido nos mapas de profundidade. Porém, esta abordagem foi capaz de produzir uma geometria detalhada e superar o ruído nos mapas de profundidades individuais.

### 2.3.1 Guia de reconstrução com o MVE

Tirando fotos: Um bom conjunto de dados é gerado se algumas regras simples forem seguidas:

- Para que o algoritmo do MVS consiga fazer uma triangulação com qualquer posição 3D, o conjunto de dados terá que ter, no mínimo, cinco fotos.
- As fotos devem ser tiradas com uma boa quantidade de sobreposição. A menos que o conjunto de dados se torne muito grande, uma grande quantidade de fotos não prejudicará a qualidade. Mas terá uma compensação do sistema, no que diz respeito à qualidade e desempenho.
- Para a triangulação funcionar, é necessário que tenha o efeito de paralaxe ?? (Aparente mudança na posição do objeto). Ou seja, é interessante que o conjunto de imagens seja duplicado.
- A câmera deverá ser reposicionada, de preferência.

Criando uma cena: Uma visualização contém dados por exibição (como imagens, mapas de profundidade ou outros dados). Uma cena é uma coleção de visualizações, que constitui um conjunto de dados. Uma nova cena pode ser criada utilizando a interface gráfica UMVE, ou por linha de comando (*makescene*).

Tecnicamente, a cena é criada como um diretório no sistema de arquivos (com o nome do conjunto de dados). Este, por sua vez, contém outro diretório (*views*), com todas as visualizações guardadas com uma extensão de arquivos em .MVE.

Criar uma nova cena, criará apenas o diretório (*views*) vazio. A importação de fotos criará arquivos .MVE para cada foto. Esse processo importará meta-dados provenientes das imagens (*tags EXIF*), que é necessário para estimar a distância focal para cada foto. Caso estes meta-dados não estejam disponíveis, uma distância focal padrão é assumida pelo sistema, porém se essa distância adotada for uma péssima suposição, com relação ao conjunto de dados utilizado, pode vir a acontecer erros no SfM.

Reconstrução SfM: Pode ser configurada e iniciada usando a interface gráfica UMVE ou por linha de comando (*sfmrecon*). A interface guia através da detecção de *features*, combinação emparelhada (*pairwise matching*) e uso incremental do SfM. Que, por sua vez, a reconstrução SfM começa a partir de um par inicial, e adiciona, de forma incremental, mais vistas à reconstrução [15](#).

Múltiplas visões estéreo – *Multi-View Stereo* (MVS): Usando as imagens junto com os parâmetros obtidos das câmeras, é possível reconstruir a geometria densa utilizando o MVS. Isso pode ser feito utilizando a interface gráfica (UMVE) ou por linha de comando (*dmrecon*).

O parâmetro mais importante é o nível de resolução em que os mapas de profundidade são reconstruídos: Caso seja nível 0 (ou L0), a reconstrução é feita usando o tamanho original das imagens. Se for nível 1 (ou L1), a reconstrução corresponde a metade do tamanho (um quarto dos números de pixels), e assim por diante.

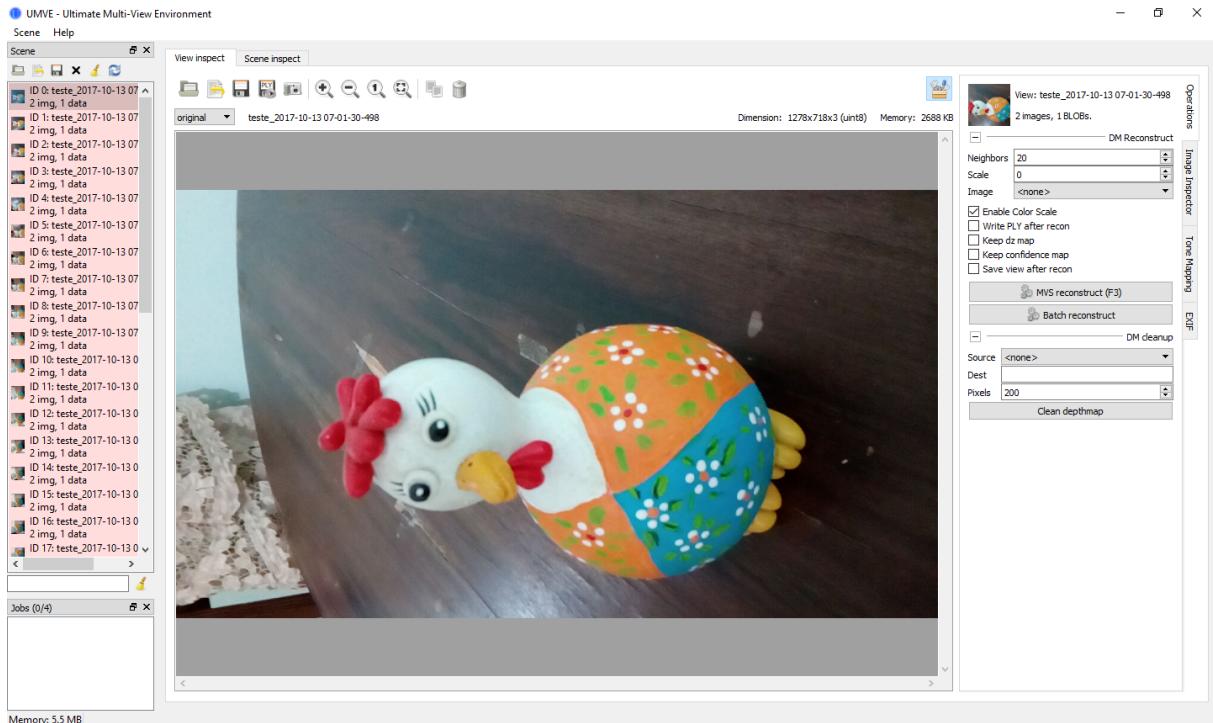


Figura 14 - Interface gráfica (UMVE)

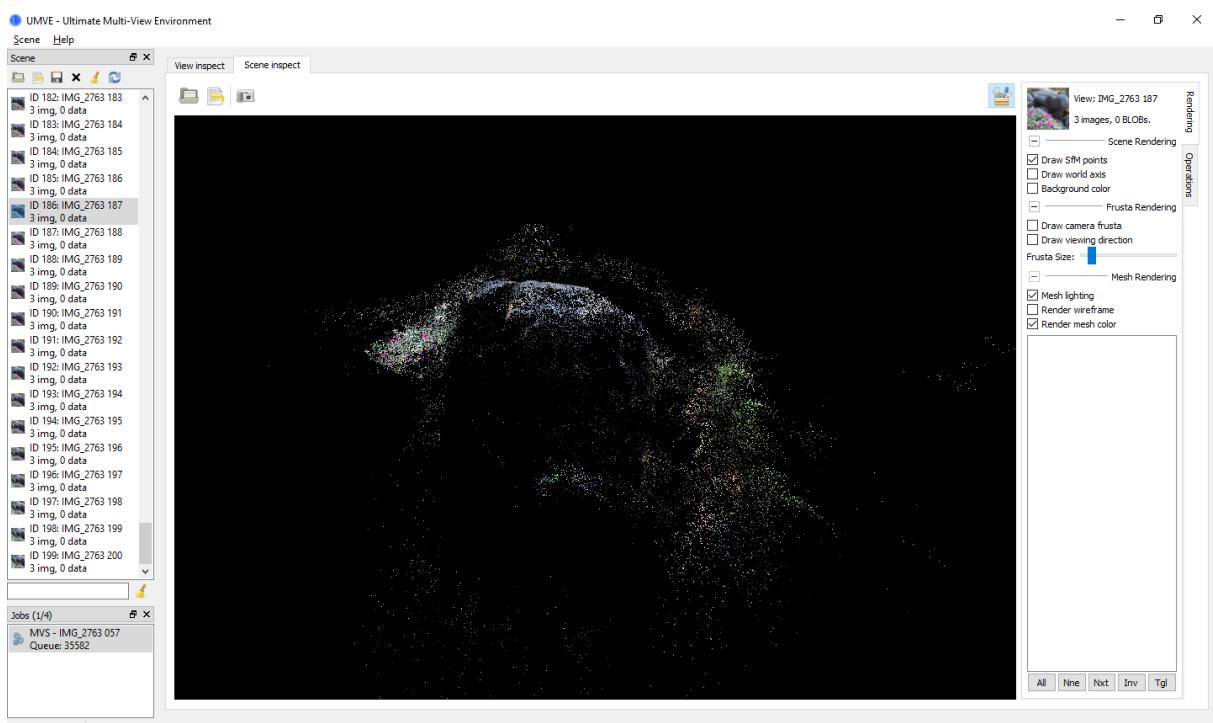


Figura 15 - Reconstrução baseada em nuvem de pontos gerada a partir da reconstrução SfM.

Com a resolução das câmeras atuais, uma reconstrução L0 é raramente usada, pois geram mapas de profundidade mais dispersos com um custo computacional elevado, o que acarreta em dificuldades para encontrar as correspondências densas das imagens.

Geralmente utiliza-se o L2, pois o processo é mais rápido, gerando mapas de profundidades completos, já que utiliza imagens menores [16](#).



Figura 16 - Uma imagem de entrada (à esquerda) e sua correspondência em mapas de profundidade (à direita), onde a parte roxa significa que não foi encontrada nenhum mapa naquela região.

Reconstrução de superfícies – Surface Reconstruction: Utiliza-se a linha de comando *scene2pet*, que combina todos os mapas de profundidade em uma única e grande nuvem de pontos. Nesta fase, um valor de escala é atribuído a cada ponto, que indica o tamanho atual da região da superfície na qual o ponto foi mensurado. Esta informação adicional permite o uso de várias propriedades benéficas usando a abordagem de reconstrução da superfície FSSR. A seguir, as ferramentas FSSR calculam uma representação volumétrica de escala múltipla a partir dos pontos (na qual não precisa de nenhum ajuste de parâmetros explícitos) e uma malha final é extraída. Esta malha pode parecer desordenada devido à regiões não confiáveis e à componentes isolados, oriundos de medidas imprecisas. Logo, a malha é limpa, retirando pequenos componentes isolados e regiões não confiáveis da superfície.

Experiência com o MVE: A utilização do *software* é bem intuitiva, seja por linha de comando ou pela interface gráfica (neste modo, fica mais fácil visualizar cada etapa da reconstrução). Amplamente configurável, podendo escolher a vizinhança, escala, manter o mapa de profundidade, ver os dados *EXIF* de cada imagem, dentre outras configurações.

Entretanto, para a aplicação proposta neste projeto, não é muito interessante, visto

que ele utiliza a informação das câmeras (*EXIF*) e das imagens e, como as imagens empregadas na reconstrução são, tecnicamente, vídeos cortados em determinados *frames*, não é possível obter a informação das câmeras [17](#), logo o *software* não tem tanta aplicabilidade neste caso, pois recai no problema dos parâmetros padrões adotados para as câmeras não serem bons o suficiente para estes conjuntos de dados, a menos que sejam tiradas fotos sequenciais de alguma escultura ou objeto que se deseja gerar a reconstrução densa, pois dessa forma, as informações necessárias das câmeras estarão armazenadas.

Com isso em mente, foi gerada uma reconstrução de um vídeo gravado de uma escultura no Jardim do Nêgo. O vídeo foi cortado em *frames* onde foram geradas 200 imagens base.

A partir disso, foi executado, todos os passos de uma reconstrução utilizando o MVE, de forma que, foram utilizadas as duas opções, tanto por linha de comando, quanto pela interface gráfica (UMVE).

Pela interface gráfica, o processo todo de reconstrução foi rápido (cerca de 30 minutos) [18](#), ao passo que por linha de comando, levou cerca de 11 horas e 30 minutos.

O UMVE não sinaliza quando o processo em execução termina, então, a explicação para essa discrepância no tempo é devido à execução de outro comando, sobrepondo o que já estava sendo executado, sem deixar terminá-lo.

Na reconstrução por linha de comando também é possível visualizar em qual etapa da execução o algoritmo está [19](#), configurar alguns parâmetros e inclusive mostrar a porcentagem de progresso do comando. Foram executados os comandos declarados nesta seção. O *sfmrecon* demorou cerca de 1 minuto e meio [20](#).

O próximo comando, *dmrecon* demorou cerca de 4 horas, usando como configuração um nível L2, com 20 vizinhos [21](#). Usando um nível L0, o algoritmo rodou durante 6 horas aproximadamente e foi cancelado devido à demora na execução.

Usando o *scene2pet*, é necessário em qual nível estamos reconstruindo e também uma saída válida. Por exemplo: "scene2pset.exe -Fnivel cena output". Onde o nível poderá ser um 0 (-F0), 1 (-F1) e assim por diante, a cena é o *input* e o *output* é um arquivo de extensão configurável, neste caso *.ply* [22](#). Este comando foi rápido, demorou cerca de 10 minutos, levando em conta todos os níveis.

Para juntar todos os níveis do *scene2pet*, foi usado o *fssrecon*, que gera uma única reconstrução. Este processo demorou bastante, cerca de 7 horas [23](#). Que teve como resultado a malha [24](#).

Finalmente, basta limpar a malha atual com o comando *meshclean*, onde foi obtido o resultado [??](#).

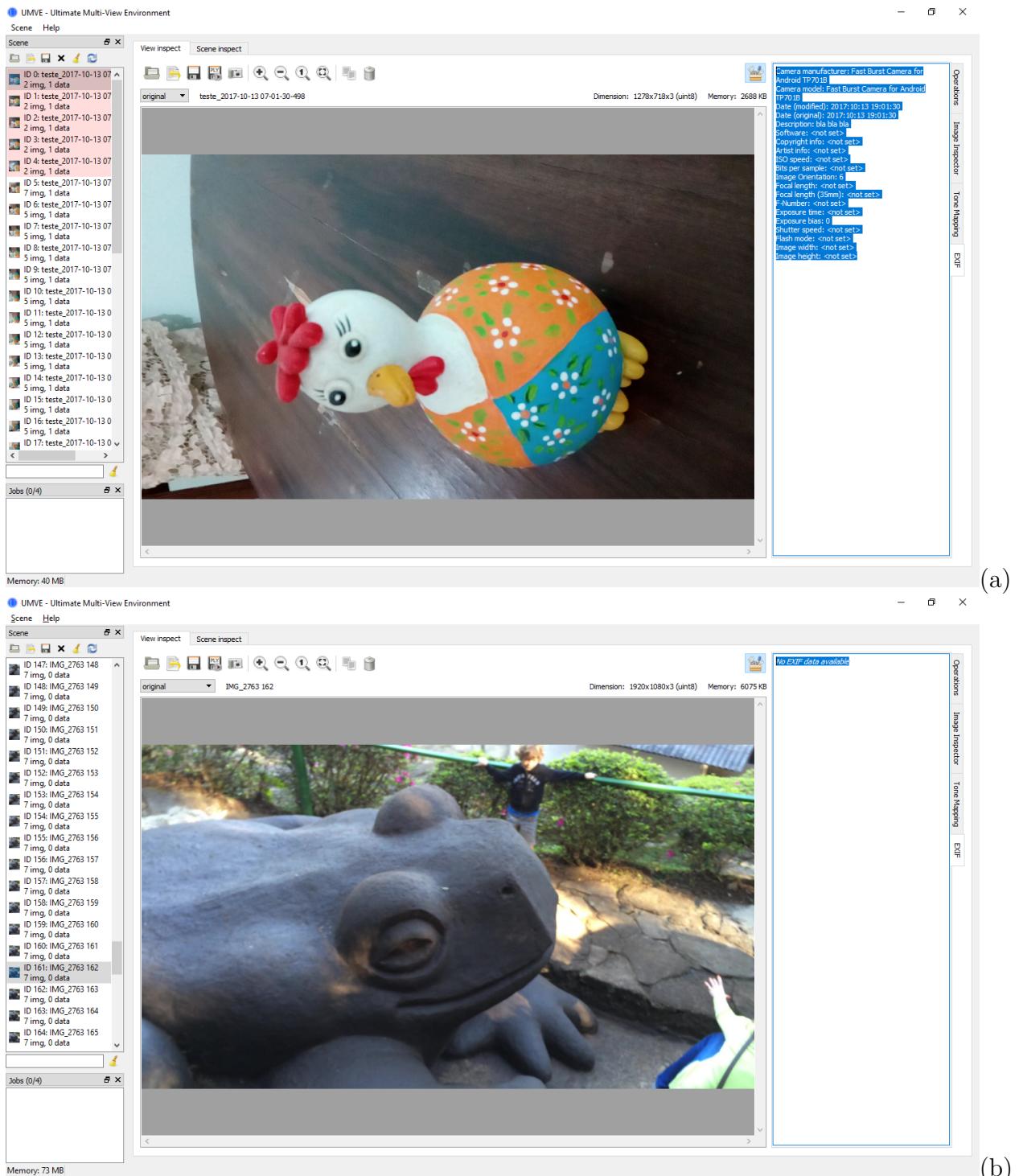


Figura 17 - A figura (a) é um exemplo onde a imagem possui dados na extensão *EXIF* (destacado em azul). Ao passo que a figura (b) é um frame de um vídeo, que não possui os dados das câmeras (destacado em azul).

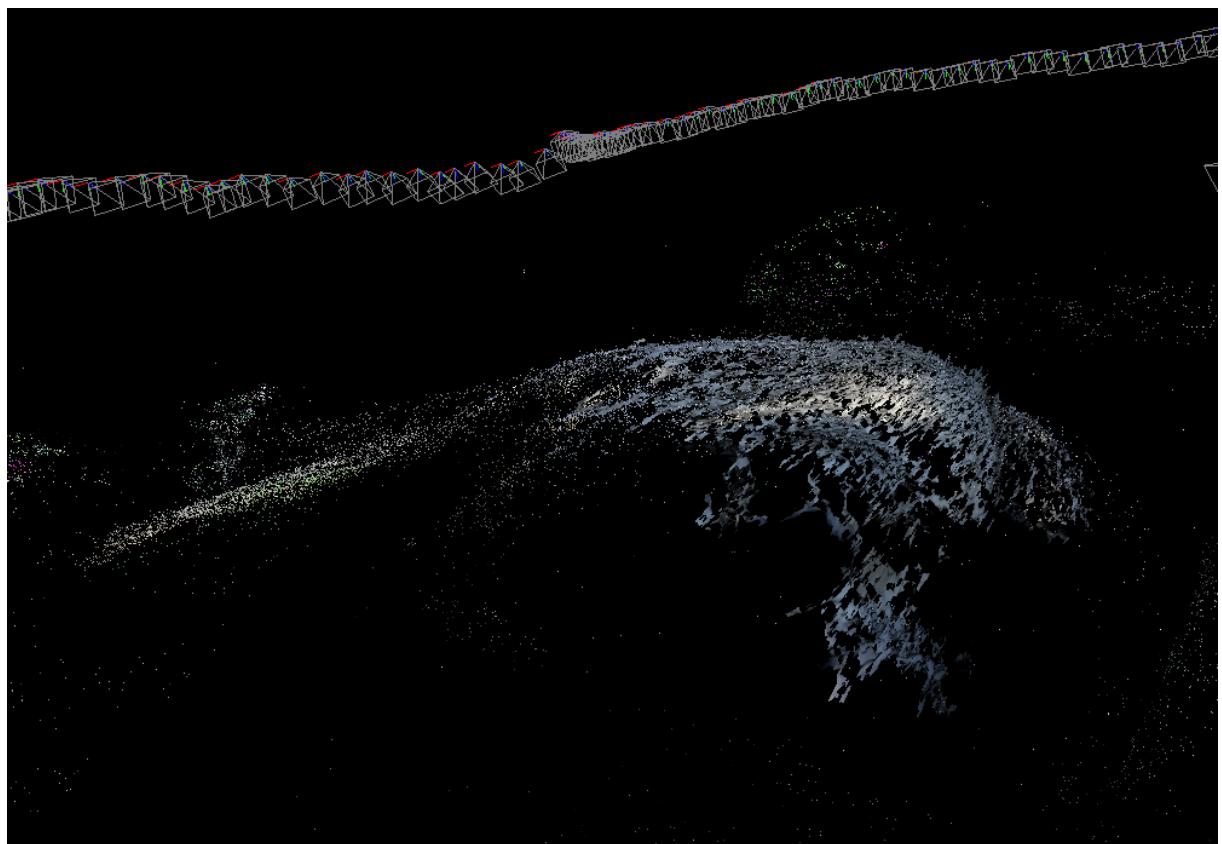
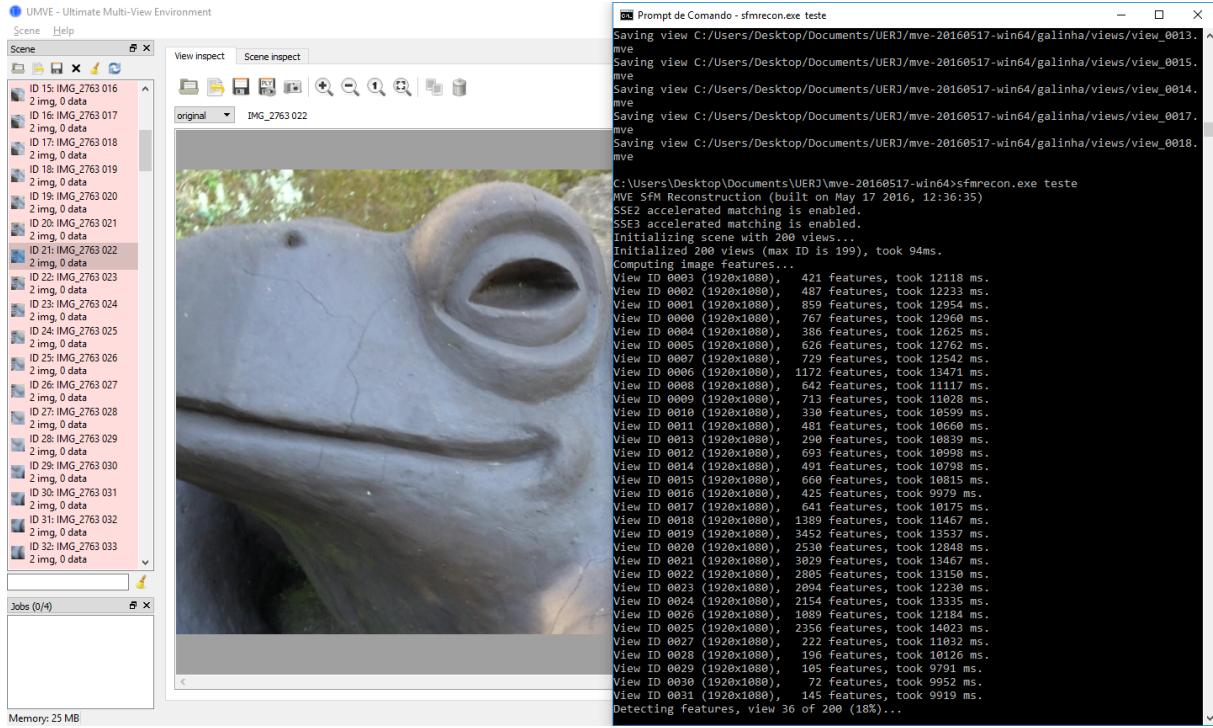
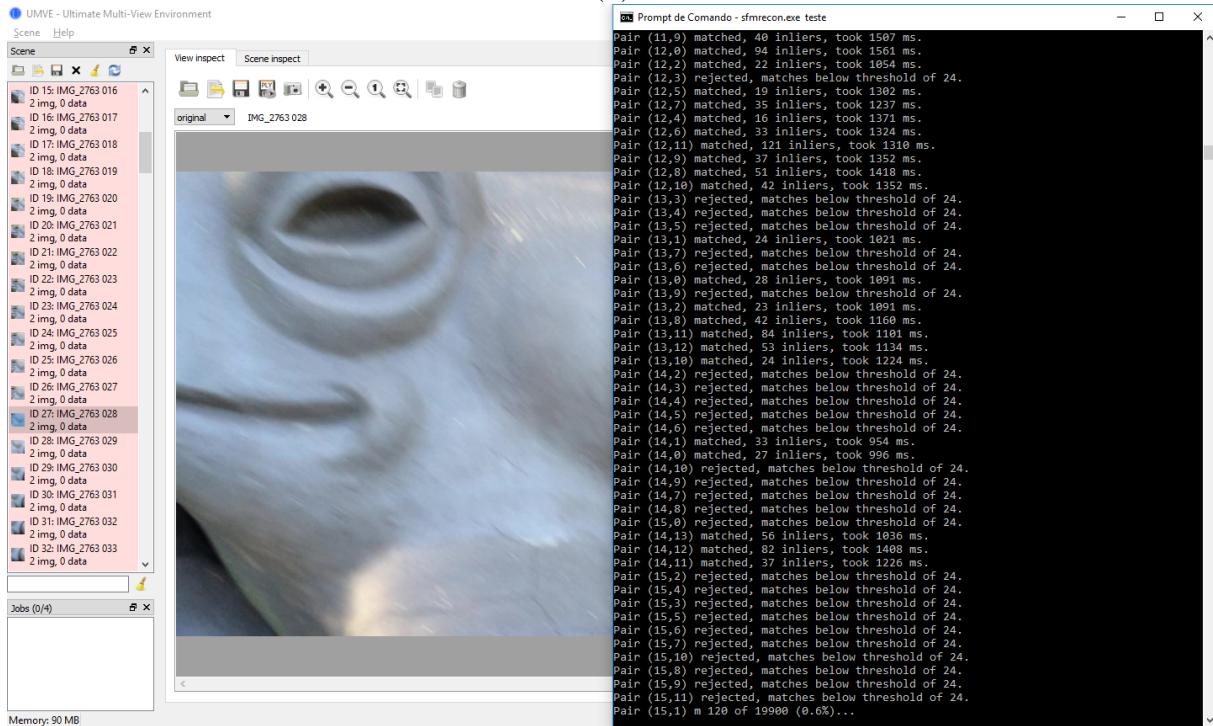


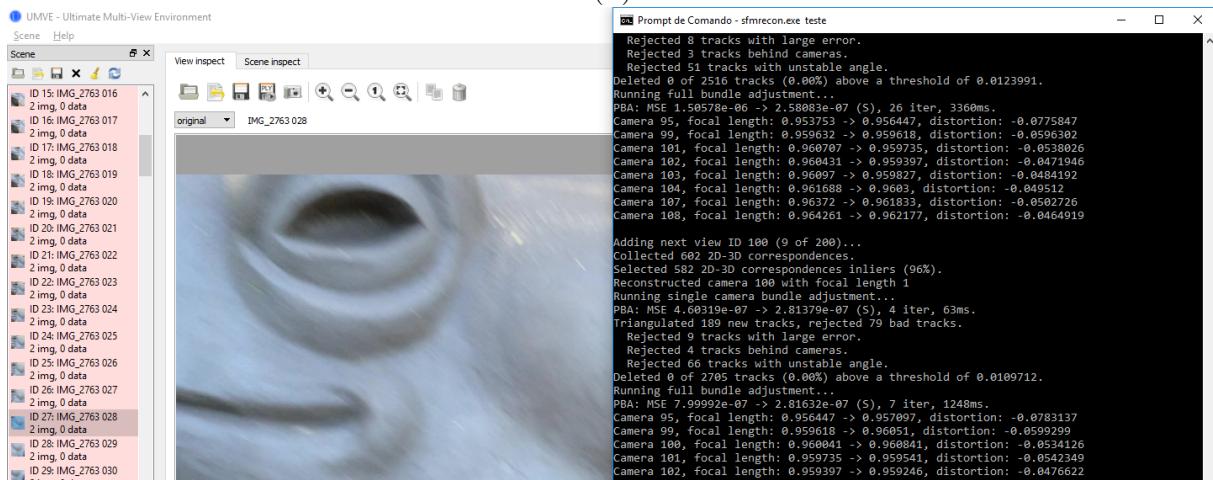
Figura 18 - Final da reconstrução via UMVE, percebe-se que alguns pontos não foram considerados, tendo como resultado uma "nuvem de pontos" mais densa, basicamente.

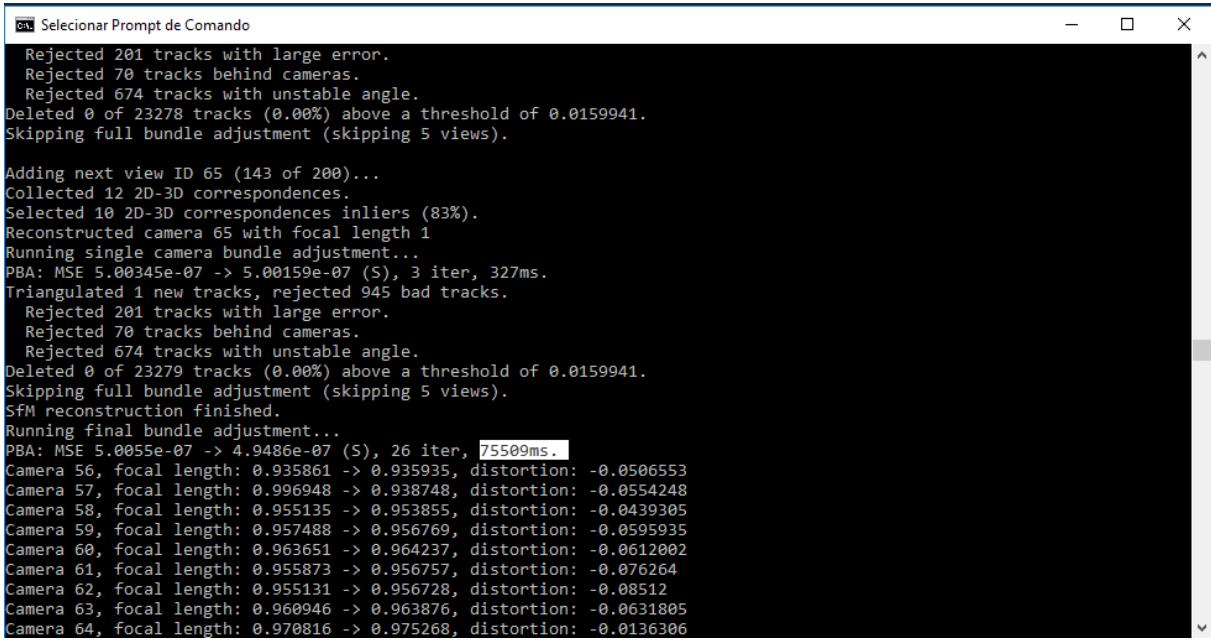


(a)



(b)





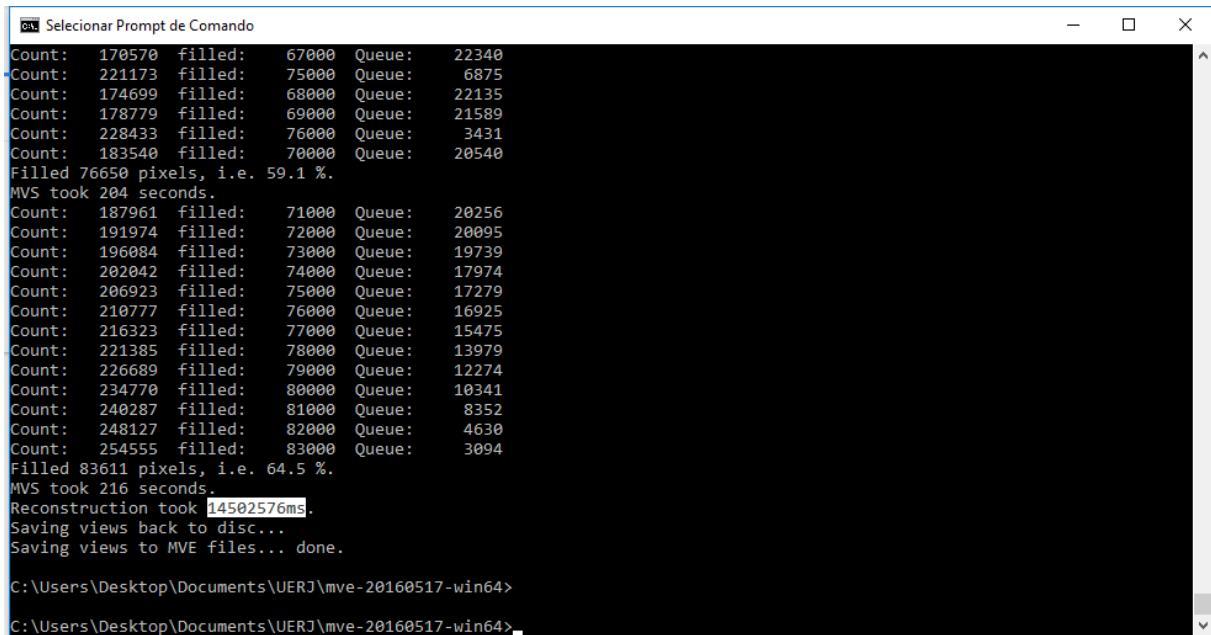
```

Selected 201 tracks with large error.
Selected 70 tracks behind cameras.
Selected 674 tracks with unstable angle.
Deleted 0 of 23278 tracks (0.00%) above a threshold of 0.0159941.
Skipping full bundle adjustment (skipping 5 views).

Adding next view ID 65 (143 of 200)...
Collected 12 2D-3D correspondences.
Selected 10 2D-3D correspondences inliers (83%).
Reconstructed camera 65 with focal length 1
Running single camera bundle adjustment...
PBA: MSE 5.00345e-07 -> 5.00159e-07 (S), 3 iter, 327ms.
Triangulated 1 new tracks, rejected 945 bad tracks.
    Rejected 201 tracks with large error.
    Rejected 70 tracks behind cameras.
    Rejected 674 tracks with unstable angle.
Deleted 0 of 23279 tracks (0.00%) above a threshold of 0.0159941.
Skipping full bundle adjustment (skipping 5 views).
SfM reconstruction finished.
Running final bundle adjustment...
PBA: MSE 5.0055e-07 -> 4.9486e-07 (S), 26 iter, 75509ms.
Camera 56, focal length: 0.935861 -> 0.935935, distortion: -0.0506553
Camera 57, focal length: 0.996948 -> 0.938748, distortion: -0.0554248
Camera 58, focal length: 0.955135 -> 0.953855, distortion: -0.0439305
Camera 59, focal length: 0.957488 -> 0.956769, distortion: -0.0595935
Camera 60, focal length: 0.963651 -> 0.964237, distortion: -0.0612002
Camera 61, focal length: 0.955873 -> 0.956757, distortion: -0.076264
Camera 62, focal length: 0.955131 -> 0.956728, distortion: -0.08512
Camera 63, focal length: 0.960946 -> 0.963876, distortion: -0.0631805
Camera 64, focal length: 0.970816 -> 0.975268, distortion: -0.0136306

```

Figura 20 - Término do comando *sfmrecon*, onde demorou cerca de 1 minuto e meio (75509 milisegundos).



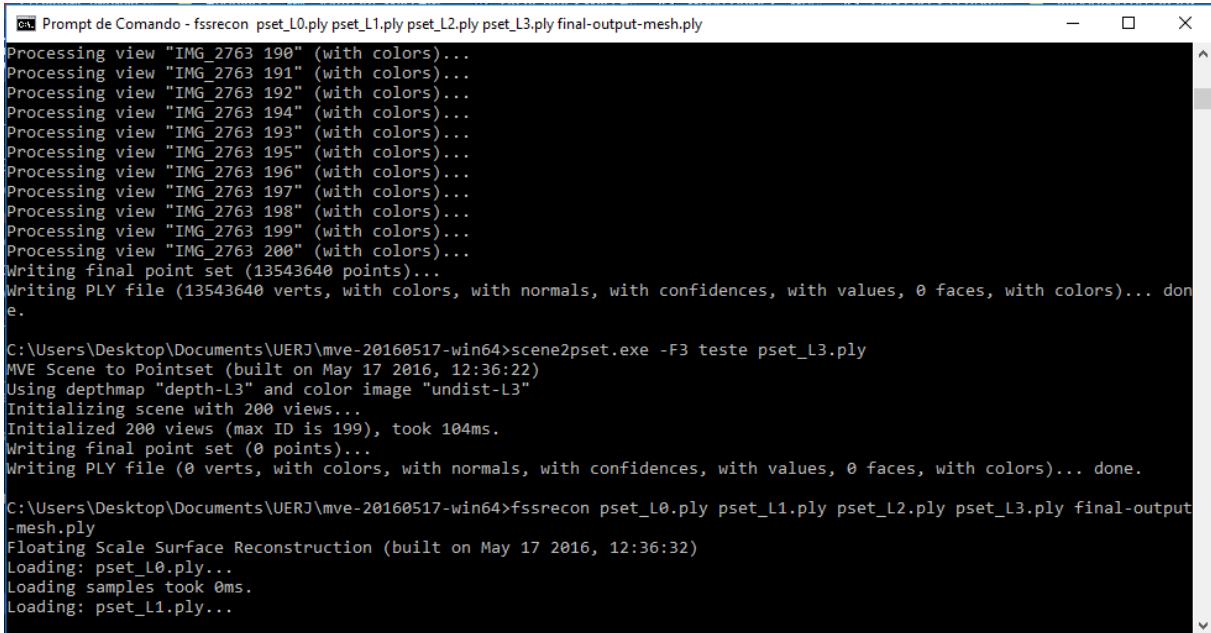
```

Count: 170570 filled: 67000 Queue: 22340
Count: 221173 filled: 75000 Queue: 6875
Count: 174699 filled: 68000 Queue: 22135
Count: 178779 filled: 69000 Queue: 21589
Count: 228433 filled: 76000 Queue: 3431
Count: 183540 filled: 70000 Queue: 20540
Filled 76650 pixels, i.e. 59.1 %.
MVS took 204 seconds.
Count: 187961 filled: 71000 Queue: 20256
Count: 191974 filled: 72000 Queue: 20095
Count: 196084 filled: 73000 Queue: 19739
Count: 202042 filled: 74000 Queue: 17974
Count: 206923 filled: 75000 Queue: 17279
Count: 210777 filled: 76000 Queue: 16925
Count: 216323 filled: 77000 Queue: 15475
Count: 221385 filled: 78000 Queue: 13979
Count: 226689 filled: 79000 Queue: 12274
Count: 234770 filled: 80000 Queue: 10341
Count: 248287 filled: 81000 Queue: 8352
Count: 248127 filled: 82000 Queue: 4630
Count: 254555 filled: 83000 Queue: 3094
Filled 83611 pixels, i.e. 64.5 %.
MVS took 216 seconds.
Reconstruction took 14502576ms.
Saving views back to disc...
Saving views to MVE files... done.

C:\Users\Desktop\Documents\UERJ\mve-20160517-win64>
C:\Users\Desktop\Documents\UERJ\mve-20160517-win64>

```

Figura 21 - Término do comando *dmrecon*, onde demorou cerca de 4 horas (14502576 milisegundos).



```

Prompt de Comando - fssrecon pset_L0.ply pset_L1.ply pset_L2.ply pset_L3.ply final-output-mesh.ply

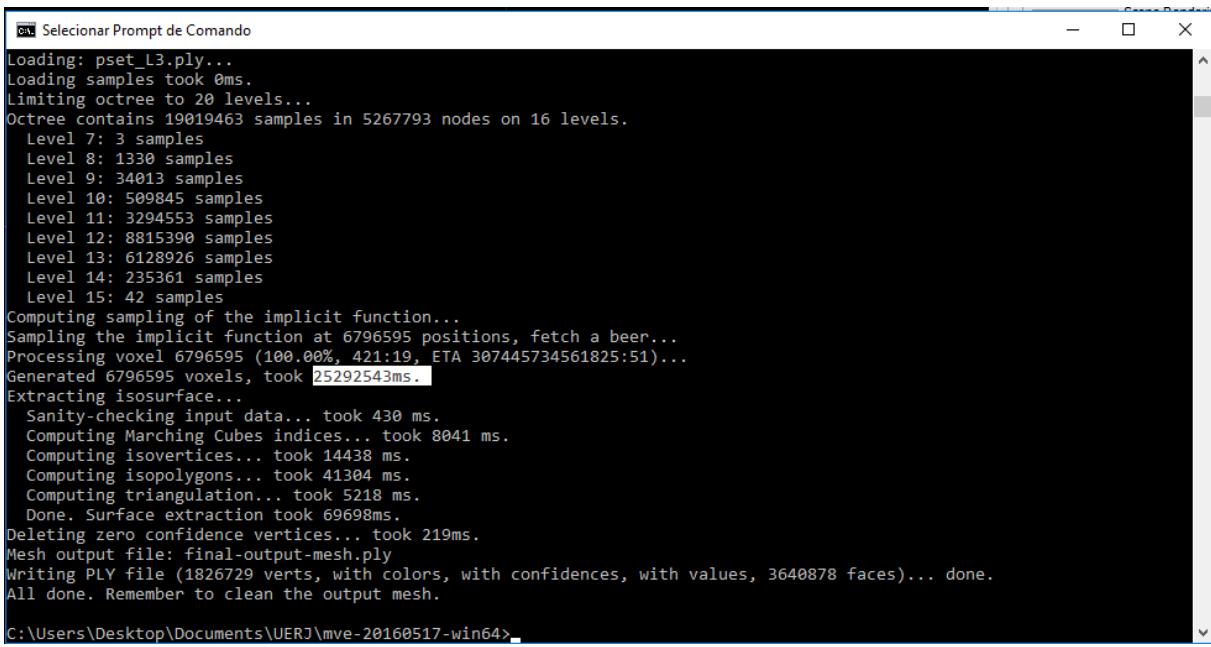
Processing view "IMG_2763_190" (with colors)...
Processing view "IMG_2763_191" (with colors)...
Processing view "IMG_2763_192" (with colors)...
Processing view "IMG_2763_194" (with colors)...
Processing view "IMG_2763_193" (with colors)...
Processing view "IMG_2763_195" (with colors)...
Processing view "IMG_2763_196" (with colors)...
Processing view "IMG_2763_197" (with colors)...
Processing view "IMG_2763_198" (with colors)...
Processing view "IMG_2763_199" (with colors)...
Processing view "IMG_2763_200" (with colors)...
Writing final point set (13543640 points)...
Writing PLY file (13543640 verts, with colors, with normals, with confidences, with values, 0 faces, with colors)... done.

C:\Users\Desktop\Documents\UERJ\mve-20160517-win64>scene2pet.exe -F3 teste pset_L3.ply
MVE Scene to Pointset (built on May 17 2016, 12:36:22)
Using depthmap "depth-L3" and color image "undist-L3"
Initializing scene with 200 views...
Initialized 200 views (max ID is 199), took 104ms.
Writing final point set (0 points)...
Writing PLY file (0 verts, with colors, with normals, with confidences, with values, 0 faces, with colors)... done.

C:\Users\Desktop\Documents\UERJ\mve-20160517-win64>fssrecon pset_L0.ply pset_L1.ply pset_L2.ply pset_L3.ply final-output-mesh.ply
Floating Scale Surface Reconstruction (built on May 17 2016, 12:36:32)
Loading: pset_L0.ply...
Loading samples took 0ms.
Loading: pset_L1.ply...

```

Figura 22 - Execução dos comandos *scene2pet*, nos níveis -F0, -F1, -F2 e -F3.



```

Selecionar Prompt de Comando

Loading: pset_L3.ply...
Loading samples took 0ms.
Limiting octree to 20 levels...
Octree contains 19019463 samples in 5267793 nodes on 16 levels.
  Level 7: 3 samples
  Level 8: 1330 samples
  Level 9: 34013 samples
  Level 10: 509845 samples
  Level 11: 3294553 samples
  Level 12: 8815390 samples
  Level 13: 6128926 samples
  Level 14: 235361 samples
  Level 15: 42 samples
Computing sampling of the implicit function...
Sampling the implicit function at 6796595 positions, fetch a beer...
Processing voxel 6796595 (100.00%, 421:19, ETA 307445734561825:51)...
Generated 6796595 voxels, took 25292543ms.
Extracting isosurface...
  Sanity-checking input data... took 430 ms.
  Computing Marching Cubes indices... took 8041 ms.
  Computing isovertices... took 14438 ms.
  Computing isopolygons... took 41304 ms.
  Computing triangulation... took 5218 ms.
  Done. Surface extraction took 69698ms.
Deleting zero confidence vertices... took 219ms.
Mesh output file: final-output-mesh.ply
Writing PLY file (1826729 verts, with colors, with confidences, with values, 3640878 faces)... done.
All done. Remember to clean the output mesh.

C:\Users\Desktop\Documents\UERJ\mve-20160517-win64>.

```

Figura 23 - Progressão do comando *fssrecon*, onde possui o ETA – *Estimated Time of Arrival*.

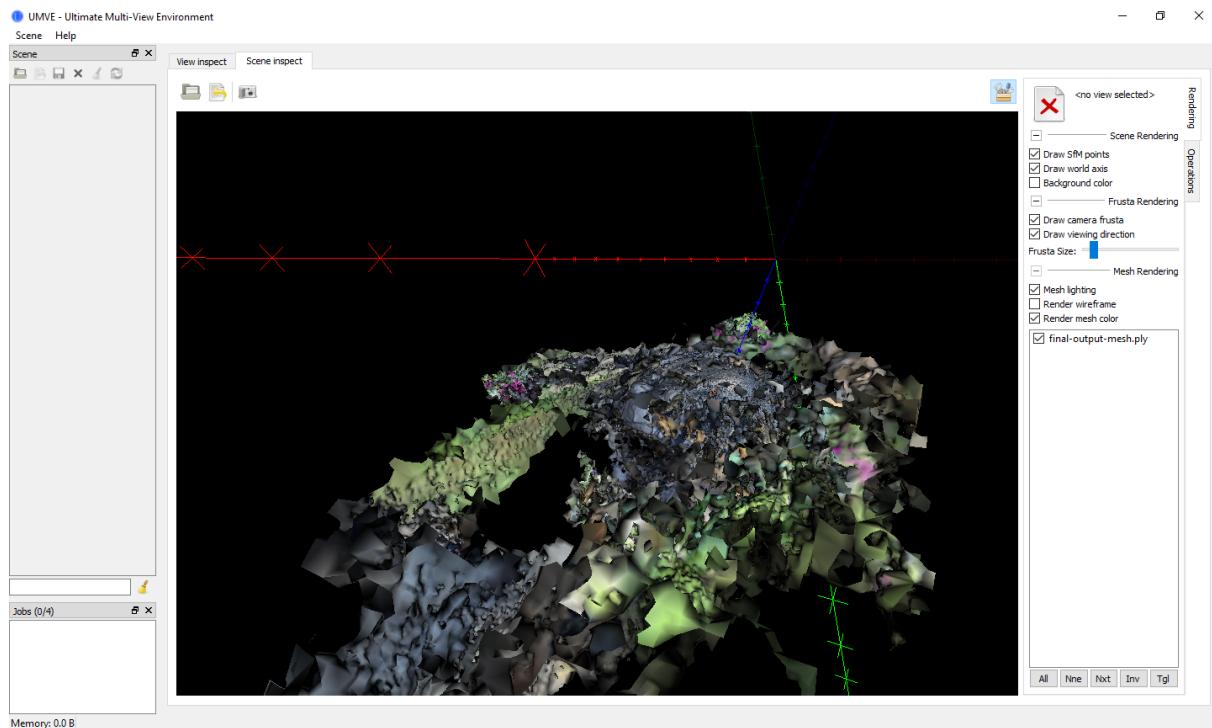


Figura 24 - Malha com ruídos proveniente do comando *fssrecon*.

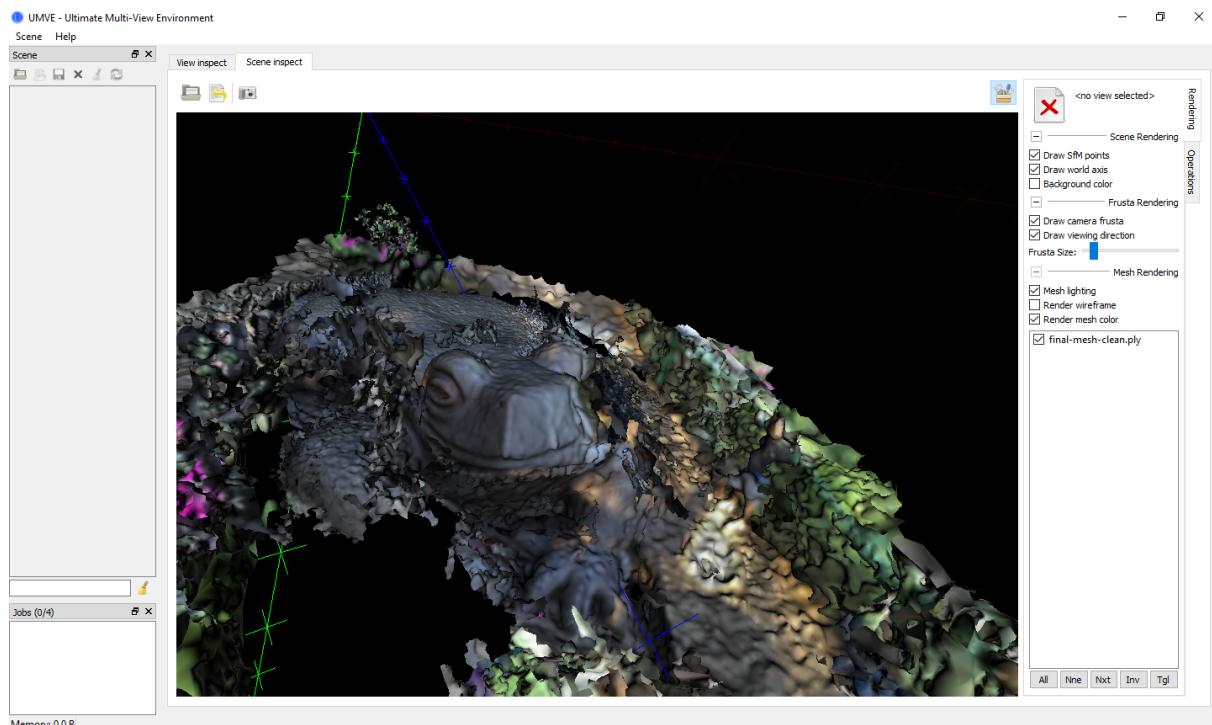


Figura 25 - Resultado final, após a remoção dos ruídos da malha.

### 3 KINECT

Um componente criado pela Microsoft para fins recreativos (como no XBox, por exemplo), virou uma das mais conhecidas ferramentas de reconstrução 3D no cenário atual. Sua primeira versão (Kinect V1) utiliza uma técnica similar à empregada no projeto da Universidade de Stanford, com luz estruturada, porém, diferentemente dos escaners à laser, o Kinect tem um custo monetário baixo e é acessível a todo público em geral (desde entusiastas, amadores até profissionais da área).

O Kinect V2 utiliza uma projeção de fóttons [...] e por isso ele já não é tão utilizado na área de reconstrução 3D como o V1.

Entretanto, uma desvantagem que diminui a aplicabilidade do Kinect é que ele foi projetado para funcionar bem em espaços fechados, com detecção de formas humanas e movimentações. Ou seja, numa aplicação *in situ* ele já não funcionaria muito bem, pois além de não conseguir projetar os detalhes em alta definição de uma escultura, ele necessita de uma fonte de energia externa, o que dificulta a acessibilidade do mesmo e como gera uma reconstrução em tempo real (não tem uma forma de salvar em *cache* ou internamente), ele precisa estar ligado a um computador para fazer o escaneamento.

#### 4 VISUALSFM

## 5 EXPERIMENTOS

## CONCLUSÃO