



Universidade Estadual de Campinas  
Instituto de Computação



# Planejamento de Trajetória de *Drones* para Reconstrução 3D

Tiago Loureiro Chaves (Aluno)  
Profa. Dra. Esther Luna Colombini (Orientadora)

Projeto FAPESP — Iniciação Científica

## Resumo

Reconstruções 3D precisas de objetos são fundamentais para diversas aplicações, desde robótica, computação gráfica e realidade virtual até medicina, geologia, agronegócio e arquitetura. Veículos aéreos não tripulados (VANTs) têm sido cada vez mais empregados para a captura de imagens aéreas a medida que modelos menores e mais acessíveis ficam disponíveis no mercado. Essas imagens podem ser utilizadas para gerar modelos 3D de alta qualidade da cena sobrevoada, mas a qualidade do modelo resultante depende muito do plano de voo e do trajeto traçado, e ainda requer pilotos experientes para ambientes complexos. Assim, a pesquisa relacionada à obtenção de imagens para a reconstrução de estruturas de grande porte vem ganhando importância, principalmente no que tange a capacidade de percepção destes veículos. No geral, devido à falta de informações precisas (e atuais) sobre o ambiente, e também por questões de segurança, as soluções automatizadas com *drones* recorrem a voos em padrões regulares a uma distância aérea segura. No entanto, os pontos de vista capturados são, em muitos casos, insuficientes para a reconstrução 3D de alta qualidade. Desse modo, a transição de sistemas automatizados para sistemas autônomos é fundamental. Neste cenário, este projeto propõe explorar uma forma de otimizar a trajetória de voo em tempo real, a partir, essencialmente, da visão adquirida através de uma câmera RGB acoplada ao robô, de modo a melhorar a reconstrução final da área de interesse.

# 1 Introdução

A resolução de problemas relacionados à reconstrução 3D sempre foi um problema difícil e amplamente abordado na literatura [1]. Não por acaso, a reconstrução de cenas a partir de pontos de vista distintos continua sendo um problema não resolvido [2]. Enquanto para os humanos, e outros seres vivos, a tarefa de determinar e discernir a estrutura tridimensional de um objeto em movimento a partir de sua projeção bidimensional (na retina) parece algo trivial — algo que bebês com menos de 1 ano já são capazes de fazer — projetar um algoritmo para que uma máquina faça isso não é uma tarefa simples.

Entretanto, apesar da complexidade, este é um tema com inúmeros estudos na área de computação, e as diversas aplicações de tal processo os justificam. Com a reconstrução 3D podemos descrever o contorno de objetos, e assim, determinar as coordenadas de qualquer ponto neles.

O interesse em técnicas de reconstrução 3D não é restrito a disciplinas da Ciência da Computação, como visão computacional e computação gráfica. Seu uso vai desde a área médica (imagiologia) e a arqueologia (fotogrametria) até a indústria agrícola e a construção civil — para cálculo de volumes, modelagem digital de terrenos, monitoramento de progresso de plantas, inspeção, mapeamento e levantamento topográfico.

Dentre tais aplicações, destaca-se o crescente uso de *drones* com câmeras acopladas para a captura de imagens aéreas de estruturas. Porém, a qualidade da reconstrução obtida ainda é muito dependente do controle manual de pilotos (voo livre), ou de trajetórias pré-programadas baseadas em grade ou em hélice (conforme Figura 1). Quando aplicadas a cenas de modelagem complexas, essas abordagens resultam em lacunas na cobertura total ou em inconsistência na reconstrução final.

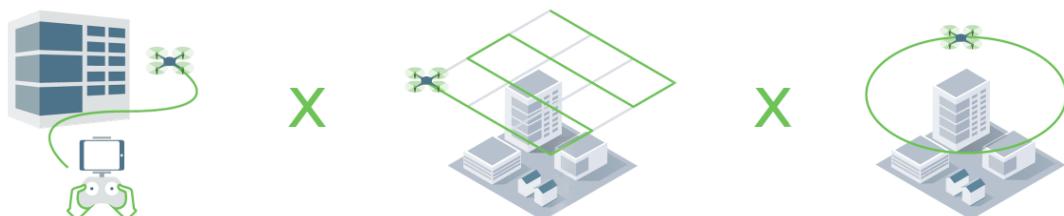


Figura 1: Ilustração esquemática do voo livre (esquerda), controlado manualmente, do voo automatizado com trajetória baseada em grade (centro), e do voo automatizado com trajetória em hélice/circular (direita)<sup>1</sup>

<sup>1</sup>Imagem extraída da página: [www.pix4d.com/product/pix4dcapture](http://www.pix4d.com/product/pix4dcapture).

Além disso, outros fatores são relevantes quando abordamos a otimização da trajetória de captura realizada por drones. Se considerarmos que o tempo máximo de voo de *drones* atualmente é de cerca de 20 a 30 minutos, — em contrapartida, a recarga da bateria leva de uma a duas horas — otimizações nesse processo são ainda mais importantes. Adicionalmente, ao considerarmos que diversos algoritmos dependem de condições de cena relativamente constantes (p. ex. iluminação) para uma boa reconstrução, a obtenção de imagens não pode perdurar por muito tempo, ou ainda, ser realizada sobre capturas tomadas em dias distintos.

Portanto, neste projeto exploraremos formas de tornar o voo de *drones* para captura de imagens mais eficiente, ou seja, de modo que os modelos 3D resultantes sejam superiores aos obtidos pela rota em grade (ou em hélice) utilizada atualmente para voos automatizados. Ainda, consideraremos a busca por um tempo de voo igual ou inferior — fazendo ajustes de trajetória (*on-the-fly*) que otimizem a reconstrução.

Assim, pretendemos desenvolver um algoritmo de planejamento de trajetória em tempo real (*real-time path planning* [3]), que iterativamente analise as nuvens de pontos que representam a reconstrução esparsa da estrutura sobrevoada para detectar regiões de densidade baixa (indicando a necessidade de mais fotos) ou muito alta (sugerindo que podemos voar mais rapidamente por elas).

Com isso, esperamos obter um modelo para o voo autônomo de *drones* que otimize as reconstruções 3D realizadas, de modo que estas possam ser empregadas em aplicações que requerem mais qualidade da reconstrução obtida. De posse destas reconstruções mais eficientes pode ser possível diminuir custos , mas também — e principalmente — diminuir o risco associado a certas atividades de inspeção e manutenção da construção civil e do setor de energia (como em estruturas de pontes e edifícios, turbinas eólicas, refinarias de petróleo, postes de alta tensão, gasodutos, etc.), e também utilizar o modelo construído e as imagens capturadas para busca, resgate e monitoramento após desastres naturais [4, 5].

## 2 Objetivos

Este projeto tem como objetivo propor um modelo de otimização da trajetória de voo de *drones* autônomos, e que ocorra em tempo real (não apenas no planejamento pré-voo), para melhoria das reconstruções 3D da área sobrevoada.

Este projeto de iniciação científica tem como principais objetivos:

1. Construir um baseline avaliando a qualidade dos modelos reconstruídos a partir das estratégias mais comuns de voo autônomo utilizadas atualmente, o voo circular e em grade (veja Figura 1).
2. Desenvolver um algoritmo capaz de desviar de sua trajetória pré-planejada para sobrevoar uma estrutura, de modo a evitar colisões, e então retomá-la.
3. Gerar incrementalmente uma nuvem de pontos da estrutura, em tempo real.
4. Analisar a nuvem de pontos para determinar que regiões devem ser capturadas mais vezes (também em tempo real), atualizando assim o plano de voo.
5. Automatizar o processo: “seguir a trajetória de voo capturando imagens da cena → reconstruir o modelo 3D → analisar a nuvem de pontos → determinar que locais devem ser sobrevoados e reformular a trajetória de voo → seguir a trajetória de voo capturando imagens da cena”, reagindo a possíveis desvios de obstáculos.
6. Confrontar os resultados do planejador de trajetória otimizado frente ao baseline.

### 3 Trabalhos Relacionados

Podemos identificar três principais sistemas operando juntos para possibilitar o planejamento autônomo da trajetória de *drones*: (a) Visão, utilizada como o sensor que permite a percepção do mundo externo, e também empregada na captura das imagens utilizadas posteriormente na reconstrução 3D; (b) Mapeamento, necessário para representar o ambiente e localizar o *drone*; e (c) Planejamento, que une a posição do *drone* e o modelo do ambiente para determinar a trajetória que deve ser seguida.

Assim, apresentamos a seguir métodos de planejamento relacionados a nossa proposta, com ênfase na atualização em tempo real do modelo da cena sobrevoada e do plano de voo com o objetivo de otimizar a trajetória.

Sirmacek et. al [6] consideram o problema da reconstrução 3D sem nenhum conhecimento *a priori* da estrutura de interesse. Ao decolar, o *drone* passa a reconstruir o local sobrevoado e identifica como área de interesse a que possui a maior quantidade de pontos na nuvem de pontos (*point cloud*), e então começa a circundá-la. A nuvem de pontos é então dividida no seu centro de massa em quatro partes, e o *drone* segue o semi-arco que

leva ao quadrante de menor entropia (calculada como a quantidade de pontos na sub-região). Entretanto, seus resultados são apresentados apenas para a reconstrução de um cilindro com textura rochosa em simulação (no Gazebo), assim não fica claro se o método é escalável para formas mais complicadas ou para o mundo real.

Peng e Isler [7] também tratam o problema do replanejamento para aperfeiçoar os modelos tridimensionais de estruturas complexas. Para isso, realizam a aquisição de imagens em duas etapas: o *drone* inicialmente explora a cena com uma trajetória fixa (em grade), gerando uma primeira aproximação da reconstrução 3D; então, eles tratam o planejamento de trajetória como um problema de cobertura de conjunto (*set cover problem, SCP* [8]) que encontra o número mínimo de visualizações para que sua região comum cubra toda a cena. Porém, ao invés de considerarem todo o espaço 3D livre, os autores introduzem um conjunto finito de planos de vista (adaptados à cena) para reduzir o espaço de busca.

Peng e Isler também mostram que a qualidade da reconstrução final pode ser consideravelmente melhorada utilizando-se mais um conjunto de imagens, obtidas em uma terceira etapa, e que três etapas são suficientes mesmo para cenas complexas simuladas na Unreal Engine. Essa abordagem *explore-then-exploit* vem sendo muito utilizada, pois a partir de conhecimento mínimo — apenas a “caixa” que envolve a região de interesse — podemos sobrevoar seguramente a cena para gerar uma primeira aproximação, que permite a elaboração de uma trajetória segura que explore as áreas desconhecidas, evitando traçar o plano de voo inicial de modo que haja colisões com a estrutura.

Hepp et. al [9] também baseiam sua trajetória em um varredura inicial da região, e apresentam resultados utilizando um quadricóptero real. Assim como em [7], o problema da seleção de *viewpoints* para capturar as imagens é abordado no contexto do *SCP*, e a otimização do caminho que percorre os pontos selecionados é tratado como uma variação do problema do caixeiro-viajante, ambos NP-difíceis. Além da nuvem de pontos, um mapa volumétrico de ocupação (*OctoMap* [10]) é construído, onde cada *voxel* identifica uma região ocupada, livre, ou desconhecida (não observada).

Dada a dificuldade da avaliação comparativa de cenas reais, devido a mudanças de iluminação, condições climáticas, objetos ao redor, e também por raramente termos os valores exatos (*ground truth*) sobre a estrutura real, Hepp et. al concentram-se em uma análise qualitativa dos exemplos físicos que apresentam. Porém, utilizam cenas sintéticas

(também com a Unreal Engine) para realizar uma comparação quantitativa com métodos anteriores, superando o estado-da-arte.

A comparação quantitativa é baseada no procedimento estabelecido por Knapitsch et al. [11, p. 9] para o cálculo de precisão (*precision*) e revocação (*recall*), definindo a porcentagem de pontos reconstruídos que estão perto de um ponto real e a porcentagem de pontos reais próximos de um ponto reconstruído, respectivamente.

Uma vez que o planejamento de trajetória tratado é um problema de otimização de difícil definição — i.e. especificar um algoritmo suficientemente bem, para que ele seja capaz de descrever precisamente a função objetivo que queremos otimizar, é uma tarefa irrealista — estudar tal problema pela ótica do aprendizado por reforço é uma abordagem que vem sendo cada vez mais explorada, especialmente devido ao aumento da quantidade e qualidade de simuladores.

As aplicações de aprendizado por reforço neste contexto vão desde a navegação [12, 13] e busca [14, 15] autônomas, até o uso de *drones* para entregas [16], cinematografia aérea [17, 18], ou como estações base móveis [19].

## 4 Metodologia Proposta

Nesta seção apresentaremos a metodologia proposta para o desenvolvimento de um algoritmo de planejamento em tempo-real, com o objetivo de elaborar uma solução autônoma para o problema. Ou seja, não queremos apenas uma trajetória pré-programada (como são oferecidas atualmente por soluções comerciais como Pix4D, DroneDeploy e 3DR Site Scan), mas sim uma que se adapte à cena explorada — desviando de obstáculos e alterando o plano de voo inicial para melhor capturar a cena. A Figura 2 apresenta o fluxo da metodologia proposta.

### 4.1 Especificidades e restrições

Limitaremos nossos testes a um ambiente simulado (utilizando o AirSim [21], na Unreal Engine), para evitar o risco de danificar *drones* reais e para acelerar os experimentos, já que podemos executar várias simulações simultaneamente e também alterar parâmetros importantes como tempo de simulação, condições de iluminação, complexidade da cena,

---

<sup>2</sup>Imagens do coelho de Stanford (modificadas a partir de [20]) ilustrando três fases finais da reconstrução 3D: *point cloud*, geração da malha poligonal, e mapeamento da textura no modelo.

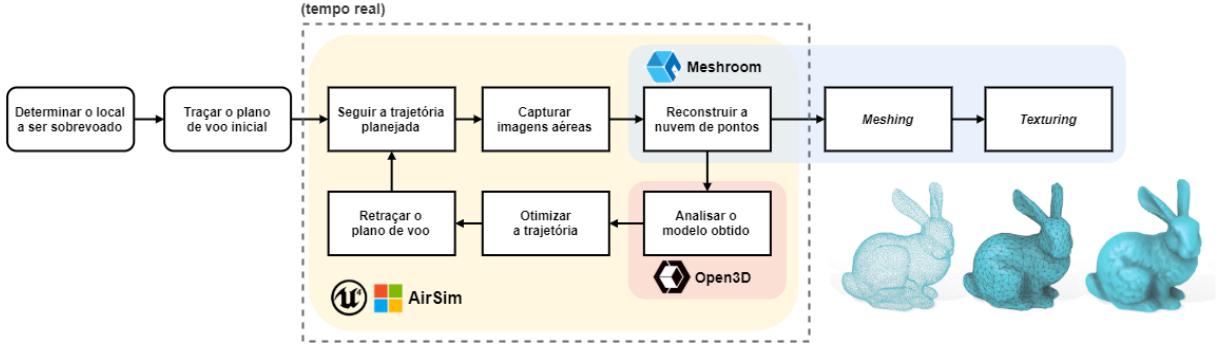


Figura 2: *Pipeline* de etapas envolvidas no processo proposto para a otimização da trajetória de *drones*.<sup>2</sup>

etc. Apesar disso, como mencionado na Seção 3, existem diversos trabalhos que estendem seus resultados obtidos em simuladores para o mundo real.

O AirSim oferece um *drone* com quatro motores — *quadrotor* — com sensores de distância, GPS, magnetômetro, barômetro, LIDAR [22], IMU [23] e cinco câmeras (com diferentes modos de captura).

Sabe-se que a navegação de *drones* em ambientes fechados ou sem sinal de GPS ainda é um grande desafio, porém, é um requisito básico para sistemas robóticos que devem operar no mundo real e para passarmos da automação para a autonomia. Assim, nos restringiremos a abordagens ao uso de sensores visuais e inerciais, como nos trabalhos [24, 25] de SLAM (*Simultaneous Localization And Mapping* [26]).

Para tornar o algoritmo generalizável a outros modelos de *drone* e com intuito de reduzir o escopo, ao invés de explicitarmos os controles que devem ser acionados ou as velocidades ( $x$ ,  $y$ ,  $z$ ) e rotações ( $roll$ ,  $yaw$ ,  $pitch$  [27]) de cada eixo, definiremos apenas o próximo ponto de onde deve-se capturar uma imagem (*viewpoint*).

Portanto, tratamos a transição entre *viewpoints* como um sistema caixa-preta. Desse modo, o controle pode ser adaptado a diferentes modelos de *drone*. Utilizamos inicialmente o `moveOnSplineAsync` da API de controle do AirSim, que ajusta uma trajetória a um conjunto de pontos de referência, e `moveToPositionAsync`, que voa reto em direção a um ponto dado.

## 4.2 Criação da *baseline*

Uma vez que não temos uma base de referência para o problema que buscamos tratar, uma importante parte inicial do projeto é a criação de uma *baseline*, contra a qual com-

pararemos os resultados obtidos posteriormente.

Escolhemos como referência as duas mais comuns trajetórias de voo automatizadas (veja Figura 1): 1) o voo em grade (ou *zigzag*), utilizado em geral para a captura regiões extensas (p. ex. prédios, florestas, plantio); e 2) o voo em hélice (ou circular), cujo objetivo é conseguir uma melhor reconstrução de estruturas mais complicadas, e isoladas.

Deste modo, avaliaremos um conjunto de *environments*<sup>3</sup> na Unreal Engine para realizarmos comparações em cenários diversos.

### 4.3 Análise dos dados

Visando o funcionamento em tempo-real, durante o voo criaremos incrementalmente uma nuvem de pontos da cena sobrevoada, realizando as etapas de *meshing* e *texturing* apenas ao final do voo — quando já temos a nuvem de pontos concluída — devido ao custo computacional dessas fases finais do processo de fotogrametria (ver Figura 2).

Podemos então computar propriedades como a densidade da nuvem de pontos para identificar as regiões das quais ainda temos pouca informação.

### 4.4 Otimização da trajetória

Partindo das informações dos sensores disponíveis e da nuvem de pontos reconstruída como dados de entrada, exploraremos como modelar um algoritmo que seja capaz de aprender a otimizar a tomada de decisão de pontos de vista (*viewpoints*) adequadamente. Neste sentido, usaremos algoritmos de Aprendizado por Reforço (AR) como base.

Uma vez que a descrição e especificação da função de reforço é muitas vezes uma das maiores dificuldades em aprendizado por reforço (AR), ressaltamos que a busca por uma função de reforço adequada também é parte central da pesquisa desenvolvida durante o projeto. Como desejamos otimizar a reconstrução 3D de estruturas, não existem métricas quantitativas bem definidas para a avaliação de diferentes soluções, tanto que a análise é ainda muito baseada em fatores qualitativos. Neste sentido, usaremos medidas de densidade, entropia, entre outras, para definir a qualidade da reconstrução realizada. Adicionalmente, como se trata de um problema de otimização multiobjetivo, dados relacionados ao tempo de bateria podem ser adicionados.

---

<sup>3</sup>Além de 12 mapas disponibilizados pelo AirSim (que incluem florestas, bairros, cidades, etc.), planejamos testar alguns dos que foram utilizados em [7] e [9] (como uma cidadela e uma refinaria de petróleo).

A qualidade final das trajetórias será avaliada segundo o procedimento proposto por Knapitsch et al. [11, p. 9] para o cálculo de precisão (*precision*) e revocação (*recall*). A Figura 3 apresenta um exemplo de *precision* e *recall* computados sobre uma reconstrução de acordo com o método proposto em [11].

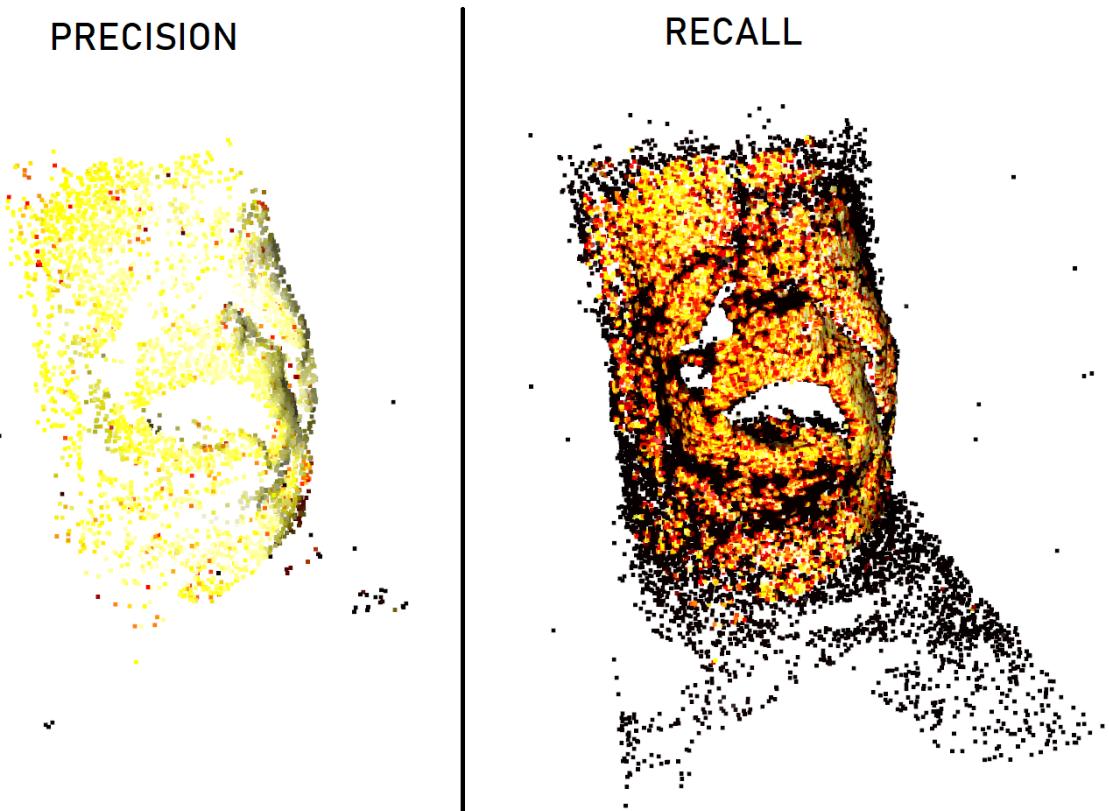


Figura 3: Precision e Recall calculados sobre uma nuvem de pontos.

## 5 Materiais

O projeto será focado na avaliação dos algoritmos desenvolvidos em ambiente simulado, utilizando o simulador AirSim [21] desenvolvido pela Microsoft, principalmente devido à sua fidelidade visual e aplicação no treinamento de diversos sistemas que atuam no mundo físico [28, 29], visando a transferência do comportamento simulado para o mundo real (*Sim2Real* [30]) como um dos desdobramentos deste projeto.

As reconstruções tridimensionais das estruturas capturadas pelas imagens obtidas em simulação serão processadas, a princípio, pelo framework AliceVision [31, 32] com o software Meshroom. Pretendemos também avaliar o conjunto de soluções OpenDroneMap

[33], possivelmente passando a utilizar o projeto PyODM ao invés da interface Python que o Meshroom expõe para utilizar o AliceVision.

Para processar e examinar os modelos gerados na etapa anterior, usaremos a biblioteca para processamento de dados tridimensionais Open3D [34]. Desenvolvida pelo *Intelligent Systems Lab* da Intel — laboratório focado na pesquisa em visão computacional, aprendizado de máquina e robótica — essa biblioteca disponibiliza um conjunto de estruturas de dados (*k-d tree*, Octree, etc.) e algoritmos (fecho convexo, DBSCAN, vizinhos mais próximos, remoção estatística de pontos *outliers*, etc.) que serão utilizados na análise das nuvens de ponto geradas para a reconstrução do modelo tridimensional, possibilitando a extração de informações em tempo real para a tomada de decisões sobre a trajetória a ser seguida pelo *drone*.

Além disso, como será utilizada principalmente a linguagem de programação Python, diversas bibliotecas que hoje são padrão na comunidade serão empregadas, para: processamento de imagens (OpenCV), visualização de dados e gráficos (Matplotlib), computação científica (NumPy e Scikit-Learn), e desenvolvimento de redes neurais (TensorFlow).

## 6 Análise de Resultados

Como destacado nas seções anteriores, a comparação objetiva de diferentes técnicas de reconstrução 3D não é trivial, e não há apenas uma métrica de avaliação bem definida e universalmente adotada.

Apesar disso, posto que usaremos um simulador, temos medidas exatas das estruturas utilizadas, o que facilita a análise comparativa.

Uma vez que devemos reconstruir a cena e, ao mesmo tempo, nos localizar nela para determinarmos o próximo ponto para o qual iremos, estamos efetivamente estudando o problema de SLAM visual [26]. Consequentemente, aproveitaremos dois métodos bem aceitos [35, 36] para a avaliação do erro associado à trajetória estimada: *absolute trajectory error* (ATE) e *relative pose error* (RPE).

Comparando a trajetória real (*ground truth*) obtida diretamente pelo simulador, com o conjunto de posições estimadas para o *drone* — usadas pelo algoritmo desenvolvido na determinação dos próximos *viewpoints* — o ATE mede diretamente a diferença entre os pontos estimados e os reais, enquanto o RPE mede a precisão local da trajetória em um

dado intervalo de tempo, que corresponde ao desvio da trajetória por unidade de tempo.

Além desses métodos, pretendemos avaliar o procedimento apresentado por Knapitsch et al. [11] para o cálculo de precisão e revocação, já que ele foi utilizado por Hepp et. al [9] em trabalho similar, especialmente nas cenas da Unreal Engine foram usadas em [9].

Ainda assim, recorreremos também à avaliação qualitativa das reconstruções realizadas, como é feito nos trabalhos citados na Seção 3.

## 7 Cronograma

Na Tabela 1 apresentamos o cronograma de 12 meses com o planejamento de atividades.

Atividades	Meses					
	1-2	3-4	5-6	7-8	9-10	11-12
Revisão bibliográfica	★	★	★	★	★	
Configuração do <i>pipeline</i> de experimentos	★	★		★		
Criação de uma <i>baseline</i>	●	●				
Prevenção de obstáculos		●	●	●		
Atualização de trajetória a partir da nuvem de pontos		●	●	●	●	
Voo livre				●	●	●
Disseminação de resultados					●	●

Tabela 1: Cronograma do projeto. (★ atividades já iniciadas; ● atividades futuras)

**Revisão bibliográfica:** Esta etapa consiste em continuar nossos estudos sobre trabalhos estado-da-arte relacionados ao projeto, explorando em particular a literatura sobre: *path planning*, otimização de trajetórias de *drones*, *struct from motion*, trabalhos utilizando simuladores para veículos aéreos e aprendizado por reforço.

**Configuração do *pipeline* de experimentos:** Prepararemos as ferramentas a serem utilizadas no projeto — AirSim (simulador), Meshroom (reconstrução 3D), Open3D (análise dos dados tridimensionais) — para que possamos automatizar os experimentos, integrando as APIs Python providas por todas as bibliotecas mencionadas, de forma a acelerar o *feedback loop* de desenvolvimento → teste → avaliação → desenvolvimento. Esta etapa também envolve a avaliação do OpenDroneMap, e a familiarização com a Unreal Engine para podermos criar e alterar diferentes cenas de teste (p. ex. para utilizarmos *domain randomization*) na segunda metade do projeto.

**Criação de uma *baseline*:** Partiremos de trajetórias simples pré-programadas, comumente utilizadas (p. ex. em grade e em hélice, conforme Figura 1), coletando imagens de um conjunto de cenas simuladas para podermos utilizá-las como base de referência para avaliar nosso trabalho, comparando as reconstruções feitas.

**Prevenção de obstáculos:** Ainda seguindo um plano de voo predeterminado, introduziremos obstáculos de modo que será necessário que o *drone* desvie do caminho planejado (sem intervenção manual de um piloto, mantendo o voo autônomo). Para isso, aplicaremos um algoritmo capaz de desviar da trajetória para evitar colisões e então retomá-la — fazendo o replanejamento e a detecção de obstáculos (a partir das imagens da câmera) em tempo real.

**Atualização de trajetória a partir da nuvem de pontos:** Nesta etapa iniciaremos a análise em tempo real da nuvem de pontos reconstruída (pelo Meshroom a partir das imagens coletadas no AirSim), com o objetivo de determinar as regiões da estrutura sobrevoada que devem ser melhor capturadas, com o objetivo de aprimorar a reconstrução 3D final.

**Voo livre:** Exploraremos como tornar o modelo implementado mais geral, removendo a limitação imposta por termos que prover um plano de voo inicial. Assim, pretendemos ter ao final desta fase um algoritmo que seja inteiramente responsável por traçar a rota de voo do *drone*, tendo conhecimento apenas de uma caixa delimitadora (*bounding box*) da área de interesse.

**Disseminação de resultados:** Na etapa final do projeto de Iniciação Científica escreveremos e submeteremos artigos científicos (e relatórios técnicos) para conferências de relevância na área, melhoraremos a documentação dos algoritmos desenvolvidos, e publicaremos o trabalho desenvolvido de forma aberta (*open-source*).

## 8 Atividades Preliminares

Nesta seção detalhamos as atividades já desenvolvidas, que tiveram como foco o levantamento, estudo, familiarização e avaliação dos algoritmos que apresentam resultados estado-da-arte nas etapas envolvidas no projeto (ilustradas na Figura 2). Nota-se que houve particular intenção na escolha por sistemas de código aberto, para que o trabalho desenvolvido também possa ser disponibilizado à comunidade.

## 8.1 Simulação

Com o objetivo de agilizar o *feedback loop* de desenvolvimento → teste → avaliação → desenvolvimento, usaremos um simulador de voo de *drones* para minimizar o tempo e os custos relacionados às etapas de teste e avaliação.

Atualmente, dois dos melhores simuladores de dinâmica utilizados na robótica são o software comercial V-REP, da empresa suíça Coppelia Robotics, e o Gazebo, um projeto *open-source* mantido pela Open Source Robotics Foundation [37].

Entretanto, apesar da fidelidade na simulação da dinâmica física, essas plataformas apresentam visuais gráficos muito simplistas, ou seja, não são capazes de gerar as renderizações fotorrealistas necessárias (como apresentado na Figura 4), uma vez que temos como objetivo utilizar as imagens do simulador para reconstruir cenas tridimensionais.

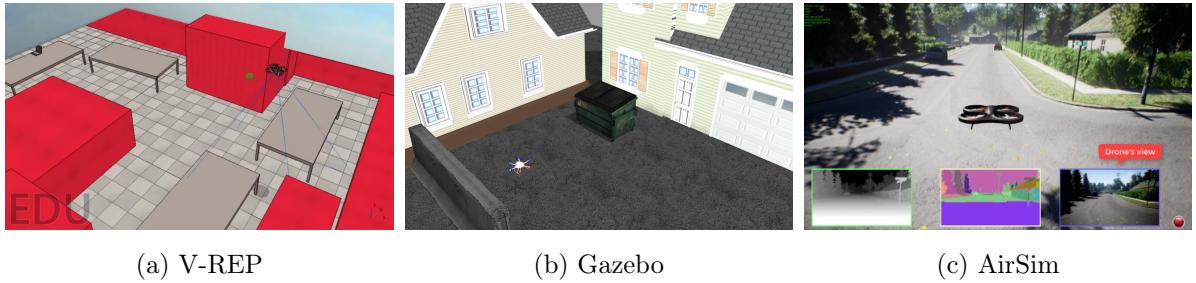


Figura 4: Renderizações em diferentes softwares de simulação

Portanto, escolhemos o simulador para veículos autônomos criado pela Microsoft AI & Research, AirSim [21]. Esse simulador foi desenvolvido na Unreal Engine — e possui atualmente uma versão experimental para a *engine* Unity — especificamente com o propósito de desenvolver simulações físicas e visualmente realistas [38]. Ressalte-se a crescente tendência no uso de mecanismos de renderização de jogos (*video game engines*) para aumentar o realismo na simulação robótica [39, 40, 41].

O AirSim disponibiliza APIs em C++ e Python que expõem funcionalidades para interagir com o *drone* — para recuperar imagens, controlar o veículo e obter informações de seu estado, como posição e ângulo de visão.

Além de 5 câmeras no *drone* (uma direcionada para baixo, uma para trás e três para frente), há 7 tipos de imagem que podem ser obtidas — dentre elas, podemos capturar imagens de profundidade e normais de superfície, que possibilitam comparações para avaliação do modelo 3D reconstruído. Também podem ser utilizados múltiplos veículos

simultaneamente, o que possibilita a extensão do estudo para *swarms* de *drones*.

## 8.2 Reconstrução

Diversos softwares de fotogrametria [42] para processamento de imagens aéreas estão disponíveis atualmente, porém muitos deles são pagos ou de código proprietário.

Dentre as soluções *open-source*, destacam-se: COLMAP [43], VisualSFM [44], Meshroom [31, 32] e OpenDroneMap.

Apesar de mais antigos, tanto o COLMAP como o VisualSFM não geram a textura para a cena reconstruída a partir das imagens utilizadas. Assim, acaba sendo necessário utilizar outros softwares complementares, como o OpenMVS, para obter-se um modelo final com malha 3D texturizada. Adicionalmente, o VisualSFM não engloba a etapa de *texture meshing*, i.e. a geração da malha poligonal.

Por outro lado, Meshroom é um software atual (iniciado em 2017) de reconstrução 3D baseado no framework de fotogrametria AliceVision, que implementa um conjunto de algoritmos estado-da-arte em visão computacional [31, 32]. O objetivo do Meshroom é prover interfaces visual, de linha de comando e em Python para a execução dos algoritmos disponíveis no AliceVision [45].

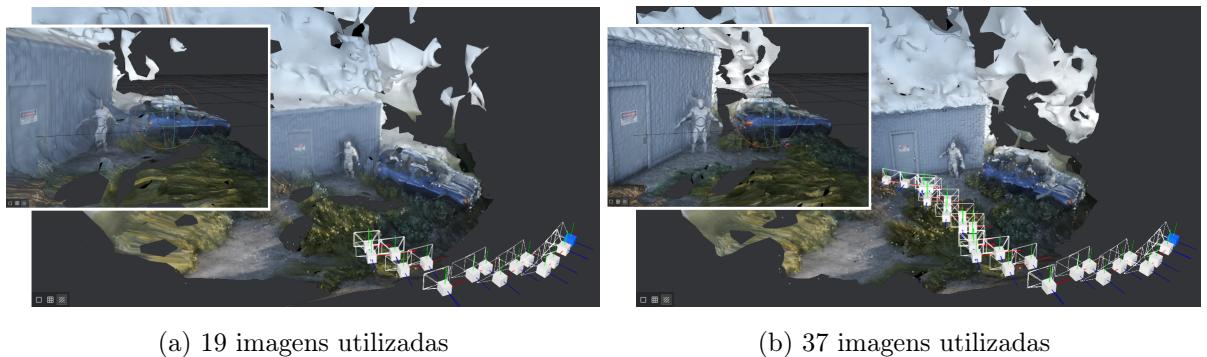


Figura 5: Reconstrução feita no Meshroom, a partir de imagens capturadas pelo AirSim

Inicialmente, realizamos testes de reconstruções das imagens obtidas em simulação com o Meshroom (veja Figura 5), pois além de compreender todo o *pipeline* necessário para a reconstrução [46], ele é bem mais amigável e direto de usar, comparado com o OpenDroneMap (que é efetivamente um “ecossistema” de aplicações para imagens aéreas composto por sete projetos, indo desde a coleta e o processamento até a análise e a exibição delas). Entretanto, para uso no contexto do projeto, foi necessário desmembrar

os elementos do sistema e criar um cliente único para controlar todo o sistema.

Por fim, ressaltamos que o OpenDroneMap parece ser também uma alternativa muito interessante, que pretendemos explorar durante o projeto, conforme citado na Seção 7.

### 8.3 Análise

Ao buscarmos por bibliotecas para o processamento de dados 3D, a Point Cloud Library (PCL) é uma das bibliotecas mais consolidadas [34]. Entretanto, além de possuir uma quantidade muito grande de funcionalidades (aumentando a curva de aprendizagem para utilizá-la corretamente), ela é feita em C++, o que diminui a agilidade com que conseguimos realizar pequenos experimentos. Há uma iniciativa de expor suas funcionalidades por meio de uma interface Python, porém, o que está disponível atualmente ainda é um conjunto muito pequeno. Assim, ao revisar a literatura encontramos dois principais projetos em Python: PyVista [47] e Open3D [48].

Apesar da facilidade de uso que tivemos com o PyVista, e de ele ser na verdade uma API de alto nível para o conhecido Visualization Toolkit (VTK [49]), o projeto possui uma comunidade menor de usuários, com quase 10 vezes menos *stars* no GitHub do que a biblioteca Open3D (277 e 2292, respectivamente).

Portanto, acabamos dando maior foco ao Open3D, que oferece um conjunto de reimplementações de estruturas de dados e algoritmos estado-da-arte em processamento de dados 3D, otimizados e configurados para paralelização.

Como nosso objetivo, além de determinar locais dos quais temos pouca informação, é também avaliar imagens capturadas desnecessariamente — pois assim evitamos voar por mais tempo do que necessário — extraímos da nuvem de pontos reconstruída pelo Meshroom a informação de quais pontos vieram de *features* de quais imagens (na etapa de *feature matching* [46]), conforme exemplificamos na Figura 6.

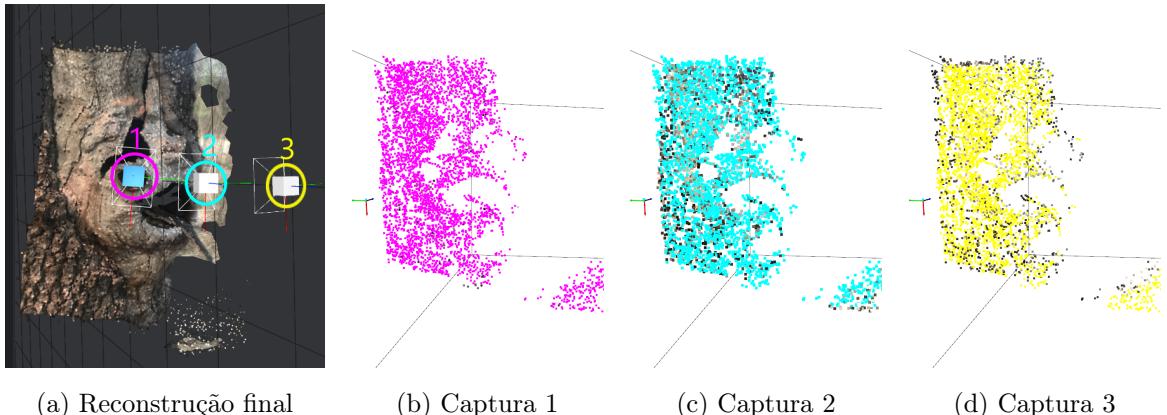


Figura 6: (a) Modelo final reconstruído pelo Meshroom a partir de 3 imagens, com suas posições de captura (b) (c) (d) Visualização da nuvem de pontos do modelo, colorida com base na contribuição de cada imagem capturada para a localização dos pontos gerados

A partir dessa informação, determinamos para cada foto os pontos visíveis que foram escolhidos como *features* pelo descriptor utilizado [50] e, reciprocamente, determinamos para cada ponto da nuvem a quais imagens ele corresponde. Assim, podemos descobrir que imagens mais colaboraram para a reconstrução final.

## 9 Sobre o Candidato

O candidato está no oitavo semestre do curso de Engenharia da Computação, sem exames ou reprovações. Possui coeficiente de rendimento igual a 0.8858 (3º entre 100 alunos, sendo o coeficiente médio da turma igual a 0.6168).

Antes de ingressar na Unicamp em 2016, o candidato concluiu curso técnico em Informática no Colégio Técnico de Campinas (Cotuca), iniciado em 2013 em conjunto com o Ensino Médio. Seu projeto de conclusão de curso envolveu um sistema embarcado de controle acesso, com autenticação em duas etapas: *tags* RFID e reconhecimento facial (utilizando a biblioteca OpenCV).

Durante todo seu primeiro ano na Unicamp, realizou estágio na empresa Movile como desenvolvedor Android, cumprindo 20 horas semanais e atuando em 8 aplicativos (atingindo um total de 1.5 milhões de usuários). Na Unicamp, o aluno já completou mais de 80% dos créditos necessários, e cursou diversas disciplinas eletivas (nos institutos de Computação, Biologia, Artes, e na faculdade de Engenharia Elétrica). Em particular, destacam-se as matérias: Introdução à Inteligência Artificial (MC906), Aprendizado de

Máquina (MC886), e Introdução ao Processamento de Imagem Digital (MC920); todas concluídas com nota superior a 9.5.

Atualmente, o candidato tem concentrado seus estudos no seu interesse pelas áreas de Computação Visual, particularmente em Processamento de Imagens e Computação Gráfica, e de Inteligência Artificial, com foco em Aprendizado de Máquina.

## Referências Bibliográficas

- [1] Wikipedia contributors, “3D reconstruction — Wikipedia, the free encyclopedia.” [2](#)
- [2] X.-F. Han, H. Laga, and M. Bennamoun, “Image-based 3D Object Reconstruction: State-of-the-Art and Trends in the Deep Learning Era,” *IEEE transactions on pattern analysis and machine intelligence*, 2019. [2](#)
- [3] Wikipedia contributors, “Real-time path planning — Wikipedia, the free encyclopedia.” [3](#)
- [4] M. Erdelj, E. Natalizio, K. Chowdhury, and I. Akyildiz, “Help from the Sky: Leveraging UAVs for Disaster Management,” *IEEE Pervasive Computing*, vol. 16, pp. 24–32, 01 2017. [3](#)
- [5] A. Restas, “Drone Applications for Supporting Disaster Management,” *World Journal of Engineering and Technology*, vol. 03, pp. 316–321, 01 2015. [3](#)
- [6] B. Sirmacek, R. Rashad, and P. Radl, “Autonomous uav-based 3d-reconstruction of structures for aerial physical interaction,” *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XLII-2/W13, pp. 601–605, 06 2019. [4](#)
- [7] C. Peng and V. Isler, “Adaptive view planning for aerial 3d reconstruction,” 05 2018. [5](#), [8](#)
- [8] Wikipedia contributors, “Set cover problem — Wikipedia, the free encyclopedia.” [5](#)
- [9] B. Hepp, M. Nießner, and O. Hilliges, “Plan3D: Viewpoint and Trajectory Optimization for Aerial Multi-View Stereo Reconstruction,” *ACM Trans. Graph.*, vol. 38, pp. 4:1–4:17, 2017. [5](#), [8](#), [11](#)

- [10] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, “Octomap: An efficient probabilistic 3d mapping framework based on octrees,” 2012. 5
- [11] A. Knapitsch, J. Park, Q.-Y. Zhou, and V. Koltun, “Tanks and temples: benchmarking large-scale scene reconstruction,” *ACM Trans. Graph.*, vol. 36, pp. 78:1–78:13, 2017. 6, 9, 11
- [12] S.-Y. Shin, Y.-W. Kang, and Y.-G. Kim, “Automatic Drone Navigation in Realistic 3D Landscapes using Deep Reinforcement Learning,” pp. 1072–1077, 04 2019. 6
- [13] B. G. Maciel-Pearson, L. Marchegiani, S. Akcay, A. Atapour-Abarghouei, J. Garforth, and T. P. Breckon, “Online Deep Reinforcement Learning for Autonomous UAV Navigation and Exploration of Outdoor Environments,” 2019. 6
- [14] M. Akhloufi, S. Arola, and A. Bonnet, “Drones Chasing Drones: Reinforcement Learning and Deep Search Area Proposal,” *Drones*, vol. 3, p. 58, 07 2019. 6
- [15] B. P. Duisterhof, S. Krishnan, J. J. Cruz, C. R. Banbury, W. Fu, A. Faust, G. C. H. E. de Croon, and V. J. Reddi, “Learning to Seek: Autonomous Source Seeking on a Nano Drone Microcontroller with Deep Reinforcement Learning,” 2019. 6
- [16] G. Muñoz, C. Barrado, E. Çetin, and E. Salamí, “Deep Reinforcement Learning for Drone Delivery,” *Drones*, vol. 3, p. 72, 09 2019. 6
- [17] M. Gschwindt, E. Camci, R. Bonatti, W. Wang, E. Kayacan, and S. Scherer, “Can a Robot Become a Movie Director? Learning Artistic Principles for Aerial Cinematography,” 2019. 6
- [18] R. Bonatti, W. Wang, C. Ho, A. Ahuja, M. Gschwindt, E. Camci, E. Kayacan, S. Choudhury, and S. Scherer, “Autonomous Aerial Cinematography In Unstructured Environments With Learned Artistic Decision-Making,” 2019. 6
- [19] H. Bayerlein, P. Kerret, and D. Gesbert, “Trajectory Optimization for Autonomous Flying Base Station via Reinforcement Learning,” pp. 1–5, 06 2018. 6
- [20] G. Barill, N. Dickson, R. Schmidt, D. I. Levin, and A. Jacobson, “Fast Winding Numbers for Soups and Clouds,” *ACM Transactions on Graphics*, 2018. 6
- [21] “AirSim.” <https://github.com/microsoft/AirSim>, 2017-2019. 6, 9, 13
- [22] Wikipedia contributors, “Lidar — Wikipedia, the free encyclopedia.” 7
- [23] Wikipedia contributors, “Inertial measurement unit — Wikipedia, the free encyclopedia.” 7

- [24] I. Alzugaray, L. Teixeira, and M. Chli, “Short-term UAV path-planning with monocular-inertial SLAM in the loop,” pp. 2739–2746, 05 2017. [7](#)
- [25] K. Celik, S.-J. Chung, M. Clausman, and A. Somani, “Monocular Vision SLAM for Indoor Aerial Vehicles,” vol. 2013, pp. 1566–1573, 10 2009. [7](#)
- [26] Wikipedia contributors, “Simultaneous localization and mapping — Wikipedia, the free encyclopedia.” [7](#), [10](#)
- [27] Wikipedia contributors, “Aircraft principal axes — Wikipedia, the free encyclopedia.” [7](#)
- [28] A. Kapoor, “Helping first responders achieve more with autonomous systems and AirSim,” <https://www.microsoft.com/en-us/research/blog/>, 2019. [9](#)
- [29] J. Langston, “How autonomous systems use AI that learns from the world around it,” <https://blogs.microsoft.com/ai/>, 2019. [9](#)
- [30] L. Weng, “Domain Randomization for Sim2Real Transfer,” [lilianweng.github.io/lil-log](http://lilianweng.github.io/lil-log), 2019. [9](#)
- [31] P. Moulon, P. Monasse, and R. Marlet, “Adaptive Structure from Motion with a Contrario Model Estimation,” in *Proceedings of the Asian Computer Vision Conference (ACCV 2012)*, pp. 257–270, Springer Berlin Heidelberg, 2012. [9](#), [14](#)
- [32] M. Jancosek and T. Pajdla, “Multi-view reconstruction preserving weakly-supported surfaces,” in *CVPR 2011*, IEEE, jun 2011. [9](#), [14](#)
- [33] T. T. Santos and L. V. Koenigkan, “Produção de ortomapas com vants e opendrone-map,” tech. rep., EMBRAPA, 2018. [10](#)
- [34] R. B. Rusu and S. Cousins, “3D is here: Point Cloud Library (PCL),” in *IEEE International Conference on Robotics and Automation (ICRA)*, (Shanghai, China), May 9-13 2011. [10](#), [15](#)
- [35] TUM Department of Informatics Computer Vision Group, “RGB-D SLAM Data-set and Benchmark.” <https://vision.in.tum.de/data/datasets/rbgd-dataset/tools#evaluation>. [10](#)
- [36] A. Kasar, “Benchmarking and Comparing Popular Visual SLAM Algorithms,” *CoRR*, vol. abs/1811.09895, 2018. [10](#)
- [37] L. Nogueira, “Comparative Analysis Between Gazebo and V-REP Robotic Simulators,” 12 2014. [13](#)

- [38] S. Shah, D. Dey, C. Lovett, and A. Kapoor, “AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles,” in *Field and Service Robotics*, 2017. 13
- [39] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, “CARLA: An open urban driving simulator,” in *Proceedings of the 1st Annual Conference on Robot Learning*, pp. 1–16, 2017. 13
- [40] W. Guerra, E. Tal, V. Murali, G. Ryou, and S. Karaman, “FlightGoggles: Photorealistic Sensor Simulation for Perception-driven Robotics using Photogrammetry and Virtual Reality,” *ArXiv*, vol. abs/1905.11377, 2019. 13
- [41] J. Greaves, M. Robinson, N. Walton, M. Mortensen, R. Pottorff, C. Christopherson, D. Hancock, J. Milne, and D. Wingate, “Holodeck: A High Fidelity Simulator,” 2018. 13
- [42] Wikipedia contributors, “Photogrammetry — Wikipedia, the free encyclopedia.” 14
- [43] J. L. Schönberger and J.-M. Frahm, “Structure-from-motion revisited,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 14
- [44] C. Wu, “VisualSFM : A Visual Structure from Motion System.” <http://ccwu.me/vsfm/>, 2019. 14
- [45] “Meshroom.” <https://github.com/alicevision/meshroom>, 2017-2019. 14
- [46] AliceVision, “Step by Step Photogrammetry Pipeline.” <https://alicevision.org/#photogrammetry>. 14, 15
- [47] C. B. Sullivan and A. Kaszynski, “PyVista: 3d plotting and mesh analysis through a streamlined interface for the visualization toolkit (VTK),” *Journal of Open Source Software*, vol. 4, p. 1450, may 2019. 15
- [48] Q.-Y. Zhou, J. Park, and V. Koltun, “Open3D: A modern library for 3D data processing,” *arXiv:1801.09847*, 2018. 15
- [49] Wikipedia contributors, “VTK — Wikipedia, the free encyclopedia.” 15
- [50] Wikipedia contributors, “Feature detection (computer vision) — Wikipedia, the free encyclopedia.” 16

Links acessados em Dezembro de 2019.