Pontifícia Universidade Católica
do Rio de Janeiro

Anderson Silva Fonseca

Sistema de anotação baseado em visualização 3D com imagens 360º de instalações industriais

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre pelo Programa de Pós–graduação em Informática da PUC-Rio.

Orientador : Prof. Marcelo Gattass
Co-orientador: Dr. Paulo Ivson Netto Santos

Rio de Janeiro
Agosto de 2022

Pontifícia Universidade Católica
do Rio de Janeiro

Anderson Silva Fonseca

Sistema de anotação baseado em visualização 3D com imagens 360º de instalações industriais

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre pelo Programa de Pós–graduação em Informática da PUC-Rio. Aprovada pela Comissão Examinadora abaixo:

Prof. Marcelo Gattass
Orientador
Departamento de Informática – PUC-Rio

Dr. Paulo Ivson Netto Santos
Co-orientador
Departamento de Informática – PUC-Rio

Prof. Waldemar Celes Filho
Departamento de Informática – PUC-Rio

Dr. Felipe Gomes de Carvalho
Departamento de Informática – PUC-Rio

Prof. Anselmo Cardoso de Paiva
UFMA

Rio de Janeiro, 29 de Agosto de 2022

Anderson Silva Fonseca

Graduado em ciência da computação pela Universidade Federal do Maranhão - UFMA.

Ficha Catalográfica

Silva Fonseca, Anderson

Sistema de anotação baseado em visualização 3D com imagens 360º de instalações industriais / Anderson Silva Fonseca; orientador: Marcelo Gattass; co-orientador: Paulo Ivson Netto Santos. – 2022.

52 f: il. color. ; 30 cm

Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2022.

Inclui bibliografia

1. Informática – Teses. 2. Gêmeo digital. 3. Modelos 3D. 4. Realidade aumentada. I. Gattass, Marcelo. II. Ivson Netto Santos, Paulo. III. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. IV. Título.

CDD: 004

Aos meus parentes e amigos
pelo apoio e encorajamento.

Agradecimentos

Primeiramente aos meus pais, Josimario Fonseca e Silva Helena, que sempre foram minha maior razão para continuar nos estudos e me incentivaram a descobrir novos caminhos.

Aos meus familiares mais próximos, em especial Ivana, Inaldo e Vitor, por terem me dado o apoio necessário para enfrentar os obstaculos e recomeçar mais uma etapa na vida.

A Nelia Reis, que me acompanhou que além de trilhar o mesmo caminho, me acompanhou durante todo o mestrado. Seja nos estudos, na preguiça e na comida.

A Mayara e Eduardo, que se aventuraram em vir ao Rio de Janeiro comigo como companheiros de casa. Agradeço por toda a companhia e pelo papo-furado de sempre.

Ao meu querido amigo Francisco, que sempre aparecia com milhões de dúvidas a serem respondidas e um tempo depois convidava para jogos que nunca vi na vida.

Aos meus companheiros de graduação, que mesmo remotamente não deixaram de me dar suporte em nenhum momento sequer.

Aos meus ex-petianos favoritos, que nunca me negaram um rolê quando voltava para São Luís.

Um carinho especial para Phillipe, Gilvan, Artur, Alexandre, Victor, Lucas e Arthur que sempre estavam disponíveis remotamente durante a pandemia para dar spoilers no meio da noite (e para perder toda e qualquer partida).

Aos companheiros do 6º Andar da Tecgraf, apesar de nunca ter participado da famosa reunião do pão de queijo, nunca deixaram que a pandemia levasse embora a animação de estar reunidos.

A Marcelo Gattass, Anselmo de Paiva e Paulo Ivson que agiram não só como orientadores, mas como amigos. Sempre dando o suporte ou o puxão de orelha necessário para eu conseguir andar com as minhas próprias pernas.

O Tecgraf e a PUC-Rio, pelos auxílios concedidos, sem os quais este trabalho não poderia ter sido realizado.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

Resumo

Silva Fonseca, Anderson; Gattass, Marcelo; Ivson Netto Santos, Paulo. **Sistema de anotação baseado em visualização 3D com imagens 360º de instalações industriais**. Rio de Janeiro, 2022. 52p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Com a chegada da Industria 4.0, empresas aderiram a usar gêmeos digitais para melhorar seus processos de produção e as condições de trabalhos de seus empregados. Os gêmeos digitais são normalmente associados a modelos tridimensionais, permitindo a realização de planejamentos, extração de dados, simulação e treinamento a partir das condições reais. Infelizmente, gêmeos digitais incorretos ou desatualizados podem induzir a erros e a desencontro de informações o que retira todas as vantagens do processo de virtualização, arruinando quaisquer comparativos com a realidade. Em contrapartida, gêmeos digitais ricos em informação permitem que simulações e extrações de dados sejam mais fieis a realidade. Atualmente, as tecnologias capazes de enriquecer as informações de gêmeos digitais são escassos, pois é um procedimento que leva tempo devido a necessidade de análises de especialistas, custos, equipamentos e ferramentas específicas. Recursos como fotografias 360°, vídeos e modelos tridimensionais podem ser usados para realizar uma avaliação e atualização nos gêmeos digitais. Porém, diferenças temporais, condições do ambiente e erros humanos entre os recursos podem gerar confusão durante a transferência e conexão da informação. Este trabalho apresenta uma ferramenta que explora as vantagens de combinar fotografias 360º com modelos 3D para gerar gêmeos digitais *as-built*. Cada imagem pode ser ajustada a uma localização dentro do sistema de coordenadas do modelo, inclusive permitindo alterações nos eixos e no campo de visão. Durante a navegação, é possível navegar livremente pelo modelo e pelas posições de interesse criadas pelo usuário. Além da visualização, a ferramenta propõe uma interação mais eficaz para realizar anotações entre modelos e fotografias 360° com o propósito de verificar consistências ou agregar novas informações ao gêmeo digital. Estas interações são importantes para a inspeção e manutenção, como avaliação de peças, análise das condições atuais ou a criação de comparativos entre o planejado e o real.

Palavras-chave

Gêmeo digital; Modelos 3D; Realidade aumentada.

Abstract

Silva Fonseca, Anderson; Gattass, Marcelo (Advisor); Ivson Netto Santos, Paulo (Co-Advisor). **Annotation system based on 3D visualization with 360º images of industrial installations**. Rio de Janeiro, 2022. 52p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

With the arrival of Industry 4.0, companies have adopted digital twins to improve their production processes and the working conditions of their employees.' Digital twins are generally associated with three-dimensional models and allow planning, data extraction, simulation, and training based on current conditions. Unfortunately, incorrect or outdated digital twins can lead to errors and information mismatch, which takes away all the advantages of the virtualization and computerization process, ruining any comparisons with reality. In contrast, information-rich digital twins allow simulations and data extraction to be more faithful to reality. Currently, technologies capable of enriching the information of digital twins are scarce, as it is a procedure that takes time due to the need for expert analysis, costs, equipment, and specific tools. Resources such as 360° photographs, videos, and 3D models can be used to perform an evaluation and update the digital twins. However, temporal differences, environmental conditions, and human errors between the images and the model can generate confusion during the transfer and connection of information. This work presents a tool that explores the advantages of combining 360º photographs with 3D models to generate as-built digital twins. Each image can be adjusted to a location within the model's coordinate system, allowing changes to axes and field of view. During navigation, it is possible to navigate the model and the user-created positions of interest freely. In addition to visualization, the tool proposes a more effective interaction to annotate between models and 360° photographs to verify consistency or add new information to the digital twin. These interactions are essential for inspection and maintenance, such as evaluating parts, analyzing current conditions, or creating comparisons between planned and actual.

Keywords

Digital twin; 3D Models; Augmented reality.

Sumário

Lista de algoritmos

Lista de Abreviaturas

1D – Unidimensional

2D – Bidimensional

3D – Tridimensional

CAD – Computer Aided Design

CPS – Sistemas Ciber Físicos (*Cyber-Physical System*)

FOV – Campo de Visão (*Field of View*)

FPSO – Unidade flutuante de armazenamento e transferência (*Floating production storage and offloading*)

GLTF – *Graphics Language Transmission Format*

IoS – Internet de Serviços (*Internet of Services*)

IoT – Internet das Coisas (*Internet of Things*)

Km – Quilômetro

OBJ – A wavefront format

SFM – *Structure from motion*

The mystery of life isn't a problem to solve,
but a reality to experience.

Frank Herbert, *Dune.*

1
Introdução

Atualmente, as empresas focadas em melhorar seu processo de produção, o ambiente de trabalho de seus funcionários, prever condições de riscos e automatizar seus procedimentos recorrem as tecnologias da Industria 4.0 (XU; XU; LI, 2018). O segmento da indústria inteligente proporcionou a virtualização, integração, análise e simulação como um fator positivo para melhorar aspectos como otimização de processos, aumento da produtividade de fabricação, redução de custos e aumento da eficiência operacional (GHOBAKHLOO, 2020).

A indústria 4.0 representa um conjunto de tecnologias de automação voltadas para a produção de recursos, muitas vezes são relacionadas a sistemas ciber físicos (CPS), Internet das Coisas (IoT), Internet de Serviços (IoS) e computação em nuvem (RODRIGUES; JESUS; SCHÜTZER, 2016). Porém, no momento atual, ainda existem inúmeros desafios científicos, tecnológicos, econômicos e sociais que impedem que uma empresa ingresse totalmente na indústria inteligente (ZHOU; LIU; ZHOU, 2015).

Um dos segmentos da indústria 4.0 é a virtualização, neste segmento são criadas cópias do mundo físico a partir de sensores de monitoramento vinculadas a modelos de simulação (RODRIGUES; JESUS; SCHÜTZER, 2016). Estes modelos, também conhecidos como Gêmeos digitais, possuem o potencial de armazenar quaisquer informações que só poderiam ser obtidas se observadas fisicamente (PIRES et al., 2019). Um gêmeo digital pode ser montado a partir de dados de sensores, modelos tridimensionais (3D), imagens ou inteligência artificial.

De maneira geral, para usar gêmeos digitais para análise, simulação ou treinamento são usados suas representações visuais, como os modelos 3D, capazes de distinguir entidades através de formas geométricas e pela possibilidade de associá-los a outros tipos de informação como análises de especialistas, informações de sensoriamento, identificadores de peças ou quaisquer informações que possam ser usadas para a virtualização.

O gêmeo digital apresenta vantagens como monitoramento remoto de processos industriais, planejamento de atividades, avaliação, diagnóstico, extração de dados, automação e outras vantagens (PIRES et al., 2019). Porém, estes benefícios são úteis apenas se os dados extraídos de sua representação estão de acordo com a realidade. Um grande desafio dos gêmeos digitais é conseguir se adaptar ao tempo, a incerteza, a imprecisão e as variações do espaço

físico. Um outro desafio é quando incoerências entre as representações visuais e a realidade aparecem, pois identificá-las e atualizá-las requerem um grande esforço dependendo do que precisa ser alterado (TAO; ZHANG, 2017).

Enriquecer um gêmeo digital com informações novas e atuais é um processo difícil, pois envolve ter acesso a outros tipos de mídia, dados de sensores, análise de especialistas, custos e equipamento. O trabalho se torna ainda mais complicado quando as informações não são atualizadas constantemente ou necessita que funcionários tenham que ir fisicamente ao local para coletar informações. Outro obstáculo é quando as informações precisam ser coletadas em locais de risco como plataformas de petróleo, usinas, pontes ou arranha-céus. Uma alternativa para não enviar um especialista a um local de risco é usar recursos que armazenam informações como vídeos, fotografias, fotografias 360º e geolocalização. Entretanto, inconsistências entre estes diferentes tipos de mídia tornam a transferência e a conexão da informação um trabalho árduo. Além de que, são poucas as aplicações capazes de lidar com vários tipos de mídia de maneira conjunta para construir um gêmeo digital consistente, pois não apresentam soluções para lidar com a variabilidade e a incerteza da realidade.

O trabalho a ser apresentado foca em explorar os benefícios de combinar fotografias 360º de instalações industriais com modelos 3D para identificar inconsistências e geração de uma base de dados anotada que possa servir de auxílio para a construção de gêmeos digitais como construído (*as-built*).

Este trabalho desenvolveu uma interface capaz de mostrar um gêmeo digital baseado em um modelo 3D de uma plataforma de petróleo em conjunto com uma fotografia 360º. E, se propôs a apresentar uma maneira eficaz de realizar anotações entre os modelos 3D e as imagens, com a finalidade de verificar a consistências do gêmeo digital para acrescentar ou atualizar novas informações a ele.

Este documento está estruturado da seguinte maneira: no capítulo 2 são apresentados alguns trabalhos relacionados que usaram abordagens similares; no capítulo 3 são apresentados os conceitos relevantes para a construção do sistema; no capítulo 4 apresentamos a solução proposta, incluindo todas as etapas necessárias até a implementação; no capitulo 5 são apresentados os resultados, junto com casos de usos; no capítulo 6 são apresentadas as conclusões e as expectativas para trabalhos futuros.

2
Trabalhos relacionados

Tendo em foco a detecção de mudanças temporais em estruturas, podemos citar o trabalho de Sakurada, Okatani e Deguchi (2013). Eles propuseram um método de detecção de mudanças em estruturas tridimensionais recriadas a partir de imagens 360º capturadas em tempos diferentes. As imagens foram adquiridas usando uma câmera omnidirecional montada em um veículo que percorreu uma área afetada por um desastre natural, cada sequência de imagens foi submetida a algoritmos de registro de imagem (*image registration*). Para obter as reconstruções geométricas foi aplicado um algoritmo de *Structure from motion* (SFM). Para encontrar as possíveis mudanças nas estruturas em diferentes tempos, as reconstruções foram comparadas usando a profundidade em relação ao posicionamento da câmera.

Voltando para corrigir problemas em modelos 3D, Taneja, Ballan e Pollefeys (2013) propuseram um método para detecção de mudanças na geometria a partir de imagens panorâmicas do *Google Street View*. Esse trabalho usou um gêmeo digital de uma cidade de baixa qualidade e um conjunto de imagens de uma área de 6 quilômetros (km). Em resumo, são apresentadas uma hipótese para lidar com inconsistências da geometria a partir da reconstrução do cenário usando *image registration*. São apresentadas também uma maneira de estimar e corrigir as posições da câmera a partir da geolocalização. E por fim, uma maneira de lidar com imagens em movimento que podem acusar falsos positivos durante a execução do algoritmo.

Logo depois, os mesmos autores apresentaram uma proposta para corrigir as geometrias incorretas usando imagens (TANEJA; BALLAN; POLLEFEYS, 2015). Nesse trabalho inicialmente detecta-se o contorno dos prédios a partir dos modelos 3D. Usando os *pixels* das imagens panorâmicas, o posicionamento da câmera é ajustado até encontrar a melhor posição que encaixe os *pixels* dos prédios ao contorno gerado pelo modelo. Para remover inconsistências, como pedestres e veículos, usou-se técnicas de *image registration* para gerar um mapa de inconsistências e realizar uma segmentação. Ao fim do algoritmo, são retornadas as posições corrigidas de cada fotografia. Em seus resultados, usou-se imagens em diferentes tempos e em diferentes posições e seu algoritmo foi capaz de gerar uma máscara que detecta as possíveis alterações nas estruturas.

Palazzolo e Stachniss (2018) apresenta um trabalho para encontrar alterações geométricas a partir de uma sequência de imagens. Seu método inicialmente estima a posição da câmera de cada imagem a partir de uma

referência geográfica. As imagens são projetadas no modelo 3D e são associadas a uma matriz de covariância. Se existir alguma alteração real no cenário durante o tempo de aquisição das fotografias, elas serão detectadas a partir dos *pixels* que se diferem dos valores armazenados anteriormente. Após identificar quais *pixels* de cada imagem não se assemelham ao original, é feito um cálculo inverso para estimar a área espacial que possui alterações. Seu trabalho foi usado para detectar mudanças geométricas de maneira rápida em aplicativos móveis.

Existem também aplicações voltadas para navegação em cenário de realidade aumentada usando imagens panorâmicas e modelos 3D. No trabalho de Pereira, Moud e Gheisari (2017) foi criado um estudo para construir panoramas interativos. No seu estudo foi usado diversos vídeos e fotos panorâmicas de um local de construção, além de usarem um modelo 3D e a planta do local. Seu protótipo propõe em mostrar um sistema de navegação onde você pode ir pra determinados locais da planta do local clicando em posições específicas do panorama. Você também tem acesso a posição geográfica e a visão do modelo 3D exibidas lado-a-lado em cantos da tela. Cada panorama também pode possuir textos e botões que podem ser adicionadas pelo usuário. Seu trabalho foi construído para ser exibido em ambiente de fácil acesso, como navegadores, sem a necessidade de possuir um equipamento adicional.

Outra aplicação voltada para navegação é mostrada em um artigo de Barboza et al. (2019). Esse trabalho é voltado para exibir um gêmeo digital de uma unidade flutuante de armazenamento e transferência (FPSO). Um ou mais usuários são inseridos dentro de um ambiente de realidade virtual colaborativo, através da representação 3D é possível observar valores reais como dados de sensores, históricos de estado e informações sobre componentes e peças. Dentro do ambiente virtual o usuário também dispõe de uma interface *point-and-click* onde ele pode emitir sinais diretamente para a peça ou realizar medidas baseada em valores reais.

Pessoa et al. (2017) apresentaram uma ferramenta para construção de ambientes de realidade virtual de uma subestação de energia. Na primeira etapa deste trabalho, foram usados um conjunto de imagens panorâmicas esféricos que eram previamente processados, cada imagem recebia um ID único e tinha determinados pontos de interesse marcados. Cada marcação era associada a um determinado equipamento da subestação, durante a conexão da associação eram disponibilizadas as informações da peça, seu posicionamento e seu estado atual, por fim as informações adicionais eram conectadas aos panoramas. Na segunda etapa, o usuário era convidado a entrar no ambiente virtual usando um *Head-mounted display* (HMD), nesse ambiente era possível nave-

gar entre os diferentes panoramas e verificar as condições dos equipamentos, mapeados durante a etapa anterior, em tempo real. Apesar de ser simulado, também era possível realizar testes e operar sobre as máquinas cadastradas.

Por fim, Abdeen e Sepasgozar (2022) apresentam uma estratégia para participação da comunidade na criação de gêmeos digitais de cidades inteligentes. Sua arquitetura se distribuem em cinco camadas, são elas: camada de aquisição de dados, onde são adquiridos modelos 3D da cidade, dados de arruamento e as informações coletadas pela comunidade sobre locais públicos e privados; camada de transmissão de dados, responsável por tratar as informações recebidas; camada de modelagem, responsável por criar um modelo digital das construções, associá-las ao gêmeo digital e conectar as informações; camada de integração, responsável por realizar a simulações, análises e extração de dados para algoritmos de redes neurais; por fim, a camada de aplicação, responsável por exibir as recomendações da comunidade, locais seguros e inseguros, eventos, etc. A construção de gêmeos digitais com essa estratégia pode ser usada para auxiliar nas decisões de policiais, desenvolvedores, funcionário de trânsito, etc.

3
Fundamentação teórica

3.1
Gêmeos digitais

Um dos princípios desejados pelas empresas que adotam a indústria 4.0 é que seus produtos, equipamentos e processos se tornem controláveis remotamente de forma autônoma, sendo possível simular situações, extrair informações reais, realizar treinamento, prever problemas ou, em caso de eventuais falhas, serem notificadas em tempo hábil. Este é o princípio da Virtualização (RODRIGUES; JESUS; SCHÜTZER, 2016) e está altamente associado a representações como os gêmeos digitais.

Um gêmeo digital é descrito por Kahlen, Flumerfelt e Alves (2017) como um conjunto de informações virtuais que descreve completamente um produto físico real ou parcialmente real de um nível micro atômico até o nível macro geométrico. Em seu melhor estado, quaisquer informações obtidas ao inspecionar este produto no mundo físico também podem ser obtidas através de seu gêmeo digital. Um gêmeo digital pode ser usado em conjunto com IoT como soluções em computação em nuvem ou para simulação e treinamento como explanados por Lee et al. (2013), como uma representação que opera dentro da nuvem em paralelo com processos reais e simula as suas condições físicas a partir de sensores, algoritmos de análise ou qualquer outro meio que provem informação dos dados físicos. Stark, Kind e Neumeyer (2017) descrevem de maneira geral que gêmeos digitais são a representação digital única de um recurso, seja ela produto, máquina, serviço, sistema ou algum recurso intangível, que descreve com exatidão suas propriedades, condição e comportamento por meio de modelos, informações e dados.

Como podemos observar na Figura 3.1, representamos um diagrama do funcionamento ideal de um gêmeo digital. No mundo físico, um determinado equipamento se comunica através de sensores através de dados. Esses dados são interpretados e agregados ao modelo virtual. O melhor resultado obtido a partir de suas análises retornam uma ação ideal executada por seus atuadores no mundo físico. Todas as funcionalidades funcionando em conjunto é chamada de integração.

Existem muitas vantagens ao usar gêmeos digitais em processos industriais, como por exemplo, avaliar decisões de produção, verificar a performance do processo como um todo, controlar e configurar máquinas remotamente, di-

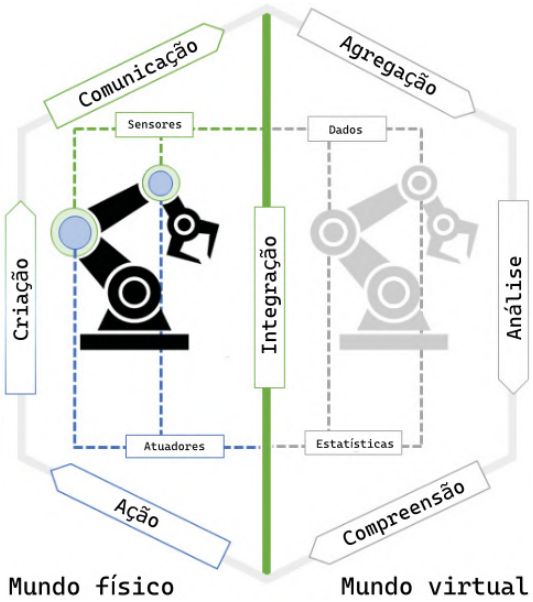


Figura 3.1: Diagrama de funcionamento de um gêmeo digital. Adaptado de: Parrott e Warshaw (2017).

agnosticar comportamentos de equipamentos e conectar sistemas e processos para monitorar e otimizar tarefas. Essas vantagens podem se estender para outras aplicações de outras áreas como análise em tempo real, simulação, treinamento, suporte para decisão, planejamento ou extração de dados (PIRES et al., 2019).

Ainda que os gêmeos digitais tragam uma diversidade de benefícios, assim como a maioria das tecnologias da indústria inteligente, ainda são necessários um conjunto de ferramentas poderosas o suficiente para alcançar e explorar com exatidão todas as possibilidades deste conceito (XU; XU; LI, 2018). Um dos desafios de usar gêmeos digitais é a necessidade de estarem sempre atualizados, um caso ideal por exemplo, é possuir conexão constante com nuvem, comunicando sobre o estado atual de seus componentes, seja a partir de um equipamento ou a partir de sensores. Um outro desafio é conseguir ligar com a variabilidade, incerteza e imprecisão do mundo físico, visto que modelos de construções devem ser altamente fieis a realidade para que os dados sejam corretos. O desafio se torna ainda mais difícil quando são encontradas inconsistências entre o real e o virtual, pois identificar, quantificar e corrigir não é algo trivial. Outro problema é conectar diversos tipos de dados de sensoriamento diferente, pois diferentes tipos de mídia de um mesmo gêmeo digital podem ter representações virtuais diferentes. Tratar cada tipo de dado diferente para incrementar o gêmeo digital e anexar novas informações de maneira automática ainda é um grande desafio (TAO; ZHANG, 2017).

Na literatura, os gêmeos digitais são frequentemente usados a partir de

sua representação visual. Sua representação mais comum são as modelagens 3D, que são associados a informações adicionais. Durante simulações e treinamentos são usados principalmente sua representação geométrica para situar os funcionários sobre a forma, a localização e o estado atual de uma determinada entidade observada. Alguns trabalhos como os de Lohtander et al. (2018) e Dang et al. (2018) usam modelos 3D como principal meio de suporte aos gêmeos digitais, como mostrado na Figura 3.2.

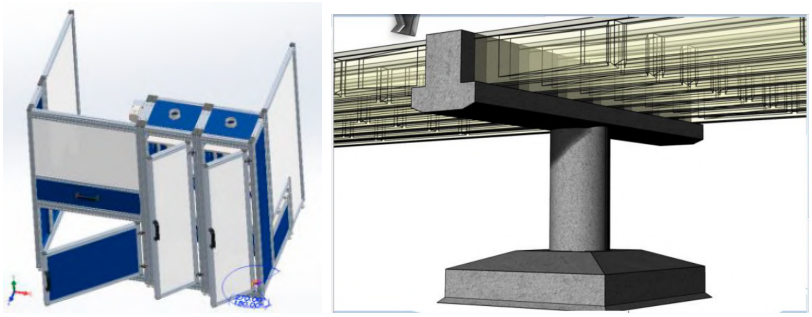


Figura 3.2: Gêmeos digitais de uma máquina e de uma ponte, respectivamente. Cada um dos modelos 3D são mostrados em seus respectivos visualizadores. Adaptado de: Lohtander et al. (2018) e Dang et al. (2018).

3.2 Definições de Câmera

Usualmente os modelos gráficos são criados com um sistema de coordenadas conveniente, um cubo por exemplo, pode ser modelado como um cubo unitário que armazena em si suas coordenadas x, y e z. Esse sistema de coordenadas é chamado de espaço do objeto ou espaço da modelagem. Uma cena, por sua vez, pode ser composta por um ou mais objetos, como representações geométricas, luzes ou modelos 3D. Cada objeto dessa cena está posicionado em algum local específico, nesse caso cada objeto é submetido a diferentes transformações em seus vértices. As coordenadas resultantes dessas transformações são chamadas de espaço de coordenados do mundo (*world space*) (HUGHES et al., 2014).

As câmeras são consideradas como um dispositivo que observa os elementos do cenário a partir de seu ponto de vista e os traduz para uma representação bidimensional (2D). A partir desse mesmo conceito existem as câmeras virtuais. Como cada objeto de uma cena virtual, ela possui uma localização e uma direção nas coordenadas do mundo. A representação dos eixos da câmera é normalmente representada a partir dos eixos x, y, z no qual x aponta para a direita da câmera (vista por trás), y aponta para cima da câmera e z aponta para a parte de trás da câmera. Logo, o ponto de observação da câmera

fica posicionado no eixo negativo de z. Todos os objetos da cena também são transformados ao serem exibidos pela câmera. Essas novas coordenadas são chamadas de espaço de coordenadas da câmera. A Figura 3.3 exibe como seria a orientação da câmera virtual fictícia.

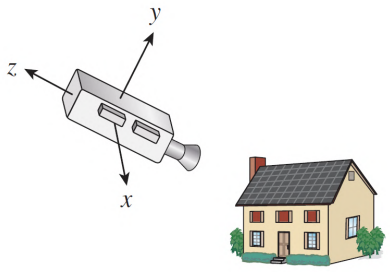


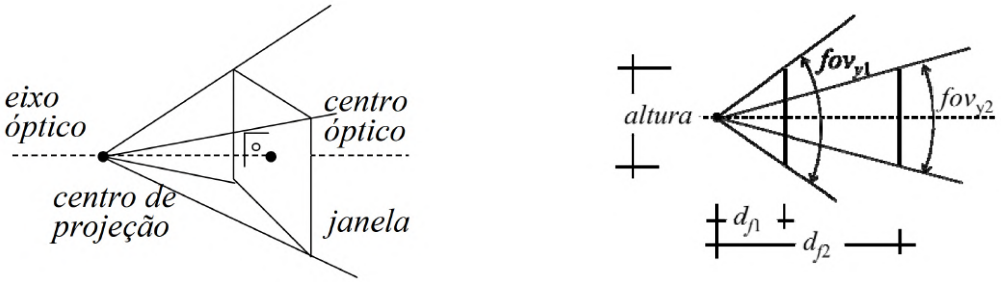
Figura 3.3: A câmera virtual observando a cena a partir de uma localização e orientação específicas. Fonte: Hughes et al. (2014).

Ao falarmos de câmeras virtuais podemos dividir seus parâmetros em dois tipos (TSAI, 1987):

- Parâmetros extrínsecos: correspondem as transformações sofridas pela geometria da câmera do espaço de coordenadas do mundo. São definidas pela sua posição no espaço (x, y, z) e pela sua orientação (definido pelos ângulos de Euler - yaw ψ, roll ϕ, pitch θ).
- Parâmetros intrínsecos: correspondem as características óticas e geométricas da câmera (distância focal, fatores de escala, distorção das lentes, coordenadas da projeção em relação ao plano da imagem).

Em suma, uma câmera virtual simples herda suas características de uma câmera *pinhole*. Cada câmera tem um centro de projeção, um eixo óptico e um plano onde se forma uma imagem de altura e largura específicos. A Figura 3.4(a) exibe um projeção cônica da câmera, o eixo óptico é perpendicular a esse plano e o intercepta exatamente em seu centro. O tamanho da área escolhida do plano e a sua distância ao centro de projeção definem a abertura da câmera ou *field of view* (fov), como podemos ver na Figura 3.4(b) (GATTASS, 2013).

Esse modelo simples de câmera é bastante usado pela comunidade acadêmica e é bastante usando por aplicações gráficas. Desafios relacionados a calibração automática de câmera usualmente se definem em encontrar a melhor combinação de parâmetros extrínsecos e intrínsecos. O grande desafio dessa área é conseguir estimar esses valores tendo conhecimento apenas das coordenadas da imagem e das coordenadas (x, y, z) dos objetos observados (TSAI, 1987).



(a) Projeção cônica. (b) Abertura da câmera

Figura 3.4: Exemplos de câmera simples. Fonte: Gattass (2013).

As câmeras virtuais simples são bastante usadas durante o algoritmo de rastreamento de raios, pois contém os principalmente elementos óticos necessários. Como podemos observar na seção abaixo.

3.3 Rastreamento de raios

Os tipos de câmera mais simples de serem reproduzidas são as câmeras *pinhole*. Neste tipo de câmera um papel fotográfico é colocado dentro de uma caixa a prova de luz. Um pequeno buraco é feito na frente da caixa do lado oposto ao papel fotográfico. Em seguida esse buraco é preenchido com algum material opaco. Para retirar a foto basta deixar a caixa imóvel e retirar o material opaco do buraco. A luz que entra pelo furo acerta o papel fotográfico causando uma reação química. Quando estiver pronto, basta colocar de volta o material opaco sobre o buraco. A Figura 3.5 exibe uma câmera *pinhole*. A necessidade de usar um pequeno furo é porque limitados a entrada de luz vinda de todas as direções, impedindo que o papel fotográfico seja exposto (GLASSNER, 1989).

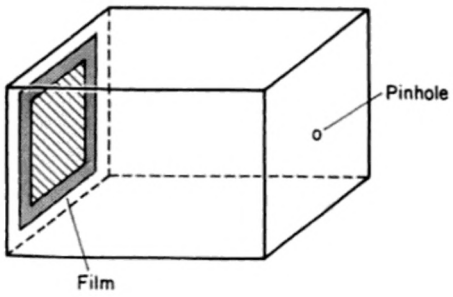
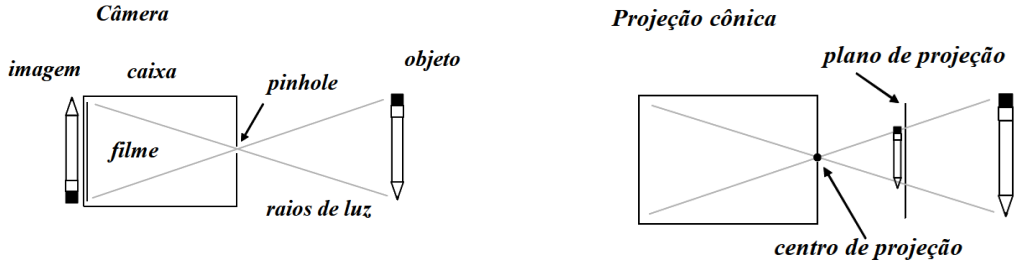


Figura 3.5: Representação de uma câmera *pinhole*. Fonte: Glassner (1989).

Ainda que existem modelos mais complicados, o processo geométrico de formação de uma imagem a partir de uma câmera *pinhole* é equivalente a projeção cônica dos objetos iluminados no plano que contêm a imagem

(GATTASS, 2013). Na câmera textitpinhole o centro da projeção é onde está localizada o furo, os raios de luz entram invertidos pelo orifício e são gravados no papel, como mostrada na Figura 3.6(a). Matematicamente, podemos ver a imagem projetada no papel fotográfico se colocarmos o plano de projeção entre o centro de projeção e o objeto, como mostrado na Figura 3.6(b).



(a) Geometria da câmera *pinhole* (b) Projeção cônica

Figura 3.6: Comparação entre a geometria da câmera e a projeção cônica. Fonte: Gattass (2013).

Essa é a representação mais simples de algoritmo de rastreamento de raios. Um observador (câmera) se posiciona em frente a uma tela plana transparente, de sua posição são emitidos diversos raios que vão atravessar cada *pixel* e acertar quaisquer objetos que estejam na cena. Quando um raio atinge um objeto, identificamos a cor do objeto e desenhamos no pixel que foi atravessado pelo raio (SILVA, 1999). O Algoritmo 1 exemplifica o funcionamento do rastreamento de raios.

Algoritmo 1: Algoritmo de rastreamento de Raios

```
para cada pixel de uma janela* faça
    Calcule uma linha reta que sai do centro de projeção até o pixel
    para cada objeto na cena faça
        Verifique a interseção desta reta com o objeto
        se se a interseção for a mais próxima do centro então
            Armazene a interseção
    se existir uma interseção então
        Calcule a cor do pixel baseado no objeto** atingido
        Pinte o pixel com a cor calculada
```

**janela* denota a área 2D do plano de projeção. ***objeto* pode ter outras influências de luzes que atingem aquele ponto.

Adaptado de: Gattass (2013) e Silva (1999).

Podemos definir os parâmetros extrínsecos da câmera no algoritmo de raios a partir das seguintes propriedades: a distância focal, correspondente a

distância entre o plano de projeção e o centro; o plano de projeção, formado por uma determinada altura e largura, como mostra Figura 3.7.

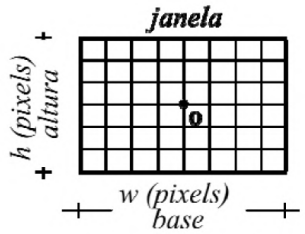


Figura 3.7: Parâmetros extrínsecos da câmera. Fonte: Gattass (2013).

O ângulo de abertura, mostrado na Figura 3.4(b), pode ser calculado seguindo a relação da Equação 3-1; esse ângulo costuma variar entre 60º a 90º em câmeras convencionais. Quanto maior o angulo, maior o espaço que a cena é enquadrada na lente.

$$\frac{altura}{distancia_f} = \tan(\frac{fov_y}{2}) \tag{3-1}$$

A janela, ou plano de projeção corresponde a imagem de saída, seu tamanho é definido pelo números de *pixels* que devem existir na horizontal e na vertical. A razão de aspecto (*aspect ratio*) é dada pela Equação 3-2.

$$aspect = \frac{base}{altura} \tag{3-2}$$

Outros algoritmos que implementam este tipo de câmera, incluem também as propriedades perto *near* e longe *far*. Estas propriedades normalmente são usadas para especificar uma profundidade do que deve ou não ser vista pelo observador.

Para definir os parâmetros intrínsecos da câmera devemos saber onde estão a posição da câmera, a direção que câmera está olhando e a rotação da câmera em torno dessa direção. Por convenção chamamos a posição do centro de projeção da câmera de *eye*, o ponto para onde a câmera está olhando de *center* e um vetor direção que indique a parte de cima da câmera de *up*, como mostra a Figura 3.8. Esse vetor *up* sempre vai estar no plano y e z.

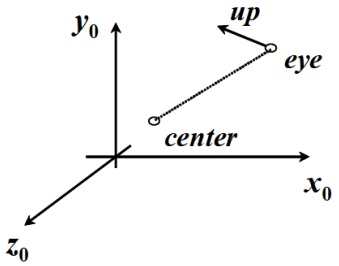


Figura 3.8: Parâmetros extrínsecos da câmera. Fonte: Gattass (2013).

Para realizar o lançamento dos raios precisamos determinar o sistema de coordenadas do *eye* que podemos definir pela valores x_e, y_e e z_e mostradas na Equação 3-3. z_e pode ser adquirida a partir da reta que passa pelo *eye* e pelo *center*. x_e pode ser calculada a partir do vetor *up*, já que ele está sobre o plano $y_e z_e$. Por fim, podemos determinar o unitário de y_e a partir de x_e e z_e. A Figura 3.9 mostra a relação entre os vetores x_e, y_e e z_e.

$$\begin{aligned} z_e =& \frac{1}{\|eye - center\|}(eye - center) \\ x_e =& \frac{1}{\|up - z_e\|}(up - z_e) \\ y_e =& x_e \times z_e \end{aligned} \tag{3-3}$$

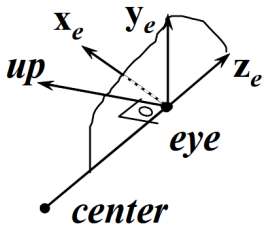


Figura 3.9: Relação entre os vetores x_e, y_e e z_e. Fonte: Gattass (2013).

Precisamos determinar quais raios vão ser lançados a partir do centro de projeção para isso precisamos determinar duas funções simples $u(x)$ e $v(y)$ que vão ser usadas para acomodar as coordenadas de imagem, a Equação 3-4 mostra como ela pode ser calculada. Esses valores determinam os *pixels* que vão ser desenhados na imagem, como ilustra a Figura 3.10(a).

$$\begin{aligned} u(x) =& \frac{x}{w}(base) \\ v(y) =& \frac{y}{h}(altura) \end{aligned} \tag{3-4}$$

O vetor do raio pode ser definido pela Equação 3-5, como ilustra a Figura 3.10(b).

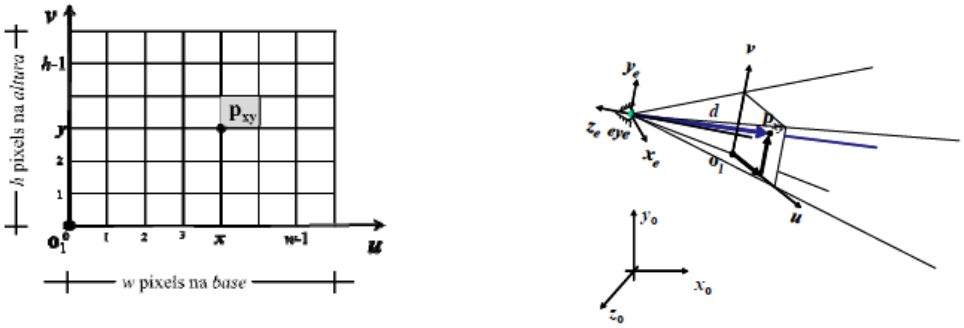
$$d = p_{xy} - eye \tag{3-5}$$

O valor do p_{xy} pode ser calculado a partir da origem do sistema do plano de projeção de uv com adição dos valores $u(x)$ e $v(y)$, como mostra a Equação 3-6. A origem o pode ser calculada a partir dos vetores unitários e o centro de projeção dada pela Equação 3-7.

$$p_{xy} = o + u(x)\hat{\mathbf{u}} + v(y)\hat{\mathbf{v}} \tag{3-6}$$

$$o = eye - d_f\hat{\mathbf{z}}_e - \frac{altura}{2}\hat{\mathbf{y}}_e - \frac{base}{2}\hat{\mathbf{x}}_e \tag{3-7}$$

Assim podemos calcular o vetor do raio substituindo os valores da Equação 3-5, resultando na Equação 3-8.



(a) Escala de uv em relação aos pixels da janela

(b) Esquema de geração do raio

Figura 3.10: Exemplo de lançamento de raio. Fonte: Gattass (2013).

$$d = -d_f \hat{\mathbf{z}}_e + altura(\frac{y}{h} - \frac{1}{2})\hat{\mathbf{y}}_e + base(\frac{x}{w} - \frac{1}{2})\hat{\mathbf{x}}_e \tag{3-8}$$

Por fim, precisamos calcular as interseções dos raios que atingem as superfícies dos objetos. O cálculo da superfície pode ser calculado de maneira diferente dependendo do tipo de modelo do objeto. Objetos do tipo implícito apresentam uma álgebra convencional, isto é, satisfazem uma condição para calcular a interseção. Geometrias básicas como esfera, cone e toro possuem uma equação bem definida. Objetos do tipo fronteira normalmente são formadas por um conjunto de malhas de triângulos, no qual sua superfície é descrita diretamente.

Seguindo o exemplo da esfera, a determinação da interseção de um raio como o objeto pode ser obtida se procurarmos os valores que satisfaçam a equação implícita da esfera, dada pela Equação 3-9. A Figura 3.11 simula a interseção do raio na esfera. Ao substituirmos, encontramos um sistema de equação de segundo grau simples. Se encontrarmos um valor que seja menor que 0 significa que o raio não atravessa a esfera. Se o valor for igual 0 significa que raio tangencia a superfície da esfera. Se for maior ou igual a 0 o raio intercepta no menor valor das raízes.

$$\begin{aligned} \|\mathbf{p}(t_i) - \mathbf{c}\|^2 &= r^2 \\ (\mathbf{d}\cdot\mathbf{d})t_i^2 + [2\mathbf{d}\cdot(\mathbf{o}-\mathbf{c})]t_i + [(\mathbf{o}-\mathbf{c})\cdot(\mathbf{o}-\mathbf{c}) - r^2] &= 0 \end{aligned} \tag{3-9}$$

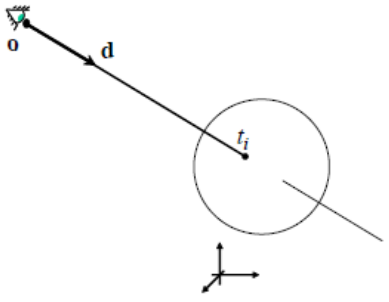


Figura 3.11: Interseção entre o raio e a esfera. Fonte: Gattass (2013).

Para determinar a interseção de um raio com um triângulo de vértices $\mathbf{p}_1\mathbf{p}_2\mathbf{p}_3$ verificamos se o raio atravessa algum ponto do triângulo e em seguida verificamos se o ponto pertence a parte interior ou exterior do triângulo, como ilustramos na Figura 3.12.

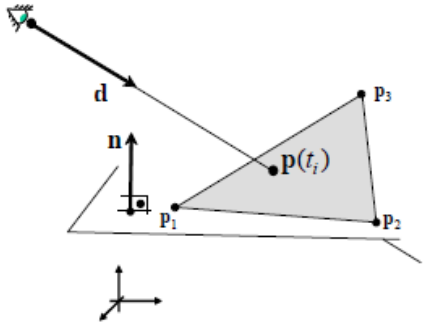


Figura 3.12: Interseção entre o raio e o triângulo. Fonte: Gattass (2013).

Por convenção, para encontrar a parte visível, isto é do exterior do triângulo, seguimos a ordem trigonométrica, dessa forma podemos calcular a parte exterior a partir da normal, como mostrada na Equação 3-10.

$$\hat{\mathbf{n}} = \frac{(p_2 - p_1) \times (p_3 - p_2)}{\|(p_2 - p_1) \times (p_3 - p_2)\|} \tag{3-10}$$

Para que um ponto pertença ao raio e esteja no mesmo plano da superfície do triângulo, ele deve satisfazer a seguinte condição de ortogonalidade demonstrado pela Equação 3-11. Ao expandirmos a equação e isolarmos o valor de t_i podemos chegar ao ponto de interseção com o plano que contém o raio que é dado pela Equação 3-12.

$$[p(t_i - p_1)] \cdot \hat{\mathbf{n}} = 0 \tag{3-11}$$

$$\begin{aligned} t_i d \cdot \hat{\mathbf{n}} + (o - p_1) \cdot \hat{\mathbf{n}} &= 0 \\ \frac{(p_1 - o) \cdot \hat{\mathbf{n}}}{d \cdot \hat{\mathbf{n}}} &= t_i \end{aligned} \tag{3-12}$$

A partir do ponto, podemos verificar se ele se encontra no interior do triângulo. Um exemplo simples é usar as coordenadas baricêntricas, basta verificar a área de cada sub triângulo gerado como demonstrado na Equação 3-13. Caso o ponto p_i esteja no interior, a divisão da área dos sub triângulos pelo área total resultará nos valores L_1, L_2 e L_3 que serão maiores ou iguais a 0 e menores ou iguais a 1, como demonstrado pela Equação 3-14. A Figura 3.13 ilustra o teste de interseção do triângulo.

$$\begin{aligned} A_1 &= \hat{\mathbf{n}} \cdot \frac{[v_{23} \times (p_i - p_2)]}{2} \\ A_2 &= \hat{\mathbf{n}} \cdot \frac{[v_{31} \times (p_i - p_3)]}{2} \\ A_3 &= \hat{\mathbf{n}} \cdot \frac{[v_{12} \times (p_i - p_1)]}{2} \end{aligned} \tag{3-13}$$

$$
\begin{aligned}
L_1 &= \frac{A_1}{A_t} \\
L_2 &= \frac{A_2}{A_t} \\
L_3 &= \frac{A_3}{A_t}
\end{aligned}
\tag{3-14}
$$

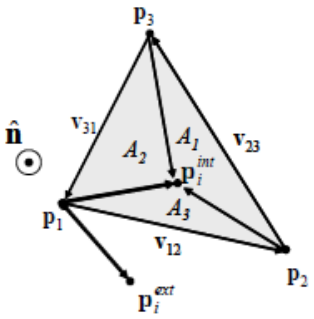


Figura 3.13: Teste do interior do triângulo usando as coordenadas baricêntricas. Fonte: Gattass (2013).

Após a determinação do ponto de interseção, deve ser determinado a cor do *pixel*. Naturalmente, quando calculamos esses valores consideramos que na cena existam modelos de iluminação. Nesses casos ao determinarmos a cor do *pixel* deve considerar como a luz é refletida na superfície até ela chegar na câmera. O modelo mais simples de iluminação usado no rastreamento de raios é o modelo de iluminação de Phong (PHONG, 1975).

3.4 Projeção equiretangular

As vezes representar a realidade através de uma única foto não é o suficiente. Artistas criam artes e vídeos que podem ser vistas pela frente e por trás simultaneamente. Desenvolvedores de jogos criam cenários imersivos que envolvem todas as direções do cenário. Usuários de mapas online podem ver em tempo real os arredores de um local. Esse tipo de imagem é conhecido como imagem panorâmica ou panorama. Um panorama é definido como uma visão ininterrupta de toda uma área circundante (SALOMON, 2006).

Toda imagem usada como panorama apresenta algum tipo de distorção. Linhas retas se tornam curvas e formas de identificar os *pixels* de uma imagem precisam ser tratadas. Em alguns casos imagens se tornam irreconhecíveis ao serem vistas de maneira casual. Existem três tipos principais de projeção panorâmica são as cilíndricas, esféricas e cúbicas.

A projeção equiretangular, também conhecida por projeção esférica, é um tipo de projeção que simula uma esfera colocada envolta de um observador, onde tudo que é visto pertence ao fundo da esfera, isto é, o material que a compõe corresponde a um cenário em sua volta. Se supormos que o material da esfera é um papel de parede, ao colocarmos dentro de um plano o material terá diversas distorções, como observado pela Figura 3.14.

Figura 3.14: Exemplo de um panorama equiretangular. Fonte: Ben Kreunen (2021)

Uma vez que um ponto cartesiano de um plano denotado por x e y é projetado na superfície da esfera, ele se torna um ponto com coordenadas polares, ou seja, passa a ser identificado pelos valores de latitude e longitude. Podemos observar a partir da Figura 3.15, que as coordenadas esféricas (θ, ϕ) correspondem às coordenadas horizontais e verticais (x,y). A coordenada θ varia de $-\pi$ a π, e a coordenada ϕ varia de $-\pi/2$ a $\pi/2$ (LIU et al., 2018).

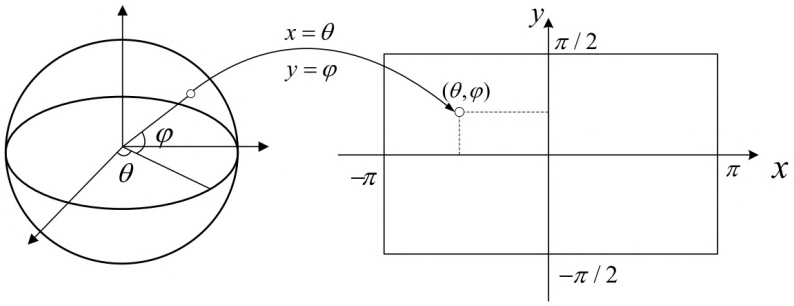


Figura 3.15: Mapeamento de superfície esférica para plano cartesiano. Fonte: Liu et al. (2018).

Em uma cena, onde temos uma câmera como centro da projeção, precisamos levar em consideração como relacionar o sistema de coordenadas do espaço para o sistema das coordenadas polares, incluindo o fato que em uma imagem não existem eixos negativos como no plano cartesiano. Observe a Figura 3.16, dado um ponto $P_1(x_1, y_1, z_1)$ que representa o ponto que está sendo observado, P_1' é o ponto de projeção de P_1 no plano $x - y$. Considere que O é o ponto de origem ou centro da projeção e o raio r é definido pela distância entre O e P_1. A longitude θ, que corresponde ao angulo do eixo x, passa a variar de 0 a 2π. A latitude ϕ, que corresponde ao angulo do eixo z, passa a variar de 0 a π. A relação das coordenadas esféricas (θ, ϕ) e coordenadas do mundo podem ser descritas pela Equação 3-15 (TSAI; KAO; LIU, 2010).

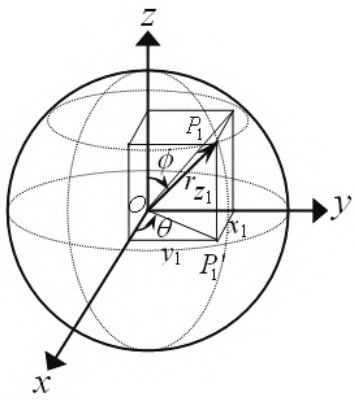


Figura 3.16: Sistema de coordenadas esférica. Fonte: Tsai, Kao e Liu (2010).

$$\begin{aligned} x_1 &= \mathbf{r} \cos\theta \sin\phi \\ y_1 &= \mathbf{r} \sin\theta \cos\phi \\ z_1 &= \mathbf{r} \cos\phi \end{aligned} \tag{3-15}$$

3.5 Mapeamento de textura

Na computação gráfica, a texturização é um processo que pega uma superfície e modifica sua aparência em cada local usando uma imagem, função ou outra fonte de dados (AKENINE-MO et al., 2018). Usualmente, texturas são usadas para descrever como a cor varia de uma posição para outra ou como uma imagem se distribui na superfície de um objeto (GATTASS, 2013).

Mapeamento de textura significa o mapeamento de uma função em uma superfície 3D. O domínio da função pode ser uma, duas, ou três dimensões e pode ser representada por uma matriz ou função matemática. Por exemplo uma textura unidimensional (1D) pode simular estratos de rocha; uma textura 2D pode simular ondas, vegetações ou elevações na superfície; uma textura 3D pode representar nuvens, madeira, mármore etc (HECKBERT, 1986).

Podemos também usar uma imagem como uma textura em uma superfície de um objeto 3D, esta imagem é mapeada para a imagem de destino (tela) pela projeção de visualização. O espaço da textura é rotulado (u, v), o espaço do objeto é rotulado (x_o, y_o, z_o) e o espaço da imagem é rotulado (x, y).

Observe as texturas 2D na Figura 3.17. Usando o exemplo da imagem com uma determinada resolução w x h, podemos associar uma textura na posição ($u = 0$, $v = 0$) a cor do pixel ($x = 0$, $y = 0$) e a posição ($u = 1$,$v = 1$) a cor do pixel ($w - 1$, $h - 1$). Precisamos definir um processo de interpolação para fornecer a textura a partir de qualquer valor real que obedeça $0 \geq u \geq 1$ e $0 \geq v \geq 1$.

Uma localização no espaço é o ponto de partida para o processo de texturização. Esse local pode estar no espaço do mundo, mas está fixado ao objeto 3D, de modo que se o objeto se move, a textura se move junto com ele. Esse ponto no espaço então tem uma função projetora aplicada a ele para obter as coordenadas de textura, que serão usadas para descobrir qual é o

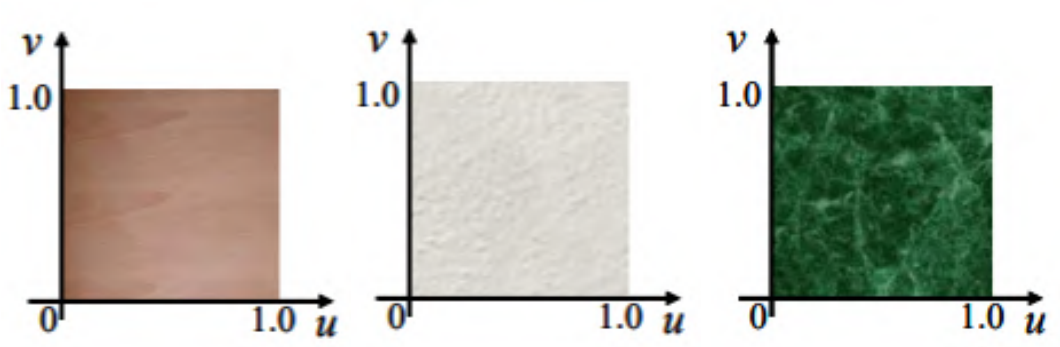


Figura 3.17: Texturas 2D. Fonte: Gattass (2013).

pixel da imagem nesse local. Tendo como exemplo a Figura 3.18, observamos um ponto (x, y, z) no espaço do mundo com valor $(-2.3, 7.1, 88.2)$, o valor retornado após a função projetora dada por u e v vai estar entre 0 e 1, para o exemplo digamos que retorne $(0.32, 0.29)$. Supondo que a nossa imagem tenha a resolução de 256x256, calculamos que a localização do *pixel* é dada por $(81.92, 74.24)$, ignorando os decimais, a partir da locação obtemos o valor do *pixel* que representa uma cor específica para a textura (AKENINE-MO et al., 2018).

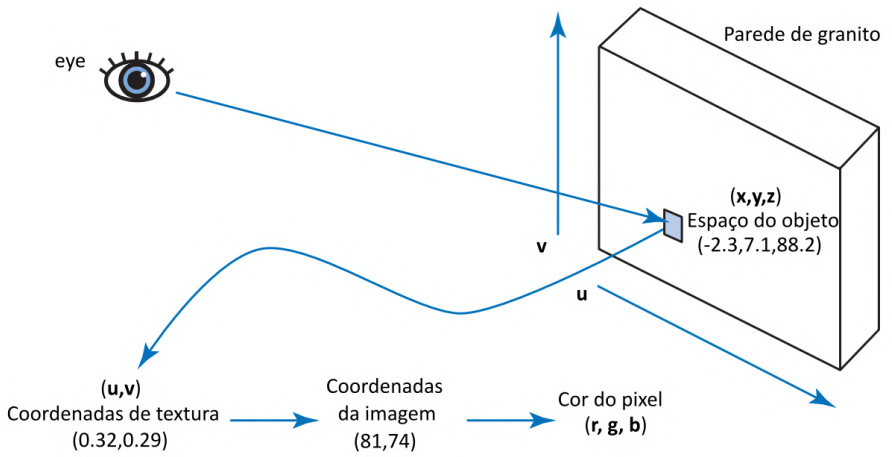


Figura 3.18: Etapas do mapeamento de textura. Adaptado de: Akenine-Mo et al. (2018).

As funções de projeção podem variar de acordo com a necessidade, objetos que possuem implícitos podem usar a sua equação interna, como as esferas. Alguns casos especiais como cubos podem dividir cada face e associa-los a um sistema de coordenadas (u, v).

Além destas funções, é possível inserir outros valores que incrementam a cor da textura. Essas funções alteram como as coordenadas de cores são lidas pelo mapeamento de textura e normalmente são usadas em domínios 3D. As mais comuns são a texturas de rugosidade (*bump textures*), que mudam a normal e consequentemente o modo como são calculados os componentes difusos e especulares; texturas de deslocamento (*displacement mapping*), que

apresentam um efeito de deslocamento nos pontos do objeto; texturas de ambiente (*environment maps*), que dão um feito de fundo a um cenário, como se fosse uma caixa de areia ou um esfera envolvente.

3.6 Textura projetiva

As texturas também podem ser utilizadas para adicionar informações visuais a partir de fontes de luz, permitir distribuição de diferentes cores e de foco, como um holofote com uma imagem desenhada. Para fontes de luz que têm toda a sua iluminação limitada ou focada em uma única cor, texturas projetivas podem ser usadas para projetar uma imagem em um objeto da cena (AKENINE-MO et al., 2018).

Produzir uma imagem de uma cena tridimensional requer encontrar a projeção dessa cena em uma tela bidimensional. No caso de uma cena que consiste em superfícies mapeadas com textura, isso envolve não apenas determinar onde os pontos projetados das superfícies devem aparecer na tela, mas também quais partes da imagem de textura devem ser associadas aos pontos projetados. Isso permite criar holofotes texturizadas, luzes padronizadas e o efeito de projeção, como observado na Figura 3.19 (SEGAL et al., 1992).

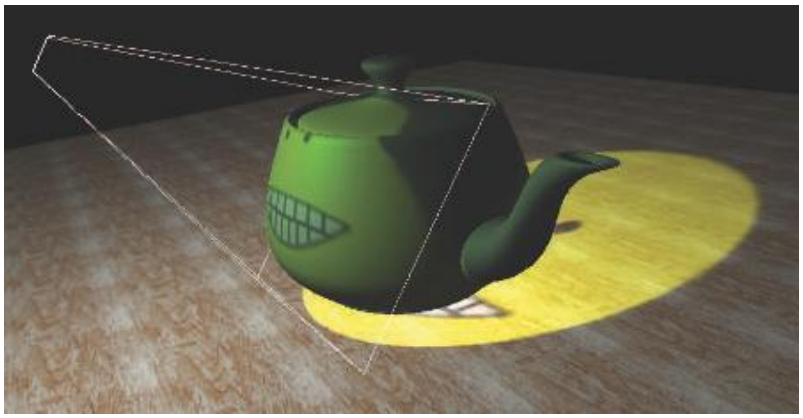


Figura 3.19: Textura projetiva de uma luz. Fonte: Akenine-Mo et al. (2018).

Para a matemática inicial precisamos introduzir um sistema de coordenada para um clipe. Essa coordenada será homogênea (x,y,z,w) e sua origem será o ponto de visão. Também será preciso introduzir o sistema de coordenadas da tela representando uma janela bidimensional com duas coordenadas (x, y). Esses são obtidos a partir das coordenadas do clipe dividindo x e y por w, de modo que as coordenadas da tela sejam dadas pelas equações 3-16. As coordenadas de luz vão ser um segundo sistema de coordenadas homogênea (x^l,y^l,z^l,w^l) onde a origem do sistema é a fonte da luz. E por fim, temos o sistema de coordenadas da textura que aqui é dada pela Equação 3-17 que irá representar a nossa imagem projetada. Nosso objetivo então será dado um ponto de (x^s, y^s), encontrar a sua correspondente em (x^t, y^t), como mostra Figura 3.20.

$$\begin{aligned} x^s &= \frac{x}{w} \\ y^s &= \frac{y}{w} \end{aligned} \tag{3-16}$$

$$\begin{aligned} x^t &= \frac{x^t}{w^t} \\ y^t &= \frac{y^t}{w^t} \end{aligned} \tag{3-17}$$

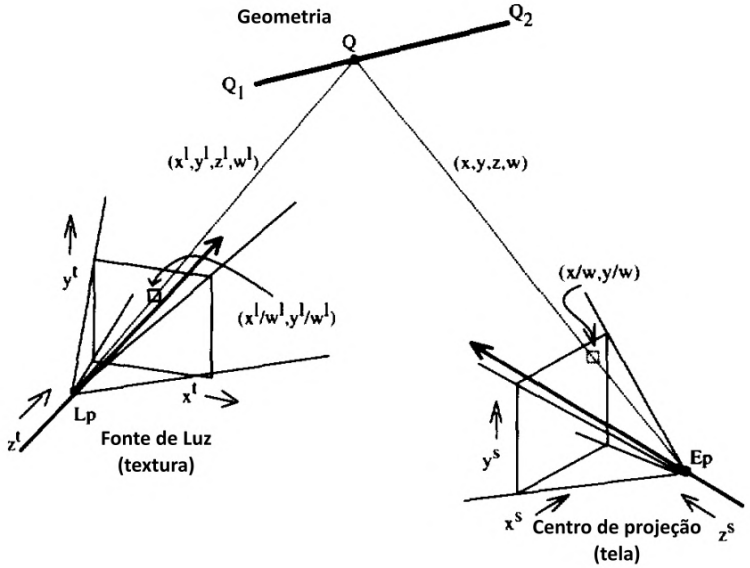


Figura 3.20: Representação dos sistemas de coordenadas ao calcular a textura projetiva. Fonte: Akenine-Mo et al. (2018).

Nas coordenadas do clipe, os pontos no final dos segmentos de linhas são dados pela Equação 3-18. Um ponto Q ao longo do segmento de linha pode ser escrito em coordenadas de clipe como a Equação 3-19, onde $t \in [0, 1]$. O mesmo vale para o ponto projetado, como mostra Equação 3-20, onde $Q_1^s = Q_1/w_1$ e $Q_2^s = Q_2/w_2$.

$$\begin{aligned} Q_1 &= (x_1, y_1, z_1, w_1) \\ Q_2 &= (x_2, y_2, z_2, w_2) \end{aligned} \tag{3-18}$$

$$Q = (1-t)Q_1 + tQ_2 \tag{3-19}$$

$$Q^s = (1-t^s)Q_1^s + t^s Q_2^s \tag{3-20}$$

Para encontrar a coordenada da luz de Q dado o valor Q^s precisamos encontrar t que corresponde a t^s, temos então a seguinte Equação 3-21. Podemos solucionar facilmente escolhendo um valor a e b tais que $1 - t^s = a/(a+b)$ e $t^s = b/(a+b)$. Também escolhemos um valor para A e B tais que $1 - t = A/(A+B)$ e $t = B/(A+B)$. A Equação 3-22 demonstra o resultado da substituição, assim conseguimos facilmente verificar se $A = aw_2$ e $B = bw_1$ satisfazem a equação, permitindo obter t e então Q.

$$\begin{aligned} Q^s =& (1-t^s)Q_1/w_1 + t^s Q_2^s \\ Q^s =& \frac{(1-t)Q_1 + tQ_2}{(1-t)w_1 + tw_2} \end{aligned} \tag{3-21}$$

$$\begin{aligned} Q^s =& \frac{aQ_1/w_1 + bQ_2/w_2}{(a+b)} \\ Q^s =& \frac{AQ_1 + BQ_2}{Aw_1 + Bw_2} \end{aligned} \tag{3-22}$$

Como a relação entre o sistema de coordenadas da luz e do clipe são afins, nós temos uma matriz homogênea M que as relaciona, como observado na Equação 3-23, onde $Q_1^l = (x_1^l, y_1^l, z_1^l, w_1^l)$ e $Q_2^l = (x_2^l, y_2^l, z_2^l, w_2^l)$ são as coordenadas das luzes dadas pelos pontos obtidos de Q_1 e Q_2.

$$\begin{aligned} Q^l =& MQ \\ Q^l =& \frac{A}{A+B}Q_1^l + \frac{B}{A+B}Q_2^l \end{aligned} \tag{3-23}$$

Por fim conseguimos obter as coordenadas de textura correspondentes a um ponto interpolado linearmente pelo segmento da reta nas coordenadas da tela, como demonstra a Equação 3-24.

$$\begin{aligned} Q^t =& Q^l/w^l \\ Q^t =& \frac{AQ_1^l + BQ_2^l}{Aw_1^l + Bw_2^l} \\ Q^t =& \frac{aQ_1^l/w_1 + bQ_2^l/w_2}{a(w_1^l/w_1) + b(w_2^l/w_2)} \end{aligned} \tag{3-24}$$

Texturas podem ser adicionadas a qualquer tipo de luz para permitir efeitos visuais adicionais. Luzes texturizadas permitem fácil controle da iluminação pelos artistas, que podem simplesmente editar a imagem usada. Alguns exemplos do uso de textura projetiva são simular projeção de slides, criar sombras a partir de imagens e holofotes texturizados.

4
Solução Proposta

Este capítulo descreve como foi elaborada a solução proposta. São explanados a natureza dos dados, a arquitetura da solução e as implementações necessárias para criar um ambiente capaz de exibir em conjunto imagens panorâmicas e modelos 3D em um cenário de realidade aumentada.

A aplicação possui três pontos relevantes: A base de dados, responsável por recuperar as informações dos modelos e transformar os modelos 3D em uma representação mais leve para exibição; arquitetura, onde são explanados a construção do ambiente para aplicação funcionar; e a implementação, voltada para a geração de cenários, onde o usuário pode criar ou editar um cenário, aplicar correções no posicionamento das imagens, navegar pelo cenário e criar anotações no modelo 3D e nas imagens 360º.

4.1
Base de dados

Foram fornecidos duas bases de dados diferentes para trabalhar. São fornecidos inicialmente os gêmeos digitais de um navio-plataforma FPSO durante a etapa de planejamento. Esses modelos foram usados no início do projeto e não são constantemente atualizados. Cada modelo pode conter algumas informações únicas como nome da peça, informações de segurança, peças de auxílio, rotas de segurança e outras informações. Outro detalhe importante, os modelos foram exportados a partir de um software CAD (*Computer Aided Design*) para o formato *Wavefront* (OBJ), um formato simples e de fácil implementação.

Os modelos 3D por sua vez possuem um grande nível de detalhes, ocasionado um arquivo muito pesado para se trabalhar em visualizadores convencionais. Foi necessário transformar o objeto original em uma versão mais leve e compacta que poderia guardar tanto a informação essencial do gêmeo digital quanto sua informação gráfica. O formato de arquivo GLTF (KHRONOS GROUP, 2021a), criado por Khronos Group (2021a), foi escolhido porque pode armazenar informações como hierarquia de nós, câmeras, luzes e materiais. Também é possível associá-los a informações geométricas como vértices, índices, *shaders*, animações e outras informações gráficas (ROBINET et al., 2014). A conversão para GLTF permite que modelos tridimensionais sejam exibidos em uma página da web, o ambiente de desenvolvimento da aplicação. A Figura 4.1 mostra uma parte do navio-plataforma após a conversão.

Além da conversão, durante o pré-processamento dos modelos são extraídas informações sobre cada peça incluída no modelo e atribuídas a um identificador único. As informações são limitadas a um identificador único da peça e um texto previamente informado pelo arquivo original. Todo objeto também mantém referências ao modelo original, para caso de consultas mais específicas.

Outro tipo de dado utilizado são as fotografias 360º, conforme a Figura 4.2. As fotografias foram capturadas a partir de uma câmera especial capaz

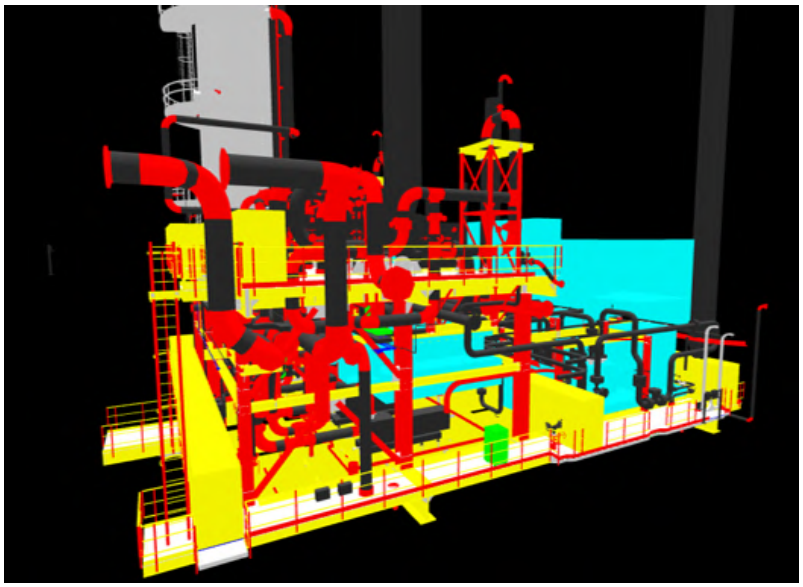


Figura 4.1: Modelo 3D de uma parte do navio-plataforma FPSO. Fonte: Próprio autor.

de criar um panorama completo. Cada fotografia mostra um local específico do navio-plataforma já em funcionamento e em alto-mar. Diferente dos modelos, as fotografias são atualizadas com mais frequências e podem apresentar alterações mais recentes. Cada imagem também recebe uma coordenada que indica onde ela está possivelmente posicionada que está no mesmo sistema de coordenadas da sua planta.

Ao total são usadas nessa aplicação dados de gêmeos digitais de dois navio-plataformas diferentes, totalizando 355 fotografias 360º e 61 modelos 3D.

4.2 Arquitetura

A aplicação proposta foi criada para ser simples aos olhos de especialistas. Onde cada usuário pode conseguir inferir anotações e simular cenários em qualquer tipo de ambiente. Por essa natureza foi escolhido usar um ambiente de desenvolvimento *web*. No qual o usuário pode acessar os modelos 3D e as imagens 360º em qualquer computador convencional com acesso à internet.

Como descrito na seção 4.1 os dados precisaram passar por um pré-processamento inicial para que o formato do dado não seja pesado ao ponto que a navegação seja prejudicada. Logo, boa parte dos recursos precisam ser tratados em um servidor dedicado. A arquitetura se divide em dois lados: o servidor, dedicado a tratar todos os recursos da aplicação, como o pré-processamento, tratamento dos modelos, tratamento das imagens e controle dos cenários criados pelos usuários; e o cliente, onde fica a aplicação responsável por toda interface da aplicação, incluindo a exibição do cenário, do modelo, da calibração e da criação de anotações.

As tecnologias usadas para o servidor foi o *framework Flask* (GRINBERG, 2018) e para a aplicação foram usadas *WebGL* (KHRONOS GROUP, 2021b) e *three.js* (DIRKSEN, 2013). A Figura 4.3 ilustra a arquitetura da apli-

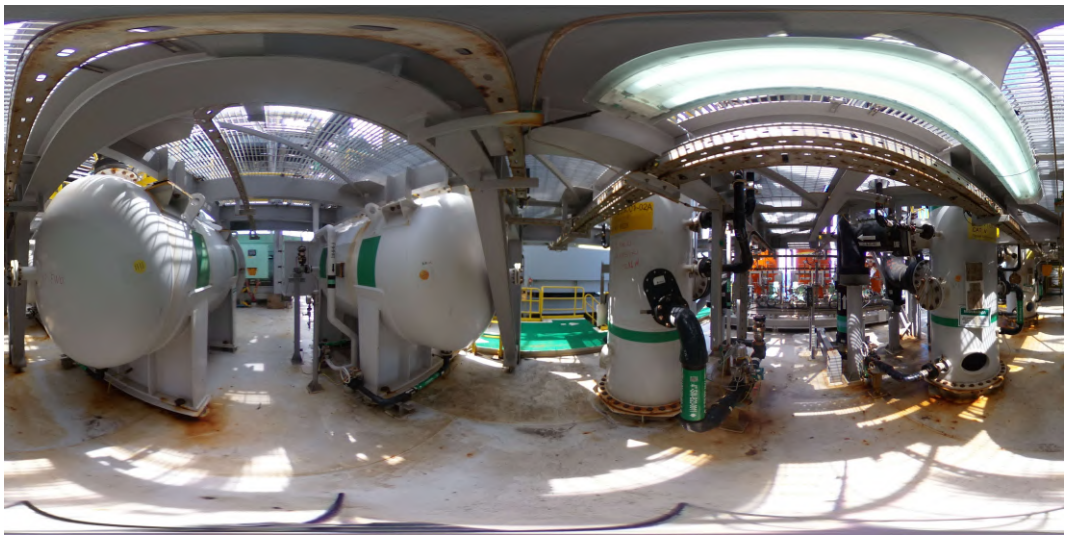


Figura 4.2: Panorama equiretangular do navio-plataforma FPSO. Fonte: Próprio autor.

cação. Em vermelho estão destacadas as operações realizadas no servidor e em azul as operações realizadas no cliente. As etapas de processamento, visualização e anotação serão descritos na seção a seguir.

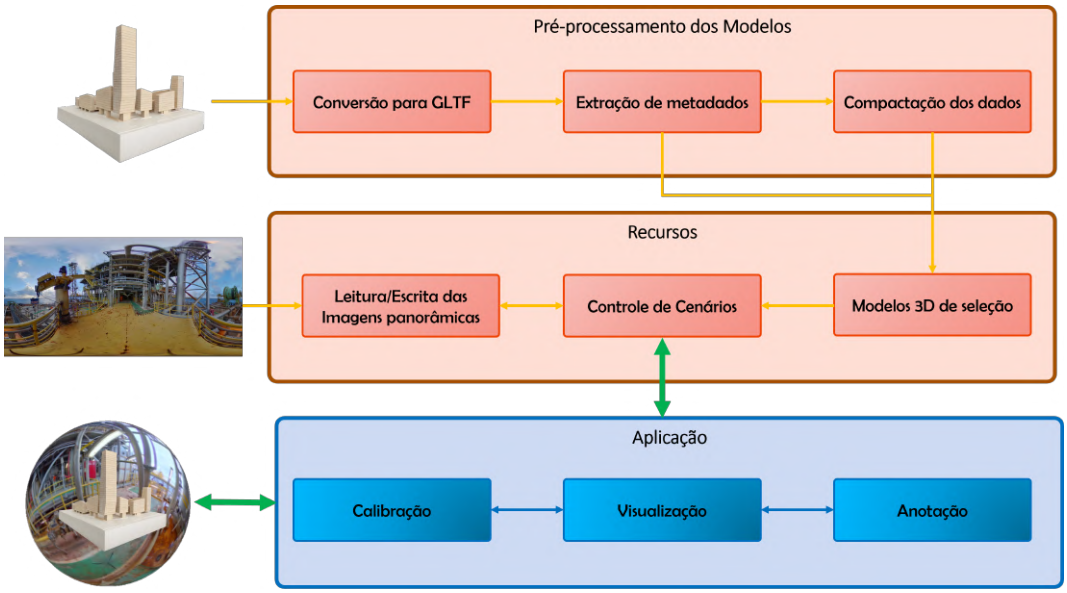


Figura 4.3: Arquitetura da aplicação. Fonte: Próprio autor.

4.3 Implementação

As subseções abaixo introduzem as funcionalidades principais da aplicação e as etapas realizadas para a produção do resultado final.

4.3.1
Etapas de processamento e renderização do 3D

Como explanado na seção de pré-processamento, os modelos 3D precisaram ser convertidos para um formato mais convencional para visualizadores comuns. O processo de conversão se traduz em ler o modelo no formato *wavefront* e transformá-lo ao padrão GLTF enviando as informações da malha e de textura. O segundo passo é a compactação do GLTF, nesse passo os materiais de cada objeto que possui semelhança são combinados, vértices em posições iguais são devidamente eliminados e por fim uma redução nas malhas visando a redução do número de triângulos. Para que os valores não sejam totalmente perdidos, são armazenadas também alguns metadados. Esses metadados guardam as informações como nomes das geometrias, identificadores únicos e a referência para o objeto original para consultas mais específicas.

Após a compactação, a malha do objeto pode ser lida pela aplicação. O material da malha não precisa de tratamentos especiais pois cada material apenas servia para diferenciar peças em uma cor diferente. Os modelos 3D também apresentam algumas geometrias adicionais como rotas de fugas ou equipamentos usados durante a etapa de planejamento.

Ao serem lidas, percebeu-se que os modelos mantinham a orientação voltada para $+z$. O ajuste escolhido foi rotacionar os modelos para orientação do projeto, $+y$. A Figura 4.4 mostra o fluxo percorrido pelo modelo 3D até a sua renderização.

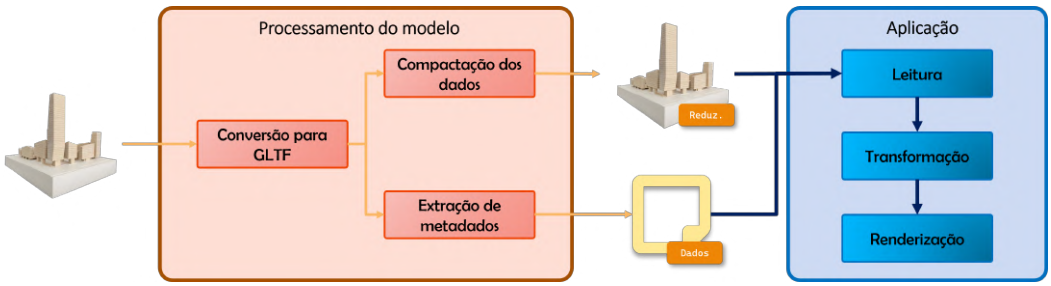


Figura 4.4: Processamento do Modelo. Fonte: Próprio autor.

4.3.2
Etapas de processamento e renderização da Imagem 360

Cada imagem 360º está representada por um panorama equiretangular. Cada imagem continha também uma coordenada de referência, que informava a posição exata de seu posicionamento na planta. Porém, por estar em alto mar durante a captura, muitas delas apresentavam um posicionamento ou rotação incorreta.

O grande desafio para projetar a imagem 360º é criar uma interface amigável para que o usuário consiga reposicionar a imagem usando a posição da câmera e rotacionar a textura de maneira livre. A solução encontrada abrangeu simular uma esfera envolvente em volta da câmera, onde o centro da esfera é o centro de projeção da câmera e a textura era voltada para o interior. Realizamos o mapeamento da textura, como demonstrado na seção 3.5, associando cada ponto (x, y, z) da superfície de acordo com as coordenadas

esféricas (θ, ϕ). Como os valores de θ e ϕ variam entre $[-\pi, \pi]$ e $[0, \pi]$, calculamos u e v a partir da Equação 4-1. Um fator desejável de usar a esfera é que podemos rotacionar a geometria de seus eixos de maneira livre, sem precisar aplicar transformações na câmera. Com a solução implementada bastava mudar os eixos da esfera, sem ter que alterar diretamente na projeção da textura, a implementação é ilustrada na Figura 4.5.

$$\begin{aligned} u &= \frac{1}{2}(1 + \frac{\phi}{\pi}) \\ v &= \frac{\theta}{\pi} \end{aligned} \tag{4-1}$$

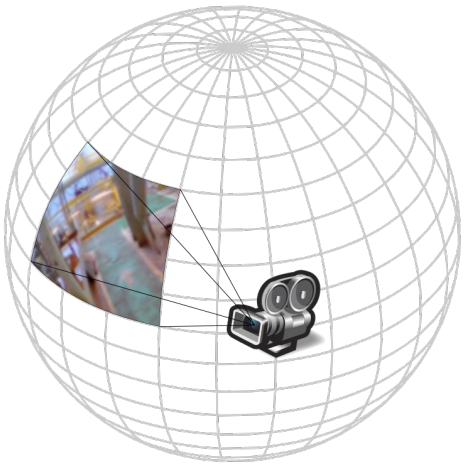


Figura 4.5: Exibição da imagem panorâmica. Fonte: Próprio autor.

4.3.3 Geração do Cenário

No contexto da aplicação, é necessário montar uma cena que envolva um conjunto de modelos e um conjunto de imagens 360º devidamente posicionadas em um local específico. Cada um desses locais deve coincidir exatamente onde a foto foi retirada, de modo que os mesmos componentes vistos na fotografias sejam encontrados no modelo 3D, como ilustra a Figura 4.6;

Inicialmente, renderizamos um ou mais modelos 3D que servirão como auxílio para geração do cenário. O usuário pode escolher cada modelo e exibi-lo na sua cena. A posição usada corresponde a posição original do modelo. A aplicação lista para o usuário os navios-plataformas e cada modelo representa um seção desse navio. A Figura 4.7 e 4.8 ilustra o processo de seleção do modelo 3D e sua exibição, respectivamente.

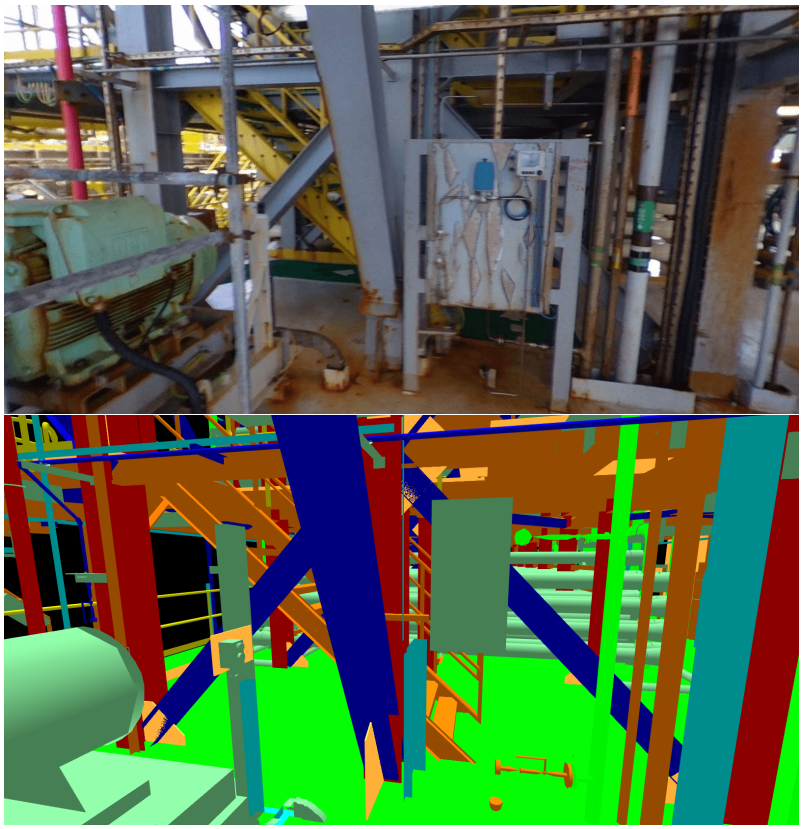


Figura 4.6: Fotografia do navio-plataforma e seu modelo correspondente. Fonte: Próprio autor.

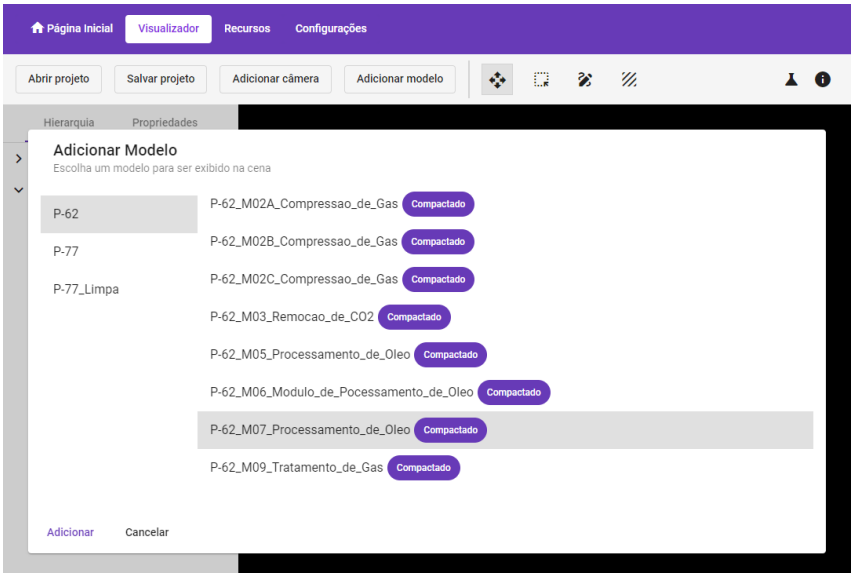


Figura 4.7: Seleção dos modelos na aplicação. Fonte: Próprio autor.

Introduzimos um objeto *SceneCamera*, ele é responsável por simular a câmera nas condições que as fotos foram tiradas durante a visita de um funcionário ao local. Nele são guardados a imagem 360º que será exibida, sua posição no mundo, a inclinação da câmera (rotação) e o seu ângulo de visão, visando simular as condições reais da câmera.

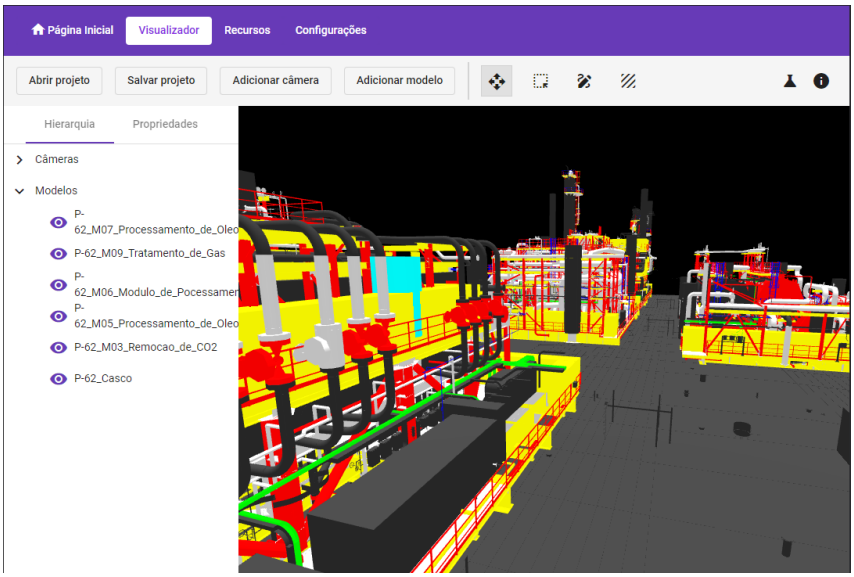


Figura 4.8: Exibição dos modelos na aplicação. Fonte: Próprio autor.

Esses valores extrínsecos e intrínsecos serão usados para realizar a calibração da *SceneCamera*. O usuário deve inicialmente escolher a textura, que corresponde a uma imagem 360º, com ilustra a Figura 4.9, em seguida ele tem acesso a uma interface onde ele pode posicionar corretamente a *SceneCamera* em uma posição específica do mundo. Ele também pode rotacionar a textura e alterar o ângulo de visão a partir da interface, como ilustra a Figura 4.10.

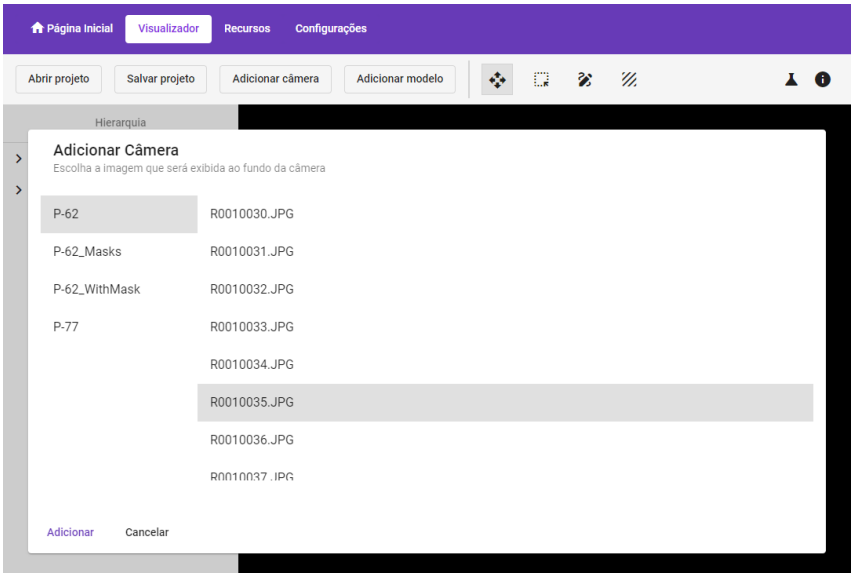


Figura 4.9: Seleção da imagem na aplicação. Fonte: Próprio autor.

Um cenário permite que o usuário adicione várias *SceneCameras* em posições diferentes, basta repetir o processo anterior, as visões podem ser trocadas ao clicar na *SceneCamera* correspondente, como ilustra a Figura 4.11.

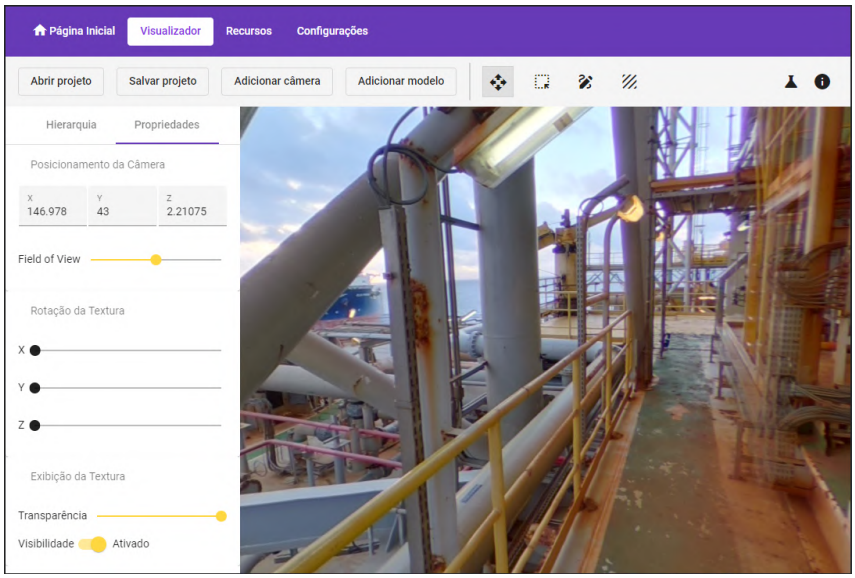


Figura 4.10: Propriedades da imagem na aplicação. Fonte: Próprio autor.

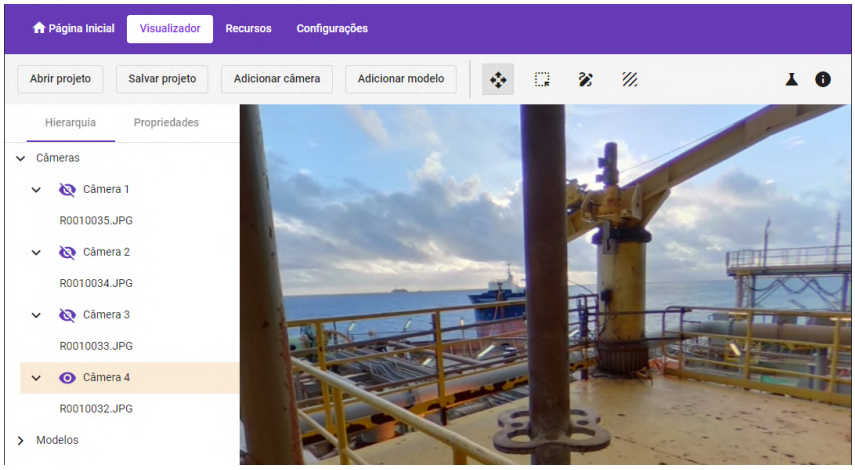


Figura 4.11: Seleção de *SceneCameras* Fonte: Próprio autor.

O processo de geração de cenário permite que o usuário simule uma visita utilizando imagens reais do local usando a representação digital das peças, porém realizar este tipo de calibração de maneira manual é um processo custoso se for feito às cegas. Para isso foram implementados alguns recursos para servir de auxílio a visualização convencional.

4.3.4 Calibração Inicial

Algumas imagens vieram acompanhadas de uma localização e uma planta que correspondiam aos locais das fotografias no mundo real. Para evitar que o usuário criasse um cenário desde o início manualmente, foi criado um cenário-base onde as *SceneCameras* já estão devidamente posicionadas. A Figura 4.12 ilustra as posições de cada fotografia na planta.

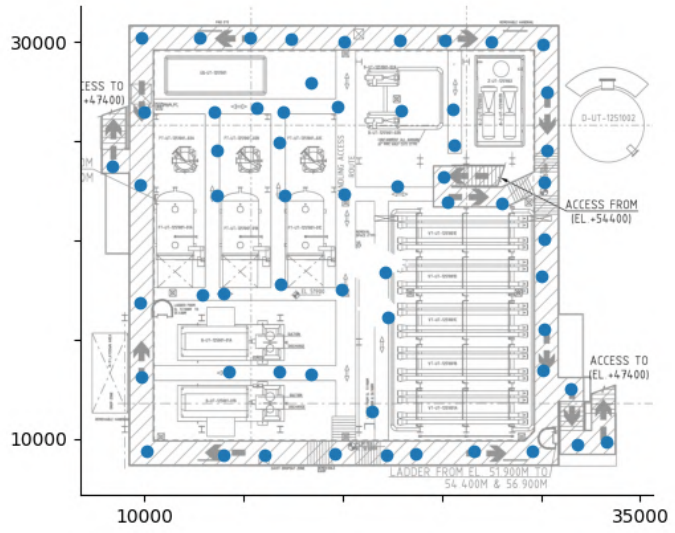


Figura 4.12: Trecho da planta do Navio-Plataforma, com os pontos de referência de cada fotografia. Fonte: Próprio autor.

Para realizar a calibração inicial, primeiro precisamos pegar dois pontos de referência de cada fotografia indicada na planta, um para cada eixo. Em seguida, usamos a aplicação para recuperar as posições correspondentes nas coordenadas do cenário, como ilustra a Figura 4.13.

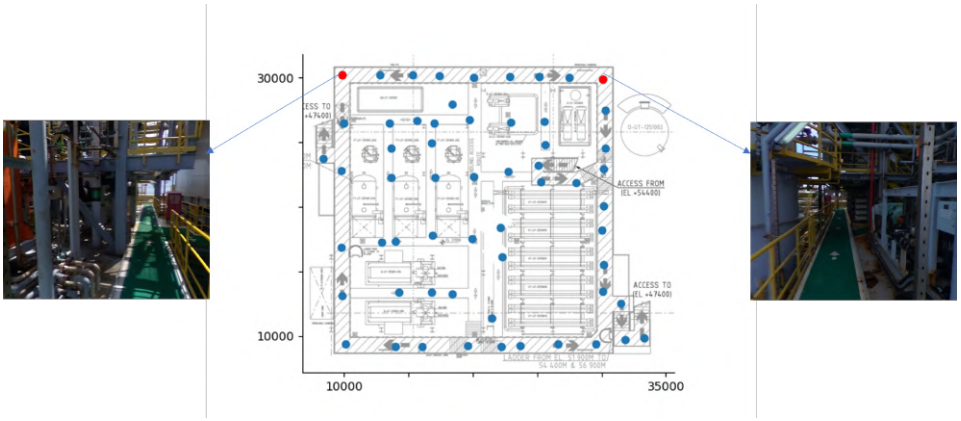


Figura 4.13: Planta do navio com fotografias dos pontos de referência. Fonte: Próprio autor.

Assumindo que cada ponto de referência esteja devidamente posicionado, calculamos as localizações intermediárias do mesmo eixo a partir de uma regra de três simples. O resultado dessa calibração é um cenário onde as *SceneCamera* estão posicionadas em um local aproximado de acordo com a planta original.

4.3.5 Tipos de Visualização

Para dar suporte a calibração manual e a geração de anotação o sistema permite dois tipos de visualização: híbrida, onde o usuário consegue ver simultaneamente a imagem 360º e o modelo 3D, e a visualização projetada, no qual os modelos 3D recebem a textura da imagem 360º.

A visualização hibrida, é a opção inicial dada para os usuários. Ela consiste em uma opção de visibilidade, onde você pode escolher se quer ou não observar a imagem, e um *slider* de transparência que varia de 0 a 100, onde 100 é completamente visível e 0 ela é totalmente invisível. A Figura 4.14 ilustra a utilização desta visualização.

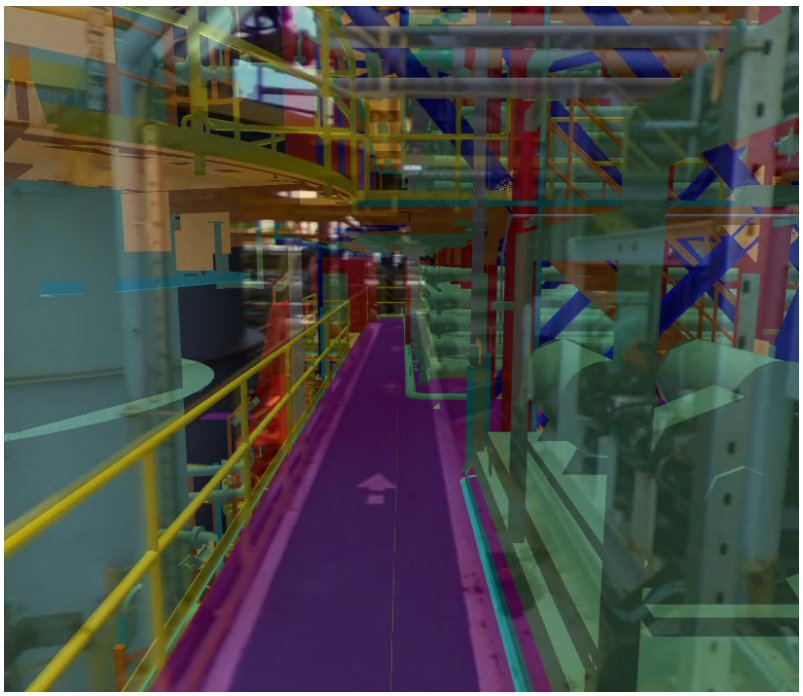


Figura 4.14: Visualização híbrida Fonte: Próprio autor.

A opção de visualização projetada consiste em executar o algoritmo de textura projetiva, demonstrados na seção 3.6, no modelo 3D. Inicialmente recuperamos a imagem panorâmica exibida no plano de projeção da câmera, guardamos os materiais originais dos modelos caso o usuário decida voltar para o estado de visualização anterior e em seguida, projetamos a textura no material do modelo 3D. A Figura 4.15 ilustra o resultado obtido.

Figura 4.15: Visualização projetada Fonte: Próprio autor.

4.3.6 Geração das anotações

Após a calibração das *SceneCameras*, o usuário é capaz de realizar comparativos entre o modelo 3D e imagens 360º usando as ferramentas de visualização apresentadas. Porém, é preciso que o usuário tenha meios de selecionar qualquer objeto conflitante durante a sua observação para indicar variáveis externas, elementos não mapeados, patologias ou qualquer informação adicional.

A geração de anotação deve abranger os dois tipos de recursos do sistema: a geometria, onde o usuário consiga identificar uma e consiga criar notas sobre ela; e a imagem 360º, onde o usuário consiga selecionar uma área alvo e adicionar uma anotação sobre a área selecionada.

Para permitir a seleção do modelo, usou-se uma técnica chamada de *Texture Picking* ou seleção por textura, no qual cada peça recebia um identificador em forma de cor. Durante a seleção, o usuário escolhe uma posição da tela e recupera a cor do *pixel* que ele selecionou. Com a cor do *pixel*, convertemos a cor de volta para o seu identificador original e procuramos nos metadados o nome original da peça. Durante o modo de seleção os objetos não selecionados são brancos e o objeto selecionado pelo usuário se torna vermelho, como ilustra a Figura 4.16.

Um modelo pode possuir dentro de si mais de 30 milhões de peças diferentes. E apenas usar o atributo de cor não é suficiente para representar todas as peças, visto que uma cor só pode atribuir no máximo $256 * 256 * 256 = 16.777.216$ cores diferentes. Para obter mais valores, usou-se pontos flutuantes como cores principais. Cada componente de cor foi dividida em $2^{16} = 65.536$ seções. Para lidar com a imprecisão do ponto flutuante durante a exibição da

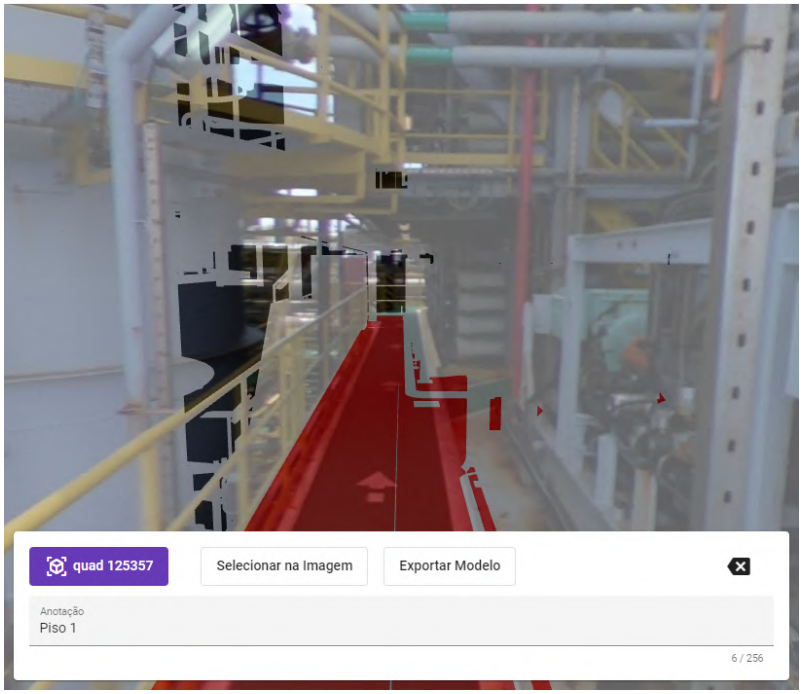


Figura 4.16: Seleção do piso da plataforma. Fonte: Próprio autor.

seleção, os *pixels* que são pintados são os fragmentos que estejam no limiar entre a cor anterior e a próxima cor da selecionada.

Após selecionar o modelo, fica disponível para o usuário uma caixa de texto, onde ele pode inserir qualquer informação relevante sobre determinada peça do modelo, como mostra a Figura 4.16.

O outro tipo de anotação que pode ser gerada pela aplicação são a criação de máscaras anotadas. Durante a criação, o usuário pode pintar uma área-alvo da imagem usando o cursor, determinando o local de atenção. Após completar a marcação ele pode inserir um texto e salvar a máscara.

Para que a máscara da anotação seja criada corretamente, usamos a posição do cursor e criamos um raio que intercede a esfera simulada, calculamos as coordenas u e v da textura e em seguida, usamos uma matriz esparsa com o mesmo tamanho da imagem original para auxiliar na criação da máscara. Para cada ponto (u, v), calculamos a coordenada da imagem a partir da coordenada de textura e inserimos o valor 1. A matriz resultante corresponderá a máscara gerada pela anotação. A Figura 4.17 ilustra o resultado da seleção.

Também é possível gerar máscaras a partir da seleção do modelo. A alteração realizada no algoritmo consiste em lançar raios em direção aos *pixels* selecionados do modelo em vez da posição do mouse, resultando em uma imagem que apresenta as formas dos modelos em suas máscaras. Para selecionar um modelo a partir das máscaras, desta vez os raios são lançados em direção aos *pixels* selecionados da imagem 360º. Cada modelo atingido tem sua cor traduzida em seu identificador único para que seja adicionado a seleção.

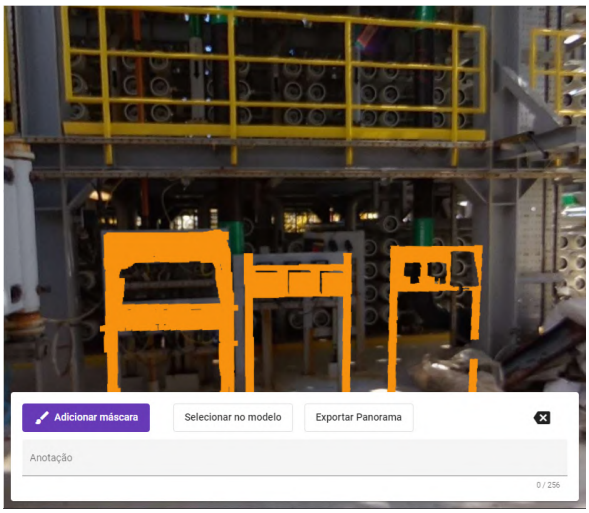


Figura 4.17: Seleção de equipamentos, usando mão livre e modelos 3D. Fonte: Próprio autor.

4.3.7 Exportação

Para que a aplicação realize a comunicação com outros sistemas e permita que seus dados sejam repassados para outros especialistas, são suportadas a exportação de imagens 360º com máscara e os pedaços dos modelos 3D que receberam anotações. Para a exportação da imagem panorâmica, usa-se a matriz esparsa criada na etapa de seleção. Como explanado, ela já possui quais *pixels* devem ser pintados. O resultado é ilustrado na Figura 4.18. De maneira semelhante, o usuário pode entrar no modo de seleção do modelo e exportar os modelos selecionados, tornando possível realizar o *download* da segmentação 3D a partir do arquivo original.

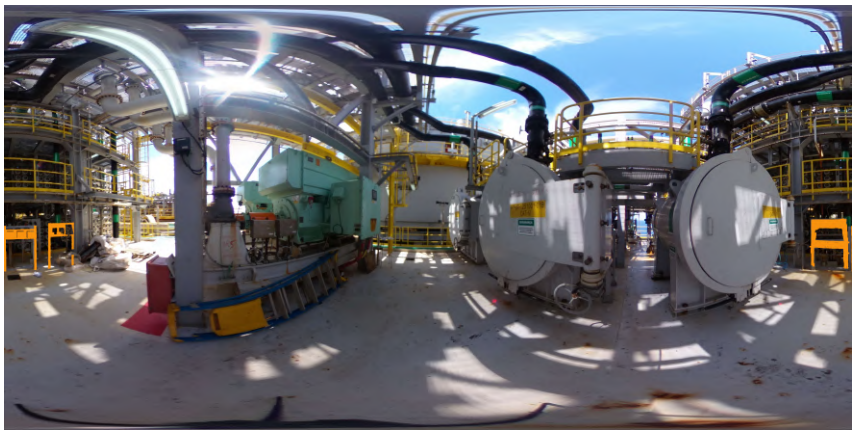


Figura 4.18: Exibição da seleção da Figura 4.17 no panorama. Fonte: Próprio autor.

5
Casos de uso

Para ressaltar a utilidade desta aplicação, apresentamos alguns casos da aplicação e seus benefícios para criação de anotação, identificar incoerências e mensurar quantitativos.

Atualmente, os modelos 3D usados na aplicação foram obtidos na etapa de planejamento, isso implica que equipamentos podem ser diferentes, podem existir peças faltando ou que exista erros humanos durante a criação da geometria. As fotografias, por sua vez, apresentam uma realidade diferente, pois foram capturadas já durante a execução e exibe uma realidade diferente, como as ações do tempo, o desgaste, novos equipamentos, etc. Usando a visualização híbrida da aplicação, é possível enxergar o que mudou desde a etapa de planejamento até os dias atuais, como é exemplificado na Figura 5.1. Na imagem é possível encontrar erros voltados para os equipamentos faltantes ou em posições incorretas, peças diferentes e erros de geometria.

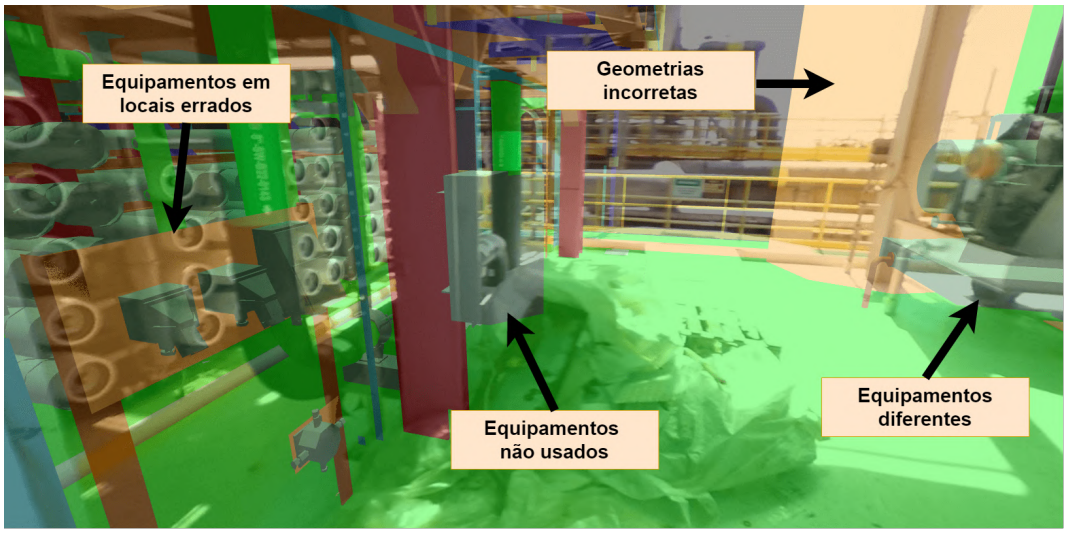


Figura 5.1: Erros encontrados na visualização híbrida Fonte: Próprio autor.

Dependendo da situação, é interessante que especialistas consigam marcar nas imagens panorâmicas de maneira correta, sem a necessidade de converterem para um formato plano. Outro aspecto importante é conseguir fazer uma seleção a partir de um conjunto de modelos, sendo possível distinguir de maneira precisa quais elementos são relevantes para sua imagem. A Figura 5.2 ilustra a seleção de dois componentes instalados em câmeras diferentes.

Outro exemplo de uso da aplicação é sua integração com outros *softwares*. Usou-se um algoritmo de inteligência artificial focado na detecção de ferrugem em estruturas metálicas. O resultado do algoritmo foi um conjunto de máscaras que informam em quais áreas da imagem apresentam algum nível de ferrugem. É possível importar as imagens junto com as máscaras para exibir as áreas afetadas pela ferrugem dentro da aplicação. Para melhor quantificar os casos de corrosão, é possível utilizar a visualização projetiva de forma que seja possível mensurar com exatidão a área afetada. A Figura 5.3 ilustra o caso de uso.

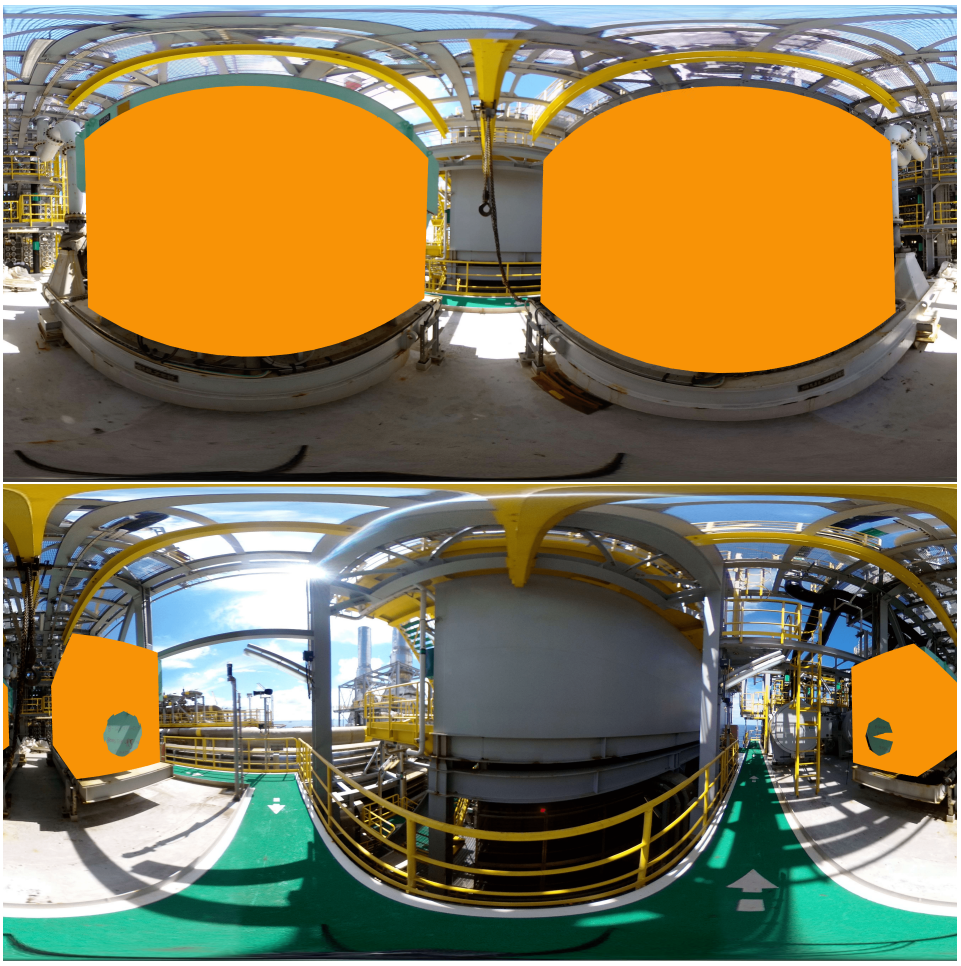


Figura 5.2: Mascará de um equipamento na imagem panorâmica Fonte: Próprio autor.

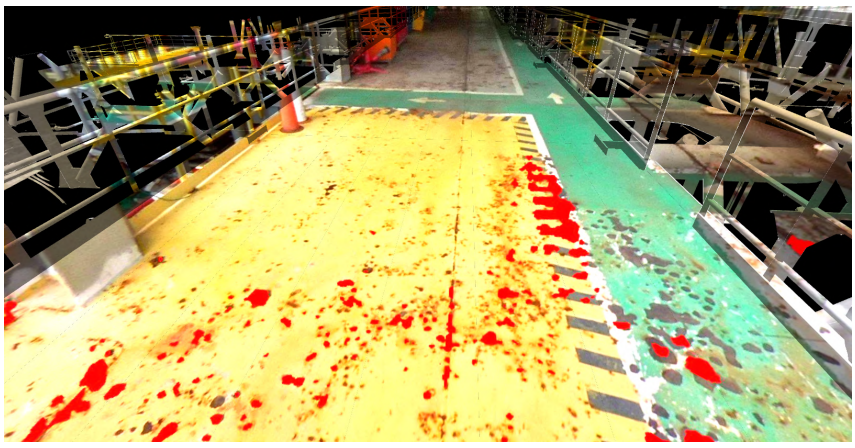


Figura 5.3: Área de corrosão sendo projetada no modelo 3D Fonte: Próprio autor.

6
Conclusão e trabalhos futuros

Este trabalho apresentou uma solução em realidade aumentada para exibir um gêmeo digital modelado e imagens 360º de maneira conjunta. Além da exibição conjunta foi apresentada funcionalidades necessárias para criar a melhor maneira de realizar anotações entre os dois tipos de recursos. A aplicação serve como auxílio para identificação de incoerências, erros e elementos incertos e permite que sejam rapidamente catalogados. Outro benefício é o fato dela ser completamente acessível a partir de um navegador, permitindo que especialistas explorem e avaliem situações reais com menos riscos e custos.

Ainda que todos os benefícios da indústria 4.0 não sejam alcançados, esta aplicação tenta lidar com o desafio de criar, editar, conectar e transferir informações usando modelos 3D e imagens panorâmicas para construir um gêmeo digital *as-built*. A base de dados geradas a partir desta aplicação pode ser usada em outras linhas como inteligência artificial, *Big Data*, computação em nuvem, etc.

A aplicação também serve de auxílio, para visualização dos efeitos do tempo em imagens, visto que é possível adicionar mais do que uma imagem por posição. A aplicação pode servir de suporte para avaliação das condições do ambiente, podendo ser usado para prevenção de acidentes.

Para trabalhos futuros, seria interessante implementar meios de otimizar a geração de anotações da aplicação, seja acelerando o processo de anotação ou criando modelos mais leves que não reduzam a qualidade do gêmeo digital. Outro trabalho futuro, seria uma maneira de associar uma câmera com uma imagem 360º a uma posição do modelo semi ou automaticamente, isto é, conseguir determinar os valores intrínsecos e extrínsecos da câmera com a mínima intervenção do usuário. Por fim, uma adição interessante seria detectar e categorizar automaticamente as possíveis diferenças e inconsistências entre a imagem 360º e o modelo 3D, com base em técnicas de processamento de imagem e IA.

7
Referências bibliográficas

ABDEEN, F. N.; SEPASGOZAR, S. M. City digital twin concepts: A vision for community participation. **Environmental Sciences Proceedings**, MDPI, v. 12, n. 1, p. 19, 2022.

AKENINE-MO, T. et al. Real-time rendering. AK Peters/CRC Press, 2018.

BARBOZA, D. et al. Virtual reality digital twin for floating production storage and offloading (fpso) units. In: SBC. **Anais Estendidos do XXI Simpósio de Realidade Virtual e Aumentada**. [S.l.], 2019. p. 31–32.

Ben Kreunen. **Big ben equirectangular**. 2021. [Online; acessado em 14 de Julho, 2022]. Disponível em: <https://wiki.panotools.org/File:Big_ben_equirectangular.jpg>.

DANG, N. et al. 3d digital twin models for bridge maintenance. In: **Proceedings of 10th International Conference on Short and Medium Span Bridges, Quebec city, Quebec, Canada**. [S.l.: s.n.], 2018.

DIRKSEN, J. **Learning Three. js: the JavaScript 3D library for WebGL**. [S.l.]: Packt Publishing Ltd, 2013.

GATTASS, M. **Rastreamento de Raios**. 2013. Access in 5/7/2022. Disponível em: <http://webserver2.tecgraf.puc-rio.br/~mgattass/LivroCG/05_Rastreamento_de_Raios.pdf>.

GHOBAKHLOO, M. Industry 4.0, digitization, and opportunities for sustainability. **Journal of cleaner production**, Elsevier, v. 252, p. 119869, 2020.

GLASSNER, A. S. **An introduction to ray tracing**. [S.l.]: Morgan Kaufmann, 1989.

GRINBERG, M. **Flask web development: developing web applications with python**. [S.l.]: "O'Reilly Media, Inc.", 2018.

HECKBERT, P. S. Survey of texture mapping. **IEEE computer graphics and applications**, IEEE, v. 6, n. 11, p. 56–67, 1986.

HUGHES, J. F. et al. **Computer graphics: principles and practice**. [S.l.]: Pearson Education, 2014.

KAHLEN, F.-J.; FLUMERFELT, S.; ALVES, A. Transdisciplinary perspectives on complex systems. **Transdisciplinary Perspectives on Complex Systems: New Findings and Approaches**, Springer, 2017.

KHRONOS GROUP. **glTF RUNTIME 3D ASSET DELIVERY**. 2021. Access in 5/7/2022. Disponível em: <https://www.khronos.org/gltf/>.

KHRONOS GROUP. **WebGL™**. 2021. Access in 5/7/2022. Disponível em: <https://www.khronos.org/webgl/>.

LEE, J. et al. Predictive manufacturing system-trends of next-generation production systems. **Ifac proceedings volumes**, Elsevier, v. 46, n. 7, p. 150–156, 2013.

LIU, D. et al. Scalable omnidirectional video coding for real-time virtual reality applications. **IEEE Access**, IEEE, v. 6, p. 56323–56332, 2018.

LOHTANDER, M. et al. Micro manufacturing unit and the corresponding 3d-model for the digital twin. **Procedia manufacturing**, Elsevier, v. 25, p. 55–61, 2018.

PALAZZOLO, E.; STACHNISS, C. Fast image-based geometric change detection given a 3d model. In: IEEE. **2018 IEEE International Conference on Robotics and Automation (ICRA)**. [S.l.], 2018. p. 6308–6315.

PARROTT, A.; WARSHAW, L. Industry 4.0 and the digital twin. **Deloitte Insights**, 2017.

PEREIRA, R. E.; MOUD, H. I.; GHEISARI, M. Using 360-degree interactive panoramas to develop virtual representation of construction sites. In: **Lean and Computing in Construction Congress (LC3)**. [S.l.: s.n.], 2017. v. 1, p. 4–7.

PESSOA, A. et al. Uma ferramenta de autoria para construção de ambientes de realidade virtual para subestações de energia baseada em panoram as aumentados. In: SOCIEDADE BRASILEIRA DE COMPUTAÇÃO PORTO ALEGRE. **Conference on Graphics, Patterns and Images**. [S.l.], 2017. v. 30.

PHONG, B. T. Illumination for computer generated pictures. **Communications of the ACM**, ACM New York, NY, USA, v. 18, n. 6, p. 311–317, 1975.

PIRES, F. et al. Digital twin in industry 4.0: Technologies, applications and challenges. In: IEEE. **2019 IEEE 17th International Conference on Industrial Informatics (INDIN)**. [S.l.], 2019. v. 1, p. 721–726.

ROBINET, F. et al. gltf: Designing an open-standard runtime asset format. **GPU Pro**, v. 5, p. 375–392, 2014.

RODRIGUES, L. F.; JESUS, R. A. de; SCHÜTZER, K. Industrie 4.0: Uma revisão da literatura. **Revista de Ciência & Tecnologia**, v. 19, n. 38, p. 33–45, 2016.

SAKURADA, K.; OKATANI, T.; DEGUCHI, K. Detecting changes in 3d structure of a scene from multi-view images captured by a vehicle-mounted camera. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2013. p. 137–144.

SALOMON, D. **Transformations and projections in computer graphics**. [S.l.]: Springer, 2006. v. 233.

SEGAL, M. et al. Fast shadows and lighting effects using texture mapping. In: **Proceedings of the 19th annual conference on Computer graphics and interactive techniques**. [S.l.: s.n.], 1992. p. 249–252.

SILVA, F. W. S. V. da. Introdução ao ray tracing. 1999.

STARK, R.; KIND, S.; NEUMEYER, S. Innovations in digital modelling for next generation manufacturing system design. **CIRP annals**, Elsevier, v. 66, n. 1, p. 169–172, 2017.

TANEJA, A.; BALLAN, L.; POLLEFEYS, M. City-scale change detection in cadastral 3d models using images. In: **Proceedings of the IEEE Conference on computer Vision and Pattern Recognition**. [S.l.: s.n.], 2013. p. 113–120.

TANEJA, A.; BALLAN, L.; POLLEFEYS, M. Geometric change detection in urban environments using images. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 37, n. 11, p. 2193–2206, 2015.

TAO, F.; ZHANG, M. Digital twin shop-floor: a new shop-floor paradigm towards smart manufacturing. **Ieee Access**, IEEE, v. 5, p. 20418–20427, 2017.

TSAI, M.-J.; KAO, C.-L.; LIU, J. The gentle spherical panorama image construction for the web navigation system. In: IEEE. **2010 IEEE International Conference on Acoustics, Speech and Signal Processing**. [S.l.], 2010. p. 1578–1581.

TSAI, R. A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf TV cameras and lenses. **IEEE Journal on Robotics and Automation**, Institute of Electrical and Electronics Engineers (IEEE), v. 3, n. 4, p. 323–344, aug 1987. Disponível em: <https://doi.org/10.1109%2Fjra.1987.1087109>.

XU, L. D.; XU, E. L.; LI, L. Industry 4.0: state of the art and future trends. **International journal of production research**, Taylor & Francis, v. 56, n. 8, p. 2941–2962, 2018.

ZHOU, K.; LIU, T.; ZHOU, L. Industry 4.0: Towards future industrial opportunities and challenges. In: IEEE. **2015 12th International conference on fuzzy systems and knowledge discovery (FSKD)**. [S.l.], 2015. p. 2147–2152.