Pós-Graduação em Ciência da Computação

Kelvin Batista da Cunha

Detecção de objetos em 6-DoF em tempo real utilizando técnicas de aprendizagem profunda

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
http://cin.ufpe.br/~posgraduacao

Recife
2019

Kelvin Batista da Cunha

Detecção de objetos em 6-DoF em tempo real utilizando técnicas de aprendizagem profunda

Trabalho apresentado ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Área de Concentração: Mídia e Interação
Orientador: Veronica Teichrieb
Coorientador: Francisco Simões

Recife
2019

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

C972d Cunha, Kelvin Batista da
Detecção de objetos em 6-DoF em tempo real utilizando técnicas de aprendizagem profunda / Kelvin Batista da Cunha. – 2019.
85 f.: il., fig., tab.

Orientadora: Veronica Teichrieb.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2019.
Inclui referências e apêndices.

1. Visão computacional. 2. Mídia e interação. I. Teichrieb, Veronica (orientadora). II. Título.

006.37 CDD (23. ed.) UFPE- MEI 2019-045

Dissertação de Mestrado apresentada por **Kelvin Batista da Cunha** à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título "**Detecção de objetos em 6-DoF em tempo real utilizando técnicas de aprendizagem profunda** " **Orientador: Veronica Teichrieb** e aprovada pela Banca Examinadora formada pelos professores:

Prof. Germano Crispim Vasconcelos
Centro de Informática/ UFPE

Prof. Bruno Motta de Carvalho
Departamento de informática e matemática aplicada / UFRN

Prof. **Coorientador**: Francisco Simões
Centro de Informática / IFPE

Prof. **Orientador**: Veronica Teichrieb
Centro de Informática / UFPE

Visto e permitida a impressão.
Recife, 22 de Fevereiro de 2019.

Prof. Ricardo B C Prudêncio
Coordenador da Pós-Graduação em Ciência da Computação do
Centro de Informática da Universidade Federal de Pernambuco.

Dedico este trabalho a minha família que foram porto seguro perante as dificuldades durante este percurso.

AGRADECIMENTOS

A Deus pela oportunidade de chegar até aqui.

A meus Pais pelos exemplos, motivação e ensinamentos que me acompanharam ao longo de toda a vida.

A Raquel por todo apoio, companheirismo e compreensão que possibilitaram a conclusão deste trabalho.

A Veronica, Francisco e ao time do Voxar pelos ensinamentos, disponibilidade e incentivo, compartilhando os conhecimentos necessários para sobrepor os desafios encontrados durante o desenvolvimento do trabalho.

RESUMO

Detecção e rastreamento em seis graus de liberdade (*6-DoF, six Degrees-of-Freedom*) são problemas amplamente estudados na área de Visão Computacional. É possível encontrar aplicações que utilizam detecção 6-DoF em áreas como realidade aumentada, robótica, interação avançada, entre outras. As técnicas desenvolvidas podem utilizar diversos tipos de sensores, com prevalência de técnicas baseadas em sensores RGB ou RGBD. Para utilizar apenas informações RGB, houve um recente avanço com a utilização de técnicas baseadas em aprendizagem profunda. Para tal, os métodos propostos geralmente utilizam modelos mais complexos para lidar com a falta da informação de profundidade. Devido a isto, a performance do algoritmo é prejudicada, realizando, em alguns casos, a estimação de pose dependente de pós-processamento que prejudica o tempo de execução do algoritmo. Nesse contexto, esta dissertação visa avaliar a aplicabilidade das recentes técnicas de aprendizagem profunda para realizar a detecção de objetos 3D com 6 graus de liberdade. O principal objetivo é o desenvolvimento de uma técnica para estimação da pose em tempo real utilizando apenas câmeras RGB com o uso de aprendizagem profunda, bem como avaliar as limitações e perspectivas de seu uso para identificação de oportunidades. Para alcançar o objetivo, foi escolhido um método base para desenvolvimento, a partir das principais características obtidas na revisão da literatura. Os resultados foram validados através da utilização da base de dados pública **LINEMOD**. Em seguida, foram analisados detalhadamente seus pontos de robustez e falhas para diferentes cenários. Posteriormente, foi gerado um conjunto de dados para avaliar como o método se comporta para cenários genéricos, variando características de iluminação, ambiente, parâmetros de câmeras e movimento da cena. Nestes cenários, o método conseguiu obter resultados compatíveis com o estado da arte para casos em que aparecem imagens borradas, ambientes poluídos e oclusão parcial do objeto. Para casos em que foram utilizadas imagens de diferentes câmeras de testes e mudanças de ambiente, o método obteve baixo desempenho, demonstrando pontos de melhoria. Para melhorar o comportamento da técnica nestes cenários, foi gerado um conjunto de imagens sintéticas, com adaptação do domínio e randomização do domínio. A utilização das imagens sintéticas possibilitou avaliar a potencial melhoria de precisão do modelo nos cenários genéricos.

Palavras-chaves: Detecção 6-DoF. Aprendizado Profundo. Visão Computacional.

ABSTRACT

Detection and tracking in six degrees of freedom (6-DoF) are tasks widely studied in computer vision. It is possible to find applications that use 6-DoF detection in areas such as augmented reality, robotics, advanced interaction, among others. The developed techniques can use several types of sensors, with prevalence of techniques based on RGB or RGBD sensors. The methods based on RGB information received a recent advance influenced by deep learning methods. To do so, the proposed methods generally use more complex models to deal with the lack of depth information. Due to this, the performance of the algorithm is impacted, requiring in some cases post-processing responsible of even more impact on the algorithm execution time. In this context, this dissertation aims to evaluate the applicability of recent deep learning techniques to detect 3D objects with six degrees of freedom. The main objective is to develop a technique for estimating a 6-DoF pose in real time using only RGB cameras with the use of deep learning, as well as to evaluate the limitations and perspectives of its use to identify future opportunities. To reach the objective, a basic method for development was chosen, based on the main characteristics obtained from the literature review. The results were validated through the use of the public dataset called **LINEMOD**. Then, its robustness points and failures for different scenarios was analyzed in detail. Later, a dataset was generated to evaluate how the method behaves for generic scenarios, varying lighting characteristics, environment, camera parameters and camera motion. In these scenarios, the method was able to obtain results compatible with the state of the art for cases in which blurred images appear, polluted environments and partial occlusion of the object. For cases where images from different test cameras and environment changes were used, the method performed poorly, demonstrating improvement points. To improve the behavior of the technique in these scenarios, a set of synthetic images was generated, with domain adaptation and domain randomization. The use of synthetic images made it possible to evaluate the potential improvement of model accuracy in the generic scenarios.

Key-words: 6-DoF Detection. Deep Learning. Computer Vision.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

6-DoF	six Degree of Freedom
AE	Auto-Enconders
CAE	Convolutional Auto-Enconder
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DA	Domain Adaptation
DCNN	Deep Convolutional Neural Networks
DNN	Deep Neural Networks
DR	Domain Randomization
EPnP	Efficient Perspective-n-Point
FOV	Field of View
fps	frames per second
GPU	Graphics Processing Unit
ICP	Iterative Closest Point
IoU	Intesection over Union
K-NN	K-Nearest Neighbor
mm	milímetros
PnP	Perspective-n-Point
PSO	Particle Swarm Optimization
px	pixel
RAM	Random access memory
RGB	Red, Green, Blue
RGBD	Red, Green, Blue, Depth
SGD	Stochastic Gradient Descent
VOC2012	Visual Object Classes Challenge 2012
XML	Extensible Markup Language
YOLO	You Only Look Once

SUMÁRIO

1 INTRODUÇÃO

Detecção e rastreamento em seis graus de liberdade (6-DoF, *six Degree-of-Freedom*) é uma tarefa amplamente estudada na área de Visão Computacional. O uso da detecção e rastreamento 6-DoF tem aplicações em diversas áreas, como por exemplo, realidade aumentada (LEE; PARK; WOO, 2011), robótica (HERNANDEZ et al., 2016; SCHWARZ et al., 2018; SCHWARZ et al., 2017), reconstrução (ONDRÚŠKA; KOHLI; IZADI, 2015; PRISACARIU et al., 2015), entre outros.

Na realidade aumentada, o ambiente real faz parte do contexto da aplicação, e o objetivo é aumentar a percepção do usuário sobre o mundo real utilizando as informações virtuais inseridas através do rastreamento, viabilizando uma interação natural para o relacionamento homem-máquina. Devido a isso, estas aplicações possuem alto requisito de precisão e devem ocorrer em tempo real, permitindo a interação fluida entre o usuário e a aplicação.

Na robótica, os movimentos e ações de robôs autônomos são realizados através do rastreamento de informações inseridas no ambiente, possibilitando a interação do robô com objetos de interesse, manipulações dos objetos e evitando colisões durante o movimento.

É possível aplicar a detecção e/ou rastreamento para realizar diversas outras tarefas que envolvem o reconhecimento e entendimento do ambiente, possibilitando a interação com objetos e cenários reais. Alguns exemplos são a localização e movimentação de carros e robôs autônomos, aplicações de reconhecimento facial, identificação e manipulação de objetos reais, renderização de modelos virtuais em ambientes, reconhecimento da cena para reconstrução virtual do ambiente, entre outros.

A tarefa de detecção de objetos em 6-DoF foca em encontrar um objeto de interesse na cena, sem nenhuma informação prévia disponível sobre a posição e orientação iniciais do alvo. No caso do rastreamento 6-DoF, onde geralmente a posição e orientação iniciais do objeto são conhecidas na cena, o objetivo do método é registrar o movimento do objeto continuamente durante uma sequência de quadros da cena. Para realizar o rastreamento e detecção de objetos em 6-DoF em tempo real, os algoritmos buscam entender as características da cena, extraindo informações sobre o objeto de interesse em tempo de execução, e realizar sua correlação com informações previamente adquiridas do objeto para estimar a pose do objeto, tanto através de informações selecionadas manualmente (técnicas comumente chamadas de Geométricas ou de Visão Computacional Clássica) (SEO et al., 2013; SEO et al., 2014; WANG et al., 2015) ou utilizando informações selecionadas através de técnicas de aprendizagem de máquina (BALNTAS et al., 2017; JATESIKTAT et al., 2018). Grande parte dos métodos propostos atualmente lidam com informações de sensores RGB (câmeras) ou RGBD (câmeras com sensor de profundidade, do inglês

Depth) (CHOI; CHRISTENSEN, 2013; CHAUDHARY et al., 2017), estes últimos, conseguindo obter melhor acurácia entre os trabalhos do estado da arte devido a informação extra de profundidade disponível (TAN; NAVAB; TOMBARI, 2017). O grande problema dos métodos RGBD está no custo em se obter o sensor, geralmente mais caro que câmeras RGB. Câmeras RGBD também possuem limitações como a influência da luz solar que pode afetar a precisão do sensor, e a distância da imagem obtida.

Métodos recentes, buscam aplicar imagens RGB juntamente com algoritmos baseados em métodos de aprendizagem de máquina (TAN; NAVAB; TOMBARI, 2017; TEJANI et al., 2018; DOUMANOGLOU et al., 2016b). Algumas abordagens conseguem obter resultados similares aos sensores RGBD utilizando apenas as informações de cores, ocasionando nos últimos anos, uma forte retomada na investigação dos métodos para câmeras RGB. A grande limitação destes métodos é relacionado ao desempenho do algoritmo. Geralmente, os algoritmos de aprendizagem que lidam com imagens RGB possuem alto tempo de execução, ou necessitam de processos de refinamento que prejudicam o tempo de execução do algoritmo.

Técnicas mais recentes baseados em redes neurais profundas (DNN, *Deep Neural Networks*) utilizam modelos capazes de manter alto desempenho no tempo de execução e obter bons resultados de precisão da estimação de pose (KEHL et al., 2017; TEKIN; SINHA; FUA, 2017), eventualmente sem a necessidade de um pós-processamento, nos modelos chamados *singleshot*.

Contudo, a limitação destes métodos reside na alta quantidade de dados necessários para treinamento dos modelos e na necessidade de ser obter dispositivos de alto poder computacional para sua execução. A obtenção e rotulação de dados é um processo que exige bastante tempo e esforço empregado. Algumas bases de dados podem ser encontradas publicamente com imagens e anotações para validação das técnicas desenvolvidas (HINTERSTOISSER et al., 2012; TEJANI et al., 2014; DOUMANOGLOU et al., 2016b; GARON; LALONDE, 2017; XIANG et al., 2017). Mas para aplicações que necessitem de objetos e informações específicas, a obtenção de dados pode ser um grande problema no desenvolvimento. Outro problema, é que geralmente os dados disponíveis são voltados para testes de aplicações de cenários conhecidos e/ou controlados, dificultando a avaliação de desempenho do modelo para diversos desafios que podem ocorrer em aplicações reais devido a mudança do sensor, ambiente, iluminação, etc.

Para sobrepor esse problema, autores buscam formas de conseguir gerar dados de treinamento sinteticamente. Gerando imagens a partir de modelos 3D dos objetos de interesse (HINTERSTOISSER et al., 2012; HINTERSTOISSER et al., 2017), utilizando técnicas de adaptação dos dados para obter de forma automática um conjunto de informações semelhantes a um pequeno conjunto de imagens reais conhecidas (ROZANTSEV; LEPETIT; FUA, 2015; ROZANTSEV; SALZMANN; FUA, 2018), ou gerando uma grande quantidade de variação nos dados de forma a induzir maior generalização dos modelos de aprendizagem

utilizados (TREMBLAY et al., 2018; TOBIN et al., 2017).

1.1 OBJETIVOS

Neste cenário, esta dissertação visa avaliar a aplicabilidade das recentes técnicas de aprendizagem profunda para realizar a detecção de objetos 3D com 6 graus de liberdade.

1.1.1 Objetivo principal

O principal objetivo dessa dissertação é o desenvolvimento de uma técnica para estimação da pose em 6 graus de liberdade em tempo real utilizando apenas câmeras RGB com o uso de aprendizagem profunda minimizando a dependência de dados reais no treinamento, bem como avaliar as limitações e perspectivas de seu uso para identificação de oportunidades.

1.1.2 Objetivos específicos

Como objetivos específicos é possível destacar:

- Implementação e validação da técnica do estado da arte.
- Estudo da área de detecção e rastreamento 6-DoF, identificando as principais abordagens utilizadas e oportunidades para desenvolvimento de melhorias do método.
- Avaliação dos principais pontos de robustez e falhas e definição de refinamentos para serem realizados.
- Geração de um conjunto de dados reais com um novo objeto, para avaliação do comportamento do método em novos cenários. Inserindo novos desafios através da variação das características de iluminação, ambiente e parâmetros de câmeras.
- Avaliação do comportamento e precisão para a mudança destes diversos parâmetros.
- Refinamento do resultado da técnica, desenvolvendo abordagens de geração de imagens sintéticas para treinamento do modelo, viabilizando sua aplicação em casos de uso de cenários genéricos. Utilizando abordagens de geração de imagens sintéticas simples para avaliar a precisão da técnica em relação aos diferentes cenários de teste e avaliar o comportamento da técnica para diferentes tipos de informação utilizada no treino, analisando a dependência do aprendizado em relação ao conjunto de imagens reais utilizadas. Também será avaliado o processo de geração de dados utilizando *transfer-learning* com adaptação de domínio, treinando o modelo com apenas informações de dados sintéticos e avaliando a possibilidade de diminuição do número de imagens reais necessárias para treinamento. E por fim, a geração de

imagens sintéticas por randomização do domínio. Misturando no treinamento imagens geradas sinteticamente através de diversas variações aleatórias e imagens reais, com o objetivo de melhorar o nível de generalização da técnica para vários casos de uso.

1.2 ESTRUTURA DO TRABALHO

No segundo capítulo são descritos os conceitos utilizados como base para o desenvolvimento desta dissertação. São descritos os conceitos sobre a tarefa de detecção e rastreamento, descrevendo os principais objetivos, tipos de técnicas e informações utilizadas por cada uma. Em seguida, é discutido o modelo utilizado para câmera virtual, representação dos parâmetros de câmera, rotação e translação. Por fim, são definidos os tipo de métodos utilizados para rastreamento e detecção, diferenciando as características das abordagens geométricas e por aprendizado de máquina.

No terceiro capítulo é discutida a revisão de literatura para rastreamento e detecção em seis graus de liberdade. Os principais trabalhos do estado da arte relacionados a modelos de aprendizagem de máquina são descritos, levando em consideração suas principais características (tipo de sensor de entrada, objetos suportados, dados de treinamento utilizados, entre outros) para entendimento dos métodos e comparação entre as técnicas. Também são avaliadas as bases de dados disponíveis para treinamento e validação dos modelos, avaliando tipos de objetos, quantidade de informação e desafios inclusos. E por fim, são discutidas as abordagens de treinamento utilizadas para melhorar o resultado do treinamento com a manipulação do conjunto de dados, através de técnicas como *transfer-learning*, *data augmentation*, geração de dados sintéticos e randomização de domínio.

O quarto capítulo descreve o método desenvolvido e, as abordagens de geração de dados utilizadas para treinamento do método.

No capítulo cinco são descritos os passos de implementação realizados, bibliotecas e parâmetros utilizados. Também são definidos os conjuntos de dados, procedimentos para treinamento e funções de avaliações utilizadas para validação do método.

No capítulo seis são mostrados e discutidos os resultados obtidos e, o capítulo sete conclui o trabalho, apresentando suas contribuições e trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 RASTREAMENTO E DETECÇÃO 6-DOF

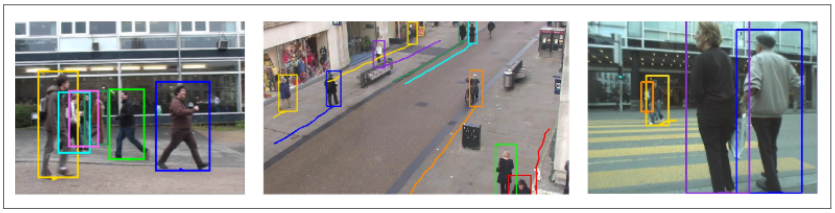
A detecção e rastreamento de objetos a partir de imagens é uma tarefa bastante estudada na área de Visão Computacional, onde o objetivo é encontrar e/ou registrar a posição e movimento de um ou mais objetos na imagem de entrada. A tarefa de detecção e rastreamento pode ser realizada considerando apenas as informações 2D ou as características 3D do objeto na cena.

Técnicas para detecção de objetos utilizam apenas a informação de uma único quadro para extrair características do objeto e realizar a estimação de sua localização, sem conhecimento prévio sobre a cena de entrada. No caso do rastreamento, a posição inicial do objeto é conhecida previamente e a informação estimada é recuperada quadro a quadro, registrando o movimento do objeto durante toda a cena. Por este motivo, as técnicas de detecção tendem a ser mais complexas computacionalmente (TJADEN; SCHWANECKE; SCHÖMER, 2017). Em geral, as técnicas de detecção e rastreamento são utilizadas em conjunto, utilizando a detecção para recuperar a pose inicial do objeto e inicializar o algoritmo de rastreamento. A detecção também pode ser utilizada para recuperar a posição do objeto nos casos de perda durante o rastreamento.

As técnicas híbridas utilizam a detecção e rastreamento em conjunto de forma a evitar problemas recorrentes em cada uma das abordagens. No caso da detecção, quando cada estimação é feita de forma independente um efeito 'tremido' (*jitter*) é percebido para o resultado retornado em diferentes quadros. No rastreamento pode ocorrer o acúmulo do erro ao longo do tempo (*drift*), fazendo com que o movimento estimado durante os quadros seja percebido com um atraso em relação ao movimento real.

Para a detecção e rastreamento 2D o objetivo é encontrar e registrar o movimento do objeto ao longo de uma sequência de quadros, considerando as coordenadas do objeto projetadas no plano 2D da imagem. Na detecção 2D são utilizadas apenas as informações de pontos chaves que definem o objeto na imagem, como por exemplo, as coordenadas da região retangular que delimita o objeto $((x1, y1), (x2, y2))$ (Figura 1).

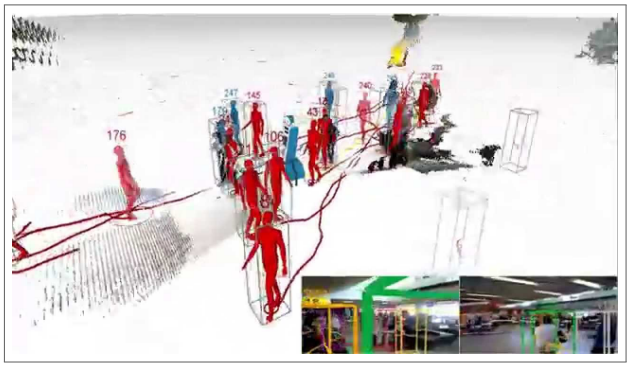
Figura 1 – Rastreamento 2D de pessoas, MOTChallenge 2015.



Fonte: (LEAL-TAIXÉ et al., 2015)

No caso 3D é necessário computar os valores de transformação que leva os pontos 3D (x, y, z) do mundo real para as coordenadas de câmera 2D (u, v), estimando-se os parâmetros de rotação e translação da câmera em relação ao objeto e vice-versa. Na detecção e rastreamento 3D são consideradas a distância dos objetos em relação a câmera e as relações entre cada objeto na cena (Figura 2).

Figura 2 – Rastreamento 3D de pessoas em ambientes populosos.



Fonte: (LINDER et al., 2016)

Atualmente, a tarefa de detectar e rastrear objetos em 3D é referido como rastreamento e detecção em seis graus de liberdade (6-DoF, do inglês *six Degree-of-Freedom*), uma vez que a rotação e a translação possuem três graus de liberdade cada uma. Tal problema, também é conhecido como estimativa de pose, dado que a rotação e a translação quando compostas numa matriz 3x4 são chamadas de matriz de pose, que será explicada nas próximas seções.

Conhecendo os parâmetros que definem o modelo da câmera e os pontos 3D que formam o objeto de interesse, é possível estimar os valores do vetor de rotação ($\mathbf{r} = [r_\alpha, r_\beta, r_\gamma]$) e translação ($\mathbf{t} = [t_x, t_y, t_z]$) que representa o movimento relativo do objeto/-câmera. Através dos valores de $\mathbf{r}$ e $\mathbf{t}$ é possível conhecer a relação entre os pontos 3D em coordenadas de mundo e os pontos 3D correspondentes em coordenadas da câmera virtual.

Considerando o modelo de câmera pinhole (STURM, 2014), possuindo as informações de calibração intrínseca da câmera e a matriz de pose, é possível levar os pontos do objeto (3D) para as coordenadas correspondentes na imagem (2D), viabilizando as aplicações de Realidade Aumentada, interação, Robótica, entre outras.

Nas próximas seções serão definidos conceitos e parâmetros utilizados para detecção e rastreamento em 6-DoF discutidos nessa dissertação.

2.2 REPRESENTAÇÃO DA CÂMERA VIRTUAL

No rastreamento e detecção 3D precisamos definir o modelo de câmera virtual utilizado para simular o dispositivo no mundo real que captura e armazena as informações da cena

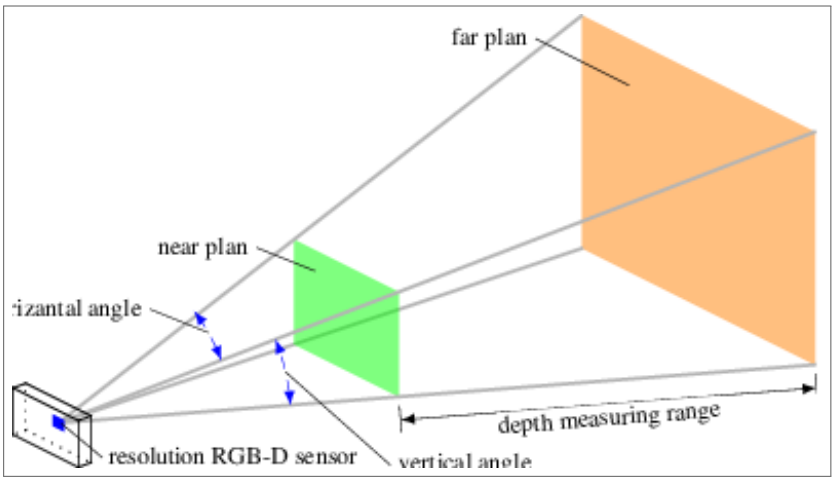
em pixel, conhecendo as transformações que levam os pontos do mundo real ao sistema de coordenadas 2D definidos na imagem digital. Um dos modelos mais utilizados para simular as câmeras em sistemas computacionais se baseia no modelo de câmera ***pinhole***. O modelo de câmera perspectiva é utilizado para simular o comportamento deste modelo. Este modelo de câmera virtual é definido em termos da distância focal (ou campo de visão, FOV), tamanho do sensor simulado e resolução da imagem gerada.

No modelo de câmera virtual, são utilizados os planos de proximidade (*near clipping plane*) e distância (*far clipping plane*) para definir a visibilidade do objeto na cena, limitando a distância permitida de cada objeto em relação a câmera (Figura 3). Objetos mais próximos que o plano *near* ou mais distantes que o plano *far clipping* serão invisíveis para a câmera. O *near* e *far clipping* não são utilizados nas câmeras reais, mas são definidos em modelos virtuais para auxiliar o controle de profundidade dos pontos mapeados em ferramentas de renderização, evitando o cálculo relacionado a pontos que seriam imperceptíveis ao usuário. Outra diferença, é que, diferentemente do modelo de câmera real, onde o plano da imagem é localizado após o centro de projeção, nos modelos virtuais os planos podem ser localizados a frente do centro de projeção e paralelos ao plano da imagem.

Para gerar a representação virtual do modelo *pinhole*, são utilizadas as relações geométricas entre o plano de projeção e da cena, para converter as coordenadas do mundo em pixel através da projeção perspectiva.

As coordenadas de mundo são transformadas em coordenadas de câmera através de uma matriz de transformação de dimensões $4x4$, chamada de matriz de projeção, e seu uso pode variar dependendo do tipo de algoritmo utilizado para gerar as imagens digitais.

Figura 3 – Modelo de câmera virtual perspectiva.



Fonte: (SCHÖNING; HEIDEMANN, 2016)

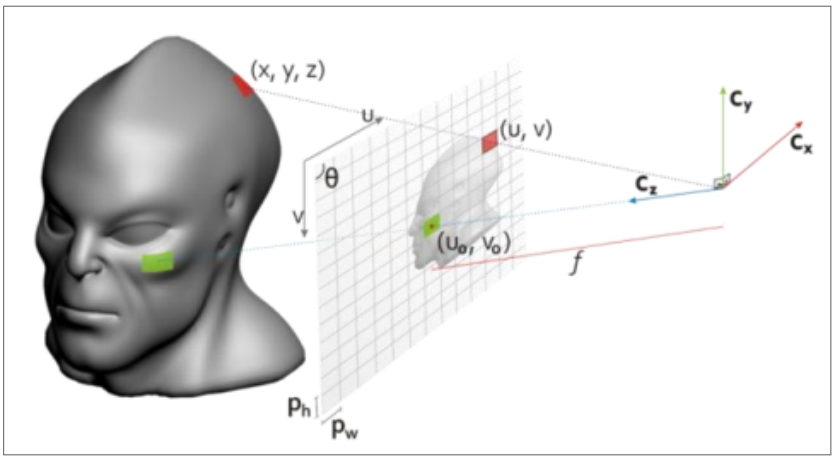
2.3 MATRIZ DE CÂMERA

A matriz de projeção é a representação matemática que leva os pontos 3D ($\mathbf{P}^w$) para as coordenadas 2D ($\mathbf{P}^c$) projetados na imagem, Equação 2.1 (Figura 4). Essa matriz $\mathbf{P}$ é formada pela composição da matriz de valores intrínsecos, derivados da calibração da câmera $\mathbf{K}$, com a matriz de valores extrínsecos $(\mathbf{R}|\mathbf{t})$, chamada de matriz de pose.

$$k\mathbf{P}^c = \mathbf{P} * \mathbf{P}^w \tag{2.1}$$

$$\mathbf{P} = \mathbf{K} * (\mathbf{R}|\mathbf{t}) \tag{2.2}$$

Figura 4 – Projeção dos pontos 3D para o plano da imagem 2D a partir da matriz de pose $\mathbf{P}$.



Fonte: (LIMA et al., 2010)

2.3.1 Matriz de parâmetros intrínsecos

A matriz de parâmetros intrínsecos $\mathbf{K}$ é definida pelos valores de distância focal em relação aos eixos x (f_x) e y (f_y) que são medidos em pixel. O valor de s representa a inclinação entre os eixos x e y, e geralmente pode ser utilizado como 0. u_0 e v_0 representam os pontos principais na imagem (ponto central).

$$\mathbf{K} = \begin{bmatrix} f_x & s & u_0 & 0 \\ 0 & f_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.3}$$

A matriz $\mathbf{K}$ é obtida através da calibração da câmera que será utilizada, conhecendo os valores da distância focal f em milímetros (mm), largura e altura do sensor S_w e S_h em mm e resolução da imagem $(W \times H)$ com largura W e altura H medidos em pixel (px), podemos obter os valores de f_x, f_y, u_0 e v_0 pela Equação 2.4. Onde δ_x e δ_y representam

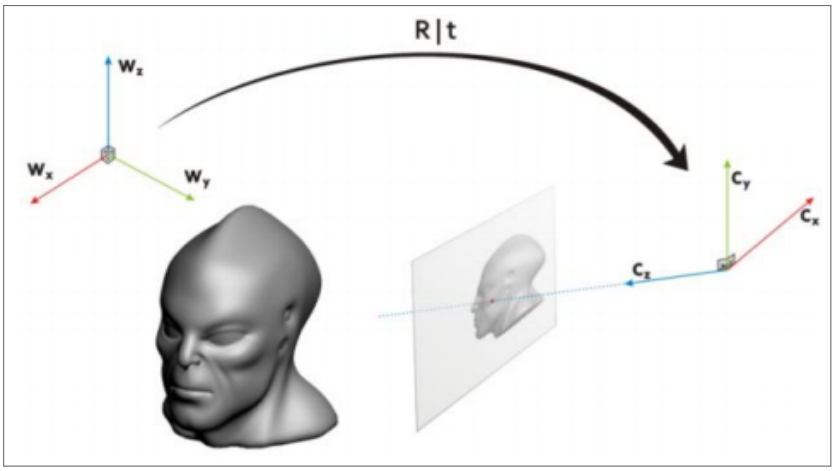
o desvio do plano de projeção em relação ao centro de projeção para os eixos x e y respectivamente.

$$\begin{aligned} f_x = \frac{f \times W}{S_w}; \quad f_x = \frac{f \times H}{S_h}, \\ u_0 = \frac{W}{2} + \delta_x; \quad v_0 = \frac{H}{2} + \delta_y. \end{aligned} \tag{2.4}$$

2.3.2 Matriz de parâmetros extrínsecos

A matriz de parâmetros extrínsecos $(\mathbf{R}|\mathbf{t})$ é formada pela matriz de rotação 3D $(\mathbf{R})$ concatenada com o vetor de translação 3D $(\mathbf{t})$, e representa a transformação das coordenadas 3D do mundo para o sistema de coordenadas 3D da câmera (Figura 5). O valor da matriz extrínseca pode ser visto como o movimento realizado pela câmera no sistema de coordenadas do mundo.

Figura 5 – Transformação das coordenadas 3D do mundo para o sistema de coordenadas 3D da câmera através da matriz $(\mathbf{R}|\mathbf{t})$.



Fonte: (LIMA et al., 2010)

A matriz de rotação $\mathbf{R}$ pode ser definida em termos de rotações nos eixos x, y e z.

$$\mathbf{R} = R_x(\alpha)R_y(\beta)R_z(\gamma) \tag{2.5}$$

onde R_x, R_y e R_z são definidos como:

$$\mathrm{R}_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}; R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}; R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.6}$$

Os ângulos α, β e γ representam *yaw*, *pitch* e *roll*, e definem os 3 graus de liberdade para a rotação 3D, essa representação é conhecida como representação em ângulos de Euler (ANG; TOURASSIS, 1987).

A rotação também pode ser expressa pela representação eixo-ângulo (vetor de rotação). Nesta representação a rotação é parametrizada em relação ao espaço euclidiano tridimensional por um vetor de três valores, composto por um vetor unitário ($r = [r_x, r_y, r_z]$) que indica a direção do eixo de rotação e um ângulo (θ) que define a magnitude da rotação em relação ao eixo (DIEBEL, 2006).

Outra representação bastante utilizada é a representação por *quartenions*. A representação em *quartenions* é composta por um vetor de quatro valores e pode ser obtida a partir de uma extensão da fórmula de Euler (KUIPERS et al., 1999).

Essas representações são formas equivalentes de descrever a matriz de rotação, sendo bastante utilizadas nos algoritmos por serem mais compactas, eficientes e estáveis.

O vetor de translação $\mathbf{t}$ é estimado através do centro da câmera $\mathbf{C}$ no sistema de coordenadas do mundo e da matriz de rotação estimada (Equação 2.7), a equação para estimação do vetor de translação é encontrada através da Equação 2.1 (LEPETIT; FUA et al., 2005).

$$\begin{aligned} \mathbf{0} &= \mathbf{RC} + \mathbf{T} \\ \mathbf{C} &= -\mathbf{R}^T\mathbf{T} \end{aligned} \tag{2.7}$$

2.4 RASTREAMENTO GEOMÉTRICO X RASTREAMENTO POR APRENDIZAGEM

Para realizar o rastreamento e/ou detecção do objeto, é preciso conhecer as características que definem e distinguem o objeto na cena. Para isso, algoritmos extratores de características podem ser utilizados, baseando-se na estrutura do objeto alvo (contorno, forma, cor, etc). Para realizar a interpretação destas informações extraídas algoritmos de Visão Computacional Clássica (geométricos) ou orientados a dados (aprendizagem de máquina) podem ser utilizados.

Os algoritmos geométricos utilizam a informação sobre a estrutura do objeto e como estas informações se relacionam com a cena observada durante a execução. Estes algoritmos também são conhecidos como algoritmos baseados em modelos, por utilizarem a informação do modelo 3D do objeto para realizar a estimação da posição do objeto na cena. Geralmente, os pontos de vista que definem a orientação do objeto são definidos previamente, extraindo informações sobre o modelo 3D em cada orientação e armazenando em uma base de dados que será comparada com as informações extraídas da cena.

As técnicas geométricas podem se dividir entre métodos de casamento de padrões, segmentação de regiões, extração de contornos e baseados em marcadores. Nos métodos de casamento de padrões, as informações são extraídas da cena através de descritores (LOWE, 1999; ABDEL-HAKIM; FARAG, 2006; LEUTENEGGER; CHLI; SIEGWART, 2011). Estas características extraídas são comparadas com um conjunto de características armazenadas do objeto de interesse extraídas a partir do modelo 3D ou de imagens de treino, que são ob-

tidas utilizando o mesmo descritor. Um processo de otimização é realizado para estimar a posição e orientação do objeto através do casamento das características mais similares extraídas da cena com o objeto. Os métodos baseados em casamento de padrões podem lidar com diversas situações como variação de escala, mudança de iluminação e fundo poluído, porém são sensíveis em casos de oclusão e borramento. Outra limitação é dependência dos descritores para objetos texturizados, ocasionando em baixa performance do algoritmo na detecção de objetos não texturizados.

Os métodos de extração de contorno utilizam apenas as informações de arestas da cena, obtidas por filtros de extração de arestas. As correspondências são realizadas encontrando o conjunto de arestas mais similares aos contornos dos objetos conhecidos através de seus modelos 3D. Estes métodos são utilizados para lidar com a falta de textura em objetos, porém são bastante sensíveis a ruídos do plano de fundo.

Nos métodos baseados em segmentação de regiões, as informações extraídas são obtidas através das regiões que compõem o objeto (contorno, interior e exterior), estes algoritmos geralmente utilizam a informação de cor do objeto para realizar a separação das regiões, realizando a identificação destas características através de métodos de otimização de forma similar aos métodos de casamento de padrões e extração de aresta.

Para os métodos de aprendizagem de máquina, um modelo é treinado, através de um conjunto de dados relacionado ao domínio de uso, para identificar o conjunto de características extraídas da cena e realizar a estimação.

Os algoritmos de aprendizagem de máquina podem realizar o treinamento supervisionado, não-supervisionado ou semi-supervisionado. No treinamento supervisionado, um conjunto de dados rotulados (pares de entrada e resposta), contendo exemplos do domínio alvo, deve ser passado para treinamento do modelo. Os dados serão utilizados para otimização dos parâmetros do modelo utilizado através da resposta conhecida para cada exemplo.

No treinamento não-supervisionado o conjunto de dados passado para treinamento não contém os rótulos associados aos exemplos, o algoritmo deve gerar uma representação que agrupe o conjunto de dados em classes coerentes com as características dos exemplos de treino, e associar novos exemplos não conhecidos a essas classes.

O treinamento semi-supervisionado é uma junção entre o treinamento supervisionado e não-supervisionado, onde um pequeno conjunto de dados rotulados é passado para treinamento para identificação e agrupamento das classes, o restante do conjunto de dados não-rotulados é agrupado de acordo com os grupos definidos previamente.

A maioria dos trabalhos relacionados a aprendizagem de máquina para detecção e rastreamento 6-DoF utilizam modelos por treinamento supervisionado, em que são necessários os rótulos de cada exemplo para otimização dos parâmetros do algoritmo. Por este motivo, nesta dissertação focamos na revisão e descrição dos métodos que realizam o treinamento supervisionado.

Métodos clássicos de aprendizagem de máquina recebem um conjunto de características definidas previamente e, realizam o treinamento de forma a otimizar o conjunto de parâmetros para estas características gerando um conjunto de hipóteses que estimam a resposta para exemplos ainda não conhecidos.

Os modelos podem realizar a estimação da resposta através da regressão ou classificação. Na regressão o modelo é treinado para estimar valores dentro do conjunto de números reais. Na classificação são definidas um conjunto de classes e o modelo deve rotular o exemplo passado como entrada para uma das classes conhecidas.

Métodos recentes de aprendizagem utilizam o paradigma de aprendizado de representação, em que não é necessário a realização do design de extratores de características. Os algoritmos de aprendizado de representação recebem a imagem diretamente como entrada e, durante o treinamento desenvolvem um modelo capaz de extrair características e interpretar estas características aprendidas em uma mesma arquitetura, retornando a resposta final por regressão ou classificação. Essa abordagem se tornou bastante popular devido ao desenvolvimento automático, através do processo de otimização do modelo, das características mais relevantes extraídas no conjunto de dados. A utilização desses métodos pode diminuir o custo de desenvolvimento, evitando a definição de extratores característicos manualmente, que muitas vezes se baseiam na experiência do desenvolvedor sobre o domínio avaliado.

A maioria dos métodos desenvolvidos com aprendizado de representação se baseiam no uso de redes neurais profundas, no caso de detecção e rastreamento de imagens as arquiteturas convolucionais profundas (DCNN, do Inglês *Deep Convolutional Neural Networks*) são os modelos mais utilizados. O grande número de trabalhos que aplicam modelos DCNN se deve ao recente avanço e popularização de sistemas capazes de suportar o treinamento e uso destes algoritmos. A dificuldade do uso desta abordagem, em métodos de treinamento supervisionado, se deve à grande quantidade de dados necessária para realização do treinamento dos modelos.

Alguns modelos são utilizados apenas para gerar uma representação das características automaticamente, como as arquiteturas AE (do Inglês *Auto-Encoders*). Nestes modelos uma rede neural é treinada para receber uma entrada e retornar como resposta a mesma informação. O conjunto de ativações intermediárias retornada por uma das camadas internas da rede é utilizada como uma nova representação para entrada com dimensionalidade reduzida, que será passada a um algoritmo que realizará a estimação da pose através da interpretação desta nova representação.

Geralmente as técnicas de rastreamento utilizam métodos baseados nas características geométricas (WANG; ZHONG; QIN, 2017; SANDY; BUCHLI, 2018; CHAUDHARY et al., 2017; PRISACARIU; REID, 2012), enquanto que para aprendizagem de máquina a maioria das técnicas são propostas para realizar a detecção dos objetos (BRACHMANN et al., 2016; RAD; LEPETIT, 2017; KEHL et al., 2017; TEKIN; SINHA; FUA, 2017; TEJANI et al., 2018). Esse

resultado se deve ao fato de que os algoritmos de aprendizagem de máquina conseguem generalizar informações através de um conjunto de dados, utilizado previamente para treinamento, podendo estimar com maior precisão a posição do objeto na imagem utilizando apenas a informação dos *pixels* de entrada durante a execução, sem o conhecimento da pose inicial do objeto na cena. No caso do rastreamento, os métodos baseados em características geométricas são bastante utilizados devido ao alto desempenho e baixo custo computacional, sendo aplicados modelos mais simples que realizam o casamento de características extraídas da imagem durante a execução, comparando as informações da cena com características já conhecidas previamente sobre o modelo, podendo ser empregados para estimação da pose em tempo real.

3 REVISÃO DA LITERATURA

Definimos o problema de pesquisa desta dissertação em "métodos para detecção e/ou rastreamento de objetos em seis graus de liberdade baseados câmeras RGB e em aprendizagem profunda". Neste trabalho, o foco de pesquisa é no desenvolvimento de uma técnica baseada em aprendizagem de máquina, com foco nas técnicas de aprendizado profundo.

Para realizar a revisão da literatura, foi realizada uma busca automática através do uso de palavras chaves conhecidas, com o objetivo de encontrar trabalhos relevantes relacionados ao tema proposto. Os trabalhos obtidos foram filtrados por critérios definidos previamente com a finalidade de manter os principais trabalhos do estado da arte. Em seguida, os trabalhos escolhidos foram avaliados e classificados. Foram analisadas as características de cada método identificando pontos de robustez, falhas, desempenho e acurácia.

Selecionamos nesse conjunto os principais trabalhos, e realizamos um *snowball* para cima (GOODMAN, 1961), buscando os trabalhos mais recentes que citam o conjunto selecionado com o objetivo de encontrar métodos que otimizem/refinem as técnicas selecionadas.

Após o *snowball* para cima, utilizamos nesse conjunto o mesmo processo de filtragem feito anteriormente, classificando cada trabalho em relação a acurácia obtida, tempo de execução, pontos de robustez e falhas. Restringimos também os resultado para artigos completos, excluindo resultados de teses, livros e resumos. No final do processo, realizamos a análise e classificação para a revisão. A lista final de trabalhos selecionados está descrita no Apêndice A.

Através da análise geral dos trabalhos selecionados foi possível entender as principais características dos métodos, verificando pontos como principais técnicas empregadas, acurácia média, performance, sensores utilizados, tipos de objetos suportados, etc.

3.1 TÉCNICAS DE APRENDIZAGEM PARA RASTREAMENTO E DETECÇÃO 6-DOF

Os métodos de aprendizagem de máquina utilizam o conhecimento prévio de um conjunto de dados para desenvolver um modelo capaz de inferir hipóteses quando novos dados não conhecidos são passados como entrada. Certas abordagens utilizam modelos tradicionais que geram as hipóteses a partir de características definidas manualmente pelo desenvolvedor. Nesses modelos as informações são extraídas da imagem de entrada e utilizadas para treinamento (DOUMANOGLOU et al., 2016b; TEJANI et al., 2018; TAN; NAVAB; TOMBARI, 2017). Em diversos cenários, os métodos baseados em características definidas manualmente utilizam sensores RGBD, podendo empregar modelos mais simples para classificação, o que o tornam mais eficientes e robustos a certas situações, como casos de ambiente poluído, movimento rápido de câmera e oclusão parcial dos objetos (SAHIN; KIM, 2018). O sensores RGBD possuem limitações para casos de alta intensidade de ilu-

minação, e variação da distância de captura da imagem, além de serem menos acessíveis e de maior custo quando comparadas com as câmeras RGB.

Recentemente, diversas abordagens buscam desenvolver métodos utilizando apenas a informação RGB (SOCK et al., 2018; KAO et al., 2018). A maioria destes modelos são mais complexos, devendo lidar com menos informações no aprendizado. A falta de informação de distância advinda dos sensores utilizados nas técnicas RGB aumenta a necessidade de passos intermediários para estimação dos valores de profundidades para cada pixel, algumas vezes modelos mais profundos são utilizados para aumentar o nível de representação e compensar a falta do valor de profundidade para os pixels, sendo estes modelos menos eficientes computacionalmente. A maior parte das abordagens que utilizam sensores RGB, aplicam modelos que aprendem a representação dos dados automaticamente, ou seja, não é necessário definir um conjunto de características para treinamento do modelo (LI; BAI; HAGER, 2018). Os métodos conseguem extrair conhecimento a partir do conjunto de dados e obter a melhor representação das características, que serão utilizadas no treinamento posteriormente.

Entre os métodos de aprendizagem, poucos são os trabalhos que realizam o rastreamento temporal de objetos em 6-DoF (TAN; NAVAB; TOMBARI, 2017; GARON; LALONDE, 2017). A maioria das técnicas desenvolvidas empregam modelos que realizam a detecção através de uma única imagem (RAD; LEPETIT, 2017; KEHL et al., 2016; TEKIN; SINHA; FUA, 2017), inferem informações adicionais a partir de uma predição realizada (RAMBACH et al., 2018; GAO et al., 2018) ou realizam refinamentos para melhorar o resultado da detecção (RAD; ROTH; LEPETIT, 2017; KEHL et al., 2017; LI et al., 2018). Por este motivo, as técnicas de aprendizagem, geralmente, sofrem com o problema de *Jitter*, devido a falta de informação temporal para estabilizar o resultado de estimação.

A seguir descrevemos alguns dos principais trabalhos encontrados, que utilizam diferentes abordagens de aprendizagem e regressão/classificação para estimação da pose.

Baseando-se em características dos métodos geométricos e de aprendizagem de máquina, Tan et al. propõe uma abordagem híbrida (TAN; NAVAB; TOMBARI, 2017), que combina pontos do processo de otimização e de aprendizagem, para realizar o rastreamento em 6-DoF. O modelo realiza o rastreamento temporal utilizando *Random Forests*, com características definidas manualmente, e processos de otimização de energia que auxiliam no processos de aprendizado das correspondências. O algoritmo de aprendizagem é capaz de lidar com diversos cenários e desafios devido a sua natureza para generalizar a informação a partir do treinamento realizado, enquanto o processo de otimização consegue ser menos sensível a casos de mínimo local, convergindo o modelo para uma melhor acurácia, reduzindo ruídos como *jitter*. A partir das informações RGBD, o algoritmo realiza o casamento de informações extraídas dos contornos do objeto (RGB) e da região interna do objeto (profundidade). As informações extraídas são passadas para um algoritmo ba-

seado em *Random Forests*, que é composto por um conjunto de classificadores simples que, quando agrupados realizam a estimação da transformação dos vetores que atualizam a pose do objeto. Após isso, o algoritmo de otimização é utilizado, minimizando a função de energia que registra a correspondência entre as projeções do pontos obtidos para o modelo do objeto e a cena, reduzindo o erro de predição que pode ocorrer na estimação da transformação. O modelo é capaz de lidar com problemas de oclusão, fundos poluídos e baixas condições de iluminação. A técnica pode ser usada em CPU e obtêm em média uma performance de 450 fps durante a detecção de diversos quadros sequenciais. O método é sensível a iluminação solar, movimento rápido e depende da distância da câmera para o objeto rastreado. Este método de rastreamento e detecção 6DoF foi agraciado com o prêmio de melhor demonstração no International Symposium on Mixed and Augmented Reality (ISMAR 2017).

Outro trabalho que se baseia em características definidas previamente e combina os métodos de otimização e aprendizagem é proposto em (TEJANI et al., 2018; TEJANI et al., 2014). As características utilizadas são obtidas através da adaptação de métodos para casamento de padrões, os descritores gerados são integrados ao algoritmo *Random Forest*, que será treinado sobre a distribuição dos dados RGBD. O treinamento pode ser realizado com imagens sintéticas dos modelos 3D dos objetos, simplificando o processo de obtenção de dados reais. Durante a inferência da pose, o processo é refinado iterativamente, sendo projetado para lidar com cenários de grande nível de oclusão e fundo poluído, gerando a detecção para múltiplos objetos simultaneamente. Como principal contribuição do trabalho, uma nova função de custo é definida para divisão dos exemplos de treinamento do modelo, baseados nas características extraídas.

Kehl et al. propõe a combinação entre aprendizado profundo e votação para realizar detecção em 6-DoF a partir de imagens RGBD (KEHL et al., 2016). O trabalho segue o paradigma de votação a partir de características locais, onde as imagens de entrada são divididas em *patches* extraídos da imagem contendo a informação de cor e profundidade. Uma arquitetura CAE (do Inglês, *Convolutional Auto-Enconder*) é treinada utilizando informações dos *patches* extraídos dos objetos de interesse, gerando uma representação de características intermediária para cada *patch* extraído de forma independente. A arquitetura é treinada para receber os *patches* e retornar a mesma informação como saída. O objetivo é gerar uma função que gere características extraídas das camadas intermediárias da arquitetura, reduzidas dimensionalmente e que serão armazenadas em um dicionário de características. Na sequência, o conjunto de informações armazenado é associado a cada anotação de pose conhecida para os dados de treinamento. Durante o processo de estimação, um sistema de votação utilizando K-NN (do Inglês *K-Nearest Neighbor*) é aplicado para cada região da imagem, selecionando a pose com maior número de votos associados às características encontradas na imagem de entrada. O treinamento desse método evita o uso de *data augmentation* (extensão do conjunto de dados realizando transformações na

imagem original), e é realizado a partir de informação de dados sintéticos. Contudo podem ocorrer falhas durante a detecção quando utilizado para dados reais, devido ao risco de sobre-ajuste (do Inglês *overfitting*) do modelo para os dados sintéticos. Outro problema ocorre devido à limitação da quantidade de objetos utilizados, ampliando a necessidade de armazenamento para as características extraídas e armazenadas no dicionário de características, o que também aumenta a complexidade da hipótese utilizada para votação entre os *patches*.

No artigo escrito por Zhang et al. (ZHANG; CAO, 2017) o procedimento de geração de características utilizando CAE para classificação de *patches* locais é utilizado juntamente com o algoritmo de otimização PSO (do Inglês *Particle Swarm Optimization*), substituindo o sistema de votação. Após a geração das características, utilizando um procedimento similar a (KEHL et al., 2016) com imagens RGBD, o processo de otimização é aplicado. As hipóteses de classificação dos *patches* são avaliadas pelo PSO. A função de custo definida leva em consideração o custo de estimação baseado na informação de profundidade e de contorno do *patch*. O algoritmo executa um processo iterativo até encontrar a solução de menor custo para a estimação dos vetores de rotação e translação. O uso do PSO torna o procedimento eficiente, evitando a otimização por gradiente descendente e funções derivativas, encontrando a solução de menor custo rapidamente. Utilizando a informação local dos objetos, o algoritmo consegue ser robusto a casos de oclusão parcial, fundo poluído e proximidade dos objetos. Contudo, ainda há o problema da definição dos parâmetros para executar o PSO.

Em (XIANG et al., 2017) é proposta a arquitetura PoseCNN que realiza a predição da pose 3D a partir de imagens RGB. O modelo é dividido em etapas, no qual a primeira etapa consiste em passar como entrada a imagem RGB para uma arquitetura CNN, responsável por extrair características da imagem de entrada. As características são passadas adiante para processamento em 4 diferentes ramificações. O primeiro realiza a segmentação dos objetos em cada região a partir de uma camada de deconvolução e, rotula o objeto detectado de acordo com a estimação das regiões para cada objeto. O segundo ramo realiza a predição dos centros nas direções x e y, e as distâncias entre os centros de cada objeto. A informação do primeiro e segundo ramo são combinadas utilizando *Hough Voting* (YAO; GALL; GOOL, 2010) e passadas adiante para o terceiro ramo junto às características extraídas, com isso, o terceiro ramo realiza a estimação das regiões 2D que delimitam cada objeto. O quarto ramo estima os vetores de rotação (em *quaternions*) e translação a partir das características extraídas junto das informações de segmentação 2D obtidas pela terceiro ramo. A segmentação em etapas permite ao modelo lidar com casos de similaridades e proximidades entre os objetos, predizendo as regiões de cada um na imagem separadamente. O modelo consegue realizar uma representação em cascata utilizando as características extraídas inicialmente combinadas ao resultado de cada etapa para o próxima tarefa, melhorando a resposta para estimação da pose no final do

processo. Adicionalmente são definidas uma nova função de *loss* para lidar com casos de objetos simétricos e um novo *dataset* de larga-escala para rastreamento de objetos 6-DoF (YCB-VideoDataset).

O DeepIM (LI et al., 2018) utiliza o resultado do PoseCNN para o desenvolvimento de uma nova técnica de refinamento da estimação de pose 6-DoF. Dada a estimação do PoseCNN, no DeepIM é feito um processo iterativo de ajuste de pose utilizando a arquitetura FlowNet (DOSOVITSKIY et al., 2015) que prediz uma transformação relativa entre o objeto observado na imagem e o objeto renderizado com a pose estimada. A imagem de entrada é passada para o modelo, segmentada na região do objeto predito pelo PoseCNN, junto a uma imagem sintética com o modelo renderizado utilizando a pose estimada. O modelo utiliza as propriedades do objeto para construir uma representação relativa (SE(3) (HOFFMAN, 1966)) e gerar uma transformação para ajustar o erro de casamento entre a imagem original e o objeto renderizado, definindo uma nova função de *loss* que realiza a regressão da pose. O processo é realizado iterativamente, compondo a pose estimada com a transformação em passos sequenciais.

O Deep 6-DoF (GARON; LALONDE, 2017) utiliza uma abordagem semelhante ao PoseCNN, realizando a estimação de pose com CNN, mas utiliza como entrada informação do sensor RGBD e, realiza o rastreamento a partir da informação de duas imagens de entrada (quadros consecutivos de um vídeo) para estimar a transformação relativa entre eles. A rede é treinada a partir de imagens sintéticas de um *dataset* gerado pelo autor, e que está disponível publicamente. O modelo é treinado para lidar com casos de oclusão severa, e é o primeiro modelo baseado em CNN que não realiza o rastreamento por detecção, utilizando a informação temporal entre os quadros consecutivos para gerar a estimação. O Deep 6-DoF utiliza uma arquitetura simples, por este motivo consegue ser bastante eficiente, possibilitando seu uso para rastreamento em tempo real. Contudo, o modelo é sensível a perda durante o rastreamento, não conseguindo se recuperar quando o objeto detectado é perdido, propagando o erro de estimação durante todo o resto da execução. Outra limitação descrita é que o modelo só pode ser treinado para rastrear um objeto por vez.

Seguindo uma abordagem diferente, a técnica conhecida como BB8 (RAD; LEPETIT, 2017) utiliza CNN para estimar a pose do objeto na forma dos pontos 2D projetados na imagem ao invés de realizar a regressão da pose diretamente na forma dos vetores de rotação e translação. Baseado na proposta de Crivellaro et al. (CRIVELLARO et al., 2015), os 8 cantos do *bounding box* 3D do objeto são estimados na cena através da detecção da posição correspondente dos seus pontos 2D projetados. No BB8, o processo de detecção é dividido entre as etapas de segmentação, predição da pose e refinamento de pose. A etapa de segmentação é realizada através de uma arquitetura CNN combinada com uma rede *ad-hoc*, onde os objetos são segmentados de modo incremental na cena 2D. Com a geração de máscaras de segmentação binária de baixa resolução aplicadas com o processo

de deslizamento de janelas, os centróides de cada objeto são encontrados em cada região e salvos para predição da pose 3D. Esse processo de segmentação permite ao modelo lidar com oclusão dos objetos e o uso de arquiteturas mais simples na primeira etapa. Para predição das poses 3D, uma outra arquitetura CNN é utilizada. A rede é treinada para detectar os objetos nas regiões definidas pelos centróides, estimando os cantos projetados na imagem para cada região. Uma nova abordagem de treinamento é utilizada para evitar o problema de ambiguidade na estimação dos pontos projetados em objetos simétricos. Durante o treinamento, as poses utilizadas para o treinamento desses objetos são limitadas, um objeto com ângulo de simetria α será definido com uma rotação 3D em módulo de α e não 2π. O passo de refinamento utiliza a máscara binária gerada para reajustar os erros entre os pontos projetados (utilizados para renderizar o modelo do objeto na cena). O BB8 consegue lidar com os casos de oclusão e falhas de estimação para objetos simétricos, contudo necessita de grande tempo de processamento, alcançando em média apenas 3 fps.

Para tentar lidar com o problema de desempenho das CNN, Kehl (KEHL et al., 2017) propõe o SSD-6D utilizando arquiteturas *singleshot* para detecção 6-DoF. As arquiteturas *singleshot* são modelos CNN propostos para detecção de objetos em um único passo de predição, recebendo como entrada a imagem e retornando diretamente a estimação da posição do objeto (sem passos intermediários bem definidos para segmentação, extração de características e/ou refinamentos). O SSD-6D se baseia na arquitetura para detecção 2D SSD (LIU et al., 2016) e utiliza apenas a informação de imagens RGB para estimação da pose, a rede é treinada através de um conjunto de dados rotulados e aproxima a estimação de pose 6-DoF por regressão. Neste trabalho, o processo de geração e aumento de dados é utilizado para o treinamento do modelo, em que os objetos são renderizados em diferentes pontos de vista a uma distância fixa da câmera, em seguida os objetos são aplicados em imagens com planos de fundos aleatórios. Para treinamento do modelo, a função de *loss* utilizada leva em consideração o valor de confiança para cada classe em diferentes regiões da imagem, a estimação do ponto de vista do objeto encontrado e da região 2D estimada. O SSD-6D foi proposto como uma técnica *singleshot* e pode lidar com diversos desafios como oclusão, ambiente poluído e mudança de iluminação, mas para conseguir resultados equivalentes com as técnicas do estado da arte ainda precisou receber uma etapa de refinamento para verificação da pose estimada baseado no algoritmo ICP. O modelo consegue alcançar em média 10 fps utilizando o passo de refinamento.

Tekin (TEKIN; SINHA; FUA, 2017) desenvolveu uma abordagem semelhante à proposta do SSD-6D, utilizando uma arquitetura *singleshot* para realizar a estimação de pose através da predição dos pontos projetados do *bounding box* 3D, como feito com o BB8. O modelo utiliza como base a arquitetura YOLO (REDMON et al., 2016; REDMON; FARHADI, 2017), e realiza a predição dos 8 cantos do *bounding box* 3D juntamente com a predição do centróide do objeto, valor de confiança da detecção e probabilidade entre as classes

(realizando todo processo com um único passo de predição da CNN). Para obter as poses em forma de rotação e translação é utilizado o algoritmo PnP, que estima os vetores de transformação conhecendo-se apenas os valores dos pontos 3D dos modelos, os pontos correspondentes projetados na imagem e a matriz de câmera utilizada. A arquitetura consegue detectar objetos para imagens de diferentes resoluções. Para treinamento as imagens dos objetos são segmentadas e aplicadas em fundos aleatórios, para evitar sobre-ajuste do modelo. No trabalho é definida a função de *loss* que considera a distância entre os pontos projetados preditos e os pontos de anotados para cada imagem, o *loss* total é definido pela predição dos pontos, classificação do objeto em cada região e classificação entre os objetos conhecidos. No modelo, o erro de predição para os objetos é mais penalizado que o erro de predição do plano de fundo. Para treinamento o modelo necessita apenas do conhecimento do *bounding box* 3D do objeto e não utiliza nenhum tipo de pós-processamento, conseguindo obter resultados equivalentes a trabalhos do estado da arte e alcançando uma taxa média de 50 fps na detecção. O modelo é robusto a desafios como oclusão, fundo poluído e orientação da câmera, mas possui algumas falhas em casos de especularidade, imagens borradas, objetos muito próximos e oclusão severa. Na Tabela 1 é possível ver as principais caraterísticas de cada trabalho descrito nesta seção.

Tabela 1 – Lista de Características dos trabalhos para detecção e rastreamento 6-DoF revisados.

Técnica	Tempo Real (>=20fps)	Sensor	Objeto texturizado	Objeto Não texturizado	Dados Sintéticos	Data augmentation	Requer GPU	Pós processamento
(DOUMANOGLOU et al., 2016b)	Não	RGBD	Não	Sim	Não	Não	Sim	Não
(TEJANI et al., 2018)	Não	RGBD	Não	Sim	Sim	Não	Não	Sim
(TAN; NAVAB; TOMBARI, 2017)	Sim	RGBD	Não	Sim	Sim	Não	Não	Não
(SOCK et al., 2018)	Não	Profundidade	Sim	Sim	Sim	Sim	Sim	Não
(KAO et al., 2018)	-	RGB	Sim	Sim	Não	Não	Sim	Não
(LI; BAI; HAGER, 2018)	-	RGBD	Sim	Sim	Sim	Não	Sim	Não
(GARON; LALONDE, 2017)	Sim	RGBD	Não	Sim	Sim	Sim	Sim	Não
(KEHL et al., 2016)	Não	RGBD	Não	Sim	Sim	Não	Sim	Não
(ZHANG; CAO, 2017)	Não	RGBD	Não	Sim	Não	Não	Sim	Não
(GAO et al., 2018)	-	Profundidade	Não	Sim	Sim	Não	Sim	Sim
(XIANG et al., 2017)	Não	RGB/RGBD	Sim	Sim	Não	Não	Sim	Sim
(LI et al., 2018)	Não	RGB	Não	Sim	Sim	Sim	Sim	Sim
(SAHIN; KIM, 2018)	-	RGBD	Não	Sim	Sim	Não	Não	Não
(RAD; LEPETIT, 2017)	Não	RGB	Sim	Sim	Não	Sim	Sim	Sim
(KEHL et al., 2017)	Não	RGB	Sim	Sim	Sim	Sim	Sim	Sim
(TEKIN; SINHA; FUA, 2017)	Sim	RGB	Sim	Sim	Não	Sim	Sim	Não

A técnica desenvolvida por Tekin foi escolhida como base para o desenvolvimento desta dissertação. Uma das principais características que levaram a escolha do modelo é a alta acurácia, sendo comparável aos trabalhos do estado da arte, mesmo sem utilizar nenhuma etapa de pós-processamento. Entre os trabalhos encontrados, o método de Tekin é o único que consegue utilizar rede neural profunda para estimação em tempo real utilizando apenas informação da câmera RGB, o que torna o método viável para utilização em diversas aplicações práticas. O método foi desenvolvido e validado através dos resultados reportados. Outra característica importante para seleção do método é a capacidade de

lidar com objetos de diferentes tipos (planar, não planar, texturizado e não texturizado) e também a escalabilidade do método para detectar múltiplos objetos sem acréscimo do tempo de processamento. A análise sobre o comportamento do modelo foi realizada como forma de entender os pontos de falhas da técnica e propor abordagens para refinamento dos resultados gerados.

3.2 DATASETS PARA RASTREAMENTO E DETECÇÃO 6-DOF

Foi investigado o conjunto de *datasets* utilizados pelas técnicas selecionadas e avaliamos suas características como número de objetos utilizados, câmeras, tamanho do *dataset*, tipo de imagem utilizada, etc. A Tabela 2 mostra o resultado dos dados obtidos.

Tabela 2 – Lista de *datasets* utilizados pelos trabalhos selecionados.

Nome	Tamanho	Num. Objetos	Câmera	Tarefa	Tipo de objeto
YCB-dataset (CALLI et al., 2015b; CALLI et al., 2015a)	>265GB	77	Asus Xtion Pro Live	Manipulação Robótica	Real Texturizado
YCB-video-dataset (XIANG et al., 2017)	>265GB	21	Asus Xtion Pro Live	Rastreamento 3D	Real Texturizado
Doumanoglou et al. (DOUMANOGLOU et al., 2016b)	400MB	6	Primesense Carmine 1.09	Detecção 3D	Real Texturizado
Tejani et al. (TEJANI et al., 2014)	6GB	6	Primesense Carmine 1.09	Detecção 3D	Real Texturizado
T-LESS (HODAN et al., 2017)	2.7GB	30	Primesense Carmine 1.09	Detecção 3D	Real e Sintético Não Texturizado
LINEMOD (HINTERSTOISSER et al., 2012)	12GB	15	Microsoft Kinect V2	Detecção e Rastreamento 3D	Real e Sintético Não Texturizado
Toyota Light	2GB	21	Primesense Carmine 1.09	Manipulação Robótica	Sintético Não Texturizado
Rutgers APC (RENNIE et al., 2016)	5.3GB	14	Microsoft Kinect V1	Detecção 3D	Real Texturizado
TUD Light	27GB	3	Microsoft Kinect V1	Detecção 3D	Real e Sintético Não Texturizado
Deep 6Dof tracking dataset (GARON; LALONDE, 2017)	7GB	4	Microsoft Kinect V2	Rastreamento 3D	Sintético Não Texturizado
Branchman et al. (BRACHMANN et al., 2016)	4.4GB	20	-	Rastreamento 3D	Real Não Texturizado
ObjectNet3D (XIANG et al., 2016)	8.0GB	100	-	Alinhamento 3D	Real Não Texturizado
PROFACTOR 3D (AKKALADEVI et al., 2016)	520MB	4	Primesense Carmine 1.09	Rastreamento 3D	Sintético Não Texturizado

Pela tabela, é possível ver que a maioria dos *datasets* utilizados são específicos para tarefas de rastreamento e detecção 3D. As câmeras mais utilizadas entre os *datasets* analisados foram a Asus Xtion Pro Live, Primensense Carmine 1.09 e o Microsoft Kinect V1/V2, todas produzem informação RGB e de profundidade que são disponibilizadas nos *datasets*. Alguns *datasets* já disponibilizam um conjunto de dados sintéticos para treinamento dos modelos, geralmente utilizados por métodos baseados em modelos, e um conjunto de dados reais para testes. As imagens sintéticas fornecidas são geradas através de diferentes pontos de vista do objeto 3D renderizado em fundo preto.

A maioria dos *datasets* podem ser utilizados para avaliar desafios como oclusão, iluminação, fundo poluído, borramento, etc. Contudo, para todos os conjuntos de dados avaliados, gerados com imagens reais, o ambiente é sempre preservado e os parâmetros de câmera permanecem os mesmos para todo o conjunto de quadros disponível.

Neste trabalho utilizamos o **LINEMOD** para validação do método desenvolvido. O LINEMOD é um dos principais *datasets* utilizados para o treinamento de modelos de detecção 3D, possuindo diversos desafios como oclusão, fundo poluído, borramento e mudança de iluminação. O LINEMOD também foi utilizado pelo trabalho de referência, permitindo uma comparação direta para validação da implementação realizada.

3.3 GERAÇÃO DE DADOS SINTÉTICOS PARA TREINAMENTO

Um dos grandes problemas no treinamento de modelos de aprendizagem é relacionado a obtenção de dados rotulados. Métodos de aprendizado profundo necessitam de uma grande quantidade de exemplos para treinamento dos modelos. A obtenção de dados e rotulação muitas vezes é realizada de forma manual, demandando muito tempo e esforço.

As técnicas de aprendizagem de máquina baseiam-se nos dados fornecidos no treinamento para realizar a predição de novos conjuntos de dados. Isto é, assume-se que um conjunto de dados utilizado para treinamento pertence a um domínio fonte (D_f) que tem uma distribuição similar ou equivalente ao domínio alvo que será utilizado para teste (D_a) (PATEL et al., 2015).

Como visto na seção anterior, diversos *datasets* de larga escala contendo diversos tipos de objetos e cenários estão disponíveis atualmente para treinamento dos modelos (CALLI et al., 2015a; CALLI et al., 2017; DENG et al., 2009; EVERINGHAM et al., 2010). Contudo, para problemas que tratam de domínios diferentes e que necessitam de informações específicas, um novo conjunto de dados deve sempre ser gerado para viabilizar o uso dos métodos de aprendizagem. Donahue et al. (DONAHUE et al., 2014) mostra que uma pequena variação do domínio pode afetar o treinamento de técnicas que usam redes neurais profundas, prejudicando a acurácia do modelo utilizado.

Algumas abordagens utilizam *transfer-learning*, abordagem em que o problema é minimizado pela transferência da informação aprendida sobre um certo domínio, adaptando o conhecimento através da obtenção de um pequeno conjunto de dados que mapeiam a informação previamente aprendida em um novo domínio de interesse (PAN; YANG et al., 2010). O *transfer-learning* é útil quando o domínio de interesse é bem conhecido, sendo possível obter um subconjunto dos dados que possua boa representatividade sobre o problema alvo, contemplando a maioria dos casos de variações que tendem a ocorrer nas situações de teste.

No *transfer-learning*, geralmente, o domínio fonte tem uma distribuição completamente diferente do domínio alvo ($D_f \neq D_a$). Essa diferença entre os dados utilizados no conjunto

de treinamento faz com que o modelo tenha um fraco desempenho nas situações de teste (CSURKA, 2017).

De forma geral, o *transfer-learning* possibilita diminuir o impacto de obtenção dos dados, oferecendo um boa alternativa para reusar as informações disponíveis. Apesar disto, realizar esse procedimento em abordagens supervisionadas ainda é requer um certo trabalho na geração de um conjunto de treinamento que mapeiam a informação do domínio fonte para o domínio alvo.

Outra possibilidade utilizada para sobrepor esse problema é dada pelo desenvolvimento de métodos que propõem a geração de dados sintéticos para ampliar o conjunto de treinamento (HINTERSTOISSER et al., 2012; WARD; MOGHADAM; HUDSON, 2018), onde os modelos 3D dos objetos são utilizados para gerar imagens em diferentes cenários automaticamente. Na tarefa de detecção 6-DoF, para gerar os dados sintéticos, as poses são mapeadas em posições específicas da cena com o objetivo de capturar informações chaves de diferentes pontos de vista do objeto para treinamento. O modelo 3D do objeto é inserido a uma distância conhecida no mundo, e então a câmera é movimentada em torno de posições definidas gerando as imagens de treino com o modelo sintético (KEHL et al., 2016). Com isso, é possível computar o movimento relativo entre câmera-objeto, obtendo as poses para cada situação gerada. Adicionalmente podem ser inseridas diversas imagens de fundo, situações de iluminação ambiente, oclusão, etc. Essa abordagem é bastante utilizada para detecção e rastreamento 6-DoF (KEHL et al., 2017; GARON; LALONDE, 2017; TEJANI et al., 2018; DOUMANOGLOU et al., 2016b) em técnicas baseadas nas características geométricas, contudo não resolve totalmente o problema de geração de dados para técnicas baseadas em aprendizagem, visto que as distribuições entre os dados gerados para treinamento (modelo 3D renderizado em fundos aleatórios) e teste (imagens capturadas com o objeto real) continuam com diferenças relevantes. Para treinamento dos métodos que usam aprendizado profundo, um ajuste com dados reais ainda deve ser feito. Mas a necessidade de ter dados representativos é reduzida devido a variação da informação fornecida pelos dados sintéticos que tem características semelhantes ao objeto real.

Métodos mais recentes propõem técnicas de geração de dados sintéticos para treinamento com adaptação de domínio (DA, do Inglês *Domain Adaptation*), assumindo que são utilizados modelos em domínios diferentes para realizar a mesma tarefa. A adaptação de domínio é um tipo de *transfer learning*, em que o conhecimento obtido anteriormente é transferido, mapeando a informação para um novo conjunto de dados apenas utilizando a informação do domínio fonte (sem ajuste para os dados do domínio objetivo). No trabalho de Hinterstoisser et al.(HINTERSTOISSER et al., 2017) é mostrada a eficiência do uso do DA realizando a estimação da posição do objeto treinando apenas a camada de classificação de uma CNN. Durante o novo treinamento, os parâmetros das camadas iniciais da arquitetura são congelados com os valores do treinamento anterior, apenas as últimas camadas são alteradas com as informações dos dados reais.

Essa abordagem de treinamento utilizando refinamento de apenas algumas camadas do modelo por *transfer-learning* é bastante utilizada para arquiteturas DNN, em que é criada uma representação hierárquica dos dados. Nestes modelos, mantém-se o conjunto de filtros aprendidos em camadas iniciais da arquitetura onde são geradas representações mais genéricas dos dados e, atualiza-se os filtros das camadas finais onde há uma representação especializada nos dados de interesse.

Para evitar o problema de diferença entre os domínios fonte e alvo, foram propostos algoritmos que ajustam os dados sintéticos gerados através dos modelos 3D do objeto. Estes algoritmos tem como objetivo entender e aplicar as similaridades entre as distribuições dos dados adequando as imagens disponíveis com variações de ambiente e informações sintéticas, de forma a gerar um conjunto de treinamento que seja adequado para as técnicas de aprendizagem ($D_f \approx D_a$) sem a necessidade de obtenção de dados e rotulação manual.

Em Rozantev et al. (ROZANTSEV; LEPETIT; FUA, 2015) um algoritmo é utilizado para aprender os parâmetros de ambiente que definem o conjunto de dados reais. A ideia principal é diminuir o *gap* entre o domínio real e o domínio sintético gerado por simulação. Nesta abordagem, são desenvolvidos diversos cenários contendo casos que simulam as possíveis variações do domínio alvo (iluminação, reflexos, oclusão) sem a necessidade de rotulação manual dos dados.

Seguindo outra direção, certos autores afirmam que a utilização de randomização dos domínios (DR, do inglês *Domain Randomization*) pode diminuir o impacto do treinamento de dados sintéticos (TOBIN et al., 2017; BORREGO et al., 2018). A ideia principal desta abordagem é gerar um maior número de variações aleatórias no conjunto de dados sintéticos, de forma que modelo considere o cenário real como apenas mais um caso de variação a ser classificada, aprendendo apenas as características mais importantes do mesmo e evitando o sobre-ajuste. O DR é bastante utilizado para detecção de objetos em 2D. Por se tratar de uma nova abordagem, não foram encontradas trabalhos que utilizam DR aplicados para rastreamento e/ou detecção 6-DoF durante o tempo de desenvolvimento deste trabalho. Adaptamos os procedimentos mencionados para implementar a abordagem de geração randômica para o caso 6-DoF.

Nesta dissertação, o treinamento com dados reais foi combinado com a geração de dados sintéticos, testando o modelo em diversos cenários, e avaliando como a técnica desenvolvida se comporta para os diversos tipos de dados inseridos durante o treinamento. Serão avaliados como a geração de dados influencia nos resultados obtidos em diferentes cenários e o quanto o modelo pode ser dependente em relação às informações de dados reais inseridas para treinamento. Para diminuir o impacto na diferença entre os domínios sintéticos e real, foi utilizada uma abordagem similar a (ROZANTSEV; LEPETIT; FUA, 2015). Foi realizada a adaptação dos parâmetros do ambiente variando os parâmetros de renderização, tentando aproximar seus valores a situações de casos reais. O objetivo é in-

vestigar a possibilidade de utilizar apenas dados sintéticos no treinamento, aproximando a distribuição do conjunto sintético com os dados reais.Foi avaliado também o procedimento aplicado por Tremblay et al. (TREMBLAY et al., 2018), adaptando o processo de geração aleatória com DR para o método de detecção 6-DoF desenvolvido, com o objetivo de melhorar a precisão do modelo em diferentes casos de usos não conhecidos.

4 RASTREAMENTO 6-DOF BASEADO EM APRENDIZAGEM PROFUNDA

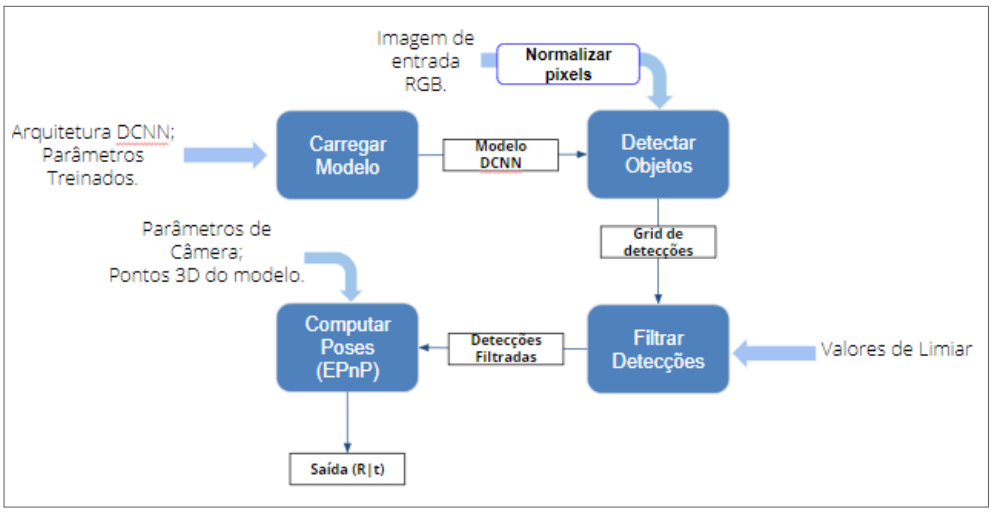
Através da revisão de literatura realizada, foi escolhido o método proposto em (TEKIN; SINHA; FUA, 2017) para ser desenvolvido como base nesta dissertação. Avaliando o método para diferentes cenários, investigando possíveis refinamentos com a aplicação de abordagens de processamento e geração de dados que auxiliam na generalização e precisão do modelo. Como visto anteriormente, a escolha foi realizada com base nas principais características apresentadas pelo trabalho como a execução em tempo real, aprendizagem utilizando apenas câmeras RGB, capacidade para lidar com diferentes tipo de objetos, escalabilidade para múltiplos objetos sem aumento do processamento e alta acurácia sem a necessidade de pós-processamento.

Neste capítulo é descrito o You Only Look Once (YOLO) (REDMON et al., 2016), trabalho utilizado como base para o desenvolvimento do método proposto por Tekin. E, em seguida serão especificadas as modificações desenvolvidas no método para realizar a detecção 6-DoF. Posteriormente serão discutidas as técnicas de geração de dados sintéticos utilizadas para refinamento dos resultados.

4.1 DETECÇÃO DE OBJETOS 6-DOF

A visão geral do método base é visto na Figura 6. O modelo utilizado para detecção foi proposto originalmente por Tekin (TEKIN; SINHA; FUA, 2017) e se baseia na ideia de utilizar modelos *singleshot* como forma de obter bom desempenho para o tempo de execução, realizando o processo de estimação de pose em um único passo de predição da rede neural profunda.

Figura 6 – Funcionamento da detecção em 6-DoF.



Fonte: Elaborado pelo autor, 2018

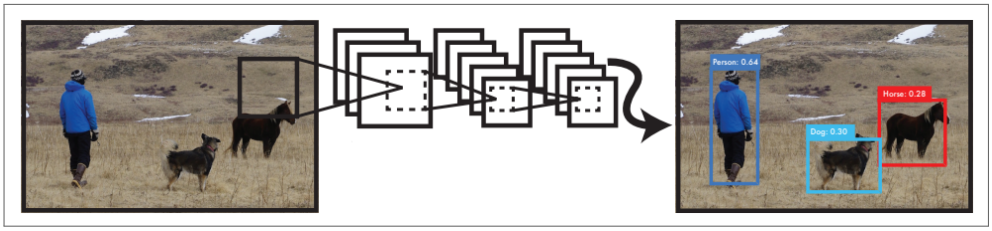
O método desenvolvido para detecção 6-DoF é baseado na segunda versão do YOLO (REDMON; FARHADI, 2017), a arquitetura implementada para este modelo é também chamada de **Darknet 19**.

4.1.1 Detecção de objetos 2D com YOLO

O YOLO (You Only Look Once) é um método desenvolvido para realizar a detecção em larga escala de objetos em imagens 2D (REDMON et al., 2016; REDMON; FARHADI, 2017). O modelo utiliza uma arquitetura DCNN, que recebe uma imagem RGB como entrada e retorna um conjunto de predições composta pela posição e valor de confiança de cada objeto na imagem (Figura 7).

O modelo foi projetado para realizar a detecção, segmentação e classificação dos objetos em um único passo de predição. Esta característica possibilita ao método alcançar taxas de até 155 fps em detecção 2D, superando técnicas propostas para a mesma tarefa, mesmo com múltiplos objetos sendo detectados. A detecção de múltiplos objetos pode ser realizada sem perda de desempenho, devido a estrutura da arquitetura que se baseia em detecção de âncoras percorrendo um *grid* de ativação obtido como saída da última camada convolucional.

Figura 7 – Visão geral do YOLO. O modelo recebe como entrada uma imagem RGB. A rede extrai e interpreta as características da imagem em um único passo de predição, retornando como resposta um conjunto de regiões que definem a posição de cada objeto detectado na imagem de entrada junto com seu valor de confiança.



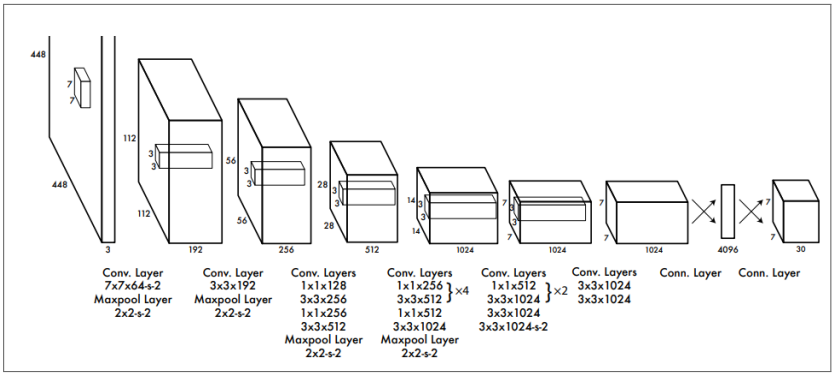
Fonte: Adaptado de (REDMON et al., 2016)

Para realizar a detecção de objetos com o YOLO em uma única etapa, é realizado um treinamento fim-a-fim com dados rotulados de acordo com a saída desejada para o modelo.

Durante o treinamento, o modelo aprende automaticamente informações relevantes sobre os objetos de interesse, gerando um extrator de características que pode recuperar as informações desejadas para toda a imagem de entrada. As características são extraídas em diversas camadas convolucionais, organizadas de forma sequencial, onde é gerada uma representação hierárquica dos dados (Figura 8). Nas camadas iniciais são extraídas informações genéricas contidas na imagem, representadas em características de alto nível.

A medida que a informação avança na arquitetura, as características são mapeadas para informações específicas sobre os objetos contidos em cada região da imagem.

Figura 8 – Arquitetura DCNN utilizada pelo YOLO.

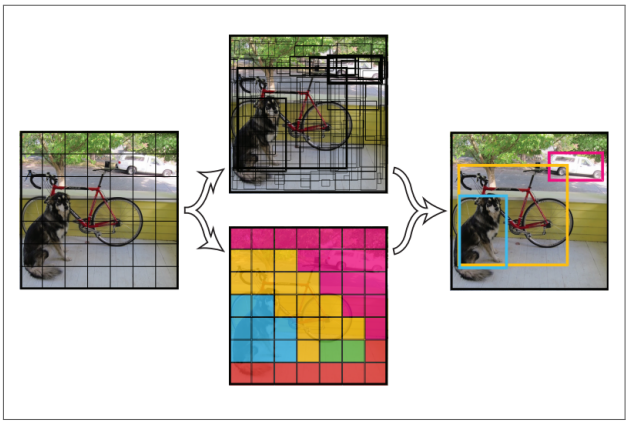


Fonte: (REDMON et al., 2016)

A organização da arquitetura é construída de forma a mapear cada região da imagem em um *grid* de ativação de tamanho $S \times S$ (Figura 9). As informações de saída estarão contidas em cada célula deste *grid*, se o centro do objeto está contido em uma célula específica do *grid*, então esta célula será responsável pela detecção do objeto na imagem.

Cada célula realiza a predição das regiões que contém o objeto, e do valor de confiança para o objeto predito em cada uma destas regiões. O valor de confiança define o nível de confiança que o modelo tem de que realmente encontrou um objeto para aquela célula do *grid*, sendo representado como uma probabilidade com valor entre 0 e 1. Adicionalmente, a rede também retorna uma distribuição de probabilidade condicional sobre todos os objetos conhecidos pelo modelo, tendo o objeto correto a maior probabilidade na distribuição.

Figura 9 – Representação do grid de saída utilizado peloYOLO.



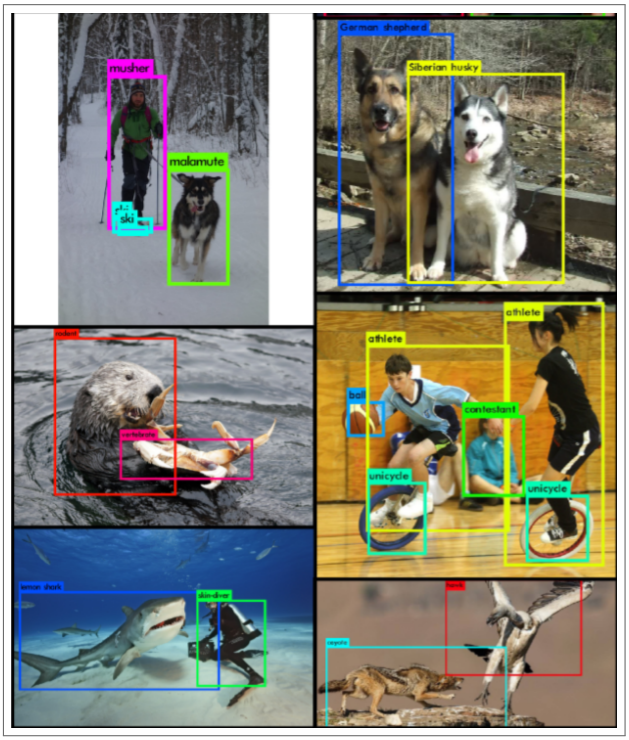
Fonte: Adaptado de (REDMON et al., 2016)

A acurácia do modelo é computada através da utilização da métrica de Intesection over Union (IoU). Tal métrica computa a taxa de sobreposição entre a intersecção da região predita pelo modelo e a região rotulada sobre a união destas regiões e quanto mais alta a

sobreposição, melhor o resultado. Define-se um limiar para considerar que a predição do modelo foi correta, geralmente um IoU acima de 0.5 é aceitável.

Cada célula no *grid* $S \times S$ computa o valor B de regiões definidas (*bounding boxes*) pelas coordenadas x, y, w, h, onde (b_x, b_y) correspondem a posição 2D do objeto na região b, (b_w, b_h) são a largura e altura da área que contém o objeto na região b, o valor de confiança C_i e C probabilidades condicionais. Então a saída da rede pode ser representada por um tensor de dimensões $S \times S \times D$, com $D = B(4 + 1) + C$. Alguns resultados obtidos pelo YOLO podem ser vistos na Figura 10.

Figura 10 – Resultados de detecção com YOLO.



Fonte: Adaptado de (REDMON; FARHADI, 2017)

Para encontrar as coordenadas (b_x, b_y), os valores das coordenadas preditas (t_x, t_y) são parametrizados em relação ao centro da célula que a contém. As ativações das coordenadas são limitadas entre o intervalo $[-1, 1]$, dado pela função sigmoid (δ). Esse valor representa o deslocamento do ponto central do objeto dentro da célula responsável por ele no *grid*. Para encontrar a posição absoluta das coordenadas, soma-se o ponto superior esquerdo da célula (C_x, C_y) com os valores de deslocamento preditos (Equação 4.1).

$$\begin{aligned} b_x &= C_x + \delta(t_x), \\ b_y &= C_y + \delta(t_y). \end{aligned} \tag{4.1}$$

Para encontrar os valores de largura e altura da região de interesse (b_w, b_h), o YOLO utiliza o método de otimização das ativações baseadas em valores de referência definidos

(valores de âncoras). Os parâmetros são otimizados de forma a realizar uma transformação de uma região conhecida previamente (p_w, p_h), para a região de interesse que delimita o objeto na imagem de teste (Equação 4.2). Com isso, é possível remover as camadas totalmente conectadas (*fully-connected*) da arquitetura, gerando a predição diretamente pela ativação das camadas convolucionais.

$$\begin{aligned} b_w &= p_w \times e^{t_w}, \\ b_h &= p_h \times e^{t_h}. \end{aligned} \tag{4.2}$$

Durante o treinamento, o valor de confiança é definido pela probabilidade do objeto (rotulado no conjunto de dados) multiplicado pelo valor de IoU computado. Quando não há objeto na célula o valor de IoU é 0, fazendo com que o valor de confiança seja o mesmo, caso contrário, o valor de confiança é limitado pela probabilidade do objeto encontrado.

A função de custo do YOLO (Equação 4.3) é definida através de uma composição de funções somadas, relativas às informações retornadas como resposta (pontos do objeto na imagem, valor de confiança para a célula e probabilidades entre as classes). A função de custo é utilizada para computar o erro do modelo em relação ao conjunto de exemplos de treinamento, fornecendo uma métrica para auxiliar no ajuste dos parâmetros durante o processo de treinamento. O cálculo da função de perda é realizado para cada célula do *grid* e para cada predição B da célula, totalizando $S \times S \times B$ operações para cada exemplo.

Para computar o erro em relação aos pontos preditos é utilizada a distância euclidiana entre as coordenadas do centro do objeto preditas (b_x, b_y) e das anotações fornecidas pelos exemplos de treinamento (b_x^{gt}, b_y^{gt}). A distância euclidiana também é utilizada para computar o erro de estimação da largura e altura (b_w, b_h) do objeto na imagem, utilizando as anotações do conjunto de treinamento (b_w^{gt}, b_h^{gt}). O nível de confiança é calculado através do valor de IoU C_i computado através das regiões preditas, sendo $C_i^{gt} = 0$ em células que não contém um objeto e $C_i^{gt} = 1$ em células que contém um objeto.

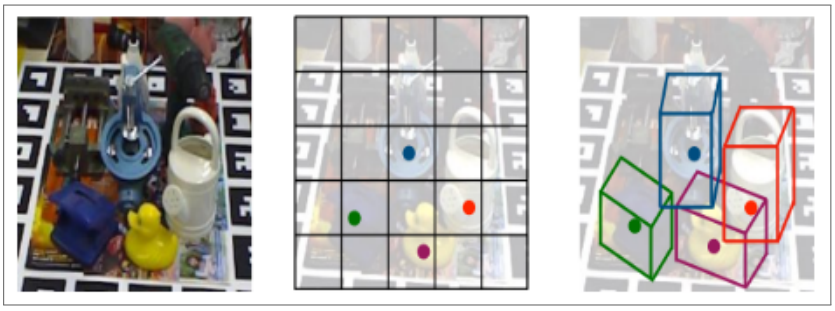
$$\begin{aligned} &\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} ((b_{x_i} - b_{x_i}^{gt})^2 + (b_{y_i} - b_{y_i}^{gt})^2) \\ &+\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} ((b_{w_i} - b_{w_i}^{gt})^2 + (b_{h_i} - b_{h_i}^{gt})^2) \\ &+\lambda_{obj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} (C_i - C_i^{gt})^2 \\ &+\lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} (C_i - C_i^{gt})^2 \\ &+\lambda_{coord} \sum_{i=0}^{S^2} \sum_{c \in classes} (p_i(c) - p_i^{gt}(c))^2 \end{aligned} \tag{4.3}$$

O parâmetro λ_{coord} representa o peso para predição dos pontos durante a otimização dos parâmetros do modelo. λ_{obj} e λ_{noobj} representam respectivamente o peso para o valor de confiança quando há um objeto na célula avaliada e quando não há um objeto na célula, sendo $\lambda_{obj} = 1$ e $\lambda_{noobj} < 1$, priorizando a informação de células que realmente contém um objeto de interesse e diminuindo a influência da informação do plano de fundo durante o aprendizado. O parâmetro λ_{prob} representa o peso entre as classes conhecidas, sendo definido como 1 para realizar o treinamento das informações de todos os objetos igualmente.

4.1.2 Arquitetura para estimação 6-DoF

Para realizar a detecção 6-DoF, a camada de saída do modelo é alterada para extrair 9 pontos da imagem, ao contrário das 4 coordenadas que definem o *bounding box 2D* do objeto na imagem. Dentro desses conjuntos de pontos estão descritos os 8 cantos do *bounding box 3D* do objeto $(x_1, y_1), (x_2, y_2), ..., (x_8, y_8)$ e o valor do centróide do objeto na imagem (x_c, y_c), com um total de 18 coordenadas (valores de ponto flutuante) a serem estimadas. Os valores preditos correspondem aos pontos do modelo (coordenadas de mundo) projetados nas coordenadas da câmera. A Figura 11 mostra a representação para a saída da rede no caso 6-DoF.

Figura 11 – Representação do grid de saída extendido para detecção 6-DoF.



Fonte: Adaptado de (TEKIN; SINHA; FUA, 2017)

As dimensões de saída da última camada convolucional pode ser então definida como $S \times S \times D$, onde $D = B(18 + 1) + C$, S = tamanho do grid de saída, B = número de âncoras utilizadas (*bounding boxes de referência*) e C o número de classes conhecidas.

Assim como no YOLO, os valores preditos dos cantos são retornados como deslocamentos $(t_{x1}, t_{y1}, ..., t_{x8}, t_{y8})$ em relação à posição do *grid* (C_x, C_y) que contém o objeto, como visto na equação 4.4. O valor do centróide é parametrizado em relação ao seu deslocamento (t_{cx}, t_{cy}) no interior da posição do grid que o contém e é limitado entre o intervalo

$[0, 1]$ pela função sigmoid δ (Equação 4.5).

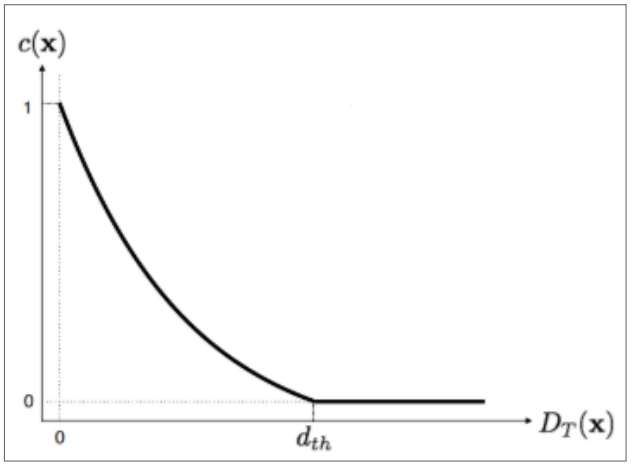
$$\begin{aligned} x_i &= C_x + t_{xi}, \\ y_i &= C_y + t_{yi}, \\ onde\ i &= 1,\ 2,\ ...,\ 8. \end{aligned} \tag{4.4}$$

$$\begin{aligned} x_c &= C_x + \delta(t_{cx}), \\ y_c &= C_y + \delta(t_{cy}). \end{aligned} \tag{4.5}$$

Para computar o valor de confiança durante o treinamento do modelo uma nova função é definida, substituindo a utilização do IoU para detecção 2D, devido ao alto custo para computar a intersecção da área de sobreposição no sistema de coordenadas 3D. A função retorna o valor de confiança $c(\mathbf{x})$ para os valores dos pontos estimados $\mathbf{x}$ baseado na distância euclidiana $D_t(\mathbf{x})$, calculada através da distância entre os pontos anotados dos exemplos no conjunto de treinamento e os pontos preditos pelo modelo para os respectivos exemplos (Equação 4.6). O nível de aceitação do valor de confiança é dado por um valor de limiar d_{th} que define o erro máximo aceito para a estimação dos pontos. A Figura 12 mostra o comportamento do valor de confiança em função da distância $D_t(\mathbf{x})$.

$$c(\mathbf{x}) = \begin{cases} e^{\alpha(1-\frac{D_t(\mathbf{x})}{d_{th}})}, & \text{se } D_t(\mathbf{x}) < d_{th}, \\ 0, & \text{caso contrário.} \end{cases} \tag{4.6}$$

Figura 12 – Função para calculo do valor de confiança. O valor de confiança depende da distância entre os pontos preditos e as anotações do conjunto de dados no treinamento. O erro máximo aceito é definido pelo valor de limiar d_{th}.



Fonte: Adaptado de (TEKIN; SINHA; FUA, 2017)

Com isso pode-se definir também os valores para as probabilidades de estimação entre as classes conhecidas pelo modelo, e a função de custo referente à distribuição pode ser calculada com o uso de *cross entropy* (LI; LEE, 1993).

Durante o treinamento, a função de custo L utilizada para otimização dos parâmetros da arquitetura pode ser definida pela equação 4.7:

$$L = \lambda_{pt} L_{pt} + \lambda_c L_c + \lambda_p L_p, \tag{4.7}$$

Onde L_{pt}, L_c e L_p representam os valores de custo retornados para a estimação dos pontos, valor de confiança e probabilidades das classes respectivamente. Os valores λ_{pt}, λ_c e λ_p representam os pesos para cada custo computado durante o processo de otimização do treinamento.

O segundo passo consiste em filtrar as informações retornadas no *grid* de saída. Nessa fase, os valores das ativações são avaliados através de um limiar definido de forma a manter apenas as informações mais confiáveis em cada posição do *grid*.

O último passo consiste em encontrar as correspondências entre os pontos 2D-3D para as posições do *grid* que contenham um objeto de interesse, utilizando o algoritmo para a estimação de pose EPnP (LEPETIT; MORENO-NOGUER; FUA, 2009).

Neste algoritmo, que recebe como entrada também os parâmetros intrínsecos da câmera em questão, são utilizados os 9 pontos projetados estimados pelo modelo e os 9 pontos correspondentes conhecidos previamente através do modelo 3D do objeto para estimar os valores de parâmetros extrínsecos da câmera. Assumindo que é conhecida a matriz de parâmetros intrínsecos da câmera K e, assumindo que os pontos de referência utilizados não são co-planares, é possível utilizar a informação dos pontos do modelo em coordenadas do mundo $P^w = [x, y, z]$ e a informação dos pontos projetados na coordenada de câmera $P^c = [u, v]$ para encontrar os vetores referentes a rotação ($r = [r_\alpha, r_\beta, r_\gamma]$) e translação 3D ($t = [t_x, t_y, t_z]$) da câmera na cena.

Os pontos P^w e P^c são descritos através da soma de um conjunto de pontos de controles virtuais, no qual gera-se um sistema linear que retorna os coeficientes para estimação dos valores de r e t. Após isso, r e t são encontrados de forma a minimizar o erro de reprojeção entre os pontos P^w e P^c, que é dado pela distância euclidiana entre o ponto 3D projetado ao correspondente 2D na imagem.

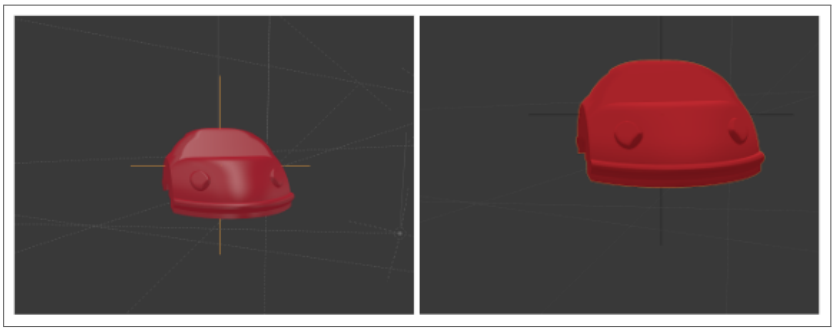
4.2 GERAÇÃO DE DADOS SINTÉTICOS

O procedimento utilizado para geração de imagens sintéticas se baseia no trabalho de Kehl et al. (KEHL et al., 2017), em que as imagens de diferentes pontos de vista são geradas a partir da rotação da câmera em pontos discretizados de uma semiesfera onde o objeto está centralizado. Diferente do trabalho original em que a variação de profundidade é reproduzida pela estimação dos pontos projetados, realizamos a variação da distância diretamente no processo de renderização. Com isso as informações das transformações são capturadas diretamente pela ferramenta de renderização, evitando perda de informação visual e erro de estimação em relação a distância do objeto para câmera.

Também foram realizadas variações dos parâmetros da câmera virtual utilizada para gerar as imagens sintéticas, evitando o sobre-ajuste (do inglês *overfitting*) do modelo para os parâmetros específicos de uma única câmera, inserindo robustez ao modelo para realizar a estimação em diferentes aberturas de câmera e coeficientes de distorção.

Na geração de imagens sintéticas foi utilizado o modelo 3D do objeto texturizado com material difuso e cores similares ao objeto original. Nesse ponto, não há preocupação com o foto realismo do modelo gerado. As imagens são geradas em fundo preto com pontos de iluminação em posições aleatórias, um exemplo do resultado pode ser visto na figura 13.

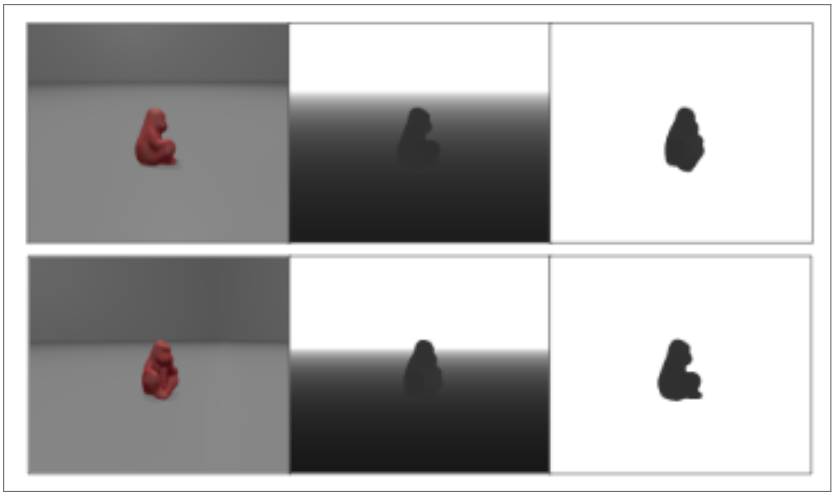
Figura 13 – Exemplo do conjunto de imagens sintéticas geradas utilizando a renderização simples pelo modelo 3D do objeto.



Fonte: Elaborado pelo autor, 2018

No processamento de cada imagem sintética, é gerada uma imagem RGB, as anotações para o modelo (pontos projetados, vetor de rotação e translação) e um mapa de profundidade para a cena (Figura 14).

Figura 14 – Conjunto de imagens de saída na geração de dados sintéticos. Na esquerda pode ser visto a imagem RGB renderizada, a imagem do centro mostra o mapa de profundidade gerado, e a direita a máscara binária filtrada com as informações o objeto de interesse na cena.



Fonte: Elaborado pelo autor, 2018

Para o treinamento, os objetos são aplicados em imagens de fundos aleatórios do dataset VOC2012 (EVERINGHAM et al., 2010; EVERINGHAM; WINN, 2011). Nesse processo é utilizada a informação sintética em conjunto com os dados reais, realizando o procedimento de transfer-learning clássico com o modelo inicializado pelos parâmetros treinados da *Darknet 19* 2D original.

4.3 GERAÇÃO DE DADOS SINTÉTICOS COM ADAPTAÇÃO DE DOMÍNIO

Para gerar as imagens aplicadas ao treinamento com DA, foi realizada a criação de imagens com técnicas de renderização foto-realistas. O objetivo da utilização dessa abordagem é investigar a elaboração de uma ferramenta de geração de dados sintéticos que se aproxime o máximo possível do conjunto de dados reais, sendo realizado o treinamento do modelo sem nenhuma informação sobre o domínio de dados reais.

Em (ROZANTSEV; LEPETIT; FUA, 2015), é desenvolvida uma técnica que recebe um conjunto de dados que representa o objetivo de classificação. O algoritmo estima, a partir dos dados, o conjunto de parâmetros de renderização para geração de imagens sintéticas. Tal ideia foi utilizada para entender o procedimento de geração de imagens sintéticas com adaptação e, com isso, analisar os melhores parâmetros para renderização. Inicialmente foram definidos os parâmetros **manualmente** para geração de imagens sintéticas, com a finalidade de avaliar o comportamento da técnica de detecção para este conjunto de dados.

Nessa tarefa, há uma preocupação em gerar um modelo virtual com alta fidelidade ao objeto real, aplicando texturas que simulam a cor, reflexão e mapas de normais. Nesta dissertação, os pontos de iluminação são definidos manualmente em pontos fixos da cena. O ambiente é variado controladamente, com cenários semelhantes (ou próximos) dos possíveis casos de uso da técnica desenvolvida. A figura 15 mostra um caso de exemplo da geração de dados com esta abordagem.

Figura 15 – Exemplo do conjunto de imagens sintéticas geradas utilizando a abordagem de renderização por adaptação de domínio.

Fonte: Elaborado pelo autor, 2018

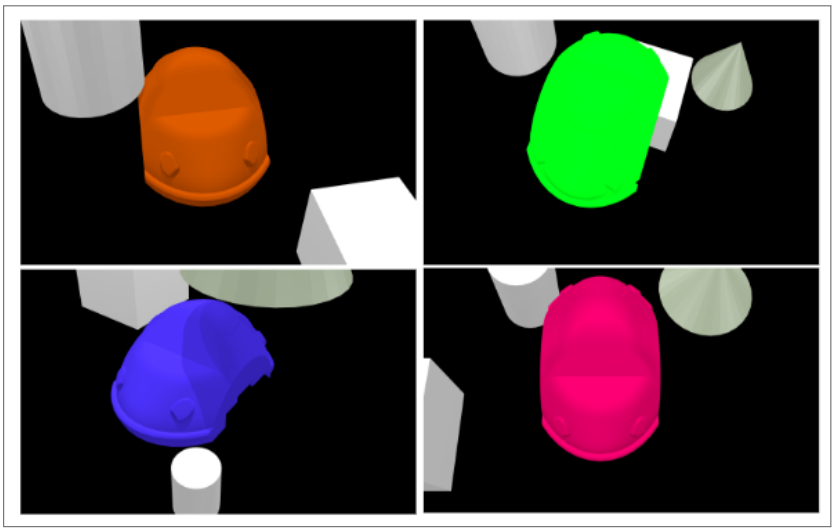
4.4 GERAÇÃO DE DADOS SINTÉTICOS UTILIZANDO RANDOMIZAÇÃO DE DOMÍNIO

Para realizar a tarefa de geração de imagens com randomização do domínio, o trabalho proposto em (TREMBLAY et al., 2018) foi utilizado. O trabalho descreve o processo de geração de imagens sintéticas variando-se a estrutura do ambiente, objetos e ruídos para realizar a detecção de objetos em 2D. Nesta dissertação, foi utilizado um procedimento semelhante, adaptando o processo descrito pelos autores para realizar a detecção em 6-DoF.

Neste conjunto de dados não há preocupação em manter a consistência dos parâmetros de renderização para casos reais. Com a randomização de domínios, devem ser criados o maior número de variação entre os cenários, de forma a aumentar o grau de generalização do modelo, fazendo com que sejam aprendidas as características mais relevantes da estrutura geométrica do objeto. Quando variamos a cor, qualidade, textura e iluminação incidente no objeto, estamos tentando induzir o modelo a entender as mudanças entre os cenários sintéticos, fazendo com que cenários reais sejam considerados apenas mais uma variação a ser estimada.

Para geração dos dados, foram utilizados objetos de distração na cena, alteração do plano de fundo e a textura do ambiente, mudança da textura e cor do objeto, inserção de ruídos aleatórios, mudança de iluminação e escala. Alguns exemplos de imagens geradas podem ser vistos na Figura 16.

Figura 16 – Exemplo do conjunto de imagens sintéticas geradas utilizando a abordagem de renderização por randomização de domínio.



Fonte: Elaborado pelo autor, 2018

5 IMPLEMENTAÇÃO

5.1 DISPOSITIVOS E CONFIGURAÇÃO

Para a captação das imagens RGB de treinamento e testes foram utilizadas 3 câmeras distintas e seus parâmetros intrínsecos foram estimados: **Iphone X** (Câmera 01), **Samsung Galaxy S8** (Câmera 02) e uma **Webcam Logitech C920** (Câmera 03). O desenvolvimento foi feito em um desktop com processador **Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz**, com memória de **8GB RAM** e placa gráfica **NVidia GeForce GTX 1080Ti 11GB RAM**, utilizando o sistema operacional **Ubuntu 16.04 OS**.

O código foi desenvolvido utilizando a linguagem de programação **Python** versão 3.6. A lista de bibliotecas utilizadas é descrita no Apêndice B.

Para geração de dados sintéticos foi utilizado o software **Blender** v2.79 com a biblioteca **Cycles Render** para renderização das imagens.

5.2 DATASETS UTILIZADOS

Para treinamento e avaliação foi utilizado inicialmente o dataset LINEMOD (HINTERSTOISSER et al., 2012), assim como no trabalho original de Tekin et al. (TEKIN; SINHA; FUA, 2017). Foi realizado o mesmo procedimento aplicado aos trabalhos que utilizam o conjunto de dados reais. O conjunto de imagens é dividido em 15% das imagens para treinamento, utilizando *data augmentation*, e 85% para testes. Na Figura 17 é possível ver os modelos dos objetos fornecidos para o conjunto de dados do LINEMOD.

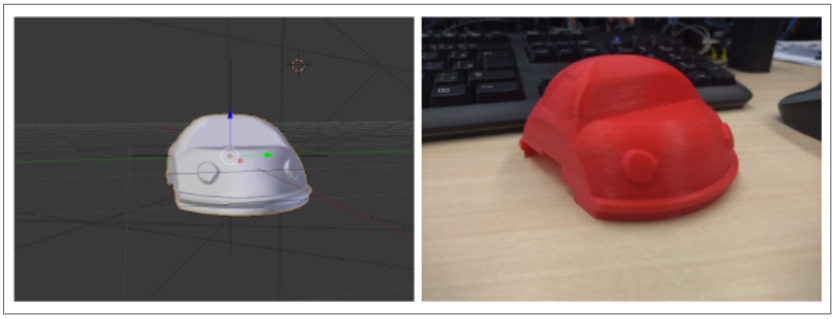
Figura 17 – Modelos 3D fornecidos para os bjetos utilizados no LINEMOD.

Fonte: Elaborado pelo autor, 2018

Adicionalmente foi gerado um dataset utilizando um novo objeto modelado e posteriormente impresso em uma impressora 3D, para avaliar sua robustez em diversos cenários e desafios (Figura 18). O objeto foi selecionado por conter uma estrutura similar aos objetos disponíveis no conjunto de dados do LINEMOD utilizado originalmente, o que possibilita

realizar uma comparação em novos cenários com um objeto que contém desafios similares aos encontrados no conjunto avaliado anteriormente. O objeto selecionado também possibilita a investigação do método para casos de uso em detecções de objetos impressos em 3D, onde o modelo 3D do objeto é utilizado previamente para realizar alguns ajustes do método com o conjunto de imagens sintéticas antes da geração do objeto pela impressora. Geramos um conjunto de treinamento gravado com o Iphone X, em que é possível aplicar o procedimento de *data augmentation*. Na rotulação dos dados aplicamos o procedimento de calibração com **OpenCV** e geração das anotações de pose com o **ArUco**. Para testes geramos um novo conjunto de vídeos com variação do ambiente, condições de iluminação, movimento e câmera utilizada. Para as imagens da câmera Iphone geramos um vídeo dedicado ao treinamento/validação e outras sequências para teste com as variações descritas. Para as câmeras S8 e a Logitech C920 100% das imagens geradas são utilizadas para testes.

Figura 18 – Na esquerda o modelo 3D modelado no Autodesk Fusion 360 e na direita o objeto impresso utilizado para construção do novo conjunto de dados.



Fonte: Elaborado pelo autor, 2018

5.3 CALIBRAÇÃO DA CÂMERA

Para calibração da câmera, foi utilizada o biblioteca Calibration and 3D Reconstruction (calib3d) do OpenCV. Através dessa biblioteca no openCV, são estimados os valores da matriz de distorção das lentes e a matriz de calibração da câmera que minimizam o erro de reprojeção dos pontos de um padrão passado através de uma imagem capturada pela câmera.

No OpenCV, é possível usar um *grid* de quadrados como um tabuleiro de xadrez ou um *grid* de círculos. Para estimar os parâmetros das câmeras nessa dissertação foi utilizado um tabuleiro de xadrez impresso em folha de papel e diversas imagens foram capturadas com as câmeras a serem calibradas, cada uma de forma independente.

Para simplificação, assumiu-se o plano do tabuleiro como $z = 0$. Após a detecção do tabuleiro as imagens são normalizadas e então convertidas para preto e branco através de um filtro de limiar adaptativo. Após isso, os pontos de cantos de cada quadrado são

encontrados (o número de cantos a ser detectado é definido previamente através de uma constante). Na biblioteca calib3d é implementada a função que consegue detectar os cantos dos quadrados no tabuleiro (cv2::FindChessboardCorners). Com os pontos do objeto (conjunto de pontos ideias para o tabuleiro) e o conjunto de corners correspondentes estimados na imagem, é possível realizar a estimação da matriz de câmera e coeficientes de distorção. Os valores cada câmera calibrada são retornados em um arquivo XML de saída posteriormente utilizados nas técnicas de rastreamento 6-DoF.

5.4 GERAÇÃO DOS DADOS REAIS

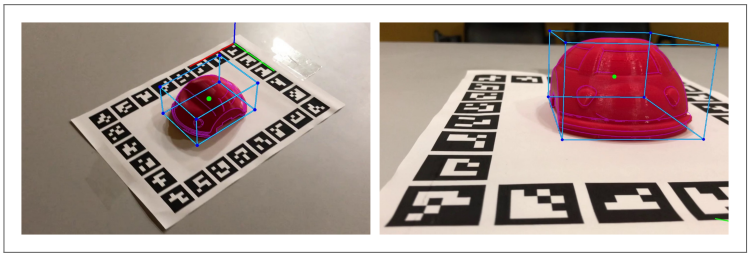
Para obter as anotações de pose de cada imagem real, foi utilizado um sistema de detecção e rastreamento de marcadores fiduciais chamado ArUco (GARRIDO-JURADO et al., 2014).No ArUco, cada marcador utilizado possui bordas pretas para facilitar o processo de detecção e contém uma matriz binária que armazena o valor de um identificador. Para estimar a posição do objeto, foi utilizado um mapa de marcadores impresso em folha A4, com identificadores compostos de 16 bits.

Para realizar a anotação das poses, o objeto de interesse é colocado no centro do mapa de marcadores, em uma posição fixa, enquanto a câmera é movimentada.

Para realizar a estimação da pose a ser utilizada, é necessário conhecer os parâmetros de calibração de câmera e os coeficientes de distorção, já obtidos através do processo de calibração descrito anteriormente.

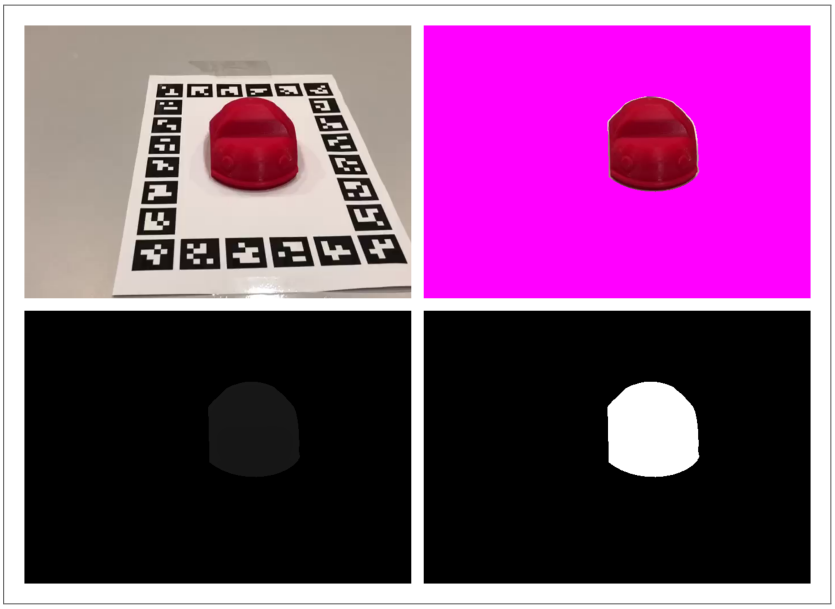
São obtidos os pontos projetados dos marcadores na imagem, e conhecendo os pontos do marcador em coordenadas 3D e os parâmetros de câmera, é possível calcular a pose utilizando a solução do algoritmo PnP. Sabendo inicialmente as posições de cada marcador e seu identificador e, sabendo a posição relativa do objeto para cada marcador, é possível estimar o movimento realizado pela câmera em relação ao objeto de interesse centrado no mapa de marcadores (Figura 19). Como saída, para cada quadro, é gerado o conjunto de pontos do *bounding box 3D* do objeto projetado na imagem, a máscara binária para o quadro, o valor do mapa de profundidade e a máscara RGB para o objeto com fundo transparente. As imagens de saída geradas podem ser vistas na Figura 20.

Figura 19 – Resultados da estimação de pose na geração de dados reais utilizando a detecção de marcadores com ArUco.



Fonte: Elaborado pelo autor, 2018

Figura 20 – Conjunto de saídas gerados para um quadro. No imagem superior esquerda a imagem RGB recebida e na direita a máscara RGB gerada para o objeto. Na parte inferior esquerda é visto o mapa de profundidade gerado pelo ArUco e na esquerda a máscara binária para o objeto alvo.



Fonte: Elaborado pelo autor, 2018

5.5 IMPLEMENTAÇÃO DA DETECÇÃO 6-DOF

Para construir o modelo para detecção 6-DoF foram utilizados os frameworks de aprendizagem de máquina **PyTorch** (PASZKE et al., 2017) e **Tensoflow** (ABADI et al., 2016).

A geração de valores de âncoras para treinamento da rede foi desenvolvida utilizando o algoritmo K-means (HARTIGAN; WONG, 1979), a partir de um conjunto de centros K que é inicializado pelo valor da largura e altura dos *bounding box 2D* normalizados para os objetos anotados na base de dados. A normalização é realizada de acordo com a Equação 5.1, onde os valores de w_k e h_k representam os valores da altura e largura do *bounding box 2D* do objeto, W_{in} e H_{in} representam a resolução da imagem de entrada da rede neural e 32 é o fator de utilizado para variação da resolução da entrada da rede durante o treinamento. Os valores do centros variam durante a execução do algoritmo até sua convergência para um valor fixo ou até alcançar o número máximo de iterações. Se o valor máximo de iterações é alcançado, o algoritmo é executado novamente, variando o valor de K até que o valor dos centros seja estabilizado para o conjunto de treinamento. Durante a avaliação, para definir o ajuste do centro ao conjunto de dados é utilizado a função de IoU como métrica de dissimilaridade para o K-means. Encontrar o valor de âncoras que melhor se ajusta ao conjunto de dados melhora a convergência do modelo durante o treinamento, diminuindo o tempo necessário para encontrar a melhor direção

de otimização dos parâmetros da arquitetura.

$$\begin{aligned} b_w &= \frac{w_k \times W_{in}}{32}, \\ b_h &= \frac{h_k \times H_{in}}{32}. \end{aligned} \tag{5.1}$$

Na fase de inicialização, os dados de entrada são passados para o *buffer* e transformados em tensores de Bytes e os valores dos pixels são normalizados para o intervalo $[0, 1]$, requisito de entrada das redes neurais. Finalmente, os dados são passados para processamento em GPU.

Durante o treinamento, a resolução da rede neural é ajustada para as dimensões [320×320, 416×416, 460×460, 544×544, 680×680]. O treinamento multi-escala permite o ajuste dos parâmetros durante o treinamento para suportar diferentes resoluções de entrada durante o uso do modelo, diferentes características para a variação de escala também são consideradas durante o ajuste do modelo.

O modelo utiliza as camadas iniciais da arquitetura Darknet.19 como inicialização dos pesos, ajustando os valores dos parâmetros nos passos seguintes do treinamento.

A função de otimização utilizada foi o SGD (do Inglês *Stochastic Gradient Descent*) (BOTTOU, 2010) com taxa de aprendizagem adaptativa de valor inicial $1e^{-3}$, decrementando a cada 300 épocas e com momento de 0.9. O valor de regularização dos pesos utilizado foi $1e^{-5}$, e o tamanho do *batch* de treinamento foi 16 (suportado pela GPU utilizada). O erro da função de custo e o passo de ajuste dos pesos é computado para cada batch. As camadas de *batch normalization* são utilizadas após cada camada convolucional e, ficam ativas apenas durante o treinamento do modelo.

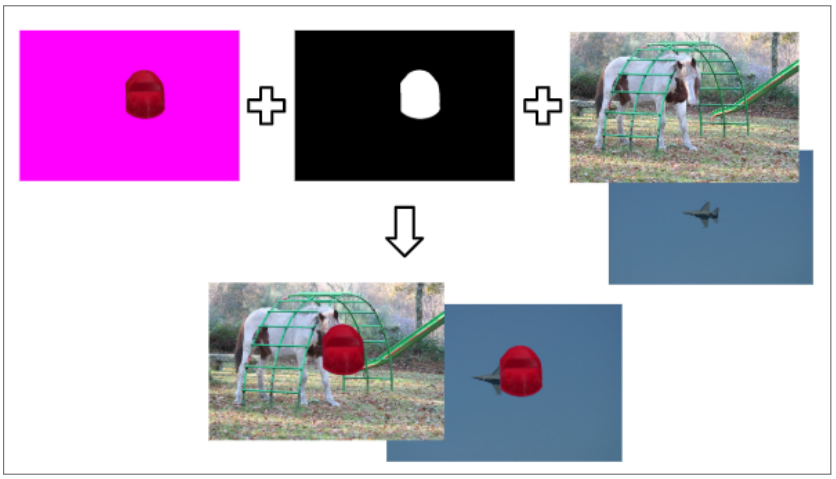
A função de custo utilizada é definida pela Equação 4.7 e utiliza a distância euclidiana para computar o erro de predição dos pontos (L_{pt}), a Equação 4.6 para a perda do valor de confiança (L_c) e *cross entropy* para o erro das probabilidades (L_p). Os coeficientes λ_c e λ_p são definidos como 1.0. O coeficiente de estimação dos pontos λ_{pt} prioriza as detecções nas regiões onde existem objetos, sendo definido para 5.0 quando um objeto é detectado e 0.1 quando não há objeto na região.

Como descrito em Tekin et al. (TEKIN; SINHA; FUA, 2017), o modelo é treinado em duas etapas, ajustando primeiro o valor de predição dos pontos de cantos ($\lambda_c = 0$) e em seguida realizando o treinamento completo com a estimação do valor de confiança ($\lambda_c = 1$). Separar o treinamento em duas etapas permite obter melhor convergência dos parâmetros, devido a estimação do valor de confiança depender da estimação dos pontos de cantos do objeto. Quando a estimação de confiança é desconsiderada na primeira etapa do treinamento, o erro total para ajuste do gradiente penaliza apenas a diferença de estimação dos pontos de cantos iniciais.

Durante o treinamento o processo de *data augmentation* é realizado para diminuir o sobreajuste do modelo em relação aos exemplos de treinamento. Utilizando a imagem

de entrada e a mascara binária do objeto na imagem é possível realizar diversas transformações nos exemplos de treinamento, como por exemplo, inserir o objeto em planos de fundos diferentes, rotacionar o objeto na imagem, mudar a escalar e mover o objeto na cena. A Figura 21 mostra um exemplo das transformações realizadas neste processo. Para realizar a transformação do plano de fundo, foram utilizadas imagens do *dataset* VOC2012 (EVERINGHAM; WINN, 2011).

Figura 21 – Exemplo do processo de *data augmentation* realizado durante o treinamento. A Figura mostra o caso de transformação do plano de fundo da imagem utilizando a mascara binária do objeto na imagem original.



Fonte: Elaborado pelo autor, 2018

5.6 GERAÇÃO DE DADOS SINTÉTICOS

Para a etapa de geração de dados sintéticos, foi utilizado o Blender, através do módulo de programação por script Python integrado à aplicação.

Para gerar as imagens sintéticas utilizando o Blender, foi necessário definir os parâmetros de sua câmera virtual. O objetivo é simular o conjunto de câmeras utilizado neste trabalho (Kinect (LINEMOD), S8, Asus, Webcam Logitech), conhecendo, a priori, os valores dos parâmetros intrínsecos f_x, f_y, u_0 e v_0 encontrados através da calibração das câmeras e os valores de resolução da imagem W e H.

Através das Equações 5.2 e 5.3 foi possível estimar a distância focal f, o tamanho do sensor s_w e s_h, a proporção da tela r e os desvios da câmera em relação aos eixos x (δ_w) e y (δ_h).

$$\begin{aligned} f_x &= \frac{f \times W}{s_w}, \\ f_y &= \frac{f \times H}{s_h}. \end{aligned} \tag{5.2}$$

$$
\begin{aligned}
u_0 &= \frac{W}{2} - \delta_w * W, \\
v_0 &= \frac{H}{2} + \delta_h * \frac{W}{r}.
\end{aligned}
\tag{5.3}
$$

No Blender o sistema de coordenadas é definido para o ambiente e câmera. Os parâmetros dos objetos de câmera podem ser modificados diretamente pela interface gráfica ou por *script*. Por *script*, foram recuperados os valores definidos e foi realizada a conversão entre as coordenadas de mundo e da câmera. Também foi realizado o mapeamento com texturas UV e para os dados sintéticos simples foram utilizadas texturas normal e difuso.

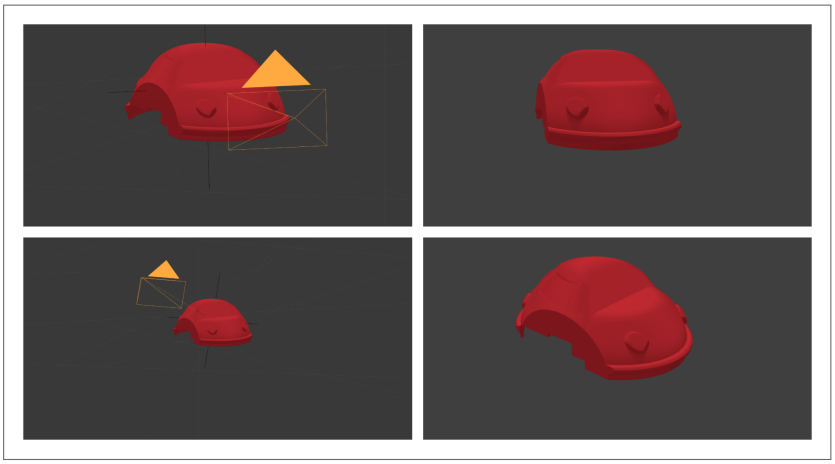
No caso da adaptação de domínio foram utilizadas texturas difusas, normal, especular e reflexiva, para simular as situações de iluminação mais próximas de casos reais. O modelo foi posicionado em ambientes fechados, com iluminação fixa e controlada. Para ajustar a renderização da iluminação e conseguir mais realismo para a cena, o gerenciamento de cor do Blender foi modificado utilizando o Filmic Blender. Tal artifício foi implementado porque as funções de transferência não lineares utilizadas no Blender não foram projetadas para renderização. A configuração com Filmic fornece uma transformação de renderização de câmera confiável e com melhor aparência.

Para o caso de randomização de domínio a textura é aplicada diretamente pelo material do modelo, que varia aleatoriamente entre os quadros. Nesse caso, há também a inserção de objetos de distração, que são formas geométricas básicas do Blender (esferas, cubos e cones). A visibilidade e posição de cada objeto são aleatórias durante a execução da animação conforme definido em (TREMBLAY et al., 2018).

O mapa de profundidade pode ser extraído diretamente pelo Blender, e pode ser utilizado para estimar o valor de profundidade da câmera para o objeto. Para gerar as imagens com diferentes pontos de vista dos objetos, a origem da câmera é alinhada a origem da cena, com uma distância variável, e a orientação da câmera é definida manualmente. Na sequência, a câmera se move ao redor do objeto, variando os valores de *yaw*, *pitch* e *roll*. Definindo o intervalo para o movimento e utilizando a ferramenta de animação do Blender é possível variar a posição e a orientação da câmera para gerar o conjunto de dados sintéticos, movendo a câmera sempre em relação a origem da cena onde o objeto está posicionado (Figura 22).

A inserção de ruído, borramento, mudança de fundo, mudança de iluminação é feita externamente ao Blender, utilizando o OpenCV como no processo de *data augmentation*. As imagens sintéticas são utilizadas **apenas** para treinamento.

Figura 22 – Exemplos do movimento de câmera para geração de dados sintéticos com Blender. As imagens da esquerda mostram a interface de visualização 3D da ferramenta com a câmera posicionada na cena, na direita são mostrados os resultados renderizados.



Fonte: Elaborado pelo autor, 2018

5.7 MÉTRICAS DE AVALIAÇÃO UTILIZADAS

A seguir são definidas as métricas utilizadas para avaliação dos modelos.

5.7.1 Erro de reprojeção

O Erro de reprojeção, E_{2D}, corresponde a distância d entre o ponto projetado x_{pr} predito pelo modelo e o ponto x_{gt} que representa o valor verdadeiro anotado no conjunto de dados. Considera-se um acerto quando, para uma imagem k de um conjunto K, o valor do erro de reprojeção médio estimado é menor ou igual a 5 pixels (Equação 5.4) (BRACHMANN et al., 2016).

$$E_{2D} = \frac{\sum_{k=0}^{K}(\Delta_{2D}(x_{gt}^{k}, x_{pr}^{k}))}{K}, \tag{5.4}$$

$$\Delta_{2D}(x1, x2) = \begin{cases} 1, & \text{se } ||x1 - x2|| \leq 5\, pixels, \\ 0, & \text{caso contrário.} \end{cases} \tag{5.5}$$

5.7.2 Erro a partir do modelo 3D

Conhecendo os pontos do modelo $x \in \mathbf{M}$ os vetores de rotação $\mathbf{r}_{gt}$ e translação $\mathbf{t}_{gt}$ anotados e, obtendo-se a estimação dos vetores de rotação e translação $\mathbf{r}_{pr}, \mathbf{t}_{pr}$.

Define-se a métrica de *6D Pose* como a média da diferença de todos os pontos do modelo transformados e os pontos correspondentes aplicados a transformação estimada

(Equação 5.6) (BRACHMANN et al., 2016; KEHL et al., 2017; RAD; LEPETIT, 2017).

$$m = avg_{x \in \mathbf{M}} ||(r_{gt}x + t_{gt}) - (r_{pr}x + t_{pr})|| \tag{5.6}$$

O modelo é considerado corretamente detectado se $m \leq k_m d$, onde k_m é uma constante definida e d representa o diâmetro do modelo $\mathbf{M}$ (HINTERSTOISSER et al., 2012). Utilizamos $k_m \in [0.1, 0.3, 0.5]$, utilizando um nível de aceitação para um erro de 10%, 30% e 50% em relação ao diâmetro do objeto.

5.7.3 Erro de translação e rotação da pose

Nessa métrica, conhecida como $5cm5dg$, é avaliada a predição da pose em relação ao erro de estimação do vetor de translação t e rotação r. Conhecendo os valores dos parâmetros anotados t_{gt} e r_{gt} para uma dada imagem k de um conjunto K. E obtendo-se a estimação t_{pr} er_{pr} do modelo para a mesma imagem. Computamos o erro de translação e erro de rotação através das equações 5.7 e 5.8, Tr representa o operador *Trace*. O valor da métrica 5cm5dg (E_{5cm5dg}) é calculado para todo conjunto K através da Equações 5.9, 5.10 e 5.11 (BRACHMANN et al., 2016; HINTERSTOISSER et al., 2012; TEKIN; SINHA; FUA, 2017).

$$e_t^k = \sqrt{(t_{gt} - t_{pr})^2} \tag{5.7}$$

$$e_r^k = \frac{\arccos \frac{Tr(r_{gt}, r_{pr}^T) - 1.}{2.} \times 180}{\pi} \tag{5.8}$$

$$E_{5cm5dg} = \frac{\sum_{k=0}^{K} (\Delta_T(e_t^k) \; and \; \Delta_R(e_r^k))}{K} \tag{5.9}$$

$$\Delta_T(x) = \begin{cases} 1, & \text{se } x \leq 5 \, cm, \\ 0, & \text{caso contrário.} \end{cases} \tag{5.10}$$

$$\Delta_R(x) = \begin{cases} 1, & \text{se } x \leq 5 \, graus, \\ 0, & \text{caso contrário.} \end{cases} \tag{5.11}$$

6 RESULTADOS E DISCUSSÕES

6.1 AVALIAÇÃO ATRAVÉS DE DATASET PÚBLICO

O primeiro passo da avaliação do método foi validar a implementação desenvolvida, realizando a reprodução dos resultados reportados pelo trabalho base (TEKIN; SINHA; FUA, 2017). Para realizar o treinamento do modelo, foi utilizado o mesmo procedimento descrito pelo trabalho.

O conjunto de imagens reais disponíveis no dataset LINEMOD é utilizado, onde 15% dos dados disponíveis são utilizados para treinamento e 85% são utilizados para os testes. Valores típicos para divisão dos dados no LINEMOD quando há a utilização de *data augmentation*, sendo o mesmo esquema aplicado por Tekin et al. (TEKIN; SINHA; FUA, 2017) e outros trabalhos analisados na revisão de literatura que utilizam o LINEMOD (KEHL et al., 2017; XIANG et al., 2017; LI et al., 2018).

O treinamento é realizado em etapas, onde primeiro foi realizado o treinamento otimizando os parâmetros para realizar a estimação dos pontos projetados, computando apenas a função de custo em função de L_{pt}. Após isso, foi feito o segundo passo do treinamento considerando o erro em relação ao valor de confiança L_c para otimização dos parâmetros.

A função que computa o valor de confiança depende dos valores estimados para os pontos projetados, como visto na Equação 4.6. Realizar o treinamento em 2 passos melhora o tempo que o modelo leva para convergir durante o treinamento, penalizando o erro de estimação em etapas separadas.

Para reduzir o sobre-ajuste durante o treinamento é utilizado o processo de aumento de dados (*data augmentation*). Para gerar o conjunto de dados de treinamento que será efetivamente utilizado pelo modelo, o objeto é extraído da cena usando uma máscara de segmentação disponível no *dataset*. Isso evita que o modelo leve em consideração características do plano de fundo no momento do treinamento. O objeto é recortado e inserido em imagens com fundos aleatórios.

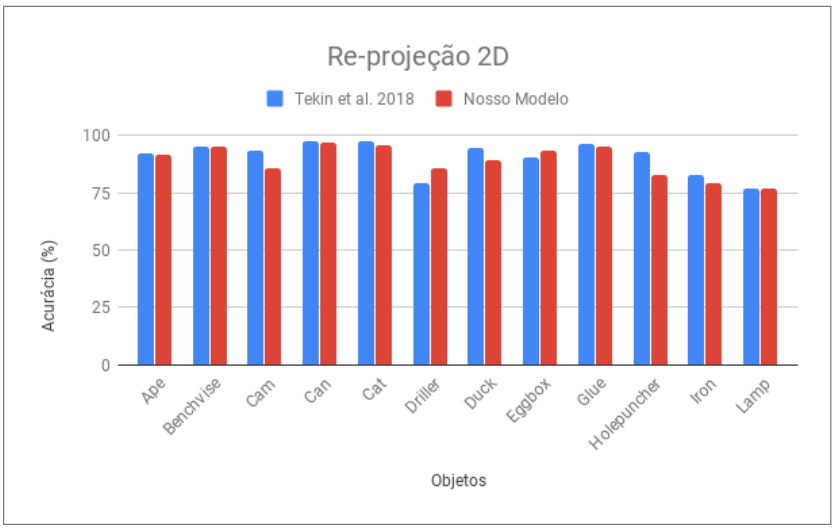
O acompanhamento do treinamento foi realizado utilizando a ferramenta de visualização do TensorFlow (Tensorboard). Verificando a convergência do modelo para cada passo do treinamento em tempo real.

Para avaliação foram computados os valores para re-projeção 2D e 6D Pose com $k_m = [0.1, 0.3, 0.5]$, representando uma tolerância de erro para $10\%, 30\%$ e 50% em relação ao diâmetro do objeto. Os limiares de k_m selecionados cobrem o nível de aceitação para diversos tipos de aplicação que utilizam a detecção 6-DoF (BRACHMANN et al., 2016). Os resultados foram calculados para cada objeto e comparado com os resultados reportados pelos autores. A comparação pode ser vista nas Figuras 23 e 24.

É possível notar que o modelo obteve resultados similares para a maior parte dos

objetos testados. Em alguns casos foram obtidos resultados levemente menores ou maiores em relação ao resultado original reportado. Considerando que, em média, os valores obtidos foram consistentes aos reportados, é possível afirmar que foi gerado um modelo equivalente ao descrito no artigo.

Figura 23 – Valor do erro de re-projeção 2D para o dataset LINEMOD.



Fonte: Elaborado pelo autor, 2018

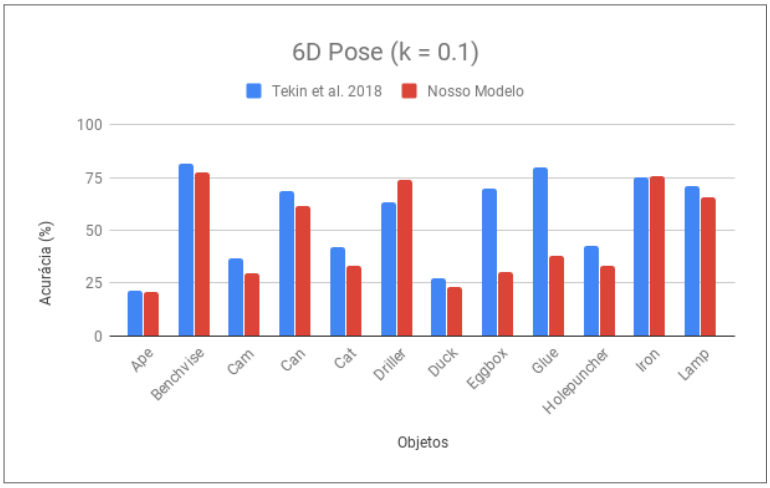
Avaliando os resultados qualitativamente, foi possível notar que o algoritmo consegue lidar com alguns casos de mudança da iluminação e borramento da imagem. As maiores fontes de erros ocorrem em casos em que há objetos similares ao alvo, principalmente em casos de cores iguais, e casos de erros da estimação da rotação para objetos simétricos. As Figuras 25 e 26 mostram exemplos dos casos para erro de detecção com objetos similares e estimação de rotação respectivamente.

O erro para objetos de cores similares acontece devido ao conjunto de características que o modelo extrai durante o treinamento, a cor do objeto é uma informação predominante em todo o conjunto de dados, e consequentemente se torna uma boa característica para identificá-lo. Porém quando aparecem objetos de cores similares, a confiança do modelo diminui para a detecção.

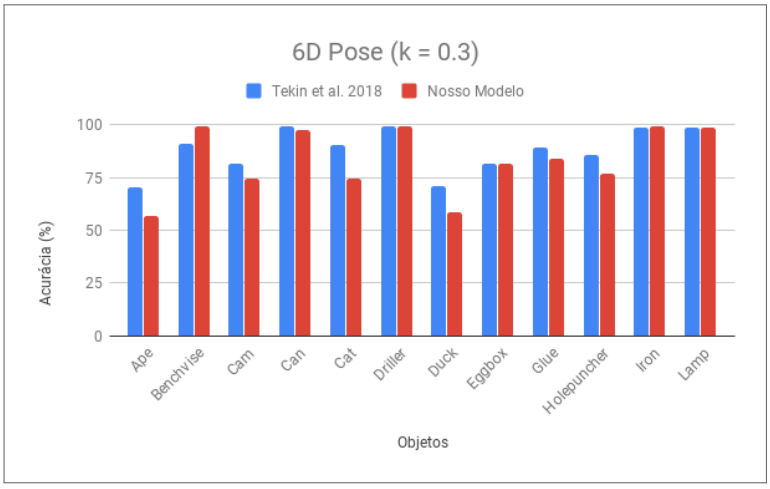
O erro na estimação da rotação para objetos simétricos, é um problema que ocorre em diversas técnicas de detecção 6-DoF, nesse caso não há um tratamento específico no modelo para lidar com a estimação do ponto de vista correto do objeto no trabalho base. Para lidar com os resultados de estimação da rotação nos casos de objeto com diferentes pontos de simetria, o autor utiliza uma variação da função que computa o acerto para a pose $6D$, considerando as diferentes estimações de rotação no eixo de simetria como uma única resposta correta. O que ocasiona a maior diferença de resultados (Figura 24a) para os casos dos objetos *Eggbox*, *Holepuncher* e *Glue* da base LINEMOD. Na medida que o limiar de aceitação é aumentado, o valor de acerto para a métrica de pose $6D$ nestes casos

se torna equivalente ao modelo original (Figura 24b e 24c).

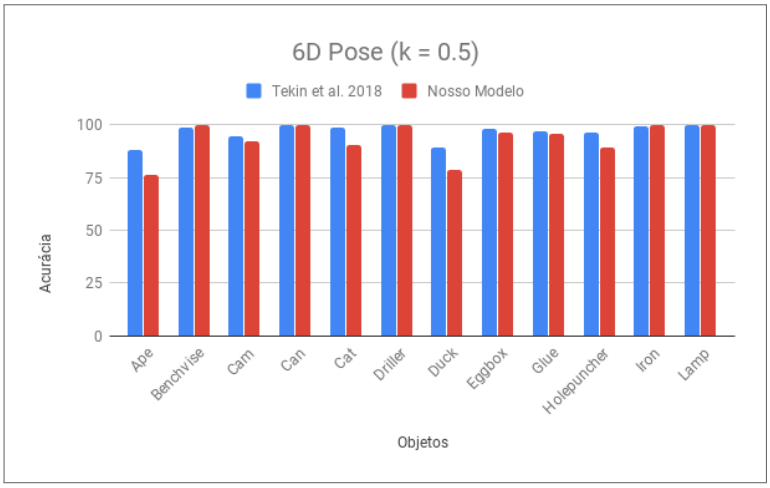
Figura 24 – Valor para 6D Pose no dataset LINEMOD.



(a) Limiar de 10% em relação ao diâmetro do objeto.



(b) Limiar de 30% em relação ao diâmetro do objeto.



(c) Limiar de 50% em relação ao diâmetro do objeto.

Fonte: Elaborado pelo autor, 2018

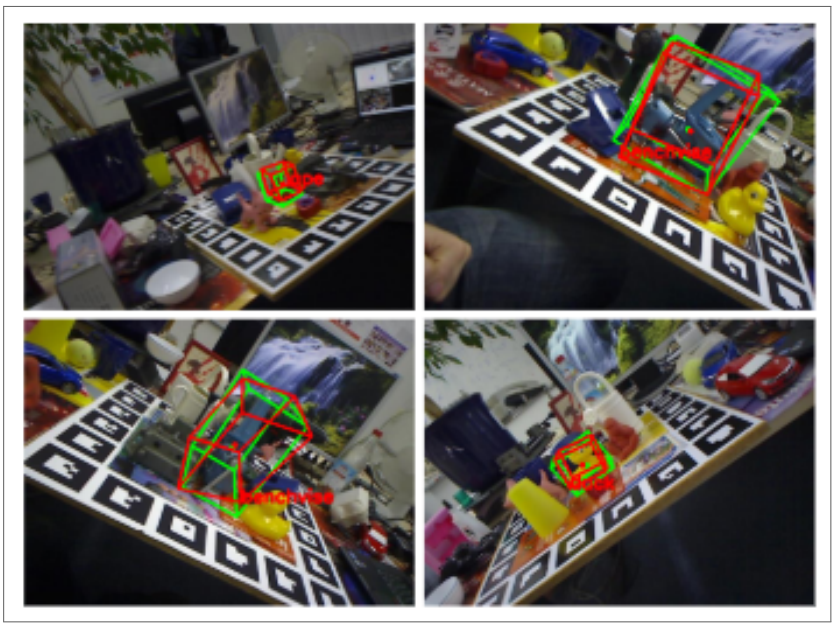
Outro problema identificado é o tremido gerado em respostas sequenciais. Por ser

um modelo que realiza o rastreamento por detecção, quando utilizado para quadros sequenciais, as respostas tendem a ser instáveis. Cada detecção realizada é independente e, quando utilizado para aproximar um rastreamento, as respostas do modelo para cada quadro possuem um desvio significativo, ocasionando na instabilidade perceptível durante a execução, fenômeno conhecido em técnicas de Realidade Aumentada como *jitter*.

Figura 25 – Casos de erros de detecção na presença de objetos similares ao objeto de interesse.

Fonte: Elaborado pelo autor, 2018

Figura 26 – Casos de erros para estimação de rotação em objetos simétricos.



Fonte: Elaborado pelo autor, 2018

6.2 AVALIAÇÃO PARA NOVOS CENÁRIOS

Para o segundo teste, foi gerado um novo conjunto de imagens. No novo conjunto, um objeto impresso em 3D foi utilizado e, foi variado o ambiente em que as imagens foram capturadas. Os parâmetros de iluminação e movimento de câmera foram semelhantes ao conjunto de treinamento, objetivando a avaliação da técnica considerando o desempenho da técnica para diferentes câmeras.

Neste cenário foi realizado o treinamento utilizando apenas os dados obtidos pela Câmera 01, e realizado testes para as Câmeras 01, 02 e 03. No conjunto de dados LINEMOD assume-se que as condições utilizadas no treinamento serão as mesmas utilizadas para teste. Com a geração de um novo conjunto de dados, o objetivo é avaliar o desempenho do modelo proposto para casos de aplicações genéricas, em que não são possuídas as informações sobre a câmera e ambiente do usuário.

Na Tabela 3 é possível ver que o modelo tem bom resultado para a Câmera 01 com dados similares ao treinamento. Quando testado para os outros conjuntos, o desempenho se torna muito baixo, conforme esperado devido às divergências entre as câmeras e o não tratamento desse fator pela técnica.

Utilizando apenas imagens de um tipo de câmera, o método é levado ao sobre-ajuste dos parâmetros relacionados a câmera. Quando utilizado uma câmera de resolução, qualidade e parâmetros específicos, o modelo não consegue generalizar a informação do treinamento para o novo conjunto de teste.

Os dados sintéticos foram gerados com a finalidade de obter a variação destes parâmetros, assumindo que não conhecemos as características dos dispositivos que serão utilizados. Queremos avaliar o método, para que seja possível obter bom desempenho para estes casos, sem a necessidade de se obter dados rotulados para treinamento de todas as câmeras avaliadas.

Tabela 3 – Resultados obtidos utilizando treinamento com dados reais obtidos pela Câmera 01.

Métrica	Câmera 01	Câmera 02	Câmera 03
6D Pose 10%	84.3	49.9	18.0
Re-projeção 2D	80.4	41.7	36.04
5cm5dg	84.81	35.8	9.61

6.3 AVALIAÇÃO PARA TREINAMENTO SINTÉTICO

Foi realizado o treinamento com imagens geradas sinteticamente, variando-se a quantidade de imagens reais misturadas ao conjunto. O objetivo é avaliar a generalização da informação utilizando um conjunto de dados sintéticos para teste do modelo e a dependência de imagens reais no aprendizado do modelo. Pela Tabela 4 é possível avaliar os

resultados. Utilizando apenas os dados sintéticos, o valor de acurácia obtido é muito baixo para as métricas utilizadas. Quando a informação do conjunto de dados reais e sintéticos são combinadas, o modelo consegue obter melhores resultados. As taxas obtidas não superam os valores do treinamento com imagens reais para a Câmera 01, mas é possível notar uma melhora nas taxas para as Câmeras 02 e 03. Quando são inseridas as informações sintéticas, variando-se os parâmetros de renderização da câmera virtual, o risco de acontecer sobre-ajuste para as características da Câmera 01 é reduzido.

Tabela 4 – Resultados obtidos utilizando treinamento com dados sintéticos e reais.

Métrica	Sintético			Sintético + real		
	Câm 01	Câm 02	Câm 03	Câm 01	Câm 02	Câm 03
6D Pose 10%	2.13	3.61	0.58	80.16	79.5	64.61
Re-projeção 2D	0.7	0.39	0.	62.9	54.5	51.04
5cm5dg	0.05	0.03	0.	51.07	52.8	50.98

O segundo teste é relativo a avaliação do treinamento através da adaptação de domínio. Gerando imagens sintéticas com técnicas de fotorrealismo, foi investigado o desempenho do modelo na transferência da informação aprendida através do conjunto sintético para o mundo real. Nesse método, é gerado um conjunto de dados sintéticos com distribuição similar ao conjunto de dados reais, utilizando esta abordagem foi investigado a possibilidade de evitar a necessidade de se dados reais para treinamento do modelo.

Pela Tabela 5 é possível ver que os resultados ficaram abaixo dos outros casos. Apesar a utilização de técnicas para aproximar o conjunto de dados sintéticos do domínio real, ainda não foi possível adaptar a informação para evitar completamente o uso de dados reais no treinamento. Contudo, é possível notar que para todo os casos, os valores obtidos em diferentes cenários se mantiveram similares. Para melhorar o resultado com esta abordagem, ainda é preciso investigar abordagens que estimem os parâmetros de renderização automaticamente, e assim desenvolver um método que não seja dependente da obtenção de dados reais rotulados, e que possa ser utilizado para estimar a pose em diversos dispositivos.

Tabela 5 – Resultados obtidos utilizando treinamento apenas com dados sintéticos, transferindo aprendizado com adaptação de domínio.

Métrica	Câmera 01	Câmera 02	Câmera 03
6D Pose 10%	54.29	52.7	31.07
Re-projeção 2D	62.55	68.29	28.12
5cm5dg	53.85	42.85	26.9

O terceiro teste realizado envolve a utilização de randomização de domínio. Foi gerado um conjunto de dados com uma grande quantidade de variação da informação. Nesse

teste, também foi avaliado o resultado para o caso em que o conjunto de dados reais é misturado ao conjunto de dados sintéticos durante o treinamento. Os resultados podem ser vistos na Tabela 6.

Foi obtido um baixo resultado utilizando apenas as imagens sintéticas com randomização de domínio. Para as diferentes câmeras o resultado foi superior ao treinamento somente utilizando imagens reais.

Quando utilizamos o conjunto de dados reais em conjunto com as imagens sintéticas, obtemos bons resultados para a Câmera 01, 02 e 03. Esse método se comportou de forma similar a abordagem de geração de dados sintéticos com *data augmentation*. As variações no conjunto de dados tornam o resultado equivalente para todas as câmeras, evitando o sobre-ajuste para as características da câmera de treinamento e dos dados sobre o objetivo apresentados no treinamento.

Tabela 6 – Resultados obtidos utilizando treinamento apenas com dados sintéticos, randomização de domínio.

Métrica	DR			DR + real		
	Câm 01	Câm 02	Câm 03	Câm 01	Câm 02	Câm 03
6D Pose 10%	4.87	5.27	2.36	86.29	66.0	55.07
Re-projeção 2D	0.15	0.68	0.2	78.8	53.85	49.98
5cm5dg	0.01	0.	0.	68.61	51.55	48.85

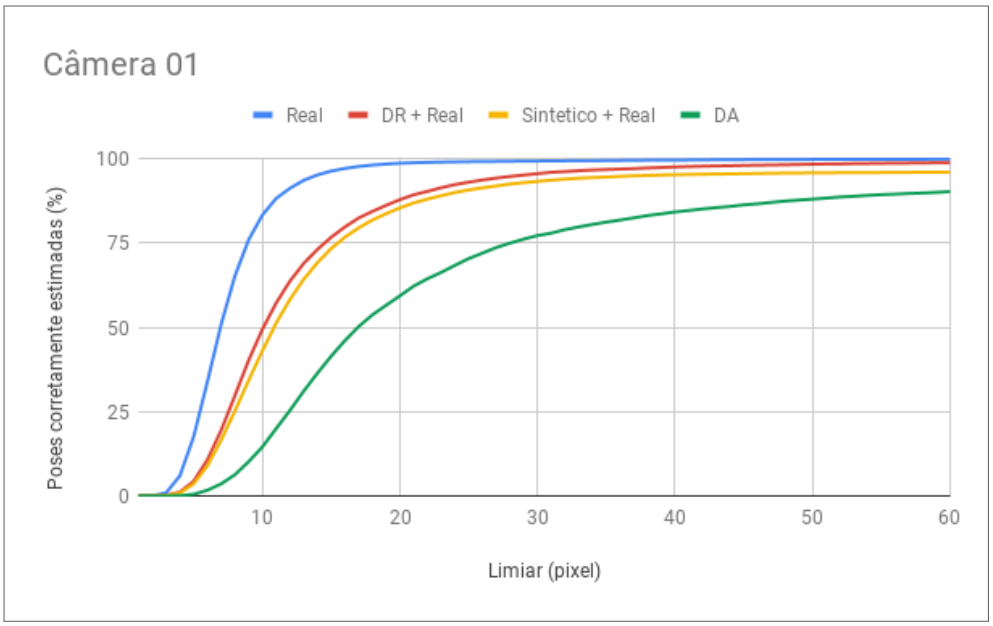
6.4 COMPARAÇÃO DOS RESULTADOS

Para avaliar o desempenho de cada abordagem em diferentes condições, foi avaliado o número de poses corretamente estimadas em função do erro de projeção, para todas as câmeras consideradas nesta dissertação. Nesta avaliação, são variadas as condições de iluminação, movimento da câmera e ambiente.

Para o teste foram considerados o treinamento com imagens reais e aumento de dados, treinamento com adaptação de domínio utilizando apenas imagens sintéticas, treinamento de imagens sintéticas com imagens reais e o treinamento de randomização de domínio com imagens reais. O treinamento real é realizado com imagens da Câmera 01, as imagens sintéticas foram geradas com as configurações das Câmeras 01, 02 e 03. Os resultados podem ser visto nas Figuras 27, 28 e 29.

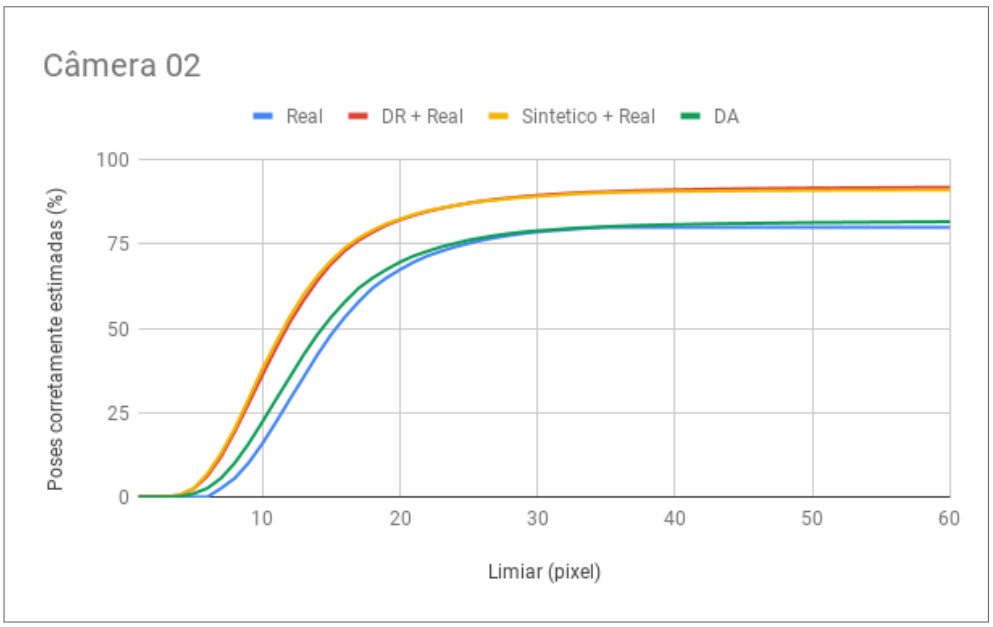
Analisando os gráficos é possível ver que o treinamento com dados reais é melhor quando a mesma câmera é utilizada para testes. Já nos casos em que o método é avaliado em outras câmeras o resultado tem uma queda em relação as outras configurações.

Figura 27 – Resultado da variação do limiar e poses corretamente classificadas em função do erro de projeção para a Câmera 01.



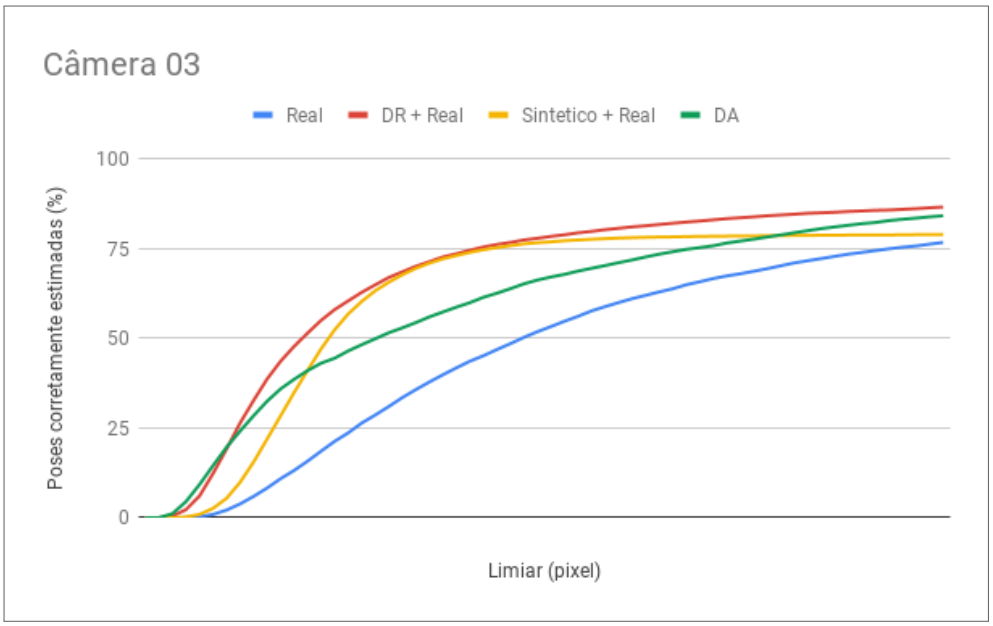
Fonte: Elaborado pelo autor, 2018

Figura 28 – Resultado da variação do limiar e poses corretamente classificadas em função do erro de projeção para a Câmera 02.



Fonte: Elaborado pelo autor, 2018

Figura 29 – Resultado da variação do limiar e poses corretamente classificadas em função do erro de projeção para a Câmera 03.



Fonte: Elaborado pelo autor, 2018

Para as imagens reais, o treinamento é realizado ajustando os parâmetros da rede para uma única configuração de câmera, fazendo com que o risco de *overfitting* seja maior.

Em relação aos conjunto de dados sintéticos, é possível notar que os valores são próximos para todas as configurações. O melhor resultado acontece para a Câmera 01, onde existem informações reais semelhantes passadas no treinamento. A adição de dados reais auxilia no aprendizado do domínio real, associando as informações do conjunto sintético aos dados reais passados no treinamento. Desta forma, é possível fazer com que a técnica realize a estimação em diferentes cenários sem a necessidade de se inserir essa informação com dados reais no treinamento.

Para a Câmera 02 o resultado do treinamento real se mantém próximo aos outros, a Câmera 02 possui semelhanças em relação a Câmera 01 como resolução e qualidade da imagem. Entretanto, a diferença entre elas é suficiente para causar um decréscimo no desempenho do modelo para o treinamento real. No cenário sintético essa diferença é amenizada, fazendo com que o resultado neste caso se mantenha similar ao anterior.

O pior resultado é obtido com a Câmera 03. A Câmera 03 é a que possui maior diferença em sua configuração, possuindo menos qualidade em relação às outras câmeras. Outro fator que piora o resultado neste cenário é a mudança de foco automática, que não pôde ser ajustada durante a captura, ocasionando na geração de diversos quadros com ruídos. Neste cenário, o treinamento real realizado com a Câmera 01 teve seu pior desempenho. As abordagens com dados sintéticos conseguem obter maior nível de generalização neste caso, tendo um desempenho melhor neste cenário com o treinamento das configurações da câmera previamente, fazendo com que o modelo precise lidar com menos situações

de variação não conhecida. Apesar disto, é possível notar que em relação as Câmeras 01 e 02 o resultado é um pouco pior, sugerindo que pode haver influência das informações reais utilizadas no treinamento, ocasionando algum *overfitting*. Contudo, a utilização de imagens reais no treino é necessária para realizar a transferência do conhecimento entre os domínios sintéticos e real.

6.4.1 Análise qualitativa de estimação

Quando utilizado para predição de múltiplos quadros no novo conjunto de dados, pequenos desvios de estimação podem ocorrer durante a inferência das poses sendo possível perceber o efeito de *jitter* na estimação das poses sequenciais, problema importante para aplicações de interação em tempo real como em Realidade Aumentada por exemplo.

Em alguns casos, o modelo obteve baixa performance devido ao desbalanceamento das informações reais de diferentes pontos de vista do objeto, ou devido a variação da luz em pontos de reflexão do objeto para diferentes quadros. Com uso de dados sintéticos, foi possível obter a diminuição da influência de reflexão do objeto e do mapeamento de pose durante o aprendizado.

Em um dos casos de teste gerado, o algoritmo conseguiu detectar o objeto utilizando apenas a iluminação de uma lanterna em movimento apontando para o objeto (Figura 30a). Para alguns casos de iluminação muito baixa o algoritmo perde o objeto, como mostra a Figura 30b.

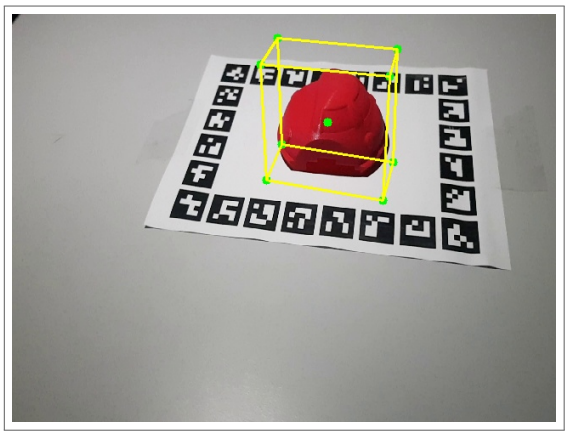
Um dos problemas ocasionados durante os testes do LINEMOD foi o erro de estimação do valor de confiança quando um objeto de características semelhantes está localizado próximo ao objeto de interesse. A maior ocorrência deste erro foi dada em relação a objetos de mesma cor na mesma cena, Figura 31a.

A cor é uma das principais características que podem ser extraídas para descrever o objeto. Quando a cor do objeto é fixa no conjunto de dados utilizado para treinamento, pode-se ocorrer um sobreajuste do modelo, gerando filtros relativos para a cor encontrada no conjunto de dados de treino. Isto pode ocasionar problemas de estimação quando utilizada outra câmera para obtenção da cena ou em casos de mudança de iluminação em que a representação de cores do objeto pode ser alterada na imagem capturada.

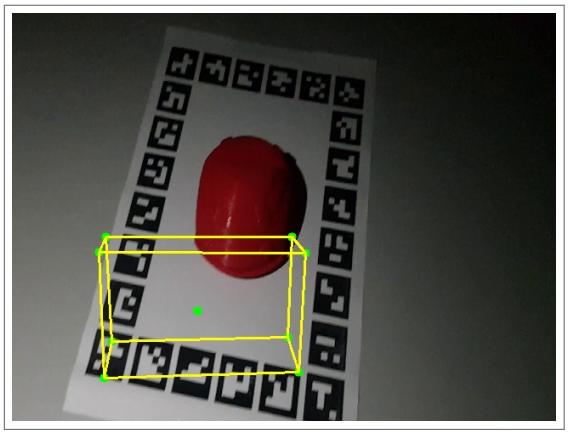
Esse cenário foi simulado em um caso de teste, inserindo um objeto de cor semelhante (lado vermelho do cubo de Rubik) próximo ao objeto alvo.

Com o processo de randomização de domínio, esse problema pôde ser minimizado. Inserindo a variação de cores e textura no objeto alvo, o modelo treinado é induzido a priorizar características mais relevantes sobre a estrutura geométrica do objeto de interesse. O resultado pode ser visto na Figura 31b com o treinamento sem utilização de imagens reais.

Figura 30 – Resultado de detecção sem iluminação ambiente, utilizando uma lanterna em movimentos aleatórios.



(a) Acerto na detecção com iluminação reduzida.



(b) Caso de falha com iluminação muito baixa.

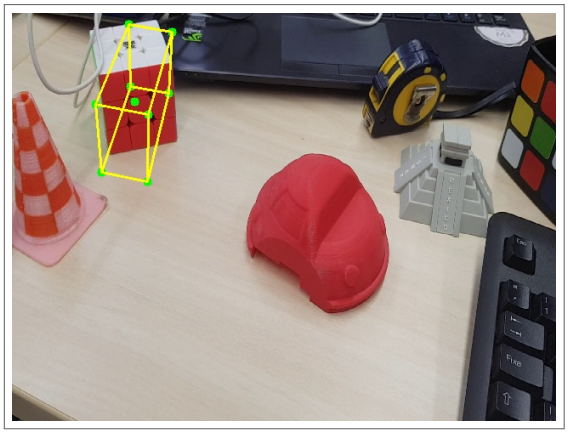
Fonte: Elaborado pelo autor, 2018

6.4.2 Treinamento

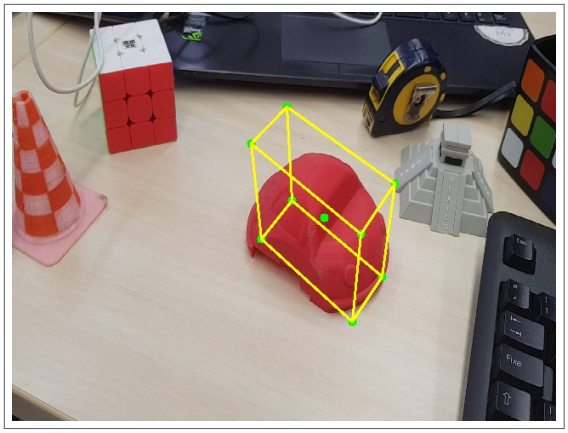
Como descrito no desenvolvimento, o treinamento do modelo deve ser realizado em dois passos. O primeiro passo para realizar a convergência da estimação dos pontos de cantos e o segundo passo para ajustar o valor de confiança baseado nestes pontos. O tempo total de treinamento, utilizando um conjunto de aproximadamente 1600 imagens por objeto (desconsiderando o aumento de dados), pode chegar a 48 horas realizando um total de 2000 épocas.

Quando o modelo é pré-treinado com o conjunto de imagens sintéticas simples, o tempo necessário para o modelo convergir pode decrescer em até 30 horas, conseguindo otimizar os parâmetros da arquitetura em aproximadamente 10 horas sem a necessidade de treinamento sem o valor de confiança. Apesar de não obter uma boa acurácia utilizando apenas as imagens sintéticas simples, o modelo pré-treinado neste conjunto pode ser re-aproveitado para os treinamentos posteriores, utilizando qualquer abordagem de geração de dados.

Figura 31 – Resultado de detecção com o erro de estimação quando objetos de características similares aparecem na cena, para as abordagens de treinamento original e por randomização de domínio.



(a) Resultado com treinamento realizado utilizando a abordagem original com imagens reais aumentadas.



(b) Resultado com treinamento realizado por randomização de domínio sem uso de imagem real.

Fonte: Elaborado pelo autor, 2018

6.4.3 Desempenho e armazenamento

Para os testes realizados com a GPU **Nvidia GTX 1080Ti**, quando utilizada uma resolução de entrada de 544×544 para a rede neural, é possível obter uma taxa de 60 fps para detecção de múltiplos quadros. Utilizando o valor de resolução de entrada em 446×446 é possível alcançar a taxa de 99 fps, contudo o valor de acurácia diminui para menores resoluções de entrada. O treinamento para otimização dos parâmetros da rede gerou um arquivo de 77 MB. Esse valor se mantém quando múltiplos objetos são utilizados, sendo uma das principais vantagens do método utilizado com relação aos métodos do estado da arte.

7 CONCLUSÕES

Neste trabalho foi estudado o problema de detecção em 6-DoF utilizando um método do estado da arte baseado em aprendizado profundo. Realizando a revisão da literatura, foram investigados os principais trabalhos do estado da arte para detecção e rastreamento 6-DoF com aprendizagem de máquina. Analisando os resultados e perspectivas de utilização, o trabalho proposto por (TEKIN; SINHA; FUA, 2017) foi escolhido como base para desenvolvimento desta dissertação, levando em consideração a robustez, acurácia e tempo de execução reportados pelo autor.

O método foi desenvolvido e validado analisando o comportamento da técnica em um conjunto de dados disponível publicamente. Os resultados reportados foram replicados e, foram investigados os pontos de robustez e falhas do modelo através da implementação desenvolvida com o objetivo de identificar limitações e pontos de melhoria no método.

Após isso, o objetivo foi avaliar o uso do método em aplicações genéricas, onde não há conhecimento sobre o ambiente e dispositivos de teste. Nos datasets e métodos encontrados, assume-se que os dados são obtidos em um cenário controlado. Por este motivo, um conjunto de dados foi gerado, inserindo diversas variações para o ambiente, iluminação e câmeras utilizadas. Foram realizados testes para o método nesta nova situação, obtendo-se um baixo valor de acurácia.

Nesta dissertação foi proposta a utilização de diferentes abordagens de treinamento, combinando a informação dos dados reais e dados sintéticos, para melhorar o desempenho do modelo.

O uso de treinamento com dados reais obteve o melhor resultado quando avaliado no conjunto de teste, onde existem características similares ao conjunto de treinamento. Para os casos de maior variação o resultado piorou, tendo em alguns casos um resultado inferior a situações de ajuste do modelo com imagens sintéticas. O treinamento apenas com imagens reais é ideal quando o modelo deve ser testado em configurações semelhantes aos dados disponíveis para treinamento, sem grandes variações de cenários.

Comparando o desempenho do modelo, foi percebido um baixo resultado quando utilizada apenas a informação sintética em todos os cenários. Apesar de não alcançar a mesma taxa para dados conhecidos, o uso de informação sintética conseguiu melhorar o resultado para características não conhecidas de outras câmeras quando utilizadas para ajustar o modelo treinado com imagens reais, evitando *overfitting* de treinamento.

O uso de apenas dados sintéticos não foi capaz de sobrepor os resultados do treinamento para os dados reais nessa pesquisa, mas o desenvolvimento destas abordagens é necessária para diminuir o custo de obtenção de dados para o desenvolvimento de técnicas de aprendizado profundo, sendo ideal para aplicações em cenários genéricos em que não são conhecidas todas as características do ambiente e dispositivos de uso. Tal tendência

é observada fortemente no estado da arte, inclusive conforme demonstrado em recente trabalho publicado (ZAKHAROV et al., 2018).

As principais limitações observadas qualitativamente foram relativas aos problemas de objetos com cores similares aos objetos de interesse e o erro de rotação para objetos com eixos simétricos, problemas que ocorrem principalmente nos casos de detecção quando o modelo é treinado apenas com dados reais. A utilização de randomização de domínio diminui a influência destes erros, inserindo diversas variações nas poses e texturas do objeto no conjunto de treinamento, fazendo com a estrutura geométrica do objeto fosse priorizada no passo de aprendizagem.

Também foram encontradas limitações em casos de iluminação muito baixa e quando ocorrem reflexos no objeto de interesse, causando confusão nas estimações realizadas pelo modelo.

7.1 CONTRIBUIÇÕES

As principais contribuições desta dissertação são resumidas na lista abaixo:

- Avaliação do método de detecção 6-DoF com aprendizagem profunda em tempo real proposto por Tekin et al.(TEKIN; SINHA; FUA, 2017).
- Proposição e avaliação de variações da técnica com datasets sintéticos para minimização de problemas como *overfitting*, capaz de lidar com diferentes câmeras e cenários reais.
- Geração de dados sintéticos para treinamento baseado em (KEHL et al., 2017) com foco na avaliação da influência do conjunto de dados sintéticos no resultado do modelo, utilização de *transfer-learning* e combinação de informação sintética e real.
- Geração e avaliação de imagens sintéticas com adaptação do domínio real, simulando casos de uso e avaliando a (in)dependência do treinamento com dados reais (ROZANTSEV; LEPETIT; FUA, 2015).
- Geração de dados sintéticos com randomização de domínio para detecção 6-DoF (TREMBLAY et al., 2018) com avaliação da influência das variações de características, cenários de uso e redução de sobre-ajuste no treinamento.
- Investigação da variação dos parâmetros de câmera na geração de dados para treinamento genérico das arquiteturas de aprendizado profundo em detecção 6-DoF, adaptando uso para utilização em técnicas de geração de dados sintéticos.

7.2 TRABALHOS FUTUROS

Para trabalhos futuros, será importante avaliar o procedimento de geração de dados, investigando formas de simplificar geração de dados sintéticos e melhorar o processo de geração com o intuito de diminuir o número de falhas. Também serão investigadas novas formas de obtenção dos parâmetros de renderização, utilizando arquiteturas de aprendizado para a geração de dados de forma automática.

Através dos resultados obtidos é possível notar que a utilização de dados sintéticos no treinamento pode auxiliar na diminuição da dependência de dados reais, otimizando o custo de desenvolvimento dos modelos de aprendizado profundo e melhorando o resultado para diferentes casos de uso. Para entender como melhorar este processo de geração de dados sintéticos, serão realizadas futuras investigações buscando encontrar e avaliar as características mais relevantes no processo de renderização, removendo a necessidade de se inserir um conjunto de imagens reais no treinamento para transferir a informação entre o domínio real e sintético, como mostra Zakharov et al. (ZAKHAROV et al., 2018) utilizando imagens de profundidade geradas sinteticamente para estimar a pose dos objetos em cenários reais através de um gerador baseado em arquitetura *adversarial*.

Em relação a randomização de domínio, serão testadas outras formas de distração e variação das informações utilizadas, buscando o entendimento do comportamento do modelo e da representação das características aprendidas em diferentes cenários. Um estudo aprofundado deve ser realizado quanto ao aprendizado dos parâmetros envolvidos no treinamento e como o modelo representa essas variações, de forma a otimizar o processo de aprendizado generalizando a informação para diversos cenários evitando o sobre-ajuste das informações reais utilizadas.

Para realizar o rastreamento temporal, serão estudadas formas de adaptação da arquitetura desenvolvida, com o objetivo de melhorar a estabilidade do modelo, reduzindo *Jitter* na resposta. O Jitter é um grande problema em aplicações que utilizam a informação do rastreamento temporal, como por exemplo, aplicações de realidade aumentada que tentam entregar a melhor experiência de interação, levando o usuário a percepção de que os objetos virtuais sejam percebidos como partes naturais do ambiente.

Abordagens que buscam utilizar aprendizado profundo para o rastreamento poderão ser investigadas, como a utilização de CNN 3D (SU et al., 2015), arquiteturas recorrentes (NING et al., 2017), e dois métodos já descritos anteriormente (GARON; LALONDE, 2017; TAN; NAVAB; TOMBARI, 2017). Será avaliado como adaptar estes modelos para uso do rastreamento em 6-DoF utilizando apenas imagens RGB e mantendo o desempenho em tempo real, necessário para aplicações de realidade aumentada.

REFERÊNCIAS

ABADI, M.; BARHAM, P.; CHEN, J.; CHEN, Z.; DAVIS, A.; DEAN, J.; DEVIN, M.; GHEMAWAT, S.; IRVING, G.; ISARD, M. et al. Tensorflow: a system for large-scale machine learning. In: *OSDI*. [S.l.: s.n.], 2016. v. 16, p. 265–283.

ABDEL-HAKIM, A. E.; FARAG, A. A. CSIFT: A SIFT descriptor with color invariant characteristics. In: IEEE. *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*. [S.l.], 2006. v. 2, p. 1978–1983.

AKKALADEVI, S.; ANKERL, M.; HEINDL, C.; PICHLER, A. Tracking multiple rigid symmetric and non-symmetric objects in real-time using depth data. In: IEEE. *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. [S.l.], 2016. p. 5644–5649.

AMMIRATO, P.; POIRSON, P.; PARK, E.; KOŠECKÁ, J.; BERG, A. C. A dataset for developing and benchmarking active vision. In: IEEE. *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. [S.l.], 2017. p. 1378–1385.

ANG, M. H.; TOURASSIS, V. D. Singularities of Euler and roll-pitch-yaw representations. *IEEE Transactions on Aerospace and Electronic Systems*, IEEE, n. 3, p. 317–324, 1987.

BALNTAS, V.; DOUMANOGLOU, A.; SAHIN, C.; SOCK, J.; KOUSKOURIDAS, R.; KIM, T.-K. Pose Guided RGBD Feature Learning for 3D Object Pose Estimation. In: *Proceedings of the IEEE International Conference on Computer Vision*. [S.l.: s.n.], 2017. p. 3856–3864.

BORREGO, J.; DEHBAN, A.; FIGUEIREDO, R.; MORENO, P.; BERNARDINO, A.; SANTOS-VICTOR, J. Applying domain randomization to synthetic data for object category detection. *arXiv preprint arXiv:1807.09834*, 2018.

BOTTOU, L. Large-scale machine learning with stochastic gradient descent. In: *Proceedings of COMPSTAT'2010*. [S.l.]: Springer, 2010. p. 177–186.

BRACHMANN, E.; MICHEL, F.; KRULL, A.; YANG, M. Y.; GUMHOLD, S. et al. Uncertainty-driven 6D pose estimation of objects and scenes from a single RGB image. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2016. p. 3364–3372.

CALLI, B.; SINGH, A.; BRUCE, J.; WALSMAN, A.; KONOLIGE, K.; SRINIVASA, S.; ABBEEL, P.; DOLLAR, A. M. Yale-CMU-Berkeley dataset for robotic manipulation research. *The International Journal of Robotics Research*, SAGE Publications Sage UK: London, England, v. 36, n. 3, p. 261–268, 2017.

CALLI, B.; SINGH, A.; WALSMAN, A.; SRINIVASA, S.; ABBEEL, P.; DOLLAR, A. M. The YCB object and model set: Towards common benchmarks for manipulation research. In: IEEE. *Advanced Robotics (ICAR), 2015 International Conference on*. [S.l.], 2015. p. 510–517.

CALLI, B.; WALSMAN, A.; SINGH, A.; SRINIVASA, S.; ABBEEL, P.; DOLLAR, A. M. Benchmarking in manipulation research: The YCB object and model set and benchmarking protocols. *arXiv preprint arXiv:1502.03143*, 2015.

CHAUDHARY, K.; CHEN, X.; CHAN, W. P.; OKADA, K.; INABA, M. STAIR3D: Simultaneous tracking and incremental registration for modeling 3D handheld objects. In: IEEE. *Advanced Intelligent Mechatronics (AIM), 2017 IEEE International Conference on*. [S.l.], 2017. p. 185–192.

CHOI, C.; CHRISTENSEN, H. I. RGB-D object tracking: A particle filter approach on GPU. In: *IROS*. [S.l.: s.n.], 2013. v. 2, n. 6, p. 7.

CRIVELLARO, A.; RAD, M.; VERDIE, Y.; YI, K. M.; FUA, P.; LEPETIT, V. A novel representation of parts for accurate 3D object detection and tracking in monocular images. In: *Proceedings of the IEEE International Conference on Computer Vision*. [S.l.: s.n.], 2015. p. 4391–4399.

CSURKA, G. Domain adaptation for visual applications: A comprehensive survey. *arXiv preprint arXiv:1702.05374*, 2017.

DENG, J.; DONG, W.; SOCHER, R.; LI, L.-J.; LI, K.; FEI-FEI, L. Imagenet: A large-scale hierarchical image database. In: IEEE. *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. [S.l.], 2009. p. 248–255.

DIEBEL, J. Representing attitude: Euler angles, unit quaternions, and rotation vectors. *Matrix*, v. 58, n. 15-16, p. 1–35, 2006.

DONAHUE, J.; JIA, Y.; VINYALS, O.; HOFFMAN, J.; ZHANG, N.; TZENG, E.; DARRELL, T. Decaf: A deep convolutional activation feature for generic visual recognition. In: *International conference on machine learning*. [S.l.: s.n.], 2014. p. 647–655.

DOSOVITSKIY, A.; FISCHER, P.; ILG, E.; HAUSSER, P.; HAZIRBAS, C.; GOLKOV, V.; SMAGT, P. V. D.; CREMERS, D.; BROX, T. Flownet: Learning optical flow with convolutional networks. In: *Proceedings of the IEEE International Conference on Computer Vision*. [S.l.: s.n.], 2015. p. 2758–2766.

DOUMANOGLOU, A.; BALNTAS, V.; KOUSKOURIDAS, R.; KIM, T.-K. Siamese regression networks with efficient mid-level feature extraction for 3D object pose estimation. *arXiv preprint arXiv:1607.02257*, 2016.

DOUMANOGLOU, A.; KOUSKOURIDAS, R.; MALASSIOTIS, S.; KIM, T.-K. Recovering 6D object pose and predicting next-best-view in the crowd. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2016. p. 3583–3592.

EVERINGHAM, M.; GOOL, L. V.; WILLIAMS, C. K.; WINN, J.; ZISSERMAN, A. The pascal visual object classes (VOC) challenge. *International journal of computer vision*, Springer, v. 88, n. 2, p. 303–338, 2010.

EVERINGHAM, M.; WINN, J. The PASCAL visual object classes challenge 2012 (VOC2012) development kit. *Pattern Analysis, Statistical Modelling and Computational Learning, Tech. Rep*, 2011.

GAO, G.; LAURI, M.; ZHANG, J.; FRINTROP, S. Occlusion Resistant Object Rotation Regression from Point Cloud Segments. *arXiv preprint arXiv:1808.05498*, 2018.

GARON, M.; LALONDE, J.-F. Deep 6-DOF tracking. *IEEE transactions on visualization and computer graphics*, IEEE, v. 23, n. 11, p. 2410–2418, 2017.

GARRIDO-JURADO, S.; MUÑOZ-SALINAS, R.; MADRID-CUEVAS, F. J.; MARÍN-JIMÉNEZ, M. J. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, Elsevier, v. 47, n. 6, p. 2280–2292, 2014.

GOODMAN, L. A. Snowball sampling. *The annals of mathematical statistics*, JSTOR, p. 148–170, 1961.

HARTIGAN, J. A.; WONG, M. A. Algorithm AS 136: A K-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, JSTOR, v. 28, n. 1, p. 100–108, 1979.

HERNANDEZ, C.; BHARATHEESHA, M.; KO, W.; GAISER, H.; TAN, J.; DEURZEN, K. van; VRIES, M. de; MIL, B. V.; EGMOND, J. van; BURGER, R. et al. Team delft's robot winner of the amazon picking challenge 2016. In: SPRINGER. *Robot World Cup*. [S.l.], 2016. p. 613–624.

HINTERSTOISSER, S.; LEPETIT, V.; ILIC, S.; HOLZER, S.; BRADSKI, G.; KONOLIGE, K.; NAVAB, N. Model based training, detection and pose estimation of texture-less 3D objects in heavily cluttered scenes. In: SPRINGER. *Asian conference on computer vision*. [S.l.], 2012. p. 548–562.

HINTERSTOISSER, S.; LEPETIT, V.; RAJKUMAR, N.; KONOLIGE, K. Going further with point pair features. In: SPRINGER. *European Conference on Computer Vision*. [S.l.], 2016. p. 834–848.

HINTERSTOISSER, S.; LEPETIT, V.; WOHLHART, P.; KONOLIGE, K. On Pre-Trained Image Features and Synthetic Images for Deep Learning. *arXiv preprint arXiv:1710.10710*, 2017.

HODAN, T.; HALUZA, P.; OBDRŽÁLEK, Š.; MATAS, J.; LOURAKIS, M.; ZABULIS, X. T-LESS: An RGB-D dataset for 6D pose estimation of texture-less objects. In: IEEE. *Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on*. [S.l.], 2017. p. 880–888.

HOFFMAN, W. C. The Lie algebra of visual perception. *Journal of mathematical Psychology*, Elsevier, v. 3, n. 1, p. 65–98, 1966.

JAFARI, O. H.; MUSTIKOVELA, S. K.; PERTSCH, K.; BRACHMANN, E.; ROTHER, C. iPose: instance-aware 6D pose estimation of partly occluded objects. *CoRR abs/1712.01924*, 2017.

JATESIKTAT, P.; FOO, M. J.; LIM, G. M.; ANG, W. T. SDF-Net: Real-Time Rigid Object Tracking Using a Deep Signed Distance Network. In: SPRINGER. *International Conference on Computational Science*. [S.l.], 2018. p. 28–42.

KAO, Y.; LI, W.; WANG, Z.; ZOU, D.; HE, R.; WANG, Q.; AHN, M.; HONG, S. et al. An Appearance-and-Structure Fusion Network for Object Viewpoint Estimation. In: *IJCAI*. [S.l.: s.n.], 2018. p. 4929–4935.

KEHL, W.; MANHARDT, F.; TOMBARI, F.; ILIC, S.; NAVAB, N. SSD-6D: Making RGB-based 3D detection and 6D pose estimation great again. In: *Proceedings of the International Conference on Computer Vision (ICCV 2017), Venice, Italy*. [S.l.: s.n.], 2017. p. 22–29.

KEHL, W.; MILLETARI, F.; TOMBARI, F.; ILIC, S.; NAVAB, N. Deep learning of local RGB-D patches for 3D object detection and 6D pose estimation. In: SPRINGER. *European Conference on Computer Vision*. [S.l.], 2016. p. 205–220.

KEHL, W.; TOMBARI, F.; ILIC, S.; NAVAB, N. Real-time 3D model tracking in color and depth on a single CPU core. In: *IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2017. p. 465–473.

KUIPERS, J. B. et al. *Quaternions and rotation sequences*. [S.l.]: Princeton university press Princeton, 1999. v. 66.

LEAL-TAIXÉ, L.; MILAN, A.; REID, I.; ROTH, S.; SCHINDLER, K. MOTchallenge 2015: Towards a benchmark for multi-target tracking. *arXiv preprint arXiv:1504.01942*, 2015.

LEE, W.; PARK, N.; WOO, W. Depth-assisted real-time 3D object detection for augmented reality. *ICAT'11*, v. 2, p. 126–132, 2011.

LEPETIT, V.; FUA, P. et al. Monocular model-based 3D tracking of rigid objects: A survey. *Foundations and Trends® in Computer Graphics and Vision*, Now Publishers, Inc., v. 1, n. 1, p. 1–89, 2005.

LEPETIT, V.; MORENO-NOGUER, F.; FUA, P. EPnP: An accurate O(n) solution to the PnP problem. *International journal of computer vision*, Springer, v. 81, n. 2, p. 155, 2009.

LEUTENEGGER, S.; CHLI, M.; SIEGWART, R. Y. BRISK: Binary robust invariant scalable keypoints. In: IEEE. *Computer Vision (ICCV), 2011 IEEE International Conference on*. [S.l.], 2011. p. 2548–2555.

LI, C.; BAI, J.; HAGER, G. D. A Unified Framework for Multi-View Multi-Class Object Pose Estimation. *arXiv preprint arXiv:1803.08103*, 2018.

LI, C. H.; LEE, C. Minimum cross entropy thresholding. *Pattern recognition*, Elsevier, v. 26, n. 4, p. 617–625, 1993.

LI, Y.; WANG, G.; JI, X.; XIANG, Y.; FOX, D. DeepIM: Deep Iterative Matching for 6D Pose Estimation. *arXiv preprint arXiv:1804.00175*, 2018.

LIMA, J. P.; SIMÕES, F.; FIGUEIREDO, L.; KELNER, J. Model based markerless 3D tracking applied to augmented reality. *Journal on 3D Interactive Systems*, v. 1, 2010.

LINDER, T.; BREUERS, S.; LEIBE, B.; ARRAS, K. O. On multi-modal people tracking from mobile platforms in very crowded and dynamic environments. In: IEEE. *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. [S.l.], 2016. p. 5512–5519.

LIU, W.; ANGUELOV, D.; ERHAN, D.; SZEGEDY, C.; REED, S.; FU, C.-Y.; BERG, A. C. SSD: Single shot multibox detector. In: SPRINGER. *European conference on computer vision*. [S.l.], 2016. p. 21–37.

LOWE, D. G. Object recognition from local scale-invariant features. In: IEEE. *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*. [S.l.], 1999. v. 2, p. 1150–1157.

NING, G.; ZHANG, Z.; HUANG, C.; REN, X.; WANG, H.; CAI, C.; HE, Z. Spatially supervised recurrent convolutional neural networks for visual object tracking. In: IEEE. *Circuits and Systems (ISCAS), 2017 IEEE International Symposium on*. [S.l.], 2017. p. 1–4.

ONDRÚŠKA, P.; KOHLI, P.; IZADI, S. Mobilefusion: Real-time volumetric surface reconstruction and dense tracking on mobile phones. *IEEE transactions on visualization and computer graphics*, IEEE, v. 21, n. 11, p. 1251–1258, 2015.

PAN, S. J.; YANG, Q. et al. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, Institute of Electrical and Electronics Engineers, Inc., 345 E. 47 th St. NY, v. 22, n. 10, p. 1345–1359, 2010.

PASZKE, A.; GROSS, S.; CHINTALA, S.; CHANAN, G. *PyTorch*. 2017.

PATEL, V. M.; GOPALAN, R.; LI, R.; CHELLAPPA, R. Visual domain adaptation: A survey of recent advances. *IEEE signal processing magazine*, IEEE, v. 32, n. 3, p. 53–69, 2015.

PAUWELS, K.; KRAGIC, D. Simtrack: A simulation-based framework for scalable real-time object pose detection and tracking. In: IEEE. *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. [S.l.], 2015. p. 1300–1307.

PAVLAKOS, G.; ZHOU, X.; CHAN, A.; DERPANIS, K. G.; DANIILIDIS, K. 6-DoF object pose from semantic keypoints. In: IEEE. *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. [S.l.], 2017. p. 2011–2018.

PRESSIGOUT, M.; MARCHAND, E. Real-time 3D model-based tracking: Combining edge and texture information. In: IEEE. *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*. [S.l.], 2006. p. 2726–2731.

PRISACARIU, V. A.; KÄHLER, O.; MURRAY, D. W.; REID, I. D. Real-time 3D tracking and reconstruction on mobile phones. *IEEE transactions on visualization and computer graphics*, IEEE, v. 21, n. 5, p. 557–570, 2015.

PRISACARIU, V. A.; REID, I. D. PWP3D: Real-time segmentation and tracking of 3D objects. *International journal of computer vision*, Springer, v. 98, n. 3, p. 335–354, 2012.

RAD, M.; LEPETIT, V. BB8: A scalable, accurate, robust to partial occlusion method for predicting the 3D poses of challenging objects without using depth. In: *International Conference on Computer Vision*. [S.l.: s.n.], 2017. v. 1, n. 4, p. 5.

RAD, M.; ROTH, P. M.; LEPETIT, V. ALCN: Adaptive Local Contrast Normalization for Robust Object Detection and 3D Pose Estimation. In: *British Machine Vision Conference*. [S.l.: s.n.], 2017.

RAMBACH, J.; DENG, C.; PAGANI, A.; STRICKER, D. Learning 6-DoF Object Poses from Synthetic Single Channel Image. In: *Proceedings of the 17th IEEE International Symposium on Mixed and Augmented Reality*. [S.l.: s.n.], 2018.

REDMON, J.; DIVVALA, S.; GIRSHICK, R.; FARHADI, A. You only look once: Unified, real-time object detection. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2016. p. 779–788.

REDMON, J.; FARHADI, A. YOLO9000: better, faster, stronger. *arXiv preprint*, 2017.

RENNIE, C.; SHOME, R.; BEKRIS, K. E.; SOUZA, A. F. D. A dataset for improved RGBD-based object detection and pose estimation for warehouse pick-and-place. *IEEE Robotics and Automation Letters*, IEEE, v. 1, n. 2, p. 1179–1185, 2016.

ROZANTSEV, A.; LEPETIT, V.; FUA, P. On rendering synthetic images for training an object detector. *Computer Vision and Image Understanding*, Elsevier, v. 137, p. 24–37, 2015.

ROZANTSEV, A.; SALZMANN, M.; FUA, P. Beyond sharing weights for deep domain adaptation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, IEEE, 2018.

SAHIN, C.; KIM, T.-K. Category-level 6D Object Pose Recovery in Depth Images. *arXiv preprint arXiv:1808.00255*, 2018.

SAHIN, C.; KOUSKOURIDAS, R.; KIM, T.-K. Iterative Hough Forest with histogram of control points for 6-DoF object registration from depth images. In: IEEE. *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. [S.l.], 2016. p. 4113–4118.

SANDY, T.; BUCHLI, J. Object-Based Visual-Inertial Tracking for Additive Fabrication. *IEEE Robotics and Automation Letters*, IEEE, v. 3, n. 3, p. 1370–1377, 2018.

SCHÖNING, J.; HEIDEMANN, G. Taxonomy of 3D Sensors-A Survey of State-of-the-Art Consumer 3D-Reconstruction Sensors and their Field of Applications. In: *VISIGRAPP (3: VISAPP)*. [S.l.: s.n.], 2016. p. 194–199.

SCHULMAN, J.; LEE, A.; HO, J.; ABBEEL, P. Tracking deformable objects with point clouds. In: IEEE. *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. [S.l.], 2013. p. 1130–1137.

SCHWARZ, M.; MILAN, A.; LENZ, C.; MUNOZ, A.; PERIYASAMY, A. S.; SCHREIBER, M.; SCHÜLLER, S.; BEHNKE, S. NimbRo Picking: Versatile part handling for warehouse automation. In: IEEE. *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. [S.l.], 2017. p. 3032–3039.

SCHWARZ, M.; MILAN, A.; PERIYASAMY, A. S.; BEHNKE, S. RGB-D object detection and semantic segmentation for autonomous manipulation in clutter. *The International Journal of Robotics Research*, SAGE Publications Sage UK: London, England, v. 37, n. 4-5, p. 437–451, 2018.

SEO, B.-K.; PARK, H.; PARK, J.-I.; HINTERSTOISSER, S.; ILIC, S. Optimal local searching for fast and robust textureless 3D object tracking in highly cluttered backgrounds. *IEEE transactions on visualization and computer graphics*, IEEE, v. 20, n. 1, p. 99–110, 2014.

SEO, B.-K.; PARK, J.; PARK, H.; PARK, J.-I. Real-time visual tracking of less textured three-dimensional objects on mobile platforms. *Optical Engineering*, International Society for Optics and Photonics, v. 51, n. 12, p. 127202, 2013.

SOCK, J.; KIM, K. I.; SAHIN, C.; KIM, T.-K. Multi-Task Deep Networks for Depth-Based 6D Object Pose and Joint Registration in Crowd Scenarios. *arXiv preprint arXiv:1806.03891*, 2018.

STURM, P. Pinhole camera model. In: *Computer Vision*. [S.l.]: Springer, 2014. p. 610–613.

SU, H.; QI, C. R.; LI, Y.; GUIBAS, L. J. Render for CNN: Viewpoint estimation in images using cnns trained with rendered 3D model views. In: *Proceedings of the IEEE International Conference on Computer Vision*. [S.l.: s.n.], 2015. p. 2686–2694.

TAN, D. J.; NAVAB, N.; TOMBARI, F. Looking beyond the simple scenarios: combining learners and optimizers in 3D temporal tracking. *IEEE Transactions on Visualization & Computer Graphics*, IEEE, n. 1, p. 1–1, 2017.

TEJANI, A.; KOUSKOURIDAS, R.; DOUMANOGLOU, A.; TANG, D.; KIM, T.-K. Latent-Class Hough Forests for 6-DoF Object Pose Estimation. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, v. 40, n. 1, p. 119–132, 2018.

TEJANI, A.; TANG, D.; KOUSKOURIDAS, R.; KIM, T.-K. Latent-class Hough Forests for 3D object detection and pose estimation. In: SPRINGER. *European Conference on Computer Vision*. [S.l.], 2014. p. 462–477.

TEKIN, B.; SINHA, S. N.; FUA, P. Real-Time Seamless Single Shot 6D Object Pose Prediction. *arXiv preprint arXiv:1711.08848*, 2017.

TJADEN, H.; SCHWANECKE, U.; SCHÖMER, E. Real-time monocular pose estimation of 3D objects using temporally consistent local color histograms. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2017. p. 124–132.

TOBIN, J.; FONG, R.; RAY, A.; SCHNEIDER, J.; ZAREMBA, W.; ABBEEL, P. Domain randomization for transferring deep neural networks from simulation to the real world. In: IEEE. *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*. [S.l.], 2017. p. 23–30.

TREMBLAY, J.; PRAKASH, A.; ACUNA, D.; BROPHY, M.; JAMPANI, V.; ANIL, C.; TO, T.; CAMERACCI, E.; BOOCHOON, S.; BIRCHFIELD, S. Training Deep Networks with Synthetic Data: Bridging the Reality Gap by Domain Randomization. In: . [S.l.: s.n.], 2018. p. 1082–10828.

TRINH, S.; SPINDLER, F.; MARCHAND, E.; CHAUMETTE, F. A modular framework for model-based visual tracking using edge, texture and depth features. In: *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS'18*. [S.l.: s.n.], 2018.

WAGNER, D.; SCHMALSTIEG, D.; BISCHOF, H. Multiple target detection and tracking with guaranteed framerates on mobile phones. IEEE, 2009.

WANG, B.; ZHONG, F.; QIN, X. Pose optimization in edge distance field for textureless 3D object tracking. In: ACM. *Proceedings of the Computer Graphics International Conference*. [S.l.], 2017. p. 32.

WANG, G.; WANG, B.; ZHONG, F.; QIN, X.; CHEN, B. Global optimal searching for textureless 3D object tracking. *The Visual Computer*, Springer, v. 31, n. 6-8, p. 979–988, 2015.

WARD, D.; MOGHADAM, P.; HUDSON, N. Deep leaf segmentation using synthetic data. *arXiv preprint arXiv:1807.10931*, 2018.

XIANG, Y.; KIM, W.; CHEN, W.; JI, J.; CHOY, C.; SU, H.; MOTTAGHI, R.; GUIBAS, L.; SAVARESE, S. Objectnet3D: A large scale database for 3D object recognition. In: SPRINGER. *European Conference on Computer Vision*. [S.l.], 2016. p. 160–176.

XIANG, Y.; SCHMIDT, T.; NARAYANAN, V.; FOX, D. Posecnn: A convolutional neural network for 6D object pose estimation in cluttered scenes. *arXiv preprint arXiv:1711.00199*, 2017.

YAO, A.; GALL, J.; GOOL, L. V. A Hough transform-based voting framework for action recognition. In: IEEE. *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. [S.l.], 2010. p. 2061–2068.

ZAKHAROV, S.; PLANCHE, B.; WU, Z.; HUTTER, A.; KOSCH, H.; ILIC, S. Keep it Unreal: Bridging the Realism Gap for 2.5D Recognition with Geometry Priors Only. *arXiv preprint arXiv:1804.09113*, 2018.

ZHANG, H.; CAO, Q. Combined Holistic and Local Patches for Recovering 6D Object Pose. In: *Proceedings of the IEEE International Conference on Computer Vision*. [S.l.: s.n.], 2017. p. 2219–2227.

APÊNDICE A – LISTA FINAL DOS TRABALHOS SELECIONADOS

- Real-time 3D Model-Based Tracking: Combining Edge and Texture Information (PRESSIGOUT; MARCHAND, 2006);
- Multiple target detection and tracking with guaranteed framerates on mobile phones (WAGNER; SCHMALSTIEG; BISCHOF, 2009);
- Depth-assisted Real-time 3D Object Detection for Augmented Reality (LEE; PARK; WOO, 2011);
- PWP3D: Real-time segmentation and tracking of 3D objects (PRISACARIU; REID, 2012);
- Real-time visual tracking of less textured three-dimensional objects on mobile platforms (SEO et al., 2013);
- RGB-D Object Tracking: A Particle Filter Approach on GPU (CHOI; CHRISTENSEN, 2013);
- Tracking Deformable Objects with Point Clouds (SCHULMAN et al., 2013);
- Optimal Local Searching for Fast and Robust Textureless 3D Object Tracking in Highly Cluttered Backgrounds (SEO et al., 2014);
- Global optimal searching for textureless 3D object tracking (WANG et al., 2015);
- SimTrack: A Simulation-based Framework for Scalable Real-time Object Pose Detection and Tracking (PAUWELS; KRAGIC, 2015);
- Real-Time 3D Tracking and Reconstruction on Mobile Phones (PRISACARIU et al., 2015);
- MobileFusion: Real-time Volumetric Surface Reconstruction and Dense Tracking On Mobile Phones (ONDRÚŠKA; KOHLI; IZADI, 2015);
- Deep learning of local RGB-D patches for 3D object detection and 6D pose estimation (KEHL et al., 2016);
- Going further with point pair features (HINTERSTOISSER et al., 2016);
- Team delft's robot winner of the amazon picking challenge 2016 (HERNANDEZ et al., 2016);
- Siamese regression networks with efficient mid-level feature extraction for 3d object pose estimation (DOUMANOGLOU et al., 2016a);

- Iterative hough forest with histogram of control points for 6 dof object registration from depth images (SAHIN; KOUSKOURIDAS; KIM, 2016);
- Recovering 6D Object Pose and Predicting Next-Best-View in the Crowd (DOUMANOGLOU et al., 2016b);
- Real-Time Monocular Pose Estimation of 3D Objects using Temporally Consistent Local Color Histograms (TJADEN; SCHWANECKE; SCHÖMER, 2017);
- Real-Time 3D Model Tracking in Color and Depth on a Single CPU Core (KEHL et al., 2017);
- Pose Optimization in Edge Distance Field for Textureless 3D Object Tracking (WANG; ZHONG; QIN, 2017);
- STAIR3D: Simultaneous Tracking And Incremental Registration For Modeling 3D Handheld Objects (CHAUDHARY et al., 2017);
- BB8: A Scalable, Accurate, Robust to Partial Occlusion Method for Predicting the 3D Poses of Challenging Objects without Using Depth (RAD; LEPETIT, 2017);
- SSD-6D: Making RGB-Based 3D Detection and 6D Pose Estimation Great Again (KEHL et al., 2017);
- iPose: Instance-Aware 6D Pose Estimation of Partly Occluded Objects (JAFARI et al., 2017);
- PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes (XIANG et al., 2017);
- Deep 6-DOF Tracking (GARON; LALONDE, 2017);
- ALCN: Adaptive Local Contrast Normalization for Robust Object Detection and 3D Pose Estimation (RAD; ROTH; LEPETIT, 2017);
- Combined Holistic and Local Patches for Recovering 6D Object Pose (ZHANG; CAO, 2017);
- 6-dof object pose from semantic keypoints (PAVLAKOS et al., 2017);
- T-LESS: An RGB-D dataset for 6D pose estimation of texture-less objects (HODAN et al., 2017);
- A dataset for developing and benchmarking active vision (AMMIRATO et al., 2017);
- Real-Time Seamless Single Shot 6D Object Pose Prediction (TEKIN; SINHA; FUA, 2017);

- Looking beyond the simple scenarios: Combining learners and optimizers in 3d temporal tracking (TAN; NAVAB; TOMBARI, 2017);
- Object-Based Visual-Inertial Tracking for Additive Fabrication (SANDY; BUCHLI, 2018);
- Latent-Class Hough Forests for 6 DoF Object Pose Estimation (TEJANI et al., 2018);
- DeepIM: Deep Iterative Matching for 6D Pose Estimation (LI et al., 2018);
- A Unified Framework for Multi-View Multi-Class Object Pose Estimation (LI; BAI; HAGER, 2018);
- Category-level 6D Object Pose Recovery in Depth Images (SAHIN; KIM, 2018);
- Multi-Task Deep Networks for Depth-Based 6D Object Pose and Joint Registration in Crowd Scenarios (SOCK et al., 2018);
- Occlusion Resistant Object Rotation Regression from Point Cloud Segments (GAO et al., 2018);
- Learning 6DoF Object Poses from Synthetic Single Channel Image (RAMBACH et al., 2018);
- An Appearance-and-Structure Fusion Network for Object Viewpoint Estimation (KAO et al., 2018);
- A modular framework for model-based visual tracking using edge, texture and depth features (TRINH et al., 2018);

APÊNDICE B – LISTA DE BIBLIOTECAS UTILIZADAS PARA DESENVOLVIMENTO

- numpy v1.14.5;
- Pillow v5.2.0;
- opencv-python v3.4.2.17;
- scipy v0.17.0;
- tensorflow-gpu v1.10.1;
- tensorboard v1.10.0;
- torch v0.3.1;
- torchvision v0.2.1;
- vispy v0.5.3;
- plyfile v0.6;
- matplotlib v1.5.1;
- glumpy v1.0.6;
- graphviz v0.8.4