UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

FELIPE PIERRE CONTER

DETECÇÃO DE OBJETOS E ESTIMATIVA DE POSIÇÃO 3D ATRAVÉS DA APLICAÇÃO DE REDES NEURAIS CONVOLUCIONAIS

CURITIBA

2022

FELIPE PIERRE CONTER

DETECÇÃO DE OBJETOS E ESTIMATIVA DE POSIÇÃO 3D ATRAVÉS DA APLICAÇÃO DE REDES NEURAIS CONVOLUCIONAIS

Object detection and 3D pose estimation through the use of convolutional neural networks

Dissertação apresentada como requisito para obtenção do grau de Mestre em Computação Aplicada, do Programa de Pós-Graduação em Computação Aplicada, da Universidade Tecnológica Federal do Paraná (UTFPR).

Orientador: Prof. Dr. João Alberto Fabro

CURITIBA

2022

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Campus Curitiba

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

FELIPE PIERRE CONTER

DETECÇÃO DE OBJETOS E ESTIMATIVA DE POSIÇÃO 3D ATRAVÉS DA APLICAÇÃO DE REDES NEURAIS CONVOLUCIONAIS

Trabalho de pesquisa de mestrado apresentado como requisito para obtenção do título de Mestre Em Computação Aplicada da Universidade Tecnológica Federal do Paraná (UTFPR). Área de concentração: Engenharia De Sistemas Computacionais.

Data de aprovação: 03 de Fevereiro de 2022

Prof Joao Alberto Fabro, Doutorado - Universidade Tecnológica Federal do Paraná

Prof Andre Schneider De Oliveira, Doutorado - Universidade Tecnológica Federal do Paraná

Prof Marcelo Nepomoceno Kapp, Doutorado - Universidade Federal da Integração Latino-Americana (Unila)

Documento gerado pelo Sistema Acadêmico da UTFPR a partir dos dados da Ata de Defesa em 07/02/2022.

Dedico este trabalho a todos os pesquisadores
da área de robótica, em especial aqueles
envolvidos com pesquisas sobre a aplicação da
visão computacional na robótica.

AGRADECIMENTOS

Agradeço primeiramente à minha família, pelo apoio incondicional que tive durante toda minha carreira acadêmica e profissional. Sua presença guiou a definição de minha ética e caráter. Em momentos onde precisei de apoio, sempre pude encontrá-lo em vocês.

Ao meu orientador, João Alberto Fabro e demais membros da UTFPR, estendo minha gratidão pelo conhecimento e apoio incansável. Destaco também um agradecimento especial aos professores e colegas do laboratório LASER e a todos os participantes da categoria @Home do LARC. Certamente não teria conseguido atingir sucesso na realização deste trabalho sem o conhecimento e o apoio que colocaram à minha disposição. Agradeço também pelos recursos, tempo e espaço que a UTFPR e seu corpo docente disponibilizaram.

A todos meus colegas de vida acadêmica e profissional de instituições por onde passei, agradeço pelas pequenas pedras que colocaram durante esta contínua e infinda edificação de meu conhecimento e caráter.

A todos os meus amigos, novos e antigos, pelo apoio nos momentos em que parecia não haver saída ou solução. Agradeço por estarem presentes em minha vida.

But maybe a better question is: "What are we going to do with these detectors now that we have them?" A lot of the people doing this research are at Google and Facebook. I guess at least we know the technology is in good hands and definitely won't be used to harvest your personal information and sell it to.... wait, you're saying that's exactly what it will be used for?? Oh. (REDMON, Joseph, 2018).

RESUMO

CONTER, Felipe P. **Detecção de objetos e estimativa de posição 3D através da aplicação de redes neurais convolucionais**. 2022. 88 f. Dissertação (Mestrado em Computação Aplicada) – Universidade Tecnológica Federal do Paraná (UTFPR). Curitiba, 2022.

Para aplicar com sucesso soluções de robótica em ambientes não controlados é necessário obter informação de qualidade sobre o ambiente de atuação do robô. A detecção, o reconhecimento e a estimativa de posição 3D de objetos fazem parte desse processo, sendo importantes para tarefas como a manipulação de objetos com um braço robótico. Métodos baseados em redes neurais convolucionais e modelos 3D dos objetos obtiveram sucesso nesta tarefa, em diferentes cenários, apesar da dificuldade de obtenção de modelos 3D em uma situação prática. Além disso, nem todos os métodos podem ser executados em sistemas embarcados, como aqueles usualmente presentes em robôs autônomos de serviço. Sensores RGB-D recentemente ganharam importância como ferramentas para o sensoriamento 3D do ambiente. Estes sensores associam a imagens coloridas RGB (*red, green and blue*) uma estimativa de profundidade para cada *pixel* da imagem, tornando possível a estimativa de posição 3D de objetos na cena com relação ao sensor. Este trabalho propõe e avalia um método que, a partir de dois registros RGB-D feitos de posições diferentes, faz a estimativa do ponto central 3D de um objeto, sem a necessidade de conhecer o formato 3D do objeto de antemão. Os experimentos demonstram que a distância euclidiana média entre o ponto central 3D estimado e o real foi de 9,4 cm, com desvio padrão de 5,4 cm, sendo que na melhor estimativa este valor foi de 8 mm.

Palavras-chave: Detecção de objetos. Estimativa de posição de objetos. Robótica. RoboCup@Home. RGB-D.

ABSTRACT

CONTER, Felipe P. **Object detection and 3D pose estimation through the use of convolutional neural networks**. 2022. 88 p. Dissertation (Master's Degree in Applied Computing) – Federal University of Technology – Parana (UTFPR). Curitiba, 2022.

To successfully apply robotics solutions in uncontrolled environments it is necessary to obtain quality information about the robot's operating environment. Detecting, recognizing and estimating 3D position of objects are part of this process and are important for tasks such as object manipulation with a robotic arm. Methods based on convolutional neural networks and 3D models of objects were successful in this task, in different scenarios, despite the difficulty of obtaining 3D models in a practical situation. Also, not all methods can be executed in embedded systems, such as those usually present in autonomous service robots. RGB-D sensors have recently gained importance as tools for environment 3D sensing. These sensors associate RGB color images (red, green and blue) with a depth estimate for each pixel of the image, making it possible to estimate the 3D position of objects in the scene with respect to the sensor. This work proposes and evaluates a method that, from two RGB-D shots made from different positions, estimates an object's 3D center point, without the need of knowing the object's 3D shape in advance. The experiments show that the average euclidian distance between the estimated and the real 3D center points was 9.4 cm, with a standard deviation of 5.4 cm and in the best estimation this value was 8 mm.

Keywords: Object detection. Object pose estimation. Robotics. RoboCup@Home. RGB-D.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A aplicação da robótica em ambientes domésticos foi por muito tempo tratada como ficção, sendo retratada em filmes e desenhos animados como utopia futurística. O custo elevado das soluções de robótica e a capacitação de recursos humanos para sua operação apresentou-se, desde os primeiros estudos, como barreiras que dificultavam a aplicação da robótica não somente em ambientes domésticos, como também em diversas áreas da indústria, o que foi mudando conforme se estabeleciam avanços tecnológicos. Com o tempo, componentes com capacidades de processamento cada vez maiores tornaram-se disponíveis a preços cada vez menores (CORKE, 2017).

Gasparetto e Scalera (2019) definem que a história da evolução dos robôs industriais pode ser classificada em quatro gerações:

a) Primeira geração de robôs industriais (1950-1967): os robôs dessa geração eram máquinas sem comunicação com algum ambiente externo. Quase todos os robôs eram baseados em atuadores pneumáticos, e era comum que fizessem muito barulho, devido à colisão dos braços com limitadores mecânicos (que serviam para limitar o movimento dos eixos). Atuadores hidráulicos também foram introduzidos posteriormente. Devido à simplicidade, eram aplicados em tarefas triviais como carga e descarga, e outras tarefas que envolvessem movimentos simples. Eram empregados principalmente na indústria automotiva.

b) Segunda geração de robôs industriais (1968-1977): Neste período os robôs receberam servo-controladores, que permitiam maior controle sobre os movimentos de seus eixos, além de microprocessadores que começaram a ser largamente aplicados em tarefas complexas e com alto custo computacional. Atuadores elétricos foram introduzidos em 1970 e gradualmente foram substituindo os atuadores pneumáticos e hidráulicos. Apesar das evoluções, os robôs eram dispositivos criados para aplicações específicas, sendo que era muito rara a aplicação de um mesmo robô em tarefas diferentes. A capacidade de reconhecer o ambiente externo e de se adaptar era básica, dando os primeiros passos para o que viria nas próximas gerações.

c) Terceira geração de robôs industriais (1978-1999): o principal diferencial na terceira geração foi um maior nível de interação com o ambiente e com o operador

através de interfaces complexas como a visão ou a voz. A capacidade de programação de alto nível ampliou o potencial de aplicação dos robôs. Relatórios de erros tornaram-se mais elaborados que apenas uma luz vermelha ou um aviso sonoro, com detalhes mais ricos de onde o erro foi detectado e quais as possíveis causas. Apesar de limitadas, capacidades adaptativas evoluíram e os robôs podiam alterar seu comportamento de acordo com novas informações advindas de sensores. A introdução do acionamento direto das juntas permitiu que motores se conectassem diretamente às juntas, eliminando elementos intermediários como sistemas de correntes e aumentando a precisão dos movimentos.

d) Quarta geração de robôs industriais (2000 em diante): Capacidades consideradas "inteligentes" foram desenvolvidas através de algoritmos de alto nível com forte influência das pesquisas em inteligência artificial. Exemplos incluem o *deep learning*, estratégias complexas e comportamento colaborativo. Componentes mais baratos e sensores mais precisos que os presentes nas gerações anteriores também favoreceram a aplicação em maior escala.

Essa evolução tecnológica possibilitou que computadores com capacidade de processamento elevado estivessem nas casas e nos bolsos das pessoas (ex: *smartphones*). Robôs para entretenimento e robôs aspiradores de pó são outros exemplos da presença da robótica em casa que já estão comercialmente disponíveis (PRASSLER; KOSUGE, 2008).

Ambientes estruturados e controlados, como fábricas, exigem um nível menor de sensoriamento do ambiente visando a adaptação da atuação do robô. Ambientes estruturados permitem que o robô atue com menor risco, com menos fatores influenciando sua tomada de decisão. Dentre os desafios para a aplicação mais ampla da robótica em ambientes domésticos está o fato de que casas são naturalmente diferentes umas das outras, com muitas diferenças de um ambiente para outro, como tipos de piso e iluminação, por exemplo. A transição dos robôs de ambientes estruturados para ambientes não-estruturados, como o ambiente doméstico, exige maior adaptabilidade do robô, o que por sua vez, exige mais complexidade no processo de tomada de decisão e no processamento de informações sobre o ambiente. Neste contexto, a visão computacional torna-se uma área crucial no que diz respeito a pesquisas aplicadas em robótica.

A visão computacional tem aplicações em diversas áreas da computação, incluindo a robótica, inteligência artificial, interação homem-máquina e veículos autônomos. Sua aplicação na robótica pode fornecer ao robô informações cruciais sobre o seu ambiente de atuação e

sobre os objetos que se encontram nesse ambiente. Para que o robô possa manipular um objeto, por exemplo, é essencial que o robô consiga detectar sua presença no ambiente e que consiga estimar sua posição com relação ao robô. O conhecimento na área da visão computacional teve grandes mudanças nos últimos anos com relação às ferramentas e métodos utilizados para o reconhecimento de objetos em imagens (CARVALHO; VON WANGENHEIM, 2019). Por volta de 2011 e 2012, métodos baseados em redes neurais convolucionais[1] (*convolutional neural networks*, ou CNN) começaram a se popularizar, resultando em estimativas mais precisas que aquelas das técnicas tradicionais aplicadas anteriormente, apesar do custo computacional mais alto (KRIZHEVSKY *et al.*, 2012). Um dos principais fatores que contribuíram para isso foi o alto poder de processamento e a velocidade de acesso à memória que as GPUs[2] (*graphical processing unit* ou unidade de processamento gráfico) disponíveis no mercado alcançaram. Estes avanços tecnológicos nas GPUs apresentaram um novo patamar de desempenho que permitiu elevar a complexidade das CNNs, processando conjuntos cada vez maiores de dados. Esse fator está relacionado ao conceito de *big data*.

O processo tradicional de aquisição de imagens nos sistemas de visão computacional era baseado em câmeras coloridas RGB (*red, green and blue*). Entretanto, sensores diferentes das câmeras RGB tradicionais também ajudaram a expandir os horizontes das pesquisas. O LIDAR (*light detection and ranging*) é um método de medição de distâncias que aplica conceitos análogos ao radar, porém com base na iluminação *laser*. Geralmente funcionam a partir da rotação uma base emissora/receptora de *laser* em torno de um eixo. Computacionalmente, as informações registradas por um sensor LIDAR podem ser representadas por um vetor de números, que representam as distâncias medidas nos diferentes ângulos durante essas rotações. Este tipo de sensor pode medir distâncias com alta precisão, sendo aplicado em veículos autônomos e problemas onde é necessário precisão milimétrica (RAJ *et al.*, 2020).

Outros sensores buscaram ampliar essa capacidade para conseguir registrar uma matriz de distâncias, permitindo ao robô uma percepção tridimensional de seu ambiente. As câmeras estéreo já produziam resultados satisfatórios ao reconstruir uma cena 3D a partir de duas ou mais imagens 2D, capacidade que foi consolidada com os sensores RGB-D. Tais sensores trouxeram uma nova camada de percepção para os robôs, sendo capazes de, além de registrar uma imagem bidimensional colorida (RGB), adicionar um canal com uma estimativa de profundidade para

[1] É um tipo específico de rede neural artificial que vêm sendo aplicada com sucesso no processamento e análise de imagens digitais.

[2] As GPUs são uma categoria específica de microprocessadores especializados em processamento gráfico.

cada um dos *pixels* da imagem. Essa profundidade, em outras palavras, é a distância do sensor ao ponto 3D no mundo real representado pelo *pixel* correspondente na imagem. Por exemplo: uma imagem RGB que tradicionalmente é armazenada como um conjunto de 3 matrizes de dados (uma para cada camada de cor), agora ganha mais uma matriz onde cada elemento guarda a profundidade estimada, geralmente expressa em milímetros, para cada *pixel* na imagem. Quando sensores de baixo custo como o *Kinect*[3] foram disponibilizados no mercado, o sensoriamento sincronizado de imagem (RGB) e de profundidade ficou mais acessível e tornou-se viável para uso disseminado (SHAO *et al.*, 2014).

O sensor RGB-D pode ser utilizado na resolução de problemas da robótica doméstica, como estimativa de posição de objetos e de esqueletos (termo que é usado em pesquisas relacionadas a estimativa de posição de seres humanos envolvendo as articulações do corpo), rastreamento de pessoas e objetos ao longo do tempo, mapeamento 3D e "localização e mapeamento simultâneos" (*simultaneous localization and mapping* ou SLAM) (CORKE, 2017).

Considerando todo o contexto apresentado, este trabalho envolve a pesquisa de abordagens que permitam a robôs autônomos identificarem a posição de objetos em ambientes pouco estruturados como os ambientes residenciais. O objetivo é ter uma estimativa de posição de objetos precisa para que um braço robótico acoplado ao robô consiga usar esta informação para capturar e manipular estes objetos.

1.1 MOTIVAÇÃO

O evento RoboCup é uma competição mundial que visa o avanço do desenvolvimento da tecnologia robótica, através de competições em diversas categorias envolvendo robôs autônomos. Uma das categorias mais recentes é a RoboCup@Home[4] que busca desenvolver tecnologia para robôs com foco em assistência e serviço para aplicações domésticas e pessoais. Nesta categoria, diversas equipes competem em um ambiente doméstico e passam por testes para avaliar o desempenho de seus robôs. As provas incluem, porém não se resumem, a: interação homem-máquina e cooperação (incluindo interpretação e síntese de voz), navegação e mapeamento em ambientes dinâmicos, visão computacional e reconhecimento de objetos sob condições de iluminação natural, manipulação de objetos, comportamentos adaptativos e integração comportamental. As equipes buscam cumprir as provas combinando essas diversas capacidades de seus robôs. Por

[3] O *Kinect* é um dispositivo popular de baixo custo lançado em 2010 originalmente para o console de videogame Xbox 360. Além de uma câmera, possui um sensor de profundidade e microfones.

[4] *Site* oficial da RoboCup@Home: https://www.robocupathome.org.

exemplo, há tarefas que envolvem interagir com pessoas, demandando a capacidade de síntese e reconhecimento de voz. Outras tarefas podem exigir que o robô locomova-se até determinadas posições no ambiente, o que envolve a navegação e mapeamento. Manipular objetos envolve o reconhecimento e a estimativa de posição 3D dos mesmos. Somente na etapa de manipulação de objetos, existem diversos problemas. Para resolvê-los o robô pode empregar alguns métodos, indo desde a visão computacional para identificar o objeto até a definição da estratégia de manipulação do objeto, envolvendo questões como: Qual o melhor ponto para agarrá-lo? Por onde o braço deve se aproximar? Portanto, para poder manipular um objeto, é essencial que o robô tenha a capacidade de detectar sua presença e que consiga estimar sua posição 3D com relação ao robô.

Atualmente, a RoboCup@Home possui 3 diferentes ligas:

a) *Domestic Standard Platform League* (DSPL): foca na solução de tarefas domésticas usando robôs com *hardware* padronizado, eliminando a influência do elemento do *hardware* nas avaliações. O robô padrão atualmente escolhido para esta liga pela *RoboCup Federation* é o *Toyota Human Support Robot* (HSR), que pode ser visto na Figura 1.

Figura 1 – O robô *Toyota Human Support Robot* (HSR).

Fonte: Toyota (2015).

b) *Social Standard Platform League* (SSPL): o foco dessa liga é nos aspectos sociais da interação do robô, também utilizando *hardware* padronizado. O robô padrão

atualmente escolhido para esta liga pela *RoboCup Federation* é o *Softbank Pepper Robot*, que pode ser visto na Figura 2.

Figura 2 – O robô *Softbank Pepper Robot*.

Fonte: SoftBank Robotics (2020).

c) *Open Platform League* (OPL): sem a imposição de nenhuma restrição de *hardware* às equipes participantes, essa liga permite maior inovação e aplicação de diversas abordagens diferentes pelas equipes. A OPL permite que as equipes participantes coloquem em prática diferentes soluções de *software* e de *hardware*, utilizando diferentes robôs com diferentes sensores, bases móveis e manipuladores. A edição latino-americana da RoboCup@Home, que ocorre anualmente como parte do *Latin American Robotics Competition* (LARC), possui apenas a OPL atualmente, não realizando competições da DSPL ou SSPL. Exemplos dos robôs que participaram no LARC em 2019 podem ser vistos na Figura 3. Nesta foto, registrada ao final do último dia da competição, é possível ver também o robô da equipe UTBots@Home, da UTFPR, que conquistou o 5º lugar na classificação geral: é primeiro robô da esquerda para a direita.

É válido enfatizar as principais diferenças entre as ligas. A DSPL e a SSPL trabalham com robôs padronizados. O alto custo destes robôs é um fator que dificulta a entrada de novas equipes. Por outro lado, limitações provenientes do *hardware* ou sensores afetam todas as equipes igualmente, uma vez que elas estão compartilhando o mesmo conjunto de *hardware*. A OPL,

Figura 3 – Robôs participantes da categoria @Home no LARC 2019.

Fonte: Autoria própria (2019).

no lado oposto desse espectro, não possui nenhuma restrição de *hardware*, bastando que o robô atenda os requisitos mínimos apresentados nas regras e seja aprovado na avaliação, que ocorre no início das provas. Por não limitar as soluções de *hardware*, cada equipe pode optar por bases móveis diferentes, com diferentes sensores e braços manipuladores. As escolhas desses elementos de *hardware* também afetam diretamente a *performance* das equipes, que devem ter cuidado não somente com as soluções de *software*, mas também com a gestão desse *hardware* a fim de evitar falhas.

Conforme as regras oficiais da edição de 2019 (MATAMOROS *et al.*, 2019a), que ocorreu em Sidney na Austrália, a competição tenta manter um conjunto de regras simples, favorecendo a comunicação das mesmas entre juízes e equipes. Todos os robôs participantes devem ser autônomos e móveis: agindo e navegando no ambiente sem qualquer tipo de controle remoto. Os testes envolvem aplicações reais com diferentes níveis de incerteza. Por exemplo, algumas interferências podem ocorrer, como cadeiras que mudaram de lugar ou pessoas transitando pela casa. O ambiente de provas também não é padronizado, variando entre as diversas edições da RoboCup@Home. Algumas provas podem ocorrer em locais desconhecidos dos participantes, como um espaço fora do ambiente de provas.

A competição busca manter-se atrativa ao público geral, dessa forma busca recompensar a originalidade e atratividade das equipes. Outro conceito que merece atenção é que a competição

busca divulgar ao público a utilidade e aplicabilidade real dos robôs autônomos em situações sociais, como guiar pessoas, servir bebidas, etc. Busca-se incentivar o desenvolvimento científico valorizando a inovação e a aplicação de novas abordagens. A competição valoriza a comunidade criada em torno de si encorajando a troca de informações entre as equipes, e a colaboração científica entre as mesmas buscando o avanço tecnológico conjunto.

Um detalhe notável é que para um robô executar tarefas precisa processar dados provenientes de diversos sensores. Normalmente a gestão da tomada de decisão do robô está dividida em mais de um processo no sistema operacional e é preciso que tais processos possam comunicar-se de forma ordenada. Para orquestrar essa comunicação existe o ROS (*Robot Operating System*)[5]. O ROS serve como uma base onde processos relacionados à robótica podem ser executados, garantindo transparência e portabilidade a soluções, tornando-as independentes de detalhes de *hardware* específicos.

O reconhecimento de objetos e pessoas faz parte das capacidades que o robô deve possuir para cumprir as tarefas. No caso específico de objetos, os testes envolvem pelo menos 30 classes diferentes. Há ainda a possibilidade da introdução de objetos previamente desconhecidos. Os objetos são divididos em categorias, por exemplo: as classes `maçã` e `banana` fazem parte da categoria `frutas`. Além disso, se dividem em 2 grupos: os padronizados, que não apresentam diferenças perceptíveis de outros da mesma classe, como `lata do refrigerante A` e `caixa do cereal B`; e os não-padronizados, aqueles que podem apresentar uma variedade de aparências dentro da própria classe, como `maçãs`, `sanduíches`, etc.

Uma lista de objetos pré-definidos é apresentada no livro de regras da competição e objetos adicionais são introduzidos pelos organizadores antes do início das provas. De maneira geral, esses objetos ficam disponíveis para as equipes por pouco tempo. A duração deste período não está definida no livro de regras, podendo variar de competição para competição. Este é o período que as equipes têm para gerar os dados relacionados a esses objetos para treinar seus algoritmos. No LARC de 2019, por exemplo, os objetos foram apresentados às equipes um dia antes do início das provas. Essas restrições dificultam o treinamento de algoritmos de reconhecimento que demandem muitos exemplos ou muito tempo para a execução dos passos inerentes à preparação dos dados. As equipes devem ser capazes de incluir objetos novos em seus algoritmos já no local da competição, de forma rápida e com baixa complexidade operacional.

Na OPL, as equipes podem usar diferentes robôs com diferentes configurações de

[5] *Site* oficial do ROS: https://www.ros.org/.

hardware. Segundo Matamoros *et al.* (2019b), em 2017 e 2018 todos os times reportaram que usaram pelo menos um sensor RGB-D, sendo que os mais recorrentes são o *Kinect 2*[6], o *Asus Xtion*[7] e a primeira versão do *Kinect*.

Sabendo-se que o reconhecimento de objetos faz parte das tarefas da RoboCup@Home, que os métodos empregados devem permitir o treinamento rápido com novos objetos, e que os sensores RGB-D podem ser utilizados nesta tarefa, pode-se destacar também quais são as técnicas mais utilizadas pelas equipes participantes para esta tarefa. Conforme apresentado na Tabela 1, 84% das equipes participantes da OPL em 2017 e 2018 reportaram as soluções que empregaram no reconhecimento de objetos. A rede neural convolucional YOLO (REDMON; FARHADI, 2018) é a mais presente, seguido de outras técnicas mais tradicionais como SIFT (LOWE, 1999) e SURF (BAY *et al.*, 2008).

Tabela 1 – Soluções de *software* adotadas para reconhecimento de objetos na RoboCup@Home OPL em 2017 e 2018, conforme reportado por 84% das equipes participantes.

Solução	Casos
YOLO	31%
SIFT	19%
SURF	16%
OpenCV Caffe	9%
Tensorflow	9%

Fonte: Adaptado de Matamoros *et al.* (2019b).

O YOLO é um detector de objetos baseado em redes neurais. O SIFT e o SURF são descritores de características de objetos muito utilizados na detecção dos mesmos. Já o OpenCV Caffe e o Tensorflow são bibliotecas que dão suporte para uma variedade de abordagens baseadas em redes neurais. Todas estas tecnologias são melhor detalhadas no Capítulo 2.

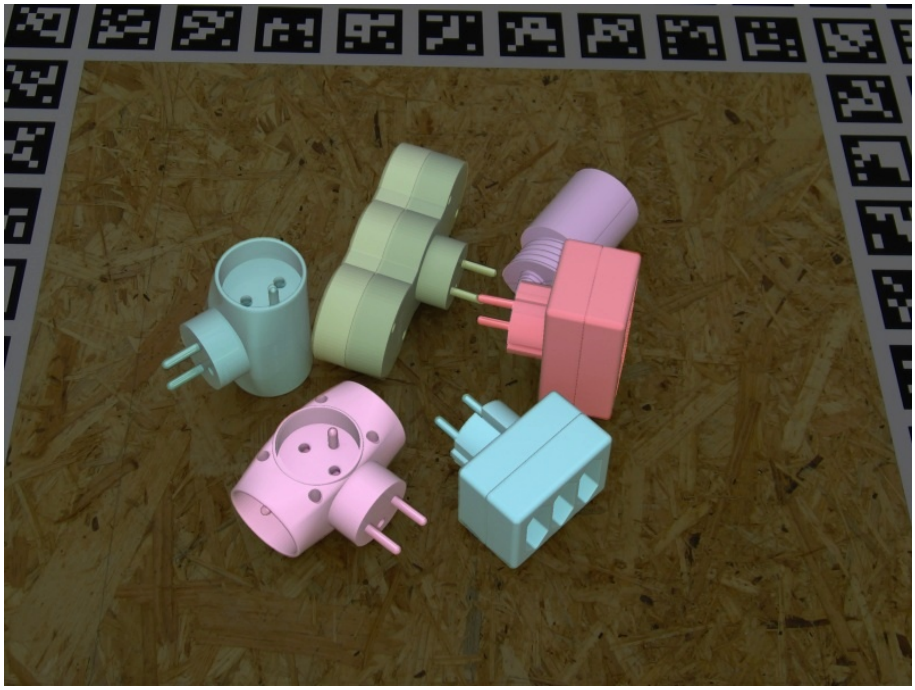
Além dessas tecnologias citadas, outras abordagens podem ser empregadas para realizar a estimativa de posição 3D de objetos. Suas características serão também detalhadas no Capítulo 2. Há um conjunto de métodos que baseiam-se em modelos 3D dos objetos. Tais modelos são representações da forma 3D de objetos. Modelos 3D de objetos rígidos podem ser gerados com um *scanner* 3D e podem servir de base para métodos que busquem essas formas 3D no ambiente registrado pelos sensores 3D do robô. Há uma etapa inerente a este processo que nem sempre pode ser aplicada na prática: a obtenção ou a criação de modelos 3D dos objetos que se deseja encontrar. Na Figura 4 é possível ver exemplos de modelos 3D. As limitações implicadas pelos

[6] Lançado em 15 de julho de 2014, o *Kinect 2* é baseado no *Kinect* do Xbox One e apresenta avanços tecnológicos comparado ao *Kinect* original.

[7] É um sensor alternativo ao *Kinect*, com as mesmas características básicas. *Site* oficial do *Asus Xtion*: https://www.asus.com/3D-Sensor/Xtion/.

métodos baseados em modelos 3D serão abordadas com mais detalhes no Capítulo 2.

Figura 4 – Modelos 3D de objetos sobrepostos em uma foto.



Fonte: Hodaň *et al.* (2017).

Objetos como maçãs, bananas, ou até mesmo objetos com forma deformável como pacotes maleáveis comumente encontrados em estabelecimentos comerciais (ex: pacotes de salgadinhos) não possuem uma forma 3D definitiva ou imutável. Tentar registrar ou prever todas as diferentes formas 3D de tais objetos seria impossível, uma vez que podem possuir variação intraclasse. O problema agrava-se quando tratamos das instâncias de objetos cuja forma é deformável como sacos e pacotes maleáveis.

Com estas constatações, percebe-se que é necessário o desenvolvimento de métodos de estimativa de posição 3D de objetos que dispensem o conhecimento prévio do formato 3D dos mesmos. Uma hipótese é que tais métodos poderiam também estimar a posição 3D de instâncias de objeto desconhecidas, nunca antes vistas pelo robô. Essa capacidade de aceitar instâncias nunca vistas antes pelo algoritmo pode ser observada em algoritmos de detecção de objetos em imagens RGB baseados em CNNs como o YOLOv3 (REDMON; FARHADI, 2018).

Uma vantagem de usar um método de estimativa de posição 3D que dispense os modelos 3D dos objetos é a simplificação de sua aplicação em situações práticas. Por exemplo, no caso de robôs de serviço atuando em ambientes domésticos, como ocorre nas competições da OPL. Métodos baseados em modelos 3D não funcionam bem com objetos de formas deformáveis ou com formatos variados. Tais métodos também não funcionam bem com instâncias de objetos

nunca vistas antes. A classificação de objetos guiada por redes neurais permite que esses objetos sejam considerados pela rede, que tentará atribuir a eles uma classe, mesmo que seja com um grau de confiança menor.

A estimativa de posição 3D de um objeto envolve estimar onde um objeto se encontra em uma cena. Um ponto 3D é expresso na forma de 3 coordenadas, representando sua posição em relação aos eixos `x`, `y` e `z`. Há diferentes formas de representação dessa estimativa, sendo uma delas o ponto central 3D do objeto: esse ponto é localizado em seu centro, e pode ser definido como o ponto central nos eixos `x`, `y` e `z`. Por exemplo, em um objeto com 5 cm de profundidade, 4 cm de largura e 10 cm de altura, o ponto central 3D poderia ser definido na metade dessas medidas: $(x; y; z) = (2{,}5; 2{,}0; 5{,}0)$, sendo a unidade de medida o centímetro.

Outra forma de estimativa de posição de um objeto que inclui ainda mais detalhes é chamada de *6D pose estimation* ou *6-Degree-Of-Freedom pose estimation* (*6-DOF pose estimation*). Essa estimativa inclui não somente a posição 3D, mas também a orientação do objeto (rotação nestes três eixos). A orientação é representada a partir de 3 valores adicionais: *roll*, *pitch* e *yaw*, conceito este importado da aviação. Anualmente, o evento *International Workshop on Recovering 6D Object Pose* dedica-se a apresentar os avanços em soluções para esta problemática. Tais algoritmos normalmente baseiam-se nos modelos 3D dos objetos: com o conhecimento prévio do formato 3D do objeto, estimam a posição e orientação do mesmo.

Restringindo a aplicação de métodos para apenas aqueles que não exigem conhecimento prévio do formato 3D dos objetos, pode-se utilizar o ponto central do objeto como representação de sua posição 3D. Sem o modelo 3D do objeto, é difícil definir a orientação do objeto, ainda mais quando trata-se de um objeto novo, não visto anteriormente pelo algoritmo. Não foram encontradas abordagens na literatura que realizem estimativas de posição para objetos de forma 3D desconhecida (aqueles para os quais não se possui um modelo 3D). Consideranto todo este contexto, são apresentados na seção a seguir os objetivos geral e específicos deste trabalho.

1.2 OBJETIVOS

O objetivo principal deste trabalho é propôr um método de estimativa do ponto central 3D de objetos para aplicação na robótica. Sua aplicação envolve a etapa da detecção de objetos. O foco é o uso em robôs autônomos de serviço para aplicação doméstica e pessoal, contribuindo também para o desempenho da equipe do laboratório LASER da UTFPR em futuras edições da RoboCup@Home. Este método não baseia-se em modelos 3D dos objetos, uma vez que a

obtenção de tais modelos é pouco prática tanto para a competição RoboCup@Home, quanto para a aplicação em um ambiente doméstico real. Portanto, o método proposto vai restringir-se apenas à estimativa de posição sem considerar a orientação. Justamente por dispensar modelos 3D, a ideia é que o método seja aplicável a objetos de forma livre, objetos deformáveis e também a objetos nunca apresentados antes ao robô (desde que esses objetos sejam pertencentes a classes já conhecidas).

É importante ressaltar que este trabalho limita-se à detecção de objetos e estimativa do ponto central 3D. A manipulação robótica é um passo posterior, fora do escopo do trabalho atual, que pode ser baseada em diferentes propostas e estratégias específicas para esse fim.

1.2.1 Objetivos Específicos

Além do objetivo principal, é importante detalhar os objetivos específicos do trabalho, que estão listados abaixo:

- Revisão bibliográfica e estudo sobre o estado da arte: De forma a apresentar conceitos que servem de base para este trabalho, é necessária a realização de uma revisão bibliográfica. O objetivo é detalhar quais são as pesquisas relacionadas com o estado da arte na detecção e na estimativa de posição 3D de objetos. Isso será melhor detalhado no Capítulo 2. Este capítulo também é importante para evidenciar os principais problemas em aberto.

- Definição de um método para detecção de objetos: Uma etapa anterior à estimativa de posição de um objeto é a detecção deste objeto no ambiente de atuação do robô. É fundamental deixar claro que o método proposto será aplicado na robótica autônoma de serviços. Para isso, é interessante que toda a tecnologia envolvida seja adaptável para a execução em plataformas embarcadas, com foco em baixo consumo de energia e cuja detecção de objetos ocorra com velocidade de processamento de pelo menos 1 quadro por segundo (*frames per second* ou FPS). O método deve permitir o treinamento com classes de objetos personalizadas e deve permitir a detecção de ocorrências de novas instâncias dentro dessas classes. Um exemplo é o da classe `maçã`: na OPL, uma maçã é fornecida à equipe para treinamento, mas durante a prova nada garante que ocorrências de objetos desta classe terão o mesmo formato e cor. Sacolas maleáveis de produtos industrializados também podem variar em formato e disposição. Até mesmo ao lidar com objetos com formato e características pouco variáveis como colheres: a capacidade de

classificar corretamente colheres não vistas anteriormente na classe `colher` é interessante. Portanto, um dos objetivos é que o método permita o reconhecimento de instâncias novas, buscando categorizá-las em alguma das classes definidas.

- Proposta de um método de estimativa de ponto central 3D de objetos: Uma vez que foi realizada a revisão bibliográfica e que foi definido um método para detecção de objetos, o trabalho propõe um novo método de estimativa de posição de objetos baseado no conceito do ponto central 3D. Este método é detalhado no Capítulo 3. É preciso que o método seja de fácil aplicação, demandando pouco esforço em sua etapa de preparação. Esta característica é conveniente para a aplicação em problemas da robótica autônoma de serviços em ambientes domésticos reais, onde os robôs muitas vezes entram em contato com objetos desconhecidos nas residências das pessoas.

- Definição e execução de experimentos para avaliação do método proposto: Experimentos específicos foram conduzidos a fim de avaliar a precisão das estimativas de posição do método proposto. Estes estão expostos, junto a seus resultados, no Capítulo 4.

1.3 RESULTADOS ESPERADOS

Espera-se que o método proposto seja capaz de detectar objetos nas cenas de atuação do robô, classificando tais objetos dentro de um conjunto de classes previamente definidas.

Espera-se também que ao identificar estes objetos, o método consiga estimar o ponto central 3D desses objetos. Com pelo menos 2 registros do objeto feitos a partir de pontos diferentes, o robô deve ser capaz de estimar uma posição 3D para o objeto, na forma de um ponto central 3D. O objetivo é permitir posteriormente, a manipulação do objeto por um robô. Entretanto, a manipulação do objeto é um passo posterior, fora do escopo do trabalho atual.

1.4 ESTRUTURA DO DOCUMENTO

Neste capítulo, foi apresentada uma introdução ao trabalho, contextualizando-o nos campos da robótica e da visão computacional, além do detalhamento dos objetivos do trabalho.

O Capítulo 2 apresenta uma revisão da literatura que serve de base para este trabalho, definindo conceitos relacionados ao reconhecimento de objetos e apresentando as técnicas fundamentais definidas nas décadas de 80 e 90 do século XX. São apresentados também os

resultados de um levantamento que foi realizado utilizando-se de técnicas do mapeamento sistemático a fim de encontrar artigos e trabalhos em conferências com foco na detecção ou reconhecimento de objetos com uso de sensores RGB-D. São listados os principais formatos de representação de objetos 3D com exemplos de métodos que os utilizam. O processo de classificação de objetos como um todo é detalhado de forma genérica, sem prender-se a um método em específico. Um foco pontual é dado aos métodos mais aplicados na Robocup@Home - OPL, apresentando seus detalhes. Por fim, são discutidos os principais problemas em aberto, sob a ótica da aplicação na robótica autônoma de serviços.

O Capítulo 3 inicia com o detalhamento da metodologia. O método proposto para estimativa do ponto central 3D do objeto é definido, além de características que o método deve atender.

O Capítulo 4 apresenta os experimentos realizados, e os resultados obtidos com o método proposto. Com tais experimentos pretende-se avaliar a precisão do método proposto e sua resiliência a erros de posicionamento do robô.

Concluindo o trabalho, o Capítulo 5 discute os resultados obtidos e apresenta perspectivas para o futuro com relação à estimativa de posição 3D de objetos através de métodos que dispensem o conhecimento prévio da forma 3D dos mesmos.

2 REVISÃO DA LITERATURA

Para detecção de objetos em imagens, e para a estimativa de posição 3D dos mesmos, há diversos métodos que foram apresentados ao longo dos últimos anos. O objetivo deste capítulo é apresentar avanços relevantes e métodos do estado da arte.

Antes da visão computacional, muito se pesquisou sobre a visão ao longo dos anos, sob a ótica de outras áreas do conhecimento, como a medicina e a biologia. Com relação à visão, é interessante a reflexão definida por Marr (1982):

> O estudo da visão deve, portanto, incluir não apenas o estudo de como extrair das imagens os vários aspectos do mundo que são úteis para nós, mas também uma investigação sobre a natureza das representações internas pelas quais capturamos esta informação e assim a tornamos disponível como base para decisões sobre nossos pensamentos e ações. Esta dualidade - a representação e o processamento da informação - está no coração da maioria das tarefas de processamento de informação e moldará profundamente a nossa investigação dos problemas específicos impostos pela visão (MARR, 1982, tradução nossa).

Anos depois, essa afirmação ainda soa atual e verdadeira. Percebe-se que na visão computacional, o mesmo também se aplica. É preciso estudar maneiras de extrair de imagens as informações que são úteis, devendo-se questionar também o formato de representação interna dessa informação sob a ótica dos algoritmos de tomada de decisões que irão consumi-las. Sem saber que tipo de informação se quer extrair dos sensores, qual o formato no qual essa informação deve ser disponibilizada para os algoritmos, é difícil definir qual será a abordagem mais adequada. A visão computacional inclui, portanto, a investigação sobre as maneiras de representação das informações obtidas num nível mais alto de abstração, e não apenas o processamento dos dados brutos disponibilizados na imagem, ou em casos mais específicos da robótica, pelos sensores RGB-D. O mesmo se aplica para o reconhecimento de objetos, que possui aplicações em diversas áreas da computação, incluindo a robótica, inteligência artificial, interação homem-máquina e veículos autônomos.

O reconhecimento de objetos é um conceito genérico, que se refere a uma coleção de tarefas relacionadas a identificar a presença de objetos em imagens digitais. Por outro lado, pode-se distinguir claramente o significado dos termos a seguir (BROWNLEE, 2019) (PANTHA, 2019) (SAXENA, 2020):

- Classificação de imagens: prever a classe de um objeto em uma imagem. Dada uma imagem, o algoritmo produzirá como saída uma classe específica podendo ter uma probabilidade

associada (geralmente expressa em porcentagem). Por exemplo, uma imagem que contém gatos terá como saída a classe `gato`, com 70% de probabilidade;

- Localização de objetos: localizar a presença de objetos em uma imagem, indicando a localidade com uma *bounding box* (“caixa delimitadora”, em tradução livre). Uma *bounding box* é um retângulo delimitador, que destaca a localização do objeto na imagem, tangenciando as bordas do mesmo. Por exemplo, uma imagem com dois objetos teria como saída ideal duas *bounding boxes*. É importante observar que a localização é um conceito que não inclui a classificação.

- Detecção de objetos: localizar a presença de objetos em uma imagem com uma *bounding box* e uma previsão de classe para cada ocorrência. Novamente, é comum que cada ocorrência de objeto venha acompanhada por uma probabilidade expressa em porcentagem, porém isso não é uma regra. Por exemplo, uma foto com um gato e um cachorro teria como saída ideal duas *bounding boxes*: uma identificada com a classe `gato` e a outra com a classe `cachorro`.

Em diferentes publicações, estes problemas são comumente chamados de “reconhecimento de objetos”. O conceito de cada termo pode variar entre trabalhos científicos diferentes. Dessa forma, neste trabalho, os termos serão utilizados com as definições que foram apresentadas.

Ao longo da história, diferentes abordagens que envolvem o reconhecimento de objetos foram apresentadas à comunidade científica, com diferentes tipos de dados. A seção 2.1 trata sobre trabalhos fundamentais da década de 80 e início da década de 90 do século XX que buscaram resolver o problema do reconhecimento de objetos em 3 dimensões (3D) ou com múltiplas visões (normalmente baseados em mais de uma imagem 2D), assim como o problema da estimativa de posição.

A seção 2.2 apresenta um levantamento bibliográfico sobre a temática da detecção e reconhecimento de objetos utilizando-se de sensores RGB-D. Os formatos de representação de objetos 3D mais utilizados são descritos na seção 2.3.

Já as seções seguintes tratam das etapas inerentes ao processo: A aquisição de dados (seção 2.4), a identificação/classificação do objeto (seção 2.5) e mais detalhes sobre como as unidades de processamento gráfico embarcadas podem ser utilizadas (seção 2.6).

A seção 2.7 detalha as tecnologias e métodos empregados pelas equipes participantes da RoboCup@Home na categoria OPL. Finalmente, a seção 2.8 trata de apresentar os principais

problemas em aberto no que diz respeito à aplicação da detecção de objetos e estimativa de posição 3D dos mesmos na robótica móvel autônoma.

2.1 ANOS 80 E 90: TÉCNICAS FUNDAMENTAIS

Em 1980, a transformada de Hough foi adaptada para detectar instâncias de contornos específicos em imagens, um conceito básico que mais tarde seria essencial para a detecção de objetos em imagens (BALLARD, 1981). Em 1985, medições locais esparsas de posição e de vetores normais de superfícies (*surface normals*) foram utilizadas para identificar e localizar objetos através das partes que se sobrepõem entre o modelo e os dados medidos pelos sensores, funcionando para dados esparsos (*sparse data*) tanto em duas quanto em três dimensões (GRIMSON; LOZANO-PEREZ, 1985). A utilização de vetores normais de superfícies foi uma constante nos próximos anos, aparecendo em trabalhos posteriores. Lowe (1987) apresentou um sistema de visão computacional capaz de reconhecer objetos tridimensionais a partir de imagens em escala de cinza registradas em pontos de vista desconhecidos.

A geração de amostras para alimentar algoritmos de aprendizagem pode ser feita a partir de dados reais (imagens reais dos objetos), ou a partir de dados sintéticos (dados gerados artificialmente por alguma técnica). Dados sintéticos são aqueles que foram gerados por algoritmos computacionais, ao invés de serem capturados diretamente do mundo real. Em 1991, uma nova abordagem de reconhecimento visual de objetos fazia a representação de um objeto 3D a partir de uma combinação linear de imagens 2D do mesmo, de forma que uma variedade de registros do objeto a partir de diferentes pontos de vista pode ser expressa através de combinações lineares desse pequeno número de imagens 2D. Ou seja, um modelo com poucas imagens 2D seria suficiente para identificar o objeto em uma grande variedade de pontos de vista, aplicando combinação linear nessas imagens 2D (ULLMAN; BASRI, 1991). Já em 1996, um trabalho propôs uma metodologia para a geração de amostras para aprendizagem em reconhecimento de objetos baseado em aparência. Este é um algoritmo que, a partir de uma quantidade pequena de imagens, pode gerar várias amostras do objeto em questão através do uso de técnicas como interpolação, deformação da imagem e a compressão de grandes conjuntos de amostras (MURASE; NAYAR, 1996). Com mais amostras extraídas de um número menor de imagens, a barreira de entrada para este tipo de pesquisa foi enfraquecida, permitindo pesquisas com cada vez menos dados brutos. Essa tendência se intensificou nos anos seguintes, com pesquisas que usam somente dados sintéticos como fonte de treinamento para os algoritmos.

2.2 LEVANTAMENTO BIBLIOGRÁFICO SOBRE A DETECÇÃO OU RECONHECIMENTO DE OBJETOS COM USO DE SENSORES RGB-D

O mapeamento sistemático (MS) (KITCHENHAM; CHARTERS, 2007) é uma ferramenta poderosa para mapear e supervisionar o conhecimento de uma área específica. Para obter uma visão mais ampla, foi realizado um levantamento bibliográfico empregando algumas das técnicas sugeridas por Kitchenham e Charters (2007). O objetivo foi realizar um levantamento bibliográfico de artigos científicos relacionados à detecção ou reconhecimento de objetos em imagens utilizando-se de sensores RGB-D, podendo incluir a estimativa de posição do objeto.

2.2.1 Planejamento

A estratégia de busca foi focada em três portais, considerados pela qualidade dos artigos que apresentam. A busca automática foi feita nas bases *IEEE Xplore*[1], *ACM Digital Library*[2] e *Springer Link*[3].

A *string* de busca precisa ser restritiva para eliminar trabalhos que não tenham relação com o tema desejado, porém não tão restritiva a ponto de eliminar trabalhos que possam ser interessantes. É uma das etapas mais importantes do mapeamento, pois uma *string* de busca mal formada pode trazer trabalhos fora do escopo desejado. A *string* de busca definida foi a seguinte:

- `("object recognition" OR "object detection") AND (pose OR position) AND (pointcloud OR rgbd OR "rgb-d")`

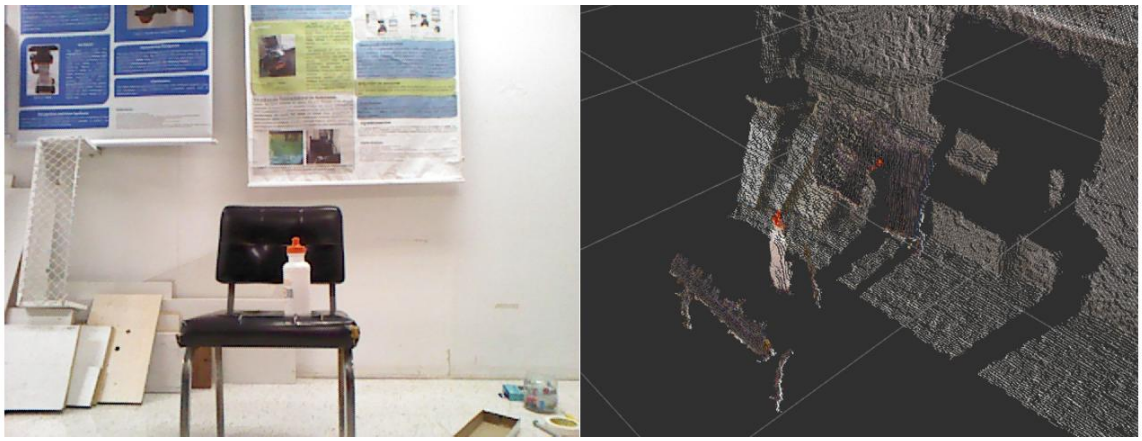
Pode-se perceber que devido à ampla definição do termo `"object recognition"` considerou-se vantajoso adicioná-lo à busca, pois nem todos os trabalhos que tratam de detecção de objetos necessariamente usam o termo `"object detection"`. A inclusão de `pose` e `position` foi essencial para filtrar apenas trabalhos onde há a presença do conceito do posicionamento de objetos. Por fim, os termos `pointcloud`, `rgbd` e `"rgb-d"` eliminou dos resultados trabalhos que não usam o tipo de representação de dados de nuvem de pontos 3D ou que não usem sensores RGB-D.

O formato *pointcloud* (ou "nuvem de pontos", em tradução livre) é comumente associado ao sensor RGB-D e sensores 3D em geral. Este formato representa um conjunto de

[1] *Site* do *IEEE Xplore*: https://ieeexplore.ieee.org/.
[2] *Site* do *ACM Digital Library*: https://dl.acm.org/.
[3] *Site* do *Springer Link*: https://link.springer.com/.

pontos em um ambiente 3D. Como trata-se de uma representação tridimensional, a posição de cada ponto é composta pela sua posição em cada um dos três eixos (`x`, `y` e `z`). Registros de profundidade feitos por sensores RGB-D podem ser armazenados neste formato, que é diferente de uma simples matriz de distâncias (que é o dado bruto captado pelo sensor RGB-D em sua camada de distâncias). A Figura 5 mostra uma imagem RGB capturada por um sensor RGB-D *Kinect* e o *pointcloud* associado visto a partir de outro ângulo.

Figura 5 – Exemplo de imagem registrada com um sensor *Kinect* e o *pointcloud* associado, visto a partir de um outro ângulo.



Fonte: Autoria própria (2022).

A *string* de busca teve que ser adaptada e buscada de forma diferente em cada uma das bases de conhecimento escolhidas devido a diferenças entre os mecanismos de busca de cada portal. O objetivo foi buscar artigos onde exista a ocorrência da *string* de busca no título, *abstract* ou *keywords*. A *Springer Link*, por exemplo, não permitia esse tipo de filtro, de forma que a busca incluiu também ocorrências das palavras no corpo do texto dos documentos.

Um dos critérios de inclusão utilizados foi o ano de publicação do artigo: entre 2008 e 2018. Foram mantidos apenas artigos publicados em *journals* ou conferências em língua inglesa.

2.2.2 Condução

No *IEEE Xplore*, a *string* `((“object recognition” OR “object detection”) AND (pose OR position) AND (pointcloud OR rgbd OR “rgb-d”))` foi buscada no campo “*metadata*”[4]. Filtros aplicados: trabalhos publicados entre 2008 e 2018.

[4] Informações do site sobre o campo *metadata*: “*includes the abstract, index terms, and bibliographic citation data, such as document title, publication title, author, etc.*”.

Na *ACM Digital Library*, a *string* `+("object recognition" "object detection") +(pose position) +(pointcloud rgbd "rgb-d")` foi buscada no campo "*any field*" (que inclui também o corpo do texto). O resultado foi filtrado por publicados entre 2008 e 2018.

No *Springer Link*, não há limitador por título, *abstract* e palavras-chave, portanto a busca teve de ser feita no campo padrão de busca (busca padrão). A *string* `("object recognition" OR "object detection") AND (pose OR position) AND (pointcloud OR rgbd OR "rgb-d")` foi buscada. Filtros aplicados: escritos em inglês, entre 2008 e 2018, com tipo de conteúdo restrito apenas a "*conference paper*" e "*article*".

2.2.3 Aplicação dos critérios de exclusão

Estabelecidos os critérios de exclusão, foram rejeitados os artigos que não estavam dentro do escopo: aqueles que apareceram em duplicidade nos resultados, aqueles que são trabalhos secundários ou *reviews* e aqueles cujo tema principal não seja relacionado com a detecção ou o reconhecimento de objetos com dispositivos RGB-D.

2.2.4 Análise dos Trabalhos

Após a leitura dos trabalhos, foi possível identificar respostas para as perguntas de pesquisa:

Q1 - Quando os estudos foram publicados? O ano de publicação dos estudos seguiu um padrão crescente, de forma que 2018 foi o ano com maior incidência de artigos, seguido por 2017.

Q2 - Onde os estudos foram publicados (qual conferência, revista ou journal)? Foi identificada grande variedade de *journals* e conferências com ocorrências únicas, de forma que apenas uma conferência teve mais de 1 artigo (*Twenty-Seventh AAAI Conference on Artificial Intelligence*). Do total, 24% foram publicados em *journals* e 76% em conferências.

Q3 - Quantas citações o trabalho tem registradas no portal Web of Science? A Tabela 2 relaciona os trabalhos com o registro da quantidade de citações constante no portal Web of Science. Os trabalhos que não foram encontrados no portal Web of Science, ou aqueles cuja quantidade de citações é 0, foram ignorados.

Tabela 2 – Distribuição dos artigos científicos analisados pela quantidade de citações que possuem no portal Web of Science.

Quantidade de Citações	Parcela dos Artigos
1	12%
2 - 4	12%
5 - 10	12%
10 - 20	8%

Fonte: Autoria própria (2022).

Q4 - Qual método foi utilizado na obtenção das imagens? Como o objetivo é buscar trabalhos focados em detecção ou reconhecimento de objetos com o uso de sensores RGB-D, pode-se identificar também quais métodos e dispositivos são utilizados para a obtenção das imagens. Nem todos os trabalhos utilizam dados originados por capturas com dispositivos, no mundo real. Um dos trabalhos em particular usou um *dataset* de imagens 3D geradas por computador (dados sintéticos). O *Kinect* foi o dispositivo de captura mais citado constando em 40% dos trabalhos, seguido do *ASUS Xtion* e *Kinect 2*.

Q5 - Que técnica foi utilizada para reconhecimento ou detecção de objetos na pesquisa? O método utilizado para o reconhecimento ou detecção de objetos é uma das principais questões. Neste tópico, há artigos que citam mais de uma técnica. Alguns artigos têm como tema principal justamente a proposta de novos métodos ou técnicas. Dentre os mais citados, destacam-se as CNNs (como o YOLO), o KNN (*k-nearest neighbours*) e o RANSAC (*RANdom SAmple Consensus*).

Q6 - No caso de trabalhos que envolvem a estimativa de distância ou posição de objetos, qual técnica foi utilizada? O método utilizado para a estimativa de distância ou posição de objetos é outra questão relevante. Nem todos os artigos analisados lidam com este problema em específico. Dentre os que lidam, muitos utilizam os mesmos métodos apresentados na questão Q5, sendo que alguns fazem a detecção do objeto e a estimativa de sua posição de forma conjunta. Dentre as técnicas mais citadas, encontram-se ICP (*iterative closest point*), RANSAC, KNN, LINE-MOD e YOLO.

Q7 - Quais os trabalhos futuros propostos? As propostas de trabalhos futuros envolvem melhorias nos métodos propostos nos artigos, assim como possibilidades de aplicação ainda não exploradas. Pode-se afirmar que a maioria tem relação com a busca por melhorias no tempo de execução ou na precisão dos algoritmos relacionados.

2.2.5 Discussão

Os artigos analisados possuem diferentes aplicações e características. Apesar de ter sido lançado em 2010, o *Kinect* é o sensor mais utilizado nas pesquisas analisadas, provavelmente devido à facilidade de acesso no mercado.

Um fato que foi confirmado foi o crescente uso de CNNs em tarefas de detecção e reconhecimento de objetos, sendo o YOLO um exemplo. Porém, devido ao fato de que as CNNs demandam alto poder de processamento, outras técnicas também ganham atenção quando se faz necessário execução em tempo real (ou pelo menos, próxima disso). A extração da informação de esqueleto das cenas também é recorrente, especialmente nas pesquisas que envolvem a interação com humanos. Destaca-se também a recorrência das técnicas KNN, ICP e RANSAC, que apareceram mais vezes que as demais.

2.2.6 Artigos complementares ao levantamento bibliográfico

O levantamento bibliográfico baseado em técnicas do mapeamento sistemático cobriu o período de 2008 a 2018. Além destes trabalhos, foram identificados alguns artigos posteriores a 2018, assim como artigos que por algum motivo não foram encontrados pela busca realizada no levantamento bibliográfico.

Especificamente sobre a problemática da estimativa da posição e orientação do objeto na cena (*6D pose estimation*), um método batizado de *Augmented Autoenconders* pretende detectar um objeto, sua posição 3D e sua orientação. O método baseia-se numa estrutura de encoder/decoder previamente treinada com imagens RGB dos objetos geradas sinteticamente a partir de modelos 3D com aleatoriedade na rotação, iluminação, *background* e oclusão. O objetivo do encoder/decoder é dar como saída um modelo padronizado de pose do objeto, livre de interferências de oclusão, iluminação, e com um *background* padrão. Estes modelos são usados para o casamento no momento da estimativa de posição em uma cena. Este método foi escolhido como o melhor método de código aberto (*best open source method*) e melhor método rápido (*best fast method*) no "*BOP Challenge 2019 Awards*" (HODAŇ *et al.*, 2019). Apesar da rapidez e *performance*, é necessário ter os modelos 3D dos objetos-alvo com antecedência. Segundo os autores, o treinamento em um dos *datasets* utilizados levou em torno de 3 horas por objeto com execução em GPU (SUNDERMEYER *et al.*, 2018).

Porém o destaque no "*BOP Challenge 2019 Awards*" (HODAŇ *et al.*, 2019), escolhido

como o melhor método de maneira geral (*best overall method*) foi apresentado por Vidal *et al.* (2018). Baseado na abordagem de votação *Point Pair Features* (DROST *et al.*, 2010), algumas alterações e novos passos são propostos, o que fez com que o método tivesse melhor desempenho que soluções consideradas estado da arte em diversos *datasets*. Assim como o método anterior, também é necessário ter os modelos 3D dos objetos-alvo com antecedência, o que é uma tendência quando se trata do *6D pose estimation*.

2.3 FORMATOS DE REPRESENTAÇÃO DE OBJETOS 3D

Pesquisas que envolvem o reconhecimento de objetos 3D utilizam diferentes formatos na representação dos objetos, usando diferentes tipos de dados. Ao longo do tempo, as características dos objetos foram exploradas como fonte de representação dos mesmos, permitindo alta discriminação com baixo consumo de recursos computacionais, o que é desejável em aplicações de tempo real na área da robótica, por exemplo. A busca por características que discriminassem melhor objetos 3D resultou na proposta de diversos descritores, construtores de vetores de características, detectores de pontos-chave (*keypoint detectors*) e métodos para aprendizado de padrões para representação de objetos, suas aparências e formas geométricas. A representação baseada em características pode ser dividida em três grupos: características locais, globais e mapas espaciais. Há também a representação baseada em visões (*view-based representation*), grafos, modelos, subespaço, tensores, além de outras formas menos recorrentes na literatura (CARVALHO; VON WANGENHEIM, 2019).

2.3.1 Características locais

A representação através de características locais é um dos tipos de representação mais recorrentes em pesquisas relacionadas com o reconhecimento e classificação de objetos 3D (CARVALHO; VON WANGENHEIM, 2019). Um desses formatos, o SIFT (*scale-invariant feature transform*), é usado para descrever pontos salientes (pontos-chave) para representar objetos (LOWE, 1999). Suas vantagens são a capacidade de permitir a identificação de objetos com invariância a escala, translação e rotação da imagem, o que fez esta forma de representação ser muito popular até hoje. Outro descritor popular é o SURF, que também é baseado na detecção de pontos-chave, com invariância a escala e rotação (BAY *et al.*, 2008).

O SHOT (*unique signatures of histograms for local surface description*) é um descritor

para casamento de superfícies baseado em histogramas. É um descritor robusto pois é resiliente a variações entre registros do objeto em um ambiente. Diferentes registros do ambiente irão capturar pontos diferentes do objeto desejado. Adicionando a isso o fato de que um sensor pode ter resolução diferente de outro, os pixels da leitura do ambiente que contém o objeto terão representações bem diferentes entre um registro e outro. O SHOT propõe um *frame* de referência 3D local que é único e não-ambíguo, independente das diferenças entre um registro e outro do objeto (TOMBARI *et al.*, 2010).

2.3.2 Características globais

As características globais são capazes de representar objetos 3D utilizando menos dimensões. Mais utilizados nos estudos iniciais de representação de objetos 3D, têm sido gradualmente substituídos pelas características locais, que ganharam mais atenção (CARVALHO; VON WANGENHEIM, 2019). Isso vêm ocorrendo pois os descritores globais não são suficientemente discriminativos para objetos com pequenas diferenças, como casos em que deseja-se discriminar 2 objetos da mesma classe, ou objetos muito similares.

Em 1999, foi apresentado *spin image* (SI) como um descritor global de um objeto 3D para utilização em reconhecimento de objetos 3D em *cluttered scenes*[5]. Envolve um esquema de compressão para imagens que representa o formato do objeto, gerado a partir do *mesh*[6] das superfícies do objeto (JOHNSON; HEBERT, 1999).

Já o VFH (*viewpoint feature histogram*) é um descritor para dados no formato *point-cloud*) que inclui informações da geometria do objeto e do ponto de vista do sensor. Este descritor permite identificar não somente o objeto, porém também estimar sua pose com 6 graus de liberdade (*6-DOF pose estimation*).

2.3.3 Mapas espaciais

Alguns métodos são capazes de capturar e preservar localizações e distâncias entre pontos físicos de um objeto, armazenando isto em um mapas espaciais que guardam informações de um objeto em termos de distância ou outro tipo de informação simbólica (CARVALHO; VON

[5] Cenas onde pode haver a presença de mais de um objeto e registros de objetos podem trazer ruído devido à oclusão.

[6] Um *mesh* 3D é um modelo 3D onde a superfície do objeto é construída a partir de uma rede de polígonos adjacentes.

WANGENHEIM, 2019).

Sen Wang, por exemplo, propõe um trabalho que simplifica a correspondência de formas 3D para um problema de casamento de imagens 2D, com a representação através de mapas. O método permite gerar mapas 2D estáveis, resilientes a mudanças de resolução, oclusão e ruído (WANG *et al.*, 2007).

2.3.4 Outros

Um formato alternativo são as visões. Neste contexto, pode-se definir uma visão como uma imagem representando como algo (um objeto, por exemplo) é visualizado. Ou seja, uma representação baseada em visões é baseada em uma ou mais imagens registradas (ou geradas sinteticamente) a partir de diferentes pontos de vista. Em 2006, por exemplo, um trabalho buscou resolver o problema da fusão de diferentes visões para identificar um objeto 3D, assim como a sugestão de um próximo ponto de vista ótimo. A abordagem consiste em usar aprendizado por reforço para treinamento e seleção de pontos de vista adicionais. Além disso, o mesmo trabalho apresenta uma abordagem para a fusão de várias visões buscando o reconhecimento do objeto (DEINZER *et al.*, 2006).

Outro formato utilizado é o grafo, uma estrutura muito utilizada em diversas aplicações da ciência da computação. Consiste basicamente de um ou mais vértices (pontos) e as conexões entre eles, que são chamadas de arestas. Os grafos são encontrados em diversas aplicações 3D, sendo o reconhecimento de objetos 3D uma delas. Um trabalho de Marini *et al.* (2007) apresentou um método para definição e construção de protótipos de formas 3D para classes de objetos. O formato destes protótipos é essencialmente um grafo com assinaturas estruturais, resumindo a topologia e geometria da forma 3D. Estes protótipos representativos de classes de objetos constituem em descrições estruturais das formas 3D e o tipo de dado é o grafo, com informações adicionais.

"Modelo" é um termo muito amplo, com definições diferentes em diferentes contextos. No campo de reconhecimento de objetos 3D, diversos tipos de modelos usados para representar objetos podem ser encontrados. Por exemplo, modelos geométricos utilizam-se de características geométricas para representar e encontrar correspondências em objetos 3D. Outro modelo é baseado no conceito matemático de variedade, aplicado à aparência e matriz de covariância (*covariance and appearance manifold*). Outros tipos de modelo incluem modelos de superfícies, modelos baseados em distribuição de feições e modelos híbridos, dentre outros (CARVALHO;

VON WANGENHEIM, 2019).

Já o subespaço é um espaço vetorial que está contido dentro de outro espaço vetorial. O espaço vetorial é formado por uma coleção de vetores. Vetores de características dos objetos são gerados, e esses vetores são a base para a definição de subespaços, que podem ser processados em operações específicas de subespaço para a computação de similaridades e pareamento.

Os tensores são objetos matemáticos que podem ser usados para descrever propriedades físicas, sendo uma generalização de escalares e vetores. Mian *et al.* (2006), por exemplo, constrói tensores de objetos 3D a partir de múltiplas imagens para fazer a correspondência entre modelos salvos posteriormente e modelos extraídos de uma cena através de um esquema de votação.

2.4 AQUISIÇÃO DE DADOS

A aquisição de dados pode utilizar diferentes sensores, incluindo câmeras digitais tradicionais (incluindo câmeras USB), câmeras estéreo, câmeras RGB-D e até mesmo sensores táteis. A geração de dados sintéticos também é uma alternativa muito empregada. Os trabalhos científicos podem ser divididos em dois grupos no que diz respeito à fonte de dados: há pesquisas que capturam ou geram os dados necessários, e há outros trabalhos que confiam em conjuntos de dados (*datasets*) registrados em algum outro momento, separadamente do trabalho em questão, sejam eles públicos ou de acesso restrito.

2.5 IDENTIFICAÇÃO/CLASSIFICAÇÃO DO OBJETO

Após a aquisição de dados, opcionalmente ocorre uma etapa de pré-processamento, o que pode envolver filtragem, segmentação e normalização. A filtragem tem por objetivo remover ruído e obter um dado mais adequado. A segmentação é uma etapa que propõe segmentos menores da imagem original, sendo normalmente empregada para sugerir segmentos com maior possibilidade de conterem um objeto. Por fim, a normalização é aplicada em alguns trabalhos onde é necessária alguma invariância nos dados.

Após a etapa opcional do pré-processamento os dados estão prontos para serem consumidos pela técnica que será empregada efetivamente na identificação ou classificação de objetos. Há aqui de maneira geral, duas linhas a serem seguidas(CARVALHO; VON WANGENHEIM, 2019):

- Na primeira linha, os dados de entrada são comparados com dados armazenados previ-

amente em algum tipo de base de dados com modelos ou descritores de objetos. Dessa forma, o método normalmente envolve algum algoritmo de correspondência ou cálculos de similaridade entre os dados de entrada e os dados registrados previamente, como métricas de distância para medir similaridade entre descritores ou objetos.

- Na segunda linha, os trabalhos de pesquisa fazem o treinamento de um classificador com dados. Normalmente isso é feito numa etapa anterior à de reconhecimento do objeto, mas alguns trabalhos também fazem o treinamento *on-line*: o algoritmo aprende na mesma etapa em que o tenta reconhecer objetos, no mesmo fluxo de execução. Este classificador aprende a realizar a classificação de forma autônoma, sem a necessidade de fazer a correspondência dos dados de entrada com uma base de modelos. Como exemplos, existem as redes neurais, incluindo as SVM (*support vector machine*), os modelos baseados em *deep learning* e as CNNs.

2.6 UNIDADES DE PROCESSAMENTO GRÁFICO EMBARCADAS

Com o avanço da indústria de jogos eletrônicos ano após ano, as unidades de processamento gráfico (GPUs) atingem capacidades de processamento e velocidades cada vez maiores. Este poder de processamento chamou a atenção de pesquisadores de outras áreas, que viram uma oportunidade de utilizá-las para outros fins. A disseminação destas GPUs em placas de processamento gráfico fez o preço desses componentes diminuir, facilitando o acesso a eles. Com isso, as GPUs começaram a ser amplamente usadas para outros fins além do processamento gráfico.

GPUs são projetadas pensando no paralelismo de dados. A empresa líder neste segmento, a NVIDIA, facilitou o acesso a estes processadores para propósitos gerais através do projeto CUDA (*Compute Unified Device Architecture*). O CUDA consiste em uma linguagem de programação e um ambiente de desenvolvimento de programas de propósito geral (*general-purpose*) para as GPUs, que começaram ser usadas de forma mais ampla para diferentes fins (PARKER, 2017).

A revolução das GPUs não ficou restrita aos computadores *desktop*, pois a NVIDIA apresentou ao mercado os módulos da família Jetson, capazes de executar em sistema embarcado os processos CUDA voltados para as GPUs. As placas Jetson são baseadas no Tegra, família de processadores da classe SoC (*system on a chip*), que inclui dentro de um único componente

eletrônico CPU, GPU e controlador de memória, dentre outros elementos computacionais. Foi desenvolvido com foco em aplicações embarcadas, tendo tamanho reduzido e baixo consumo de memória. Um exemplo é o Jetson TX1[7].

2.7 MÉTODOS UTILIZADOS PELAS EQUIPES NA ROBOCUP@HOME - OPL

Tendo em vista a grande variedade de métodos para reconhecimento de objetos, é preciso evidenciar quais são mais adequados aos problemas da robótica autônoma em ambientes domésticos, que incluem a estimativa de posição 3D do objeto. Uma forma de identificar métodos mais interessantes é observando aquilo que já é empregado pelas equipes participantes da OPL hoje.

Antes de apresentar os métodos, é fundamental apresentar o ROS (*Robot Operating System*). O ROS é um arcabouço onde processos relacionados à robótica podem ser executados de forma transparente e integrada. Dentre os conceitos relacionados ao ROS, podemos destacar dois: nós e tópicos.

Os nós são processos que manipulam informações. Pode ser inclusive algum processo responsável pela entrada e saída de um dispositivo conectado. Os nós leem e escrevem informações nos tópicos.

Os tópicos, por sua vez, são barramentos tipados através dos quais os nós trocam informações. Um nó que produz dados pode publicar informações em um tópico, enquanto que outro nó pode se subscrever para consumir informações deste tópico. Tópicos podem ter vários nós publicadores e vários nós subscritos.

Os métodos mais utilizados pelas equipes participantes da OPL foram apresentados na Tabela 1, na seção 1.1. Nos próximos parágrafos, cada um deles será apresentado de forma mais detalhada.

O *tensorflow* é uma biblioteca de código aberto para desenvolver e criar modelos de *machine learning*. Como ele não é uma técnica em si, e sim um framework que permite a implementação de diferentes técnicas, cada equipe utiliza o *tensorflow* de uma maneira. Dessa forma, não há um comportamento padrão para essa categoria de *software*.

Similar ao *tensorflow*, o *Caffe* é um framework para *deep learning* desenvolvido pelo grupo *Berkeley AI Research* e contribuidores da comunidade. A biblioteca *OpenCV* permite

[7] Mais detalhes sobre o Jetson TX1 podem ser encontrados no *site* da fabricante NVIDIA: https://developer.nvidia.com/embedded/jetson-tx1.

importar modelos de redes neurais criadas no *Caffe*, e isso permite que diferentes redes executem no *OpenCV*. Cada equipe desenvolve suas redes neurais de uma maneira, não havendo um comportamento padrão para essa categoria de *software*.

O SURF é um descritor baseado em pontos-chave detectados na imagem, que também possui sua própria implementação de um detector. Usa uma técnica chamada de *box filters* para detectar esses pontos-chave e gera estatísticas de gradientes que representam a informação local (BAY *et al.*, 2008).

Similar ao SURF, o SIFT é um descritor que baseia-se em pontos salientes, sendo resiliente a mudanças de escala, rotação e translação. O método apresentado cria um descritor baseado em histogramas locais de orientações de gradiente (LOWE, 1999).

Já o YOLO é um sistema de detecção de objetos em imagens. Consiste basicamente de uma CNN que divide a imagem em regiões e para cada região, prevê uma quantidade de *bounding boxes* candidatas com suas respectivas probabilidades de que contenham um objeto. A rede têm como saída diversas *bounding boxes*, cada uma com uma previsão de classe e probabilidade. Um exemplo de detecção de objetos pelo YOLO pode ser visto na Figura 6. A vantagem do YOLO é que com apenas um ciclo de processamento da imagem na CNN, a saída já provê várias *bounding boxes* com diferentes ocorrências de objetos, cada um com sua probabilidade. Devido a isso, pode ser empregada em situações onde há muitos objetos de interesse em uma mesma imagem. O YOLO, mais especificamente na versão YOLOv3, é uma das CNNs com a melhor *performance* no que diz respeito à detecção de objetos quando testada em diferentes *datasets* (REDMON; FARHADI, 2018).

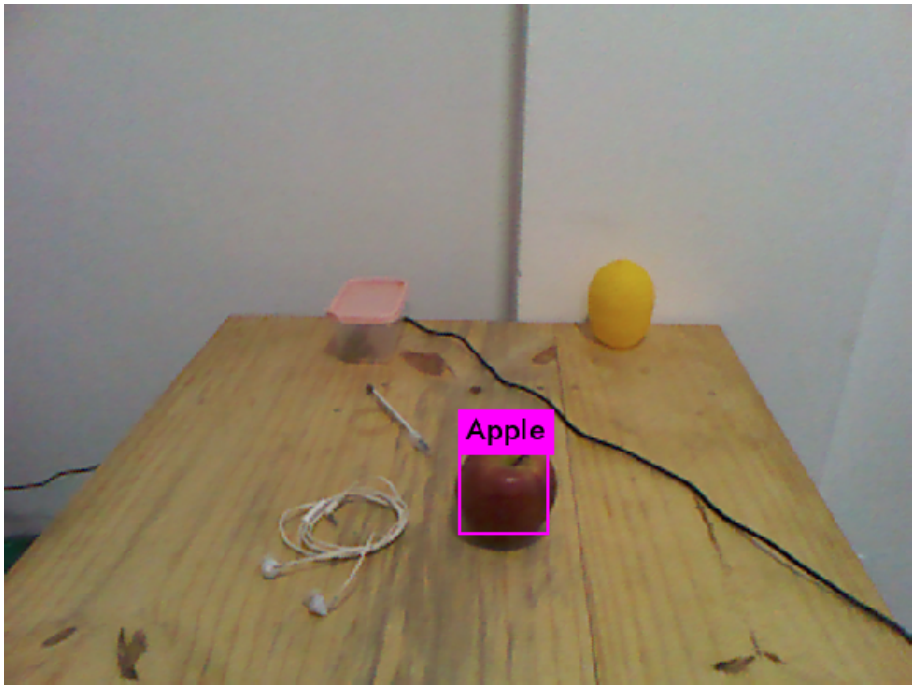
O YOLO se restringe à atuação em imagens RGB, e não busca resolver o problema da estimativa de posição 3D do objeto. Uma vez identificado onde o objeto está na imagem, pode-se utilizar essa informação como entrada para outro método capaz de identificar a posição 3D do objeto.

2.8 PRINCIPAIS PROBLEMAS EM ABERTO

Percebe-se que métodos baseados no formato 3D dos objetos funcionam muito bem para objetos com forma bem definida, como caixas de cereal, porém não são ideais para objetos de forma livre ou objetos com forma variável, como bananas e sacos deformáveis.

A aplicação na RoboCup@Home, assim como em outros cenários, é baseada ao redor do termo que designa o objeto que o robô deve procurar. Há dois casos:

Figura 6 – Detecção de uma maçã na imagem, com o *bounding box* destacado.



Fonte: Autoria própria (2022).

- No primeiro caso, um interlocutor pode ordenar o robô a buscar um objeto de uma classe específica, como no exemplo: "Robô, pegue uma `xícara` da mesa e traga para mim". Nesse caso não está claro qual instância da classe `xícara` em específico o robô deveria buscar;

- No segundo caso, o interlocutor pode pedir uma instância específica de um objeto, como no exemplo: "Robô, pegue a minha escova de dentes azul no banheiro e traga para mim."Nesse caso, é necessário identificar não somente a classe do objeto (`escova de dentes`), mas a instância específica (aquela escova de dentes azul em específico).

Para tarefas relacionadas com o primeiro caso, prender-se a uma forma 3D ou a instâncias específicas é algo que vai restringir a capacidade de identificação de objetos do robô. Por exemplo, um algoritmo de busca baseado fortemente no modelo 3D poderia ter sido treinado com 3 modelos 3D da classe `xícara`, e só considerará que são da classe `xícara`, aqueles objetos que casem com essas 3 formas.

Um algoritmo generalista, como os baseados em redes neurais convolucionais, poderia aprender a identificar a classe `xícara` baseado num treinamento com diversas imagens de xícaras diferentes, em diferentes contextos, com diferentes níveis de iluminação. Tal treinamento, se feito com exemplos o suficiente, poderia permitir ao algoritmo identificar não somente as

xícaras que já foram apresentadas ao algoritmo anteriormente, porém também ocorrências novas semelhantes àquelas vistas durante o treinamento. Essa possibilidade de trabalhar com uma classificação mais maleável e ampla, ao contrário de uma classificação específica e rígida, poderia permitir explorar ainda mais as capacidades do robô nas tarefas de detecção de objetos e estimativa de posição 3D dos mesmos.

Portanto, por mais que a área da visão computacional tenha alcançado avanços significativos nos últimos anos, percebe-se que ainda há problemas que demandam muita pesquisa, como essa capacidade de classificar corretamente novas instâncias de objetos, não vistas anteriormente. Sobre a problemática da detecção de objetos, Matamoros *et al.* (2019b) afirmou que é necessário começar a integrar o reconhecimento de objetos com o planejamento de ações de alto nível de forma que relacionamentos semânticos, espaciais e temporais também possam ser reconhecidos. Um exemplo dado, como um marco distante ainda do estado da arte atual, seria a capacidade do robô extrair informações sobre objetos que mudam com o tempo. Por exemplo, a comida, que ao longo do tempo, com a deterioração, pode mofar, mudar de coloração e até mesmo de cheiro.

Considerando este contexto mais amplo, os robôs ainda estão longe de extrair de objetos a mesma variedade de informações que os humanos conseguem. Outros sensores como microfones, sensores de aromas e sensores capazes de medir o peso do objeto nos manipuladores podem enriquecer ainda mais a capacidade do robô de extrair diferentes informações sobre os objetos manipulados. Porém esses desafios ainda estão mais distantes de serem resolvidos, tendo em vista que problemáticas mais básicas como a estimativa de posição 3D dos objetos, ainda demandam pesquisa e inovação.

3 MATERIAL E MÉTODOS

A revisão bibliográfica, apresentada no Capítulo 2, introduziu os principais conceitos que servem de base para o trabalho atual, além de evidenciar os problemas em aberto. Um resgate histórico das técnicas fundamentais para detecção de objetos e estimativa de posição 3D dos mesmos foi feito. Além disso, pesquisas mais recentes de considerável relevância puderam ser identificadas aplicando técnicas sugeridas por Kitchenham e Charters (2007) e os principais formatos de representação de objetos 3D foram apresentados.

Ainda no Capítulo 2, foram estabelecidas as principais etapas, foram apresentadas as unidades de processamento gráfico embarcadas, e discutidos os principais métodos utilizados pelas equipes na RoboCup@Home.

A estimativa de posição 3D do objeto é uma etapa posterior à detecção do objeto na imagem. A revisão bibliográfica permitiu a escolha de um método para a primeira etapa (a detecção de objetos), o que é detalhado na seção 3.1. Percebeu-se a necessidade de criar um novo método para a estimativa de posição 3D (segunda etapa) após a análise dos problemas em aberto. Este método é apresentado na seção 3.2. As premissas apresentadas nos itens a seguir dizem respeito às duas etapas (detecção e estimativa de posição 3D do objeto):

- Os métodos devem permitir sua aplicação na robótica autônoma de serviços. Para isso, é interessante que permitam sua execução em plataformas embarcadas, com foco em baixo consumo de energia e cuja detecção de objetos ocorra com velocidade de processamento de pelo menos 1 quadro por segundo (*frames per second* ou FPS).

- O foco é que os métodos sejam de fácil aplicação, demandando pouco esforço em sua etapa de preparação. Como mencionado anteriormente, na categoria OPL (Open Platform League) da RoboCup@Home há pouco tempo para extração de dados sobre os objetos que serão utilizados em uma edição da competição, pois os mesmos são apresentados às equipes pouco antes do início das provas. Esta característica também é desejada para a aplicação na solução de problemas da robótica autônoma de serviços em ambientes domésticos reais, onde os robôs muitas vezes entram em contato com objetos desconhecidos nas residências das pessoas.

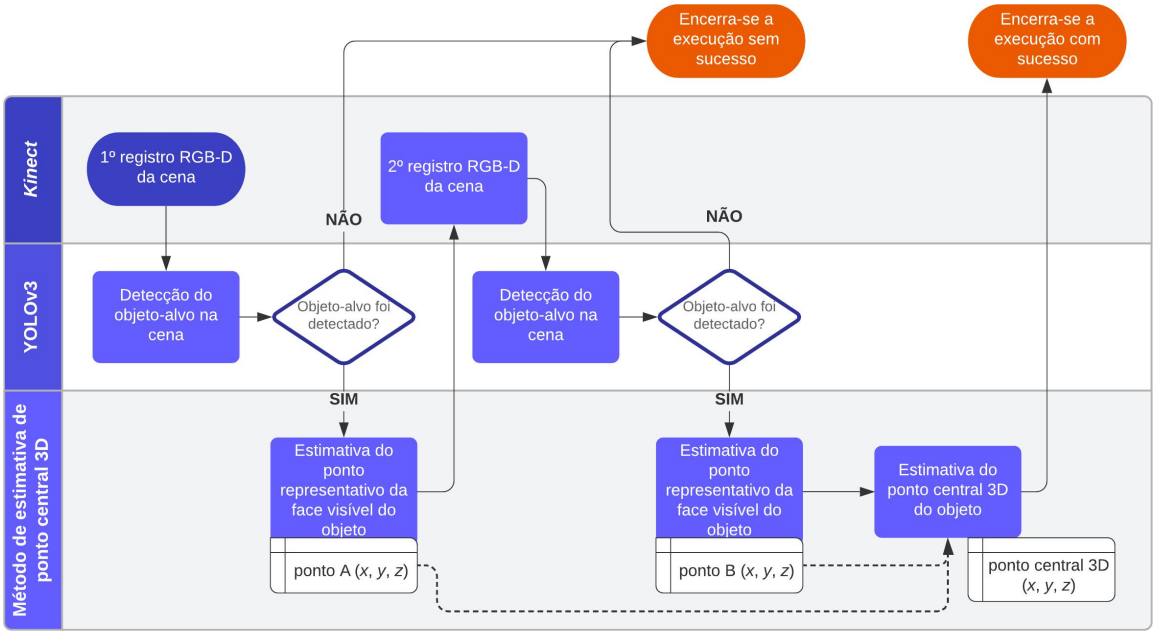
- Os métodos devem permitir o treinamento com classes de objeto personalizadas e é

desejável que sejam capazes de detectar instâncias novas, classificando-as dentro das classes previamente definidas.

Experimentos específicos foram conduzidos a fim de avaliar a precisão das estimativas de posição do método proposto. Tais experimentos são definidos no Capítulo 4. Posteriormente, o ponto central 3D estimado pelo método pode ser usado para alimentar um algoritmo responsável pela estratégia de manipulação do objeto com um braço robótico, por exemplo.

De maneira geral, as etapas necessárias para que o robô faça a estimativa de posição 3D de um objeto utilizando o método proposto neste trabalho estão resumidas em um fluxograma na Figura 7.

Figura 7 – Fluxograma de etapas da estimativa de ponto central 3D de um objeto.



Fonte: Autoria própria (2022).

Os detalhes de cada uma das etapas envolvidas serão apresentados a seguir. A seção 3.1 apresenta as etapas para a realização da detecção do objeto na cena. A seguir, a seção 3.2 apresenta a última etapa: o método proposto neste trabalho, que é responsável pela estimativa do ponto central 3D do objeto. Ao final do capítulo, a seção 3.3 apresenta premissas e regras que precisam ser respeitadas para que o método possa atingir sucesso em sua execução.

3.1 MÉTODO PARA DETECÇÃO DO OBJETO DE INTERESSE NA IMAGEM RGB

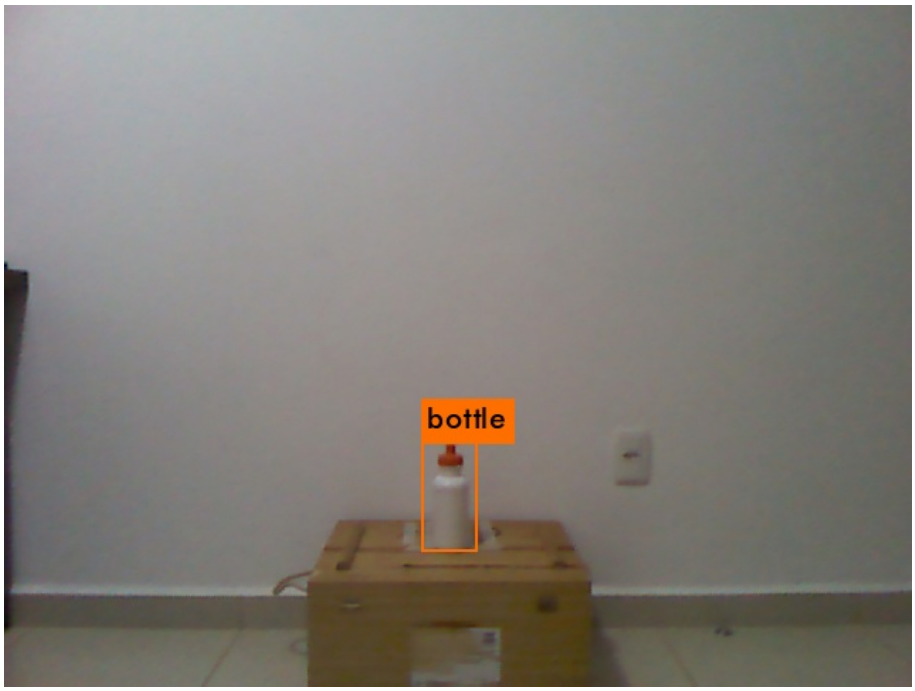
O método de detecção de objetos utilizado é baseado no YOLOv3, e apesar de o robô possuir um sensor *Kinect*, que é um sensor RGB-D, este método executa apenas com a imagem

RGB, ignorando nesta etapa a camada de profundidade. Assim, é fornecida uma imagem RGB de resolução 640 por 480 pixels. Busca-se aproveitar da robustez do YOLOv3, que é um dos métodos do estado da arte na detecção de objetos em imagens RGB.

Ao aproveitar a robustez deste método, já testado em diversos *datasets*, fortalece-se a confiabilidade da detecção das *bounding boxes* dos objetos nas imagens, com a atribuição de uma classe e um grau de confiança (expresso em porcentagem) a cada classificação.

Outra vantagem do YOLOv3 é que, por basear-se numa CNN que deve ser treinada com diversos exemplos diferentes, este método permite detectar objetos novos e classificá-los dentro de uma das classes definidas anteriormente. Por exemplo, por mais que a CNN do YOLOv3 seja treinada com fotos de um conjunto restrito de instâncias diferentes de xícaras, posteriormente em sua etapa de execução o método é capaz de classificar corretamente como xícaras instâncias novas nunca vistas anteriormente, simplesmente por considerar que esta é a classe à qual o objeto mais se aproxima. Esta capacidade de generalizar classes faz do YOLOv3 uma ferramenta poderosa para a detecção de objetos em uma cena. Na Figura 8 é possível ver uma garrafa que nunca foi apresentada antes ao algoritmo sendo detectada corretamente como pertencente à classe `bottle`.

Figura 8 – Exemplo de detecção de uma garrafa.



Fonte: Autoria própria (2022).

O treinamento do modelo da CNN do YOLOv3 é uma etapa crucial. Uma rede treinada com poucos exemplos ou com exemplos ruins pode apresentar uma precisão ruim na classifi-

cação, especialmente tratando-se de instâncias de objetos desconhecidas ou então de objetos em situações ou posições muito diferentes daquelas apresentadas durante o treinamento. Por basear-se numa CNN, as regras de classificação que distinguirão uma classe das demais estão codificadas no modelo da rede neural, com seus pesos definidos. A arquitetura de rede neural do YOLOv3 permite que, com apenas uma execução, possa-se identificar vários objetos na mesma imagem, cada um com uma *bounding box*, uma previsão de classe e um grau de confiança.

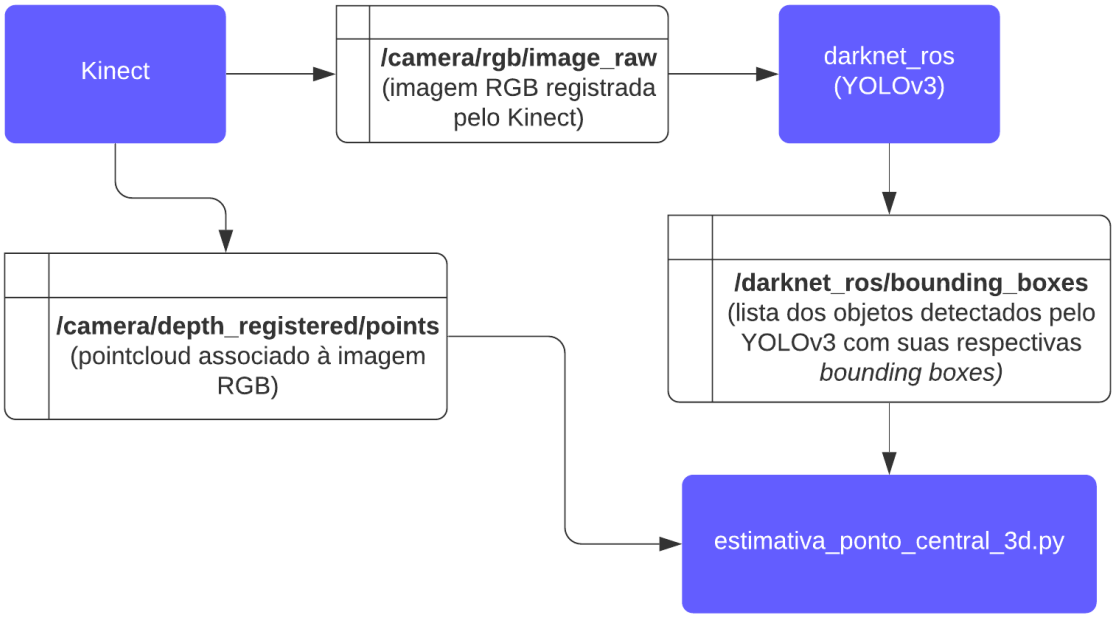
Na RoboCup@Home, a intenção é que o treinamento possa ser realizado no curto espaço de tempo em que os objetos ficam disponíveis para a equipe. O YOLOv3 permite que seu modelo seja treinado com classes customizadas de objetos. Para cada imagem a ser usada na etapa de treinamento, deve ser criado um arquivo texto onde deve anotar-se para cada ocorrência de objeto, as medidas da *bounding box* e a classe do objeto. Existem ferramentas gráficas que podem acelerar este processo, bastando desenhar as *bounding boxes* e informar as classes para que as anotações sejam geradas de forma automática[1].

Para a aplicação final no robô, existe a possibilidade do uso em uma GPU voltada para sistemas embarcados. No laboratório LASER da UTFPR, o robô móvel projetado para as competições da categoria "@Home" no LARC possui uma Jetson TX1, que é uma plataforma embarcada capaz de executar processos em uma arquitetura baseada em GPU (seção 2.6). Processos cuja execução é mais adequada numa GPU encontram no Jetson TX1 uma plataforma para execução embarcada com eficiência computacional e baixo consumo de energia. Um computador PC de arquitetura convencional que não possui placa aceleradora gráfica pode executar o YOLOv3 em um processador comum, porém dificilmente consegue-se uma taxa maior que 1 FPS. Porém o YOLOv3 foi projetado visando sua execução em uma arquitetura baseada em GPU, como é o caso do Jetson TX1. De forma a verificar a viabilidade da execução do YOLOv3 na Jetson TX1, um experimento foi realizado e foi possível obter taxas de processamento de pelo menos 30 FPS. Outros processos que não precisem necessariamente de uma execução em GPU podem continuar sua execução em um PC com processador convencional e a comunicação com a Jetson TX1 pode ser feita através do ROS.

Outra informação relevante é o fluxo de dados existente entre os diferentes elementos da solução. Essa troca de informações ocorre através dos tópicos do ROS. A Figura 9 detalha o fluxo de dados para que finalmente o algoritmo de estimativa de ponto central 3D faça sua estimativa.

[1] Um exemplo de ferramenta que pode ser usada para gerar as anotações necessárias é a YOLO-Label, disponível em https://github.com/developer0hye/Yolo_Label.

Figura 9 – Fluxo de dados entre os nós do ROS envolvidos na estimativa de ponto central 3D de um objeto. Os nós são destacados com fundo azul e os tópicos, com fundo branco.



Fonte: Autoria própria (2022).

Conforme apresentado no Capítulo 2, os nós são processos que manipulam dados e os tópicos são barramentos onde os nós publicam dados para que outros nós possam consumi-los. O nó do ROS responsável pela entrada e saída de dados do *Kinect* escreve informações e dados em vários tópicos. Destacamos aqui 2 tópicos:

- `/camera/rgb/image_raw`: tópico que contém a imagem RGB que foi disponibilizada mais recentemente pelo *Kinect*.
- `/camera/depth_registered/points`: tópico que contém o *pointcloud* associado à imagem RGB. Para cada pixel da imagem RGB, há um ponto 3D associado.

Os dados desses tópicos, por sua vez, são lidos por outros nós do ROS. O YOLOv3 foi integrado ao ROS no projeto `darknet_ros`[2]. Tal projeto permite ao YOLOv3 executar tendo como entrada um tópico de imagem, que pode ser a imagem capturada pelo *Kinect*, por exemplo. O nó `darknet_ros` consome portanto a imagem RGB disponível em `/camera/rgb/image_raw`.

Sobre esta imagem é que ocorre a detecção de objetos. Devido à integração com o ROS, as *bounding boxes* detectadas são registradas em um tópico exclusivo para isso: `/darknet_ros/bounding_boxes`.

[2] *Site* do projeto `darknet_ros` no *github*: https://github.com/leggedrobotics/darknet_ros

Como o ROS trabalha com tipos de dados padronizados numa estrutura de tópicos, uma futura atualização ou mudança do método utilizado para fazer a detecção do objeto é possível de maneira simples. Basta que as *bounding boxes* obtidas como saída deste método estejam no mesmo formato e nos mesmos tópicos ROS utilizados atualmente. De forma análoga, caso exista a necessidade de trocar o sensor RGB-D por outro modelo futuramente, basta que este sensor gere dados no mesmo formato que o *Kinect* gera. Esta é uma das principais vantagens de integrar as soluções de robótica ao ROS: a possibilidade de padronização de tipos de dados e sua organização em tópicos.

Desta forma, o ROS permite que o *Kinect* e o `darknet_ros` executem de forma integrada. O `darknet_ros` consulta a imagem RGB mais recente do *Kinect* em um tópico do ROS. O `darknet_ros` então processa a imagem e informa os objetos detectados e suas *bounding boxes* no tópico de saída específico. Todo este processo é cíclico e contínuo, de forma que as informações mais recentes vão sendo disponibilizadas nos tópicos. É dessa forma que se estabelece o processo de detecção do objeto na imagem.

A última etapa do método, que é a estimativa do ponto central 3D, lê os tópicos `/camera/depth_registered/points` e `/darknet_ros/bounding_boxes`. De posse das informações sobre os objetos detectados, suas *bounding boxes* e do *pointcloud* registrado pelo *Kinect*, este algoritmo é capaz de realizar a estimativa de ponto central 3D do objeto.

3.2 MÉTODO PARA DETECÇÃO DA POSIÇÃO 3D DO OBJETO NA CENA

A detecção da posição 3D do objeto é a etapa final do método, que estimará a posição do ponto central 3D do objeto. O processo envolve a captura de no mínimo dois registros RGB-D da cena, a partir de pontos de vista diferentes. Este processo é definido em quatro etapas:

3.2.1 Etapa 1: Primeiro registro

O robô move-se até um ponto inicial, tendo seu sensor RGB-D *Kinect* apontado para uma superfície onde deseja buscar um objeto da classe-alvo e permanece parado. É importante frisar que o *Kinect* está montado no robô e sua posição e orientação em relação ao robô são estáticas. Devido a isso, a posição do *Kinect* pode ser representada como um ponto 3D fixo no robô. Se a base do robô se move, o *Kinect* e seu ponto 3D movem-se juntos. O ponto onde

o *Kinect* está localizado neste momento é definido como **ponto A**. O robô executa a etapa de detecção de objeto, apresentada na seção 3.1. Aqui, o procedimento é encerrado nos seguintes casos:

- Caso não seja identificado nenhum objeto da classe-alvo o procedimento se encerra, com o registro em *log* de que nenhum objeto pertencente à classe-alvo foi detectado.
- Caso mais de uma ocorrência do objeto seja identificada, o procedimento se encerra, com o registro em *log* de que não é possível identificar o ponto central 3D do objeto quando há mais de uma ocorrência de objeto da classe-alvo na cena.

Caso contrário, isso significa que uma única instância de objeto da classe-alvo foi identificada. Este objeto é identificado a partir deste momento no texto como objeto-alvo. O *Kinect* pode ser integrado ao ROS pelo uso do pacote `freenect_launch`[3], que além de publicar em tópicos as camadas RGB e a camada de profundidade, ainda disponibiliza um *pointcloud* do ambiente produzido a partir do processamento das informações da camada de profundidade. As coordenadas deste *pointcloud* estão definidas em milímetros, no sistema de coordenadas 3D baseado na posição do *Kinect*. Para cada *pixel* da imagem existe um ponto correspondente no *pointcloud*. A partir do segmento que a *bounding box* do objeto-alvo define sobre a imagem original, é possível recortar também o *pointcloud*, observando apenas a região de interesse. Em outras palavras, obtém-se uma região do *pointcloud* correspondente à região de interesse a partir da *bounding box* desenhada pelo YOLOv3 ao redor do objeto-alvo.

O objetivo agora é encontrar o vetor 3D que passa pelo **ponto A** (ponto onde o sensor *Kinect* está localizado) e pelo centro da face visível do objeto-alvo. Para isso é preciso elencar um ponto 3D representativo da superfície do objeto-alvo. Uma alternativa seria, a partir do *pixel* central da *bounding box*, buscar o ponto 3D associado no *pointcloud*. Infelizmente, devido a ruídos nas informações capturadas pelo *Kinect*, essa associação simples e direta pode trazer dados não confiáveis. Outro fato é que, devido à maneira como o *Kinect* funciona, nem todos os *pixels* da imagem possuem um ponto 3D válido associado. Para contornar estes problemas, outras soluções podem ser aplicadas.

Uma delas é usar mais de um ponto 3D da *pointcloud* para obter um ponto 3D derivado. Os *pixels* vizinhos ao *pixel* central da *bounding box* podem ter pontos 3D associados a eles no *pointcloud*. Um ponto definido a partir da mediana dos valores de x, y e z nestes pontos 3D pode

[3] *Site* do pacote `freenect_launch`: http://wiki.ros.org/freenect_launch

ajudar a eliminar ruído nas medições, resultando em um ponto 3D mais confiável. Ao invés de usar uma simples média, a mediana foi selecionada pois ajuda a diminuir a influência que possíveis *outliers* poderiam ter sobre o ponto definido. É importante destacar que este ponto não representa o centro do objeto-alvo. Ele é um ponto 3D representativo do centro da face visível do objeto, derivado dos pontos 3D localizados em sua superfície.

O ponto 3D que representa o centro da face visível do objeto-alvo é, portanto, definido através da mediana dos valores de *x*, *y* e z dos pontos vizinhos ao ponto central. A sequência de comandos executada para obter este ponto 3D é detalhada no Algoritmo 1. É importante destacar o trecho que vai das linhas 4 à 7. Através de experimentação com algumas variações, decidiu-se por definir um subconjunto dos *pixels* da *bounding box*, definido por uma nova *bounding box* menor, na proporção de 1/3 da altura e largura, centralizada no *pixel* central. Os pontos 3D associados a estes *pixels* compõem o conjunto de onde serão extraídas as medianas de *x*, *y* e z para a definição do ponto 3D representativo do centro da face visível do objeto.

Algoritmo 1 – Algoritmo para definição do ponto 3D representativo do centro da face visível do objeto-alvo.

1: $conjunto\_de\_valores\_em\_x = [\,]$
2: $conjunto\_de\_valores\_em\_y = [\,]$
3: $conjunto\_de\_valores\_em\_z = [\,]$
4: $centro\_da\_bounding\_box[x] = (x\_maximo\_da\_boundingbox + x\_minimo\_da\_boundingbox)/2$
5: $centro\_da\_bounding\_box[y] = (y\_maximo\_da\_boundingbox + y\_minimo\_da\_boundingbox)/2$
6: $offset\_x = (x\_maximo\_da\_boundingbox - x\_minimo\_da\_boundingbox)/6$
7: $offset\_y = (y\_maximo\_da\_boundingbox - y\_minimo\_da\_boundingbox)/6$
8: **para** $x = (centro\_da\_bounding\_box[x] - offset\_x)$ **até** $(centro\_da\_bounding\_box[x] + offset\_x)$ **faça**
9: **para** $y = (centro\_da\_bounding\_box[y] - offset\_y)$ **até** $(centro\_da\_bounding\_box[y] + offset\_y)$ **faça**
10: $ponto = pointcloud[x, y]$
11: **se** os valores x, y e z do $ponto$ possuem informação válida **então**
12: $conjunto\_de\_valores\_em\_x.append(ponto[x])$
13: $conjunto\_de\_valores\_em\_y.append(ponto[y])$
14: $conjunto\_de\_valores\_em\_z.append(ponto[z])$
15: **finaliza se**
16: **finaliza para**
17: **finaliza para**
18: $ponto\_na\_superficie[x]$ =mediana do $conjunto\_de\_valores\_em\_x$
19: $ponto\_na\_superficie[y]$ =mediana do $conjunto\_de\_valores\_em\_y$
20: $ponto\_na\_superficie[z]$ =mediana do $conjunto\_de\_valores\_em\_z$
21: **retorna** $ponto\_na\_superficie$

Fonte: Autoria própria (2021).

Dessa forma, tendo o **ponto A** e o ponto 3D da superfície do objeto-alvo (calculado pelo Algoritmo 1), é possível traçar um vetor no espaço 3D que passe pelos dois pontos. A definição deste vetor será essencial para a identificação posterior do ponto central 3D do objeto. Define-se este vetor como **vetor A**. O **ponto A**, o ponto na superfície do objeto-alvo e o **vetor A** são então registrados na memória antes que o procedimento continue na próxima etapa.

3.2.2 Etapa 2: Movimentação do robô até um segundo ponto

Uma vez que as informações levantadas no primeiro registro foram salvas, o robô deve mover-se para um segundo ponto. A intenção é fazer um outro registro do objeto a partir de outro ponto de vista. Obstáculos e a configuração física do ambiente também podem limitar esta movimentação e não existe uma regra rígida para esta etapa, desde que a distância do *Kinect* até o objeto não ultrapasse os limites mínimos e máximos recomendados para seu funcionamento. Essa movimentação pode, portanto, variar entre diferentes execuções. De forma geral, a distância do *Kinect* ao objeto deve estar sempre entre 0,5 m e 5,0 m para garantir que o objeto esteja dentro dos limites operacionais ideais definidos nas especificações do sensor (KHOSHELHAM; ELBERINK, 2012).

3.2.3 Etapa 3: Segundo registro

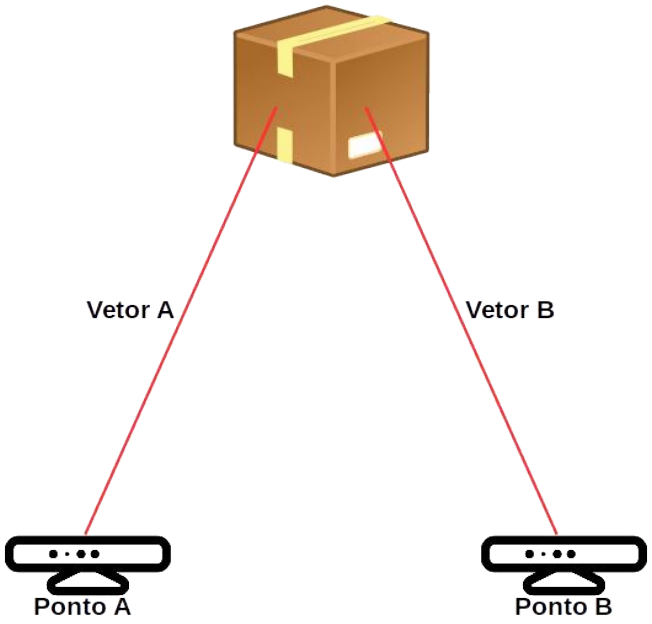
Uma vez que o robô atinja o segundo ponto, deve apontar seu sensor *Kinect* para o ponto 3D identificado na superfície do objeto durante o primeiro registro. Isto é feito para aumentar as possibilidades de que o sensor consiga acomodar o objeto-alvo dentro de seu campo de visão. Em outras palavras, como ainda não se sabe exatamente a localização do objeto-alvo, um ponto em sua superfície pode ser usado como uma referência ao mesmo. Uma vez que esteja voltado para este ponto, o robô permanece parado. O ponto onde o *Kinect* está localizado neste momento é definido como **ponto B**.

A detecção de objetos entra em execução de forma análoga ao que foi feito no primeiro registro e novamente, o procedimento pode ser encerrado caso nenhuma ocorrência de objetos da classe-alvo tenha sido identificada, ou caso mais de uma ocorrência de objetos da classe-alvo tenham sido identificadas.

Caso apenas uma ocorrência do objeto-alvo tenha sido identificada, o procedimento continua. Novamente, o *pointcloud* da cena é segmentado com base na *bounding box* do objeto-alvo fornecido pelo YOLOv3 e o ponto 3D representativo da superfície do objeto-alvo é definido pelo mesmo método aplicado anteriormente (o Algoritmo 1). É traçado um vetor 3D que passa pelos dois pontos 3D: o **ponto B** (localizado no sensor *Kinect*) e o ponto no centro da superfície do objeto-alvo. Este vetor é definido como **vetor B**.

O primeiro registro e o segundo registro são exemplificados na Figura 10.

Figura 10 – Exemplo de cenário onde o robô faz o primeiro registro, identificando o vetor A, e posteriormente, a partir de um segundo ponto de vista, faz o segundo registro, identificando o vetor B.



Fonte: Autoria própria (2022).

3.2.4 Etapa 4: Estimativa do ponto central 3D

O robô e seus diversos pontos de articulação, incluindo o manipulador e o *Kinect* podem possuir sistemas de coordenadas registrados no ROS. O ROS possui uma estrutura de dados denominada `tf`. A `tf` armazena os sistemas de coordenadas e as relações entre eles numa estrutura de árvore, comumente referenciada como "árvore de transformações". O nó pai desta árvore de transformações (o sistema de coordenadas de onde todos os demais derivam) é chamado de "sistema de coordenadas global". Os nós, portanto são sistemas de coordenadas, e entre um nó e outro existe a definição da transformação de um sistema de coordenadas para o outro.

O robô possui um ponto 3D definido virtualmente chamado *footprint*. O *footprint* é um ponto estático localizado ao nível do solo, no centro dos eixos x e y do robô. Por ser estático com relação ao robô, quando o robô se move, o *footprint* move-se com ele. Este ponto é criado com o objetivo de facilitar a identificação da posição e orientação do robô em um ambiente.

O *Kinect* já possui uma posição estática no robô, que por sua vez, possui uma árvore de transformações bem definida no ROS que inclui o *footprint*. Ou seja, sabendo onde o *footprint* do robô está, no sistema de coordenadas global, é possível saber onde o *Kinect* está, pois as

distâncias do *Kinect* até o *footprint* do robô são estáticas, sem variações de posicionamento e orientação entre um e outro.

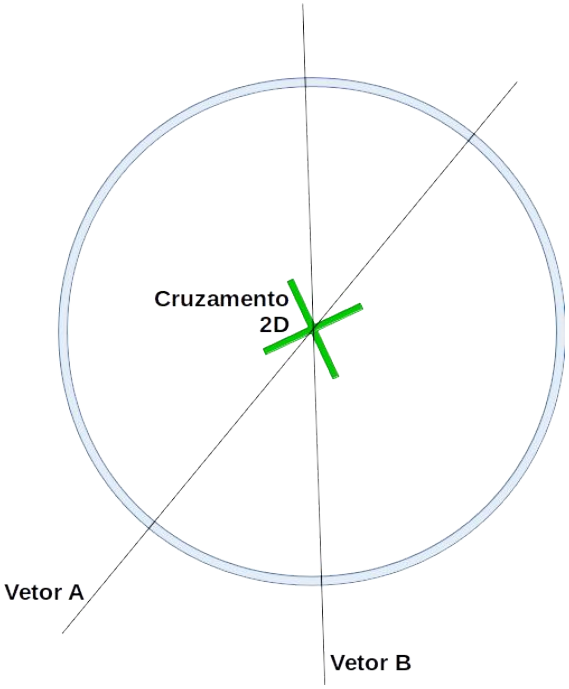
Todos os pontos e vetores levantados devem ser transformados ao sistema de coordenadas global, de forma a permitir que comparações e operações matemáticas possam ser executadas sobre os mesmos.

Neste momento estão registrados o **vetor A**, que passa pelo ponto **ponto A** e a superfície do objeto-alvo registrados no primeiro registro, e o **vetor B**, que passa pelo **ponto B** e a superfície do objeto-alvo registrados no segundo registro. A partir deste momento, pode-se determinar se tais vetores se cruzam em algum ponto do sistema de coordenadas 3D global. O problema é que diferenças entre o primeiro e o segundo registro podem trazer aos vetores variações de angulação no eixo *z*. Idealmente, os dois vetores cruzariam-se nos 3 eixos, porém a variação na definição das *bounding boxes* e a variação na distância entre o objeto e o *Kinect* influenciam a angulação deste vetor no eixo *z*. De forma a reduzir a complexidade do problema para um problema 2D, apenas os eixos *x* e *y* são considerados inicialmente. Em algum ponto do sistema de coordenadas, os dois vetores irão se cruzar em um ponto *(x, y)*. Este ponto é chamado de "cruzamento 2D": este cruzamento define o componente *x* e *y* da estimativa do centro 3D do objeto. Na Figura 11 é apresentado de forma gráfica como é definido o "cruzamento 2D".

É necessário destacar que em raras ocasiões os vetores podem ser paralelos. Quando o robô deslocou-se apenas no eixo *x*, por exemplo, ou quando o robô está observando o objeto a partir de um ponto de vista localizado exatamente na posição oposta do objeto. Essas situações podem gerar vetores paralelos, o que pode gerar infinitos pontos de cruzamento (em caso de sobreposição dos vetores), ou até mesmo nenhum ponto de cruzamento (no caso de vetores paralelos). Isso significa que seria impossível determinar um único ponto de cruzamento. Nestes casos o algoritmo não conseguirá estimar o ponto central 3D e finalizará a execução sem sucesso.

Caso contrário, a execução continua. Pode-se afirmar que, para cada um dos vetores, é possível definir no espaço 3D um único ponto 3D para quaisquer valores de *x* e *y* fornecidos. Para definir o componente *z* do ponto 3D, basta encontrar o valor de *z* no "cruzamento 2D" em ambos os vetores. Ou seja, fixando os valores de *x* e *y* para os mesmos presentes no "cruzamento 2D", encontraremos um valor para *z* no **vetor A** e outro no **vetor B**. Com isso, obtém-se 2 valores de *z*: um localizado no **vetor A** e outro no **vetor B**. Este método considera a média desses 2 valores como o valor de *z* para o ponto central 3D do objeto.

Figura 11 – Ilustração da definição do "cruzamento 2D". Ignorando-se o eixo *z* (altura), a figura demonstra uma visão superior do objeto, de cima para baixo. Os limites do objeto são representados pelo anel azulado. O "cruzamento 2D" é definido no ponto onde os vetores A e B se cruzam.



Fonte: Autoria própria (2022).

3.2.5 Casos com mais de dois registros do objeto

O método permite também a execução com mais de dois registros do objeto. Desta forma, o registro do objeto pode repetir-se *n* vezes, sendo *n* a quantidade desejada de iterações. Para cada iteração adicional, o processo é similar: o robô movimenta sua base, faz um novo registro do objeto e um novo vetor é adicionado à memória. Com mais de dois vetores, o ponto central 3D é definido de forma diferente: pontos 3D intermediários são definidos através do cruzamento de todos os vetores entre si. O ponto central 3D é definido pela média destes pontos 3D intermediários.

Para "*n = 3*", por exemplo, definem-se três pontos 3D intermediários: resultantes das combinações dos vetores (1, 2), (1, 3) e (2, 3). Para "*n = 4*", seis pontos 3D intermediários: resultantes das combinações dos vetores (1, 2), (1, 3), (1, 4), (2, 3), (2, 4) e (3, 4). Para outros valores de *n*, segue-se a mesma lógica.

Em outras palavras, para "*n > 2*" o ponto central 3D *(x, y, z)* tem seus componentes *x, y e z* definidos a partir da média aritmética simples dos valores *x, y e z* dos pontos 3D intermediários.

Algumas premissas básicas para a execução do método são apresentadas na subseção a

seguir. Elas são essenciais para o bom funcionamento do método.

3.3 PREMISSAS

a) Os níveis de iluminação do ambiente devem ser adequados. Busca-se uma iluminação homogênea e constante, sem variações bruscas que possam influenciar a qualidade dos dados do *Kinect*. É indesejado por exemplo, influências de iluminação causadas pelo sol ou condições climáticas externas. Dessa forma, recomenda-se fechar janelas. Sombras de outros elementos internos como pessoas, mesas e até mesmo do próprio robô também podem influenciar na qualidade das imagens e nos dados de profundidade obtidos. Dessa forma, os experimentos foram realizados apenas em ambientes com iluminação constante e de boa qualidade.

b) A *bounding box* deve ser bem ajustada ao objeto, tangenciando suas bordas. *Bounding boxes* que deixam parte do objeto fora de seus limites, ou que sejam maiores que o necessário, estando distantes das bordas do objeto tendem a impactar negativamente no processo da definição do ponto central 3D. Uma etapa que pode influenciar negativamente na definição das *bounding boxes* é a demarcação incorreta das mesmas nas imagens usadas para o treinamento da rede YOLO. Portanto, é essencial que as *bounding boxes* sejam demarcadas com qualidade durante a anotação das imagens usadas na etapa de treinamento do YOLO.

c) Para que a estimativa de posição 3D seja feita, a partir do momento em que o processo se iniciar o objeto não pode ser movido de seu local. Qualquer mudança de localização do objeto levará o processo a resultados inesperados.

d) Não deve haver variação significativa de altura do piso entre os pontos a partir dos quais o robô fará os registros do objeto. Considera-se, para todos os fins, que o piso é plano.

e) O *Kinect* deve estar fixo no robô móvel, de forma a não sofrer variações de localização e orientação. As distâncias entre o *footprint* do robô e o *Kinect* devem ser estáticas, não sofrendo variações entre as medições.

f) Erros de posicionamento do robô podem influenciar no resultado final. O algoritmo de navegação utilizado pelo robô usualmente é o SLAM (*Simultaneous Localization and Mapping)*, que atualiza constantemente a estimativa de posição do robô, além

de atualizar o registro interno que o robô possui do mapa do ambiente. É desejável que este algoritmo esteja produzindo o mínimo de erros de localização possível, estimando com precisão a localização do robô com relação aos pontos de referência no ambiente.

No Capítulo 4 são apresentados os experimentos realizados para avaliar a precisão do método proposto e sua resiliência a erros de posicionamento, pois mesmo algoritmos de alta qualidade podem apresentar erros na estimativa de posicionamento do robô. Dois experimentos são definidos: **experimento 1** e **experimento 2**. A finalidade do **experimento 1** é avaliar a exatidão da estimativa do ponto central 3D. Já o **experimento 2** envolve a introdução de deslocamentos propositais na posição e orientação dos pontos de fixação para avaliar sua resiliência a erros de posicionamento do robô.

4 EXPERIMENTOS, RESULTADOS E DISCUSSÃO

Para a execução dos experimentos o sensor *Kinect* foi fixado em uma haste móvel, de forma a simular sua posição no robô. Como o *Kinect* possui posição e rotação estáticas com relação à haste onde está fixado, sua posição pode ser definida com base na posição da haste. A opção por usar uma haste móvel no lugar do robô físico objetiva eliminar a influência que eventuais erros de odometria do robô poderiam causar no resultado do experimento, assim como demais erros induzidos pelo algoritmo de localização do robô. Como a intenção é avaliar o método de estimativa de ponto central 3D de forma isolada, em vez de confiar na odometria do robô, as posições e as distâncias entre os pontos de fixação serão todas medidas e informadas ao algoritmo.

Para cada execução, um objeto foi colocado em um local previamente definido sobre uma caixa e os pontos de fixação da haste móvel com o *Kinect* foram predefinidos e marcados no chão. Apesar do método permitir a definição de n pontos de fixação, a maioria dos experimentos definiu apenas 2 pontos de fixação.

Para iniciar a execução, devem estar definidos os seguintes pontos: o ponto onde está o objeto, o **ponto de fixação A** e os demais pontos de fixação. Para os experimentos com apenas 2 registros por execução, há apenas mais um ponto de fixação, o **ponto de fixação B**. É necessário medir as distâncias nos eixos x, y e z entre o **ponto de fixação A** (que é o ponto inicial da execução) e o objeto. Essa medição reflete a posição real dos elementos na cena e também é chamada de *ground truth*. Nos experimentos realizados, tais medidas foram registradas manualmente com o auxílio de uma trena a laser e transferidor de grau. A Figura 12 é uma fotografia registrada no ambiente onde os experimentos foram realizados. É importante destacar que a caixa está numa posição pré-definida. Uma vez que o objeto é posicionado, sua posição é medida e permanece inalterada até o fim do experimento.

Conforme exposto na subseção 3.2.4, o ROS possui uma árvore de transformações chamada `tf`. Cada ponto de fixação deve ser representado dentro da árvore da `tf` como um novo sistema de coordenadas. Isso facilita a posterior conversão de todos os dados e informações para o sistema de coordenadas global. Somente com todas as informações registradas em um único sistema de coordenadas é que é possível fazer operações geométricas e comparações entre os valores.

Por ser o ponto onde a haste móvel está ao iniciar o experimento, considera-se que o

Figura 12 – Fotografia do ambiente de realização dos experimentos. Pode-se observar a haste móvel com o Kinect acoplado, o transferidor de grau usado para medir as rotações da haste sobre o eixo *z*, além do objeto-alvo sobre a caixa.

Fonte: Autoria própria (2022).

ponto de fixação A é a origem do sistema de coordenadas global. As posições são expressas em 3 dimensões (*x*, *y*, e *z*), e as movimentações necessárias para realizar a transformação de um sistema de coordenadas para outro são expressas em um formato específico, de acordo com o seguinte padrão definido pelo pacote `tf` do ROS: (deslocamento em x, deslocamento em y, deslocamento em z, rotação sobre o eixo x, rotação sobre o eixo y, rotação sobre o eixo z). Os deslocamentos são expressos em metros e as rotações em graus.

A translação necessária para deslocar-se do **ponto de fixação A** até o ponto central do objeto, expressa no formato (*x*, *y*, *z*), é o *ground truth* do ponto central 3D, que será confrontado com a estimativa fornecida posteriormente pelo método de estimativa de ponto central 3D. Como a origem do sistema de coordenadas global é o **ponto de fixação A**, (*0, 0, 0*), o ponto central 3D é definido pela sua posição 3D neste sistema de coordenadas. Todas as medidas *ground truth* no trabalho atual foram apuradas manualmente no mundo real.

Devem ser medidas e anotadas também as transformações do **ponto de fixação A**, que é a origem do sistema de coordenadas global, até os demais pontos de fixação. Em outras palavras, para cada ponto de fixação a partir de onde é feito um registro do objeto, há uma linha em um arquivo texto (`tf.txt`) contendo a transformação a partir do **ponto de fixação A** até este ponto de fixação. Isso serve de base para que o algoritmo saiba a localização dos pontos de

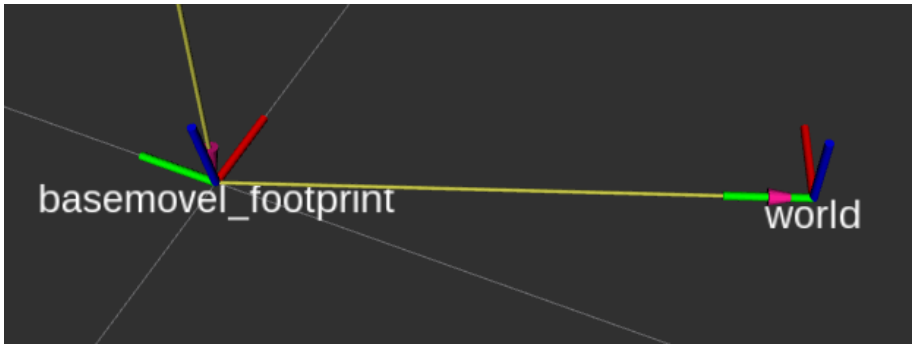
fixação no sistema de coordenadas global e possa registrar corretamente os vetores 3D assim que identificados. Um exemplo de arquivo (`tf.txt`):

- `0,0,0,0,0,0`
- `0,0.4,0,0,0,-20`

A primeira linha é sempre "`0,0,0,0,0,0`". É a linha que informa a transformação necessária para que os pontos no sistema de coordenadas do **ponto de fixação A** sejam transformados para o sistema de coordenadas global. Como o sistema de coordenadas global é definido justamente a partir do **ponto de fixação A**, não há variações na translação nem na orientação entre eles.

A segunda linha detalha uma transformação onde "x = `0`, y = `0.4` e z = `0`". Isso significa que partindo do sistema de coordenadas global para o sistema de coordenadas do **ponto de fixação B**, há um deslocamento de 40 centímetros no eixo y, porém nenhum deslocamento em x ou z. Além disso, a segunda linha indica também que há rotações a serem feitas nesta transformação: "rotação em x de 0º, em y de 0º e em z de -20º". Ou seja, há uma rotação de 20º realizada em torno do eixo z em sentido horário. A disposição geométrica desta transformação pode ser vista na Figura 13: o sistema de coordenadas do **ponto de fixação B** está localizado a 40 cm de deslocamento no eixo y do sistema de coordenadas global (movimento para a esquerda), e há uma rotação de 20º em torno do eixo z em sentido horário.

Figura 13 – Exemplo de transformação entre dois sistemas de coordenadas: o sistema de coordenadas global (representado na figura como `world`) e o sistema de coordenadas do ponto de fixação B (representado na figura como `basemovel_footprint`).

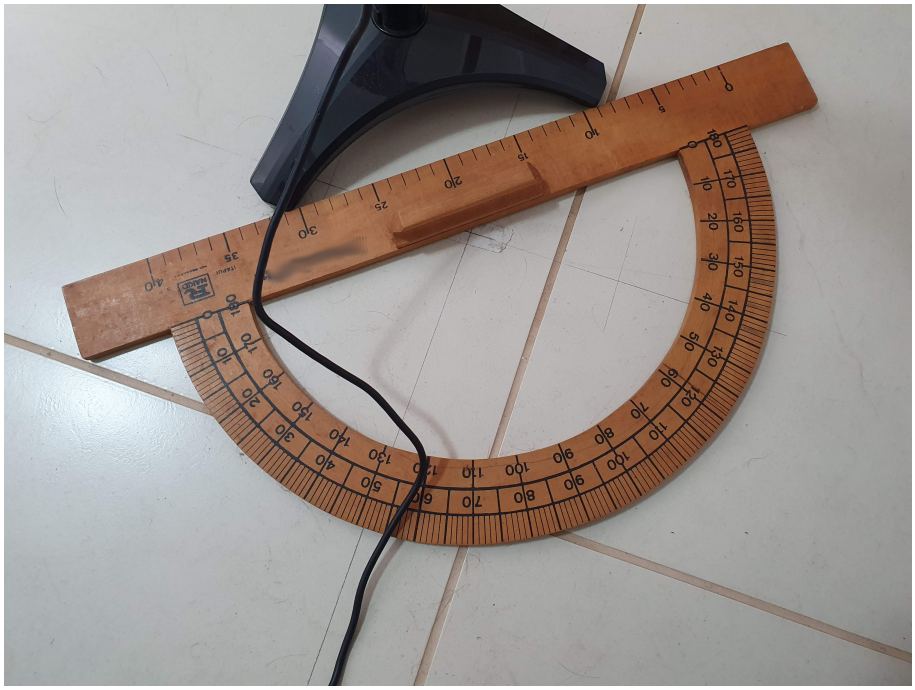


Fonte: Autoria própria (2022).

A rotação da haste móvel sobre o eixo z, quando feita no mundo real, foi medida com o auxílio de um transferidor de grau. Na Figura 14 é possível ver a medição de uma rotação de 40º no sentido anti-horário. É importante destacar que a rotação é feita sobre o ponto central

do transferidor. Este ponto é considerado o *footprint* da haste móvel, justamente para facilitar a medição deste tipo de rotação.

Figura 14 – Rotação da haste móvel de 40º sobre o eixo z no sentido anti-horário. A rotação foi feita com o auxílio do transferidor de grau. Como o transferidor gira sobre seu próprio eixo, o *footprint* da haste móvel foi definido no centro do transferidor.



Fonte: Autoria própria (2022).

Na Figura 15 é possível ver marcações feitas no chão para identificar os pontos de fixação, além de uma linha demarcando o eixo *y* do sistema de coordenadas do **ponto de fixação A**. Essas marcações auxiliaram no momento do posicionamento da haste móvel. Alinhando o centro do transferidor de grau com essas marcações e com a linha do eixo *y*, a haste móvel pôde ser posicionada e rotacionada com precisão.

Como é ideal que os experimentos possam ser repetidos, para cada registro do objeto feito a partir dos pontos de fixação definidos, uma gravação dos tópicos ROS é realizada através da ferramenta `rosbag`[1]. Dessa forma obtém-se uma gravação em tempo real do que estava acontecendo nos tópicos do ROS no momento do registro. Isso permite que posteriormente o experimento possa ser reexecutado. Basta reproduzir o arquivo que foi gravado com a ferramenta `rosbag` que os tópicos ROS são populados com as mesmas informações que o experimento real produziu no momento em que foi gravado.

Os experimentos em geral seguiram portanto as etapas a seguir: O primeiro passo para a execução é colocar a haste móvel no **ponto de fixação A**, com o *Kinect* voltado para o objeto.

[1] Mais informações sobre a ferramenta `rosbag`: http://wiki.ros.org/rosbag.

Figura 15 – Marcações feitas no ambiente de realização dos experimentos destacando os pontos de fixação iniciais de dois experimentos diferentes, identificados com 00 e 01. Além disso, é possível verificar uma linha que foi desenhada para demarcar o eixo *y* do sistema de coordenadas de ambos os pontos.

Fonte: Autoria própria (2022).

A partir daí, o YOLOv3 é executado sobre as imagens RGB obtidas a partir do sensor *Kinect* e a ferramenta `rosbag` é acionada para gravar o conteúdo dos tópicos, gerando um registro deste momento. Após isso, a haste móvel pode ser posicionada no ponto de fixação seguinte e repete-se o processo: o registro é gravado com a ferramenta `rosbag` e é anotada a transformação do *ponto de fixação A* até esta nova posição. Este processo repete-se até que os registros tenham sido feitos a partir de todos os pontos de fixação definidos.

Uma vez que todas as transformações tenham sido anotadas e os arquivos gravados com a ferramenta `rosbag`, pode ser executado o método de estimativa de ponto central 3D sem que o *Kinect* esteja sequer ligado ao computador.

É colocada em execução então a etapa 1 do procedimento de detecção da posição 3D do objeto na cena, conforme definido na subseção 3.2.1. O processo do primeiro registro segue exatamente como apresentado anteriormente, com a única diferença de que em vez de trabalhar com um robô real, o que ocorre é a reprodução das informações que foram gravadas anteriormente com a ferramenta `rosbag`. Ao final da etapa 1 têm-se registrados na memória o **ponto A**, o ponto identificado na superfície do objeto e o **vetor A**.

Para as etapas 2 e 3, que envolveriam a movimentação do robô até o segundo ponto,

basta reproduzir os tópicos registrados com a ferramenta `rosbag`. Os algoritmos consultam a transformação correspondente no arquivo `tf.txt` e aplicam essa transformação sobre as coordenadas de forma a permitir o registro dos novos valores no sistema de coordenadas global.

Com base nas informações levantadas, o algoritmo de estimativa de ponto central 3D pode calcular e retornar um resultado. A transformação de todas as coordenadas para o sistema de coordenadas global permite que o algoritmo estime o ponto central 3D. Através da comparação do *ground truth* com a estimativa de ponto central 3D, pode-se comparar os resultados e calcular o erro na exatidão da estimativa.

Dois experimentos serão detalhados a seguir: **experimento 1** e **experimento 2**. A finalidade do **experimento 1** é avaliar a exatidão da estimativa do método sem nenhuma introdução de erros propositais no processo. De forma a mensurar a influência que erros de localização do robô podem induzir no processo, o **experimento 2** envolve a introdução de deslocamentos propositais na posição e orientação dos pontos de fixação, o que será explicado melhor durante a definição dos experimentos.

4.1 EXPERIMENTO 1: AVALIAÇÃO DA EXATIDÃO DA ESTIMATIVA DO PONTO CENTRAL 3D

A exatidão da estimativa de ponto central 3D foi medida através do primeiro experimento, que envolve a execução do método a partir de diferentes posições com o objetivo de introduzir maior aleatoriedade. Este experimento foi executado sobre 5 objetos diferentes e cada objeto passou pelo experimento 2 vezes, totalizando 10 execuções.

Este experimento foi limitado a duas iterações, o que resultou em apenas dois pontos de observação por execução. Estes pontos são chamados de **ponto de fixação A** e **ponto de fixação B**. A distância entre os pontos de fixação foi variável entre as diferentes execuções. O objetivo foi verificar se existe alguma influência das diferentes variações de posicionamento no resultado final. A distância do *Kinect* ao objeto foi mantida sempre entre 0,5 m e 5,0 m para garantir que o objeto estivesse dentro dos limites operacionais ideais definidos nas especificações do sensor (KHOSHELHAM; ELBERINK, 2012).

A distância entre o **ponto de fixação A** e **ponto de fixação B** variou entre estes cinco níveis pré-definidos: 20 cm, 40 cm, 60 cm, 80 cm e 1 m. Como apresentado anteriormente, foram 2 execuções por objeto, totalizando 10 execuções. Esses 5 níveis de distância foram sorteados entre 10 execuções, de forma que para cada nível de distância existiam 2 execuções.

Cada execução foi identificada por uma letra. O resultado do sorteio e o objeto-alvo de cada execução estão na Tabela 3.

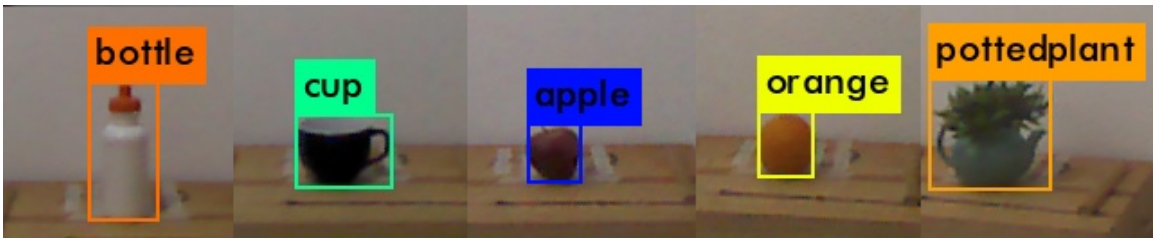
Tabela 3 – Detalhes das execuções do experimento 1.

Identificador	Objeto	Distância entre pontos de fixação
A	garrafa	40 cm
B	garrafa	100 cm
C	xícara azul	80 cm
D	xícara azul	100 cm
E	maçã	80 cm
F	maçã	60 cm
G	laranja	40 cm
H	laranja	20 cm
I	vaso de plantas	20 cm
J	vaso de plantas	60 cm

Fonte: Autoria própria (2022).

O YOLOv3 foi executado com seu modelo padrão, que foi treinado com imagens do *dataset* COCO[2]. É muito improvável que esta instância específica da garrafa utilizada neste experimento estava presente neste dataset. O mesmo se aplica pra as outras classes de objeto: xícara azul, maçã, laranja e vaso de plantas. Apesar disso, o YOLOv3 conseguiu detectar estes objetos. A Figura 16 mostra um exemplo de cada um dos objetos utilizados, com a *bounding box* definida pelo YOLOv3. Essas fotografias dos objetos foram extraídas de imagens capturadas pelo *Kinect*, que passaram pela etapa de detecção de objetos do YOLOv3 e que foram utilizados no experimento 1.

Figura 16 – Exemplos de objetos utilizados no experimento 1. Essa imagem é uma composição feita a partir das imagens RGB originais, que foram registradas pelo *Kinect* e tiveram suas *bounding boxes* definidas pelo YOLOv3.



Fonte: Autoria própria (2022).

Cada execução individual segue um procedimento similar ao apresentado anteriormente, no início do Capítulo 4. Para iniciar a execução, devem estar definidos os seguintes pontos: o ponto onde está o objeto, o **ponto de fixação A** e o **ponto de fixação B**. Como o *Kinect* possui posição e rotação estáticas com relação à haste onde está fixado, sua posição é definida a partir da posição da haste. A cada execução, é necessário medir as distâncias e posições da haste e do

[2] O dataset COCO está disponível em https://cocodataset.org/.

objeto (*ground truth*). É preciso medir as distâncias físicas entre os pontos pré-definidos e as rotações necessárias para informar no arquivo `tf.txt`. Além disso, a posição do ponto central 3D do objeto com relação ao **ponto de fixação A** também deve ser medida. Esse é o *ground truth* do ponto central 3D, que serve de base para a avaliação do método. Uma vez que for calculado o ponto central 3D estimado, poderemos calcular também a distância euclidiana 3D entre este ponto e o ponto real.

O primeiro passo é colocar a haste móvel no **ponto de fixação A**, com o *Kinect* voltado para o objeto. O **ponto de fixação A** é considerado o ponto $(x, y, z) = (0, 0, 0)$, sendo o centro do sistema de coordenadas global das medições. A partir daí, definem-se os demais pontos. Um registro é feito e armazenado através da aplicação da ferramenta `rosbag`.

A execução segue com a movimentação da haste móvel até sua nova posição (que é o **ponto de fixação B**). A translação e rotação entre a origem do sistema de coordenadas global e o **ponto de fixação B** devem ser anotados e informados no arquivo `tf.txt`. Novamente, o registro é gravado e armazenado com a ferramenta `rosbag`.

Com esses registros feitos, é possível executar o algoritmo de estimativa de ponto central 3D em si. Com base nas anotações feitas no arquivo `tf.txt`, o método consegue converter todas as coordenadas para o sistema de coordenadas global, a fim de poder executar operações geométricas entre os dois vetores 3D. O algoritmo gera então a estimativa do ponto central 3D expressa no formato (*x*, *y*, *z*), no sistema de coordenadas do **ponto de fixação A** (o sistema de coordenadas global).

Através da comparação do *ground truth* com a estimativa de ponto central 3D do método apresentado nesta pesquisa, pode-se comparar os resultados e calcular o erro na exatidão da estimativa.

Os resultados, as comparações e discussões do experimento 1 são apresentados a seguir.

4.1.1 Resultados do Experimento 1

Os resultados do experimento 1 são apresentados na Tabela 4. Os resultados são exibidos em metros e foram arredondados na terceira casa decimal (milímetros). A coluna “dist. euclidiana” exibe a distância euclidiana entre o ponto central 3D *ground truth* (medido no mundo real) e o ponto central 3D estimado pelo método. É a distância absoluta entre os dois pontos 3D[3]. Quanto maior é essa distância, maior o erro da estimativa. Perceba que este erro variou entre 0,8 cm no

[3] A distância euclidiana 3D (`d`) é obtida com a aplicação da fórmula: `d = sqrt(dx²+dy²+dz²)`.

caso onde o algoritmo faz a estimativa mais exata (destacado em itálico na Tabela) e 21,5 cm no caso onde o algoritmo teve a menor exatidão (destacado em negrito na Tabela).

Tabela 4 – Resultado das execuções do experimento 1. A coluna "dist. entre P.F." mostra a distância entre os pontos de fixação. GTx, GTy e GTz mostram os componentes *x*, *y* e *z* do *ground truth* do ponto central 3D. ESTx, ESTy e ESTz, por sua vez, mostram os componentes *x*, *y* e *z* do ponto central 3D estimado. A distância euclidiana 3D entre o ponto *ground truth* e o ponto estimado é mostrada na coluna "dist. euclidiana". Unidade de medida: metro(m)

ID	objeto	dist. entre P.F.	GTx	GTy	GTz	ESTx	ESTy	ESTz	dist. euclidiana
A	garrafa	0,4	1,574	0	0,332	1,484	0,007	0,370	0,098
B	garrafa	1,0	1,574	0	0,332	1,574	0,008	0,331	*0,008*
C	xícara azul	0,8	1,574	0	0,275	1,632	-0,030	0,261	0,066
D	xícara azul	1,0	1,574	0	0,275	1,524	-0,021	0,283	0,055
E	maçã	0,8	1,574	0	0,274	1,476	-0,026	0,319	0,111
F	maçã	0,6	1,574	0	0,274	1,446	-0,025	0,330	0,142
G	laranja	0,4	1,581	0	0,274	1,459	0,012	0,307	0,127
H	laranja	0,2	1,581	0	0,274	1,375	0,012	0,333	**0,215**
I	vaso de plantas	0,2	1,574	0	0,327	1,500	-0,021	0,334	0,077
J	vaso de plantas	0,6	1,574	0	0,327	1,519	0,005	0,355	0,062

Fonte: Autoria própria (2022).

Numa análise inicial dos resultados, percebe-se que quanto maior a distância entre os pontos de fixação, menor é a "dist. euclidiana". Isso pode ser percebido analisando a média da distância euclidiana nas execuções quando agrupadas pela distância entre os pontos de fixação. Esta informação pode ser vista na Tabela 5.

Tabela 5 – Média da distância euclidiana agrupando as execuções pela distância entre os pontos de fixação. Unidade de medida: metro(m)

distância entre pontos de fixação	média da distância euclidiana
0,2	0,146
0,4	0,112
0,6	0,102
0,8	0,089
1,0	0,032

Fonte: Autoria própria (2022).

Além disso, percebe-se que para alguns objetos o método conseguiu fazer a estimativa do ponto central 3D com maior exatidão que outros. A Tabela 6 agrupa as execuções por objeto e mostra a média da distância euclidiana para cada objeto.

Tabela 6 – Média da distância euclidiana agrupando as execuções por objeto. Unidade de medida: metro(m)

objeto	média da distância euclidiana
garrafa	0,053
xícara azul	0,061
vaso de plantas	0,070
maçã	0,127
laranja	0,171

Fonte: Autoria própria (2022).

Uma discussão mais abrangente desses resultados, considerando também os resultados do experimento 2, é apresentada na seção 4.3. A seguir, será detalhado o experimento 2.

4.2 EXPERIMENTO 2: AVALIAÇÃO DA INFLUÊNCIA DE ERROS DE POSICIONAMENTO NA ESTIMATIVA DO PONTO CENTRAL 3D

O objetivo deste experimento foi verificar qual a influência que possíveis erros de odometria e posicionamento do robô poderiam ter no método proposto. A ideia foi fazer uma execução com o registro do objeto sem nenhuma alteração. Após isso, fazer novas execuções utilizando este mesmo registro como base, porém com a introdução de erros propositais, simulando o comportamento de um robô móvel cujo algoritmo de estimativa de posicionamento (SLAM) tenha estimado erroneamente sua posição no momento de fazer o segundo registro do objeto. Ao introduzir um erro proposital no processo, é possível mensurar exatamente o impacto que este erro irá ter sobre a estimativa do ponto central 3D.

Neste contexto, foram feitas execuções com pequenas variações no posicionamento físico da haste móvel nos pontos de fixação, como por exemplo: 5 cm a mais ou a menos no eixo x ou eixo y, 10° de rotação adicional no sentido horário ou anti-horário. A ideia é fazer essas alterações no mundo físico, porém mantendo o arquivo `tf.txt` inalterado. Dessa forma, o algoritmo responsável pela execução do método ainda leva em conta os parâmetros anteriores, apesar de terem ocorrido variações no mundo real. O objetivo é verificar quais impactos esses "erros de odometria" introduzidos de forma proposital gerarão no resultado final. Apesar de indesejados, os erros de odometria ocorrem com frequência na robótica móvel. Introduzindo erros controlados no processo, pode-se mensurar o impacto que estes erros causam sobre a estimativa do ponto central 3D.

A introdução de erros propositais no mundo físico é algo que demanda precisão na medição das distâncias e rotações, além de envolver trabalho manual. Para cada novo registro no meio físico, é necessário realizar todo o processo de captura da imagem com o `rosbag`. Uma variação do experimento 2 também foi conduzida em paralelo: ao invés de introduzir erros no mundo real, mantendo o arquivo `tf.txt` inalterado, o que foi feito foi a introdução de erros no arquivo `tf.txt`, mantendo a captura que havia sido feita no mundo real.

Dessa forma há experimentos com a introdução de erros físicos reais (porém sem alterar a posição percebida pelo algoritmo) e experimentos com a introdução de erros na posição percebida pelo algoritmo (porém sem alterar a posição física real). Assim, é possível também

comparar os impactos da introdução de erros em cada um desses meios. Deixando claro que a introdução de erros é sempre realizada no segundo registro do objeto, mantendo o primeiro registro inalterado. A posição da haste móvel no primeiro registro determina também o centro do sistema de coordenadas global. Mantendo este primeiro registro inalterado, podemos mensurar então o impacto da introdução de erros no momento do segundo registro.

Cada execução foi nomeada com um identificador, iniciando pelas letras "KR" e um número sequencial. A lista de execuções, com a informação sobre os erros introduzidos está na Tabela 7. O objeto escolhido foi a garrafa, pois foi o objeto que apresentou melhor precisão no experimento 1. Todas as demais execuções são baseadas na primeira execução (identificada como KR1), que é a única que foi feita sem nenhuma introdução de erros. As rotações, quando informadas, são sempre sobre o eixo z.

Tabela 7 – Detalhes das execuções do experimento 2.

Identificador	Erro introduzido no mundo real	Erro introduzido no arquivo `tf.txt`
KR1	-	-
KR2	rotação 5° horário	-
KR3	-	rotação 5° horário
KR4	rotação 10° anti-horário	-
KR5	-	rotação 10° anti-horário
KR6	2 cm a mais no eixo y	-
KR7	-	2 cm a mais no eixo y
KR8	1 cm a mais no eixo x	-
KR9	-	1 cm a mais no eixo x
KR10	rotação 5° horário e 2 cm a mais no eixo y	-
KR11	-	rotação 5° horário e 2 cm a mais no eixo y
KR12	rotação 2,5° horário	-
KR13	rotação 2,5° anti-horário	-
KR14	-	rotação 2,5° horário
KR15	-	rotação 2,5° anti-horário
KR16	rotação 15° horário	-
KR17	rotação 15° anti-horário	-
KR18	-	rotação 15° horário
KR19	-	rotação 15° anti-horário
KR20	10 cm a mais no eixo y	-
KR21	-	10 cm a mais no eixo y
KR22	10 cm a menos no eixo y	-
KR23	-	10 cm a menos no eixo y

Fonte: Autoria própria (2022).

Todos os registros foram feitos com o mesmo objeto, na mesma posição no mundo real, e com a mesma distância entre os pontos de fixação até que se aplicasse a introdução dos erros propositais no processo. A distância entre os pontos de fixação é sempre de 100 cm, pois foi a distância que apresentou a estimativa média mais exata no experimento 1. O ponto central 3D

ground truth do objeto é (x; y; z) = (1,614; 0; 0,332), em metros. Destaca-se novamente que o ponto (x, y, z) = (0, 0, 0) é o *footprint* da haste móvel, localizado em sua base, mais exatamente no centro da posição onde o transferidor de grau é fixado na mesma. Os resultados do experimento 2 são apresentados a seguir.

4.2.1 Resultados do Experimento 2

Os resultados do experimento 2 são apresentados na Tabela 8. Os resultados são exibidos em metros e foram arredondados na terceira casa decimal (milímetros). A coluna "dist. euclidiana" exibe a distância euclidiana entre o ponto central 3D *ground truth* (medido no mundo real) e o ponto central 3D estimado pelo método. É a distância absoluta entre os dois pontos 3D. Quanto maior é essa distância, maior o erro da estimativa.

Tabela 8 – Resultado das execuções do experimento 2. A coluna "erro real" exibe o erro que foi introduzido no mundo real, quando aplicável. A coluna "erro arquivo" exibe o erro que foi introduzido no arquivo `tf.txt`, quando aplicável. As colunas ESTx, ESTy e ESTz mostram os componentes *x*, *y* e *z* do ponto central 3D estimado. A distância euclidiana 3D entre o ponto *ground truth* e o ponto estimado é mostrada na coluna "dist. euclidiana". Unidade de medida: metro (m)

ID	erro real	erro arquivo	ESTx	ESTy	ESTz	dist. euclidiana
KR1	-	-	-0,069	-0,003	0,027	0,074
KR2	rotação 5° horário	-	-0,207	0,003	0,055	0,214
KR3	-	rotação 5° horário	0,117	-0,002	-0,015	0,118
KR4	rotação 10° anti-horário	-	0,417	0,000	-0,083	0,426
KR5	-	rotação 10° anti-horário	-0,345	-0,005	0,086	0,356
KR6	2 cm a mais no eixo *y*	-	-0,037	-0,003	0,017	0,041
KR7	-	2 cm a mais no eixo *y*	-0,092	-0,003	0,034	0,098
KR8	1 cm a mais no eixo *x*	-	-0,082	-0,003	0,028	0,086
KR9	-	1 cm a mais no eixo *x*	-0,056	-0,003	0,025	0,061
KR10	rotação 5° horário e 2 cm a mais no eixo *y*	-	-0,200	-0,004	0,053	0,207
KR11	-	rotação 5° horário e 2 cm a mais no eixo *y*	0,089	-0,002	-0,007	0,089
KR12	rotação 2,5° horário	-	-0,152	-0,003	0,048	0,160
KR13	rotação 2,5° anti-horário	-	0,029	0,006	0,002	0,030
KR14	-	rotação 2,5° horário	0,019	-0,002	0,007	0,020
KR15	-	rotação 2,5° anti-horário	-0,148	-0,003	0,044	0,155
KR16	rotação 15° horário	-	-0,444	-0,005	0,108	0,457
KR17	rotação 15° anti-horário	-	0,771	0,016	-0,170	0,790
KR18	-	rotação 15° horário	0,685	0,002	-0,149	0,701
KR19	-	rotação 15° anti-horário	-0,457	-0,005	0,108	0,470
KR20	10 cm a mais no eixo *y*	-	0,051	-0,002	-0,011	0,052
KR21	-	10 cm a mais no eixo *y*	-0,177	0,004	0,058	0,187
KR22	10 cm a menos no eixo *y*	-	-0,150	-0,003	0,057	0,160
KR23	-	10 cm a menos no eixo *y*	0,051	-0,002	-0,009	0,052

Fonte: Autoria própria (2022).

Uma análise dos resultados permite identificar que movimentos correlatos no mundo físico e no arquivo produzem um erro semelhante na estimativa do ponto central 3D. Por

exemplo, rotações horárias no mundo real e rotações anti-horárias de mesma proporção no arquivo, produzem efeitos similares na distância euclidiana. Isso pode ser observado nos pares (KR12, KR15), (KR13, KR14), (KR16, KR19) e (KR17, KR18). De forma semelhante, deslocar a haste móvel no mundo real para uma direção no eixo *y* tem efeitos similares a um deslocamento de igual proporção na direção oposta no arquivo. Exemplos disso são os pares (KR20, KR23) e (KR21 e KR22).

Os resultados dos experimentos que envolvem apenas rotação mostram que quando a rotação é de até 5°, os efeitos na estimativa do ponto central 3D estão dentro da faixa de erro apresentada no experimento 1, onde não há erros introduzidos propositalmente no processo. Dentre estas execuções, que envolvem apenas rotação menor que 5°, a que apresentou maior variação na distância euclidiana em comparação com a distância euclidiana da execução KR1 foi a KR2: a diferença entre o resultado dos dois experimentos foi de 14 cm. Já a menor variação foi de 4,4 cm, percebida quando comparando as execuções KR3 com a KR1 e, novamente, ao comparar a KR13 com a KR1.

Por outro lado, alterações angulares maiores que 5 graus produzem um aumento significativo na distância euclidiana entre os pontos. No caso mais extremo, a distância euclidiana entre o ponto real e o estimado foi de 79 cm: foi na execução KR17, que envolvia a aplicação de uma rotação de 15°.

As médias e desvio padrão da distância euclidiana entre o ponto central 3D estimado e o ponto central 3D real nestes dois grupos são as seguintes:

- O grupo de execuções que envolvem apenas a rotação, cuja rotação tenha sido menor ou igual a 5° apresentou como resultado uma distância euclidiana média de 11,6 cm, com desvio padrão de 7,7 cm. Este grupo inclui KR2, KR3, KR12, KR13, KR14 E KR15.

- O grupo de execuções que envolvem apenas a rotação, cuja rotação tenha sido maior que 5° apresentou como resultado uma distância euclidiana média de 53,3 cm, com desvio padrão de 17,1 cm. Este grupo inclui KR4, KR5, KR16, KR17, KR18 E KR19.

Com relação ao deslocamento nos eixos *x* e *y*, o efeito deste deslocamento também é percebido na distância euclidiana. A Tabela 9 mostra os deslocamentos aplicados e a variação efetiva que estes deslocamentos causaram na distância euclidiana com relação à primeira execução que não teve nenhuma introdução de erro proposital, KR1. Os resultados são exibidos em metros e foram arredondados na terceira casa decimal (milímetros).

Tabela 9 – Variação da distância euclidiana com relação à execução KR1, de acordo com variações de posicionamento nos eixos *x* e *y*. Unidade de medida: metro (m)

ID	erro real	erro arquivo	variação da distância euclidiana
KR6	2 cm a mais no eixo *y*	-	-0,033
KR7	-	2 cm a mais no eixo *y*	0,024
KR8	1 cm a mais no eixo *x*	-	0,012
KR9	-	1 cm a mais no eixo *x*	-0,013
KR20	10 cm a mais no eixo *y*	-	-0,022
KR21	-	10 cm a mais no eixo *y*	0,113
KR22	10 cm a menos no eixo *y*	-	0,086
KR23	-	10 cm a menos no eixo *y*	-0,022

Fonte: Autoria própria (2022).

Ainda sobre a Tabela 9, a terceira coluna foi mantida com sinais negativos e positivos justamente pra indicar se houve melhora (diminuiu a distância euclidiana) ou piora (aumentou a distância euclidiana) com relação à execução KR1. É interessante porém, apontar qual é a variação média e o desvio padrão desta amostra. Para isso, considerou-se o valor absoluto da coluna "variação da distância euclidiana", ignorando os sinais negativos da amostra.

- O grupo com até 2 cm de deslocamento apresentou na distância euclidiana uma diferença média com relação ao registro KR1 de 2,1 cm, com desvio padrão de 1 cm. Este grupo inclui KR6, KR7, KR8 E KR9.

- O grupo com 10 cm de deslocamento apresentou na distância euclidiana uma diferença média com relação ao registro KR1 de 6,1 cm, com desvio padrão de 4,6 cm. Este grupo inclui KR20, KR21, KR22 E KR23.

Percebe-se que deslocamentos pequenos de até 2 cm causam efeito similar no ponto 3D estimado. Destes registros, o menos associado a este efeito foi o KR6, onde o deslocamento real foi de 2 cm, porém o efeito na distância euclidiana foi de 3,3 cm. Deslocamentos maiores, como é o caso dos exemplos KR20, KR21, KR22 e KR23, já não sustentam este efeito de forma tão clara. Os registros KR20 e KR23 por exemplo, tiveram a introdução proposital de erros de deslocamento de 10 cm, porém o efeito na estimativa do ponto central 3D foi menor, de 2,2 cm. Sugestões de novos experimentos que podem ajudar a definir com mais clareza o efeito dos deslocamentos, assim como outras sugestões de trabalhos futuros são apresentadas no Capítulo 5.

4.3 DISCUSSÃO SOBRE OS RESULTADOS

As execuções do experimento 1 não tiveram introdução de erros propositais, como as do experimento 2. Há porém uma execução no experimento 2 onde não houve introdução de erros, identificada como KR1. Analisando a exatidão dessas 11 execuções onde não houve introdução de erros, é possível extrair algumas informações (informações numéricas foram arredondadas na primeira casa decimal):

- A média da distância euclidiana entre o ponto central 3D *ground truth* e o ponto central 3D estimado é 9,4 cm.
- O desvio padrão da mesma amostra é 5,4 cm.

O experimento 1 também demonstrou em seus resultados que, conforme a distância entre os pontos de fixação aumenta, diminui a distância euclidiana entre o ponto central 3D estimado e o real.

Outra informação que deve ser explorada com mais cuidado nos trabalhos futuros é a singularidade de cada objeto e o efeito que isso pode ter sobre os resultados. A garrafa foi o objeto que gerou os melhores resultados, com a menor distância euclidiana média dentre os objetos: 5,3 cm. No lado oposto do espectro, a laranja apresentou a pior performance, com distância euclidiana média de 17,1 cm. O Capítulo 5 apresenta sugestões de trabalhos futuros relacionados a essa questão.

Quanto às execuções com a introdução de erros propositais, percebe-se que pequenas variações de posicionamento podem impactar a distância euclidiana com proporções similares. Com relação às variações de 10 cm, o mesmo não pode ser afirmado. Nesses casos, foi impossível perceber relação direta de proporcionalidade entre o erro introduzido e o a variação causada na distância euclidiana. Já a rotação, quando menor que 5°, causa impactos de menor proporção à distância euclidiana, porque percebe-se que a distância euclidiana continua dentro da faixa dos resultados apresentados no experimento 1. Rotações de maior amplitude, como de 10° por exemplo, já trazem impactos maiores ao resultado.

Os resultados desta pesquisa não puderam ser comparados diretamente com outras, por não se tratar de um método baseado em modelos 3D dos objetos. A proposta de trabalhar com objetos independente de seus formatos, e a abstração da posição do objeto para o ponto central 3D limitaram a comparabilidade dos resultados.

5 CONCLUSÃO E TRABALHOS FUTUROS

Neste trabalho é proposto um método de estimativa de ponto central 3D, visando sua aplicação em tarefas de manipulação de objetos por robôs de serviço. Este método envolve a obtenção de registros RGB-D a partir de pelo menos dois pontos de vista diferentes. Isto foi possível através da integração do YOLOv3 com um novo método de estimativa do ponto central 3D de objetos através do ROS. Além disso, este método é aplicável em objetos nunca antes vistos pelo robô, desde que estejam dentro de alguma das classes detectadas pelo YOLOv3. O objetivo é aplicar este método na robótica autônoma de serviços, mais especificamente nos casos presentes na competição RoboCup@Home - OPL, como a manipulação de objetos. Como o método foi projetado para funcionar com objetos de forma 3D desconhecida, a posição do objeto é representada por seu ponto central 3D, ignorando portanto sua orientação.

A revisão bibliográfica permitiu identificar que existia uma necessidade de mais pesquisas envolvendo métodos de estimativa de posição de objetos que atendessem justamente às situações encontradas na competição RoboCup@Home, onde é interessante que o robô consiga estimar a posição 3D de instâncias novas de objetos pertencentes às classes conhecidas. Permitiu também identificar que o YOLOv3 é um método de detecção de objetos do estado da arte capaz de detectar objetos nesta situação. Além disso, é possível executá-lo na Jetson TX1 com taxa de processamento maior que 30 FPS.

O método de estimativa de posição 3D de objetos proposto foi desenvolvido e testado. Este método permite a obtenção de uma estimativa de ponto central 3D de um objeto sem a necessidade de conhecer a forma 3D do mesmo. O erro da estimativa envolve uma distância euclidiana média de 9,4 cm entre o ponto central 3D real e o estimado. Apesar de ainda não ser o ideal para que um braço robótico possa manipular o objeto, percebe-se que o método consegue estimar um ponto central 3D até mesmo para instâncias de objetos não vistas anteriormente pelo robô, como foi o caso por exemplo do vaso de plantas usado no experimento 1. Ainda assim, o método teve bom desempenho em algumas execuções realizadas, sendo que os dois melhores resultados apresentaram-se em registros feitos com 1 metro de distância entre os pontos de fixação: 8 mm e 5,5 cm de distância euclidiana. Nos casos em que seja necessário detectar classes de objetos diferentes daquelas presentes no modelo padrão do YOLOv3, é possível treinar seu próprio modelo, realizando a anotação em um *dataset* customizado de imagens.

Com relação aos experimentos propostos, os mesmos foram realizados com sucesso

e seus resultados foram apresentados. Com relação às execuções sem introdução de erros, foi possível perceber que quanto maior a distância entre os pontos de fixação, menor foi a distância euclidiana entre o ponto central 3D estimado e o ponto central 3D real (*ground truth*). Percebe-se também que alguns objetos tiveram seus pontos centrais 3D estimados com maior exatidão que outros. Características inerentes ao objeto podem ter influenciado os resultados de forma positiva (como possivelmente foi o caso da garrafa) ou negativa (como o caso da laranja).

Já o experimento 2, que envolveu a introdução de erros de posicionamento propositais no momento do segundo registro, ajudou a perceber a resiliência do método a estes erros. Variações de posicionamento de até 2 cm impactaram a distância euclidiana resultante de forma proporcional. Rotações menores e iguais a 5° geraram estimativas com distância euclidiana média de 11,6 cm e desvio padrão de 7,7 cm. Percebeu-se também que rotações mais intensas podem fazer com que o ponto 3D estimado distancie-se muito do ponto real. As execuções com rotação maior que 5° geraram os resultados menos satisfatórios dentre todos os registrados no experimento 2: distância euclidiana média de 53,3 cm, com desvio padrão de 17,1 cm.

5.1 TRABALHOS FUTUROS

Com relação aos resultados dos experimentos, há algumas hipóteses que podem ser levantadas e juntamente a elas, a sugestão de trabalhos futuros que poderiam ajudar a responder algumas perguntas:

- A primeira delas é relacionada com a melhoria da estimativa do ponto central 3D de acordo com o aumento da distância entre os pontos de fixação. Um experimento que pode trazer mais conclusões seria a realização de várias execuções envolvendo apenas o deslocamento no eixo *y*, variando gradativamente a distância entre os pontos de fixação. Como no experimento 1 tivemos variadas formas de deslocamento, há casos em que houve combinação de deslocamento no eixo *x*, *y* e rotação. Isolando o deslocamento apenas ao eixo *y* e evitando a rotação, os efeitos desse tipo de deslocamento no resultado final poderiam ser mensurados com maior clareza.

- É preciso avaliar se a resolução do pointcloud não está limitando a discretização dos dados capturados pelo sensor RGB-D. Isto pode estar impactando negativamente os resultados. Uma forma de comprovar esta hipótese seria a realização de experimentos com o uso de um sensor RBG-D com maior resolução (o *Kinect* tem uma resolução de 640x480 pixels).

O objeto às vezes está numa *bounding box* muito restrita. Com mais pontos no *pointcloud* pode ser que o método tenha informações mais precisas para realizar seus cálculos.

- Outra questão é: qual a influência que o objeto escolhido tem sobre o resultado? Testes com a mesma variação de posicionamento da haste móvel, mudando apenas o objeto, poderiam trazer à tona essa resposta. Com este tipo de experimento, ficaria mais claro se alguma característica intrínseca do objeto pode afetar a estimativa. Alguns objetos possuem superfícies homogêneas e outros não. Mudando apenas o objeto, porém mantendo todas as outras variáveis, poderíamos ter um entendimento melhor sobre qual impacto que as características de cada objeto tem sobre o método. Por exemplo, no experimento 1 a laranja gerou a estimativa com a menor precisão, enquanto que a garrafa gerou o resultado com a maior precisão.

- O trabalho atual foi feito utilizando-se do YOLOv3 na etapa de detecção do objeto. Testar o método substituindo este elemento por outros detectores de objetos pode trazer resultados relevantes.

- Uma ideia que pode ajudar a melhorar a precisão das estimativas é a utilização de sensores adicionais que poderiam ajudar a refinar a estimativa do ponto central 3D, como por exemplo um sensor laser de apenas um ponto. Ajustando seu foco para o centro do objeto, poderíamos ter uma informação adicional que pode ser usada para compôr o cálculo e possívelmente melhorar as estimativas.

- Apesar do fato de que este trabalho não apresenta nenhuma execução onde foram feitos mais de 2 registros do objeto, o algoritmo foi feito pensando em tais situações. Inclusive, testes envolvendo 3 pontos de vista foram realizados para comprovar essa funcionalidade. Para estes casos, com "n" etapas de registro do objeto, alterações no método poderiam considerar pesos diferentes para cada um dos vetores, ou pesos diferentes para cada um dos cruzamentos 2D. Experimentações e testes neste sentido poderiam levar a diferentes abordagens, com resultados diferentes. Como o número de cruzamentos cresce exponencialmente a cada etapa adicionada, seria interessante considerar diferentes abordagens.

Como trata-se da proposição de uma solução nova, certamente há uma variedade de possibilidades de estudo e experimentos que não foram citados. Métodos capazes de estimar a posição de objetos desconhecidos ainda demandam inovação e pesquisa. Quanto trata-se de

objetos cuja forma 3D é desconhecida, ainda há outros fatores a serem considerados. Justamente por não ter o conhecimento prévio da forma 3D dos objetos buscados, a precisão ainda é um fator que impõe-se como um desafio. Outros métodos baseados em modelos 3D recebem mais atenção da comunidade científica. Apesar disso, acredita-se que com inovações na área do *machine learning* e da visão computacional, os métodos que não exigem conhecimento prévio da forma 3D do objeto possam se igualar em precisão àqueles que hoje são o estado da arte no *6D pose estimation*.

REFERÊNCIAS

BALLARD, D. H. Generalizing the hough transform to detect arbitrary shapes. **Pattern Recognition**, v. 13, n. 2, p. 111–122, 1981. ISSN 0031-3203. Disponível em: http://www.sciencedirect.com/science/article/pii/0031320381900091.

BAY, Herbert; ESS, Andreas; TUYTELAARS, Tinne; VAN GOOL, Luc. Speeded-up robust features (SURF). **Computer Vision and Image Understanding**, v. 110, n. 3, p. 346–359, 2008. ISSN 1077-3142. Disponível em: http://www.sciencedirect.com/science/article/pii/S1077314207001555.

BROWNLEE, J. **A gentle introduction to object recognition with deep learning**. 2019. Disponível em: https://machinelearningmastery.com/object-recognition-with-deep-learning/.

CARVALHO, L. E.; VON WANGENHEIM, A. 3D object recognition and classification: a systematic literature review. *In*: **Pattern Analysis and Applications**. Springer London, 2019. v. 22, n. 4, p. 1243–1292. Disponível em: https://doi.org/10.1007/s10044-019-00804-4.

CORKE, Peter. **Robotics, Vision and Control - Fundamental Algorithms In MATLAB® Second, Completely Revised, Extended And Updated Edition**. *[s.n.]*, 2017. v. 75. 693 p. ISSN 15737470. ISBN 978-3-319-54413-7. Disponível em: https://www.springer.com/gp/book/9783319544120.

DEINZER, Frank; DENZLER, Joachim; DERICHS, Christian; NIEMANN, Heinrich. Aspects of optimal viewpoint selection and viewpoint fusion. *In*: NARAYANAN, P. J.; NAYAR, Shree K.; SHUM, Heung-Yeung (Ed.). **Asian Conference on Computer Vision (ACCV 2006), Lecture Notes in Computer Science (LNCS)**. *[S.l.]*: Springer Berlin Heidelberg, 2006. v. 3852, p. 902–912. ISBN 978-3-540-32432-4.

DROST, Bertram; ULRICH, Markus; NAVAB, Nassir; ILIC, Slobodan. Model globally, match locally: efficient and robust 3D object recognition. *In*: **2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition**. *[S.l.: s.n.]*, 2010. p. 998–1005. ISSN 1063-6919.

GASPARETTO, A.; SCALERA, L. A brief history of industrial robotics in the 20th century. **Advances in Historical Studies**, v. 8, p. 24–35, mar. 2019. Disponível em: https://doi.org/10.4236/ahs.2019.81002DOI10.4236/ahs.2019.81002.

GRIMSON, W.; LOZANO-PEREZ, T. Recognition and localization of overlapping parts from sparse data in two and three dimensions. *In*: **Proceedings. 1985 IEEE International Conference on Robotics and Automation**. *[S.l.: s.n.]*, 1985. v. 2, p. 61–66.

HODAŇ, Tomáš; HALUZA, Pavel; OBDRŽÁLEK, Štěpán; MATAS, Jiří; LOURAKIS, Manolis; ZABULIS, Xenophon. T-LESS: An RGB-D dataset for 6D pose estimation of texture-less objects. *In*: **IEEE Winter Conference on Applications of Computer Vision (WACV)**. *[S.l.: s.n.]*, 2017.

HODAŇ, Tomáš; SUNDERMEYER, Martin; BRACHMANN, Eric; DROST, Bertram; MICHEL, Frank; MATAS, Jiří; ROTHER, Carsten. **BOP: benchmark for 6D object pose estimation**. 2019. Disponível em: https://bop.felk.cvut.cz/challenges/.

JOHNSON, Andrew E.; HEBERT, Martial. Using spin images for efficient object recognition in cluttered 3D scenes. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 21, n. 5, p. 433–449, 1999. ISSN 01628828.

KHOSHELHAM, Kourosh; ELBERINK, Sander Oude. Accuracy and resolution of kinect depth data for indoor mapping applications. **Sensors**, MDPI AG, v. 12, n. 2, p. 1437–1454, fev. 2012. ISSN 1424-8220. Disponível em: http://dx.doi.org/10.3390/s120201437.

KITCHENHAM, B. A.; CHARTERS, Stuart. **Guidelines for performing systematic literature reviews in software engineering**. *[S.l.]*, 2007. Technical Report EBSE-2007-01, School of Computer Science and Mathematics, Keele University.

KRIZHEVSKY, Alex; SUTSKEVER, Ilya; HINTON, Geoffrey E. Imagenet classification with deep convolutional neural networks. *In*: **Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1**. Red Hook, NY, USA: Curran Associates Inc., 2012. (NIPS'12), p. 1097–1105.

LOWE, David G. Three-dimensional object recognition from single two-dimensional images. **Artificial Intelligence**, v. 31, n. 3, p. 355–395, 1987. ISSN 0004-3702. Disponível em: http://www.sciencedirect.com/science/article/pii/0004370287900701.

LOWE, David G. Object recognition from local scale-invariant features. **Proceedings of the IEEE International Conference on Computer Vision**, v. 2, p. 1150–1157, 1999.

MARINI, S.; SPAGNUOLO, M.; FALCIDIENO, B. Structural shape prototypes for the automatic classification of 3D objects. **IEEE Computer Graphics and Applications**, v. 27, n. 4, p. 28–37, jul. 2007. ISSN 1558-1756. Disponível em: http://doi.org/10.1109/MCG.2007.89.

MARR, David. **Vision: A Computational Investigation into the Human Representation and Processing of Visual Information**. USA: Henry Holt and Co., Inc., 1982. ISBN 0716715678.

MATAMOROS, Mauricio; RASCON, Caleb; WACHSMUTH, Sven; MORIARTY, Alexander William; KUMMERT, Johannes; HART, Justin; PFEIFFER, Sammy; VAN DER

BRUGH, Matthijs; ST-PIERRE, Maxime. **Robocup@Home 2019: Rules and regulations**. 2019. Disponível em: http://www.robocupathome.org/rules/2019_rulebook.pdf.

MATAMOROS, Mauricio; SEIB, Viktor; PAULUS, Dietrich. Trends, challenges and adopted strategies in RoboCup@Home. *In*: **2019 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)**. *[S.l.: s.n.]*, 2019. p. 1–6.

MIAN, A. S.; BENNAMOUN, M.; OWENS, R. Three-dimensional model-based object recognition and segmentation in cluttered scenes. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 28, n. 10, p. 1584–1601, out. 2006. ISSN 1939-3539. Disponível em: http://doi.org/10.1109/TPAMI.2006.213.

MURASE, H.; NAYAR, S. K. Learning by a generation approach to appearance-based object recognition. *In*: **Proceedings of 13th International Conference on Pattern Recognition**. *[S.l.: s.n.]*, 1996. v. 1, p. 24–29.

PANTHA, N. **Understanding object detection using YOLO**. 2019. Disponível em: https://dzone.com/articles/understanding-object-detection-using-yolo.

PARKER, Michael. Chapter 29 - implementation with gpus. *In*: PARKER, Michael (Ed.). **Digital Signal Processing 101 (Second Edition)**. Second edition. Newnes, 2017. cap. 29, p. 387–393. ISBN 978-0-12-811453-7. Disponível em: https://doi.org/10.1016/B978-0-12-811453-7.00029-9.

PRASSLER, Erwin; KOSUGE, Kazuhiro. Domestic robotics. *In*: SICILIANO, Bruno; KHATIB, Oussama (Ed.). **Springer Handbook of Robotics**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. p. 1253–1281. ISBN 978-3-540-30301-5. Disponível em: https://doi.org/10.1007/978-3-540-30301-5_55.

RAJ, Thinal; HASHIM, Fazida Hanim; HUDDIN, Aqilah Baseri; IBRAHIM, Mohd Faisal; HUSSAIN, Aini. A survey on lidar scanning mechanisms. **Electronics**, MDPI AG, v. 9, n. 5, p. 741, abr. 2020. ISSN 2079-9292. Disponível em: http://dx.doi.org/10.3390/electronics9050741.

REDMON, Joseph; FARHADI, Ali. **YOLOv3: an incremental improvement**. *[S.l.]*, 2018. Disponível em: https://pjreddie.com/media/files/papers/YOLOv3.pdf.

SAXENA, P. **Object detection vs object recognition vs image segmentation**. 2020. Disponível em: https://www.geeksforgeeks.org/object-detection-vs-object-recognition-vs-image-segmentation/.

SHAO, Ling; HAN, Jungong; KOHLI, Pushmeet; ZHANG, Zhengyou. **Computer Vision and Machine Learning with RGB-D Sensors**. *[s.n.]*, 2014. 316 p. ISSN 2191-6586. ISBN 978-3-319-08650-7. Disponível em: https://doi.org/10.1007/978-3-319-08651-4.

SOFTBANK ROBOTICS. **Pepper the humanoid and programmable robot**. 2020. Disponível em: https://www.softbankrobotics.com/emea/en/pepper.

SUNDERMEYER, Martin; MARTON, Zoltan-Csaba; DURNER, Maximilian; BRUCKER, Manuel; TRIEBEL, Rudolph. Implicit 3D orientation learning for 6D object detection from RGB images. *In*: **The European Conference on Computer Vision (ECCV)**. *[S.l.: s.n.]*, 2018.

TOMBARI, Federico; SALTI, Samuele; STEFANO, Luigi Di. Unique signatures of histograms for local surface description. *In*: **European Conference on Computer Vision (ECCV 2010)**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. v. 6313, p. 356–369. ISBN 978-3-642-15558-1.

TOYOTA. **Toyota Shifts Home Helper Robot R&D into High Gear with New Developer Community and Upgraded Prototype**. 2015. Disponível em: https://global.toyota/en/detail/8709541/.

ULLMAN, S.; BASRI, R. Recognition by linear combinations of models. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 13, n. 10, p. 992–1006, out. 1991. ISSN 1939-3539.

VIDAL, Joel; LIN, Chyi-Yeu; LLADÓ, Xavier; MARTÍ, Robert. A method for 6D pose estimation of free-form rigid objects using point pair features on range data. **Sensors**, MDPI AG, v. 18, n. 8, p. 2678, ago. 2018. ISSN 1424-8220. Disponível em: http://dx.doi.org/10.3390/s18082678.

WANG, S.; WANG, Y.; JIN, M.; GU, X. D.; SAMARAS, D. Conformal geometry and its applications on 3D shape matching, recognition, and stitching. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 29, n. 7, p. 1209–1220, jul. 2007. ISSN 1939-3539.

APÊNDICES

APÊNDICE A – CÓDIGO-FONTE DO MÉTODO DE ESTIMATIVA DO PONTO CENTRAL 3D DO OBJETO

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

#### Codigo-fonte do metodo de estimativa de ponto central 3D. ####
#   Codigo feito no Ubuntu 16.04, com o ROS kinetic (versao 1.12.17).
#   Versao do python: 2.7.12.

from __future__ import division

import sys
import math
import numpy as np

import rospy
import roslib
import std_msgs
import ros_numpy
import tf

import sensor_msgs.point_cloud2 as pc2
from sensor_msgs.msg import PointCloud2
from sensor_msgs.msg import Image
from geometry_msgs.msg import PointStamped
from geometry_msgs.msg import Point
from darknet_ros_msgs.msg import BoundingBoxes

# Etapa atual de execucao:
etapa = 0
# Com estas duas listas, temos para cada execucao um ponto no Kinect e um ponto no
    objeto:
# Lista de pontos registrados na superficie do objeto
pontos_na_superficie = list()
# Lista de pontos do Kinect
pontos_no_kinect = list()

def print_boundingbox_info(bbox):
    rospy.loginfo('BoundingBox %s:', bbox.id)
    rospy.loginfo('    probability: %f', bbox.probability)
    rospy.loginfo('    box(x,y,x,y): (%i,%i),(%i,%i)', bbox.xmin, bbox.ymin,
                                                       bbox.xmax, bbox.ymax)
    rospy.loginfo('    class: %s', bbox.Class)

def get_desired_boundingbox(bboxes, classe):
```

```
43      boundingbox_probability = 0
44      boundingbox_desejada = None
45
46      for boundingbox in bboxes.bounding_boxes:
47          if boundingbox.Class == classe:
48              if boundingbox.probability > boundingbox_probability:
49                  boundingbox_probability = boundingbox.probability
50                  boundingbox_desejada = boundingbox
51      return boundingbox_desejada
52
53  # o vetor 1 vai do ponto p0 ao ponto p1.
54  # o vetor 2 vai do ponto p2 ao ponto p3.
55  # formulas adaptadas de https://en.wikipedia.org/wiki/Line%E2%80%93line_intersection
56  def interseccao_2d(p0, p1, p2, p3):
57      a1 = p1.y - p0.y
58      b1 = p0.x - p1.x
59      c1 = a1 * p0.x + b1 * p0.y
60
61      a2 = p3.y - p2.y
62      b2 = p2.x - p3.x
63      c2 = a2 * p2.x + b2 * p2.y
64
65      denominador = a1 * b2 - a2 * b1
66      try:
67          x = (b2 * c1 - b1 * c2) / denominador
68          y = (a1 * c2 - a2 * c1) / denominador
69          return x, y
70      except Exception as e:
71          print("Divisao por 0 no calculo da interseccao_2d. As linhas devem ser
        paralelas.")
72          print(e)
73          return None
74
75  def get_z_from_vetor(x_cruzamento, y_cruzamento, p1, p2):
76      # determinar o slope de cada eixo
77      x_slope = p2.x - p1.x
78      y_slope = p2.y - p1.y
79      z_slope = p2.z - p1.z
80
81      print("x slope: {x}".format(x=x_slope))
82      print("y slope: {x}".format(x=y_slope))
83      print("z slope: {x}".format(x=z_slope))
84
85      # vamos tentar obter o z a partir da variacao do eixo x
86      try:
87          t = (x_cruzamento - p1.x) / x_slope
88          print("encontrado t: {t}".format(t=t))
```

```
89      except Exception as e:
90          print("Nao foi possivel obter z a partir da variacao em x, vamos tentar a
        partir de y. Excecao:")
91          print(e)
92          try:
93              t = (y_cruzamento - p1.y) / y_slope
94              print("encontrado t: {t}".format(t=t))
95          except Exception as e:
96              print("Nao foi possivel obter z a partir da variacao em y. Excecao:")
97              print(e)
98              return None
99
100     z = p1.z + z_slope * t
101     print("z: ", z)
102     return z
103
104 def calcula_ponto_central():
105     global pontos_na_superficie
106     global pontos_no_kinect
107
108     rospy.loginfo("CALCULO DO PONTO CENTRAL 3D:")
109
110     rospy.loginfo("Pontos na superficie acumulados ate agora")
111     rospy.loginfo(pontos_na_superficie)
112     rospy.loginfo("Pontos no kinect acumulados ate agora")
113     rospy.loginfo(pontos_no_kinect)
114
115     cruzamentos_3d_x = list()
116     cruzamentos_3d_y = list()
117     cruzamentos_3d_z = list()
118
119     # Teste de seguranca para garantir que os dois vetores tem o mesmo tamanho
120     if len(pontos_na_superficie) != len(pontos_no_kinect):
121         rospy.loginfo("ERRO: vetores nao tem o mesmo tamanho.")
122         return None
123     try:
124         for pos1 in range(len(pontos_na_superficie)):
125             pos2 = pos1 + 1
126             while pos2 < len(pontos_na_superficie):
127                 print("")
128                 print("Cruzamento dos vetores", pos1, pos2)
129
130                 # dado esse par de vetores, encontre o cruzamento 2D (x, y)
131                 cruzamento_2d = interseccao_2d( pontos_no_kinect[pos1],
132                                                 pontos_na_superficie[pos1],
133                                                 pontos_no_kinect[pos2],
134                                                 pontos_na_superficie[pos2] )
```

```
135
136                 print("Cruzamento 2D: ")
137                 print(cruzamento_2d)
138
139                 if cruzamento_2d is None:
140                     print("Erro no calculo do cruzamento 2D.")
141                 cruzamentos_3d_x.append(cruzamento_2d[0])
142                 cruzamentos_3d_y.append(cruzamento_2d[1])
143                 # deve-se encontrar o Z no ponto do cruzamento.
144                 # primeiramente, o z no vetor 1.
145                 z_cruzamento_1 = get_z_from_vetor(cruzamento_2d[0],
146                     cruzamento_2d[1], pontos_no_kinect[pos1],
147                     pontos_na_superficie[pos1])
148                 z_cruzamento_2 = get_z_from_vetor(cruzamento_2d[0],
149                     cruzamento_2d[1], pontos_no_kinect[pos2],
150                     pontos_na_superficie[pos2])
151                 z_cruzamento = (z_cruzamento_1 + z_cruzamento_2) / 2.0
152                 cruzamentos_3d_z.append(z_cruzamento)
153
154                 print("Cruzamento 3D: ")
155                 print(cruzamento_2d[0], cruzamento_2d[1], z_cruzamento)
156
157                 pos2 = pos2 + 1
158
159         print("")
160         print("Cruzamentos 3D encontrados (1a linha X, 2a linha Y, 3a linha Z): ")
161         print (cruzamentos_3d_x)
162         print (cruzamentos_3d_y)
163         print (cruzamentos_3d_z)
164
165         media_x = np.mean(cruzamentos_3d_x)
166         media_y = np.mean(cruzamentos_3d_y)
167         media_z = np.mean(cruzamentos_3d_z)
168
169         ponto_central_3d = [media_x, media_y, media_z]
170         print("")
171         print("Ponto central 3D: ")
172         print(ponto_central_3d)
173         return ponto_central_3d
174     except Exception as e:
175         print(e)
176         return None
177
178 def processa_bounding_boxes(object_class):
179     global pontos_na_superficie
180     global pontos_no_kinect
181
```

```
182     bboxes = rospy.wait_for_message('/darknet_ros/bounding_boxes', BoundingBoxes)
183     cloud = rospy.wait_for_message('/camera/depth_registered/points', PointCloud2)
184     pointcloud_numpy = ros_numpy.numpify(cloud)
185 
186     boundingbox = get_desired_boundingbox(bboxes, object_class)
187 
188     if boundingbox is None:
189         rospy.loginfo("Nenhuma boundingbox da classe {c} encontrada.".format(c=
    object_class))
190         return
191 
192     rospy.loginfo("Bounding Box da classe {c} encontrada:".format(c=object_class))
193     print_boundingbox_info(boundingbox)
194 
195     centro_da_boundingbox = [ int(round(((boundingbox.xmax + boundingbox.xmin) / 2))
    ), int(round(((boundingbox.ymax + boundingbox.ymin) / 2))) ]
196 
197     # range do "quadrado central", que sera considerado para achar as medianas:
198     # a partir do ponto central, vamos utilizar o "1/6" anterior e posterior
199     range_da_boundingbox_x = boundingbox.xmax - boundingbox.xmin
200     range_da_boundingbox_y = boundingbox.ymax - boundingbox.ymin
201 
202     offset_x = int(range_da_boundingbox_x/6)
203     offset_y = int(range_da_boundingbox_y/6)
204 
205     rospy.loginfo("Centro da bounding box {c}".format(
206         c=centro_da_boundingbox))
207     rospy.loginfo("Range x: {rx}, Range Y: {ry}".format(
208         rx=range_da_boundingbox_x, ry=range_da_boundingbox_y))
209     rospy.loginfo("Offset x: {rx}, Offset Y: {ry}\n".format(
210         rx=offset_x, ry=offset_y))
211 
212     # As coordenadas x e y da boundingbox e do pointcloud numpy estao invertidas.
213     #   o x da bounding box vai ate 640 (colunas) e o y ate 480 (linhas)
214     #   o pointcloud numpy tem dimensoes (x,y) = (480, 640).
215     #   Ou seja, no lugar do x (linhas), deve-se usar o y da boundingbox e vice-
    versa.
216 
217     vx = list()
218     vy = list()
219     vz = list()
220 
221     # Vamos percorrer os pontos que fazem parte do "quadrado central".
222     #   O +1 e pra corrigir e manter o ponto central no meio, capturando a mesma
223     #   distancia em pixels para frente e para tras:
224     for x in range(centro_da_boundingbox[1]-offset_y, centro_da_boundingbox[1]+
    offset_y+1):
```

```
225         for y in range(centro_da_boundingbox[0]-offset_x, centro_da_boundingbox[0]+
    offset_x+1):
226             # se x, y e z sao numeros validos
227             if (not ((math.isnan(pointcloud_numpy[x, y]['x'])) or
228                 (math.isnan(pointcloud_numpy[x, y]['y'])) or
229                 (math.isnan(pointcloud_numpy[x, y]['z'])))):
230                 vx.append(pointcloud_numpy[x, y]['x'])
231                 vy.append(pointcloud_numpy[x, y]['y'])
232                 vz.append(pointcloud_numpy[x, y]['z'])
233 
234     # O ponto na superficie do objeto e a mediana dos valores encontrados em x, y e
    z:
235     ponto_na_superficie = [np.median(vx), np.median(vy), np.median(vz)]
236 
237     print ("O ponto 3D representativo da face visivel do objeto, definido pelas"
238         +"medianas dos valores em x, y e z, e: ({x}, {y}, {z})\n".format(
239         x=ponto_na_superficie[0], y=ponto_na_superficie[1],
240         z=ponto_na_superficie[2]))
241 
242     # Agora e preciso transformar todos os pontos para o frame global 'world'.
243     #   FRAME origem: camera_rgb_optical_frame
244     #   FRAME destino: world
245 
246     # No arquivo rgbd_launch/basemovel_kinect_frames.launch e que e definida a
247     #   transformacao do frame do kinect para o frame basemovel_footprint, conforme
248     #   o exemplo:
249     #   <node pkg="tf2_ros" type="static_transform_publisher" name="$(arg camera)
    _basemovel_link"
250     #   args="-0.1 0 1 0 0 0 /basemovel_footprint $(arg tf_prefix)/$(arg camera)
    _link" />
251 
252     tfListener = tf.TransformListener()
253 
254     ponto_objeto = PointStamped()
255     ponto_objeto.header.frame_id = "camera_rgb_optical_frame"
256     ponto_objeto.header.stamp =rospy.Time(0)
257     ponto_objeto.point = Point(ponto_na_superficie[0], ponto_na_superficie[1],
    ponto_na_superficie[2])
258 
259     ponto_kinect = PointStamped()
260     ponto_kinect.header.frame_id = "camera_rgb_optical_frame"
261     ponto_kinect.header.stamp =rospy.Time(0)
262     ponto_kinect.point = Point(0, 0, 0)
263 
264     tfListener.waitForTransform("/camera_rgb_optical_frame", "/world",
265         rospy.Time(0), rospy.Duration(4.0))
266     ponto_objeto_global = tfListener.transformPoint("world", ponto_objeto)
```

```
267     ponto_kinect_global = tfListener.transformPoint("world", ponto_kinect)
268 
269     rospy.loginfo("Ponto na superficie do objeto:")
270     rospy.loginfo(ponto_objeto_global)
271     print("")
272     rospy.loginfo("Ponto do kinect (ponto (0,0,0) do 'camera_rgb_optical_frame'):")
273     rospy.loginfo(ponto_kinect_global)
274     print("")
275 
276     # de posse dos 2 pontos no frame 'world', podemos salva-los para calcular o
277     #   ponto central 3D
278     pontos_na_superficie.append(ponto_objeto_global.point)
279     pontos_no_kinect.append(ponto_kinect_global.point)
280 
281 def run_metodo_loop():
282     global pontos_na_superficie
283     global pontos_no_kinect
284     global etapa
285 
286     # primeiro registro
287     etapa = 0
288     # A etapa atual e publicada no topico /etapa. Outro no (
289     #   basemovel_tf2_broadcaster.py) fica monitorando este topico. Quando ha
290     #   mudanca na etapa, ele e responsavel por ler a linha correspondente do
291     #   arquivo tf.txt e publicar as mudancas correspondentes na arvore de
292     #   transformacoes do ROS (topico /tf)
293     etapa_publisher = rospy.Publisher("/etapa", std_msgs.msg.Int8, queue_size=1)
294 
295     rospy.init_node('estimativa_ponto_central_3d')
296     rate = rospy.Rate(10) # 10hz
297     while not rospy.is_shutdown():
298         print("Etapa %i: deseja continuar? (S para sim e N para nao)" % etapa)
299         s = sys.stdin.readline()
300         if (s.strip().lower() == 's'):
301 
302             etapa_publisher.publish(std_msgs.msg.Int8(etapa))
303 
304             # aqui e informada a classe do objeto que se esta buscando
305             processa_bounding_boxes("bottle")
306 
307             # a partir da etapa 1 ja temos pelo menos 2 vetores salvos
308             if etapa>0:
309                 ponto_central_3d = calcula_ponto_central()
310                 rospy.loginfo("Ponto central 3D estimado:")
311                 rospy.loginfo(ponto_central_3d)
312 
313             etapa+=1
```

```
314             rate.sleep()
315         else:
316             break
317 
318 if __name__ == '__main__':
319     try:
320         run_metodo_loop()
321     except rospy.ROSInterruptException:
322         pass
```