Projeto de uma rede neural SpaceYNet voltada a obter estimação de pose e profundidade em robôs móveis

Anderson José da Silva Frazão

CENTRO DE INFORMÁTICA
UNIVERSIDADE FEDERAL DA PARAÍBA

João Pessoa, 2023

Anderson José da Silva Frazão

Projeto de uma rede neural SpaceYNet voltada a obter estimação de pose e profundidade em robôs móveis

Monografia apresentada ao curso Engenharia da Computação
do Centro de Informática, da Universidade Federal da Paraíba,
como requisito para a obtenção do grau de Bacharel

Orientador: Tiago P. do Nascimento

Janeiro de 2023

Catalogação na publicação
Seção de Catalogação e Classificação

F848p Frazao, Anderson Jose da Silva.
Projeto de uma rede neural SpaceYNet voltada a obter estimação de pose e profundidade em robôs móveis / Anderson Jose da Silva Frazao. - João Pessoa, 2022.
33 f. : il.

Orientação: Tiago Pereira do Nascimento.
TCC (Graduação) - UFPB/CI.

1. Rede neural. 2. Robôs moveis. 3. Dataset. 4. Depth-Scene. 5. SpaceYNet. I. Nascimento, Tiago Pereira do. II. Título.

UFPB/CI CDU 004.032.26

Elaborado por ROGERIO FERREIRA MARQUES - CRB-15/690

CENTRO DE INFORMÁTICA
UNIVERSIDADE FEDERAL DA PARAÍBA

ATA DE DEFESA PÚBLICA DO TRABALHO DE CONCLUSÃO DE CURSO

Aos **23** dia do mês de **Novembro** de **2022**, em sessão pública, na presença da banca examinadora presidida pelo professor orientador **TIAGO PEREIRA DO NASCIMENTO** e pelos professores **Alisson Vasconcelos de Brito** e **Davi Henrique dos Santos**, o aluno **ANDERSON JOSE DA SILVA FRAZAO** apresentou o trabalho de conclusão de curso intitulado: **Projeto de uma rede neural SpaceYNet voltada a obter estimação de pose e profundidade em robôs móveis** como requisito curricular indispensável para a integralização do Curso de **Engenharia da Computação**.

Após a exposição oral, o candidato foi arguido pelos componentes da banca que reuniram-se reservadamente, e decidiram, **APROVAR** a monografia, com nota **8,8 (dez)**. Divulgando o resultado formalmente ao aluno e demais presentes, eu, na qualidade de Presidente da Banca, lavrei a presente ata que será assinada por mim, pelos demais examinadores e pelo aluno.

gov.br Documento assinado digitalmente
TIAGO PEREIRA DO NASCIMENTO
Data: 23/11/2022 11:51:09-0300
Verifique em https://verificador.iti.br

TIAGO PEREIRA DO NASCIMENTO

Alisson Vasconcelos de Brito

Davi Henrique dos Santos

gov.br Documento assinado digitalmente
ANDERSON JOSE DA SILVA FRAZAO
Data: 23/11/2022 16:36:16-0300
Verifique em https://verificador.iti.br

ANDERSON JOSE DA SILVA FRAZAO

Thaís Gaudencio do Rêgo
Vice-Coordenadora do Curso de Engenharia da Computação

RESUMO

Uma das necessidades principais ao trabalhar com redes neurais é uma boa qualidade de dados de treinamento. Pensando nisso, executamos um projeto de criação de base de dados para treinamento de redes neurais a procura de gerar localização no meio. A base contém imagens RGB e de profundidade, relacionadas com dados de odometria devidamente rotulados. A base então foi usada para o treinamento da rede neural SpaceY-Net, inspirada na GoogLeNet e PoseNet, que tem como objetivo gerara dados de posição do robô no meio através de um framework unificado. A comparação da rede com suas inspirações mostra não só uma adição de funcionalidade, mas também uma melhoria nos resultados, tornando ela uma promissora ideia a ser investida.

Palavras-chave: $Dataset$, $Depth - Scene$, $Pose$, $regression$, $robot$.

ABSTRACT

One of the main needs when working with neural networks is a good quality of training data. With that in mind, we carried out a project to create a database for training neural networks to predict 6-DoF pose. The database contains RGB and depth images, related to properly labeled odometry data. The database was then used to train SpaceYNet, inspired by the GoogLeNet and PoseNet network, which aims to regress robot position data in the environment. Comparing the network with its inspirations shows not only an improvement in functionality, but also a promising idea to be invested in.

Key-words: *Dataset*, *Depth − Scene*, *Pose*, *regression*, *robot*.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS

UFPB - Universidade Federal da Paraíba

CI - Centro de Informática

3D - Three-dimensional

6-DoF - Six Degrees of Freedom

CNN - Convolutional Neural Network

LaSER - Laboratory of Systems Engineering and Robotics

ReLU - Rectified Linear Unit

RGB - Red-Green-Blue

RGB-D - Red-Green-Blue-Depth

SGD - Stochastic Gradient Descent

SIFT - Scale-invariant Feature Transform

SLAM - Simultaneous Localization and Mapping

Conteúdo

1 INTRODUÇÃO

A área da robótica móvel existe desde a época de 1940, onde a tecnologia era usada basicamente para guiar bombas usadas na segunda guerra mundial. Com a invenção do transistor e o avanço das técnicas de criação do mesmo, diminuindo cada vez mais seu tamanho, houve um incentivo nato de ampliar as capacidades dos robôs a cada nova geração.

Com pouco tempo houve então a tentativa de replicar o sistema cerebral humano, criando-se assim neurônios artificiais. Alguns anos depois temos sistemas inteiros constituídos de simulações de neurônios capazes de replicar processamentos similares aos que acontecem no cérebro humano. Através das técnicas de aprendizagem de máquina, hoje temos robôs e sistemas que podem conversar com humanos, identificar imagens e até se guiar sozinho em ambientes.

Sendo assim, o objetivo de gerar soluções unificadas com técnicas de inteligência artificial fica mais próximo da realidade, servindo então de inspiração para o nosso projeto: um framework unificado para solução do problema de localização de um robô móvel em um ambiente.

O projeto cujo este trabalho faz parte pode ser dividido em duas etapas, para melhor compreensão. Uma das etapas envolve a criação e teste da rede neural responsável por guiar o robô, fazendo o mesmo identificar-se no meio. Esta etapa foi feita principalmente pelo então aluno de mestrado Dunfrey Pires Aragão em [1].

A segunda etapa diz respeito a criação de um dataset próprio, representando o ambiente do laboratório LaSER, para que seja possível executar o treinamento da rede em ambiente local. Esta etapa do projeto envolveu o uso de um robô turtlebot, códigos em python para sistema ROS e técnicas de limpeza de imagem.

Neste trabalho será apresentado primariamente à segunda etapa do projeto, feito pelo autor do trabalho. Sendo assim, foi preciso agir de forma colaborativa para obter os melhores resultados.

1.1 Objetivos

Este trabalho tem como objetivo geral mostrar como foi feito o processo de criação de um dataset, contendo informações sobre uma ambiente fechado, com imagens atreladas a informações de localização e angulação de câmera, auxiliando assim o desenvolvimento da rede neural SpaceYNet, porém também servindo para o desenvolvimento de futuras soluções na área.

Para obter a melhor qualidade de informações, foi aplicado o uso de um robô

móvel com sensores de odometria, câmeras com informações de profundidade e imagens RGB, assim como análise de angulação de câmera representado pelo quatérnio Q, W, R, P. Gerando assim informação suficiente para o treinamento de redes neurais procurando aplicar soluções unificadas de localização em ambientes fechados.

Todo o código foi escrito em python para comunicação com o sistema ROS [2], facilitando a manutenção e modificação do código em caso de necessidade.

Por fim este dataset foi usado para treinamento e teste da rede neural SpaceYNet, auxiliando o refinamento da rede para aplicação em ambientes fechados.

1.2 Motivação

Não é novidade que existam sistemas que já fazem uso de redes neurais para se guiar. Alguns sistemas propõem até traçar um caminho utilizando como entrada de dados apenas uma imagem [3]. Porém muitas dessas soluções estão segmentadas, de forma que algumas redes neurais conseguem gerar apenas o caminho, sem conseguir se adaptar, ou então não tem noção de profundidade o que pode causar que o robô esbarre em alterações do ambiente.

A SpaceYNet [1] surge com o objetivo de gerar uma solução simplificada para este problema, unificando em um sistema a capacidade de obter posicionamento, profundidade e angulação da câmera em um ambiente, utilizando uma imagem RGB.

Complementando o projeto SpaceYNet, temos o nosso projeto de criação de um dataset. Dada a importância de um bom dataset no processo de criação de redes neurais, a ideia do projeto é facilitar a implementações dessas soluções, disponibilizando para o futuro uma base de dados robusta. Além do mais fazendo uso de um ambiente local, possibilitando que futuros projetos tenham facilidade de serem implementados no laboratório da própria universidade.

1.3 Metodologia

Para o desenvolvimento e testes da rede SpaceYNet foi criado um dataset com imagens do Laboratório de Sistemas Embarcados e Robótica (LaSER). A criação foi feita com uso de um robô modelo Turtlebot 2, sendo controlado por código python para o sistema ROS. Essas imagens foram então usadas como treinamento e teste da acurácia da rede SpaceYNet.

A criação da SpaceYNet foi feita pelo então mestrando Dunfrey Pires Aragão [1] com inspirações na rede PoseNet, criada por Kendall e Cipolla [4], que é uma Rede Neural Convolucional (CNN) baseada na GoogLeNet 1V1, para extrair características

2D de imagens e usar métodos de regressão para obter os padrões de posicionamento da câmera, principalmente em áreas internas. Algumas adaptações para diminuir o tempo de treinamento e melhorar a geração de imagens foram feitas, inspiradas pela U-Net [5].

2 Conceitos Gerais

2.1 Localização de robôs

Avanços na tecnologia de hardware e software estão habilitando um mundo onde os robôs podem auxiliar os humanos em tarefas do dia a dia. Aplicações no ramo de transporte, industria e até hospitalar estão cada dia mais reais, dado que existe a possibilidade de que robôs humanoides consigam se identificar no local, transportar produtos de forma geral ou até auxiliar em cirurgias.

Em armazéns de distribuição pelo mundo inteiro, pertencentes a empresas como a Amazon, temos robôs trabalhando ininterruptamente para organizar e transportar produtos em estoque de forma agilizar as encomendas e suprir a demanda cada vez maior dos pedidos. Tudo isso feito de forma automática, onde o robô faz parte de uma colmeia ou um membro da colmeia, evitando se esbarrar com os milhares de outros robôs presentes no ambiente ao mesmo tempo sem precisar diminuir seu tempo de conclusão de tarefa.

Apesar de todo o avanço que temos, ainda existe a dificuldade de um robô se identificar em um ambiente. Sensores podem ajudá-lo a evitar encontros indesejáveis, mas para obter uma consciência do local é preciso um pouco mais. Somado à dificuldade que um robô tem ao navegar por um ambiente desconhecido, temos também o acúmulo de erros que sistemas com sensores sofrem ao tentar construir mapas de regiões. Esse acúmulo de erros podem ser multiplicados ao longo do tempo, gerando incertezas na localização o que causa um mapeamento defeituoso após períodos longos de navegação.

De forma geral podemos dizer que ainda há espaço para melhoria na área. Vimos isso nos dias de hoje com o avanço da área de inteligência artificial e aprendizagem de máquina, que iremos abordar nesse trabalho. Na tentativa de unir as melhores características de vários tipos de sistemas, hoje é possível criar redes neurais que nos gerem dados suficientes para auxiliar na criação de um método de controle unificado.

2.1.1 Mapeamento

Para que o robô possa se encontrar no ambiente é preciso que ele primeiro se identifique nele. Para isso é preciso construir uma representação do ambiente com dados extraídos do mesmo. Com isso é possível que o robô construa um modelo do mundo ao seu redor em sua própria memória, auxiliando a forma que ele irá se locomover.

Notar os objetos ao seu redor é uma das principais finalidades necessárias para esse tipo de tarefa. Isso pode ser obtido através de módulos que consigam criar dados a partir do mundo ao redor, como por exemplo sonares, que trabalham com o tempo de resposta de uma onda sonora emitida no ambiente, ou então sensores dependentes de luz, como

câmeras ou sensores infravermelhos.

Sempre é possível e recomendado que um sistema de mapeamento faça uso de mais de uma técnica, a fim de evitar que o erro de uma prejudique o sistema como um todo. Dessa forma podemos usar técnicas mais simples, como a odometria, juntas com técnicas avançadas como computação visual para obter sistemas cada vez mais eficientes.

2.1.2 Odometria

A odometria é um método de localização que faz uso de sensores de movimento para determinar a movimentação em um meio de algo móvel, podendo ser um robô, um automóvel ou qualquer forma de transporte terrestre com rodas. Esse método pode ser usado em robôs para tentar criar mapas do ambiente ou se guiar pelo mesmo. A odometria funciona em uma forma similar a "contar os passos"enquanto mapeia o quanto foi andado em relação a cada direção. Para a odometria funcionar precisamos ter um ponto de partida que servirá de base para o resto da medição.

Porém, essa técnica é melhor utilizada em circuitos de curta distância. É comum ocorrer acúmulo de erros ao contar os passos, devido a problemas no piso, engrenagens travadas ou elevações indesejadas no ambiente. Por isso, uma boa prática é manter a calibração em todos os sistemas que fazem uso dessa técnica de mapeamento, de forma a não negligenciar que o diâmetro das rodas sejam iguais ou que elas estejam desalinhadas, por exemplo.

Apesar das dificuldades, a odometria pode ser bem usada para experimentos e sistemas que precisam executar tarefas em ambientes pequenos, contando sempre com a garantia que o ambiente seja bem controlado para que não ocorra problemas com erros acumulados ou outras características já citadas.

2.1.3 SLAM

O SLAM (Simultaneous Localization and Mapping) [6], ou Localização e Mapeamento Simultâneos é um algorítmo de localização bastante usado que busca resolver uns dos grandes problemas da robótica móvel. O desafio em questão seria proporcionar ao robô a possibilidade de mapear o ambiente em que se encontra, ao mesmo tempo que utilizasse esse mapa para definir sua posição e localização nesse meio desconhecido.

O SLAM tem como objetivo gerar dados de conhecimento sobre o ambiente, gerando assim uma informação sobre a localidade do robô no ambiente que foi mapeado. Essas informações podem ser obtidas por diferentes tipos de sensores, como sonares ou câmeras. O algoritmo então tenta obter pontos de referência no ambiente, assim calculando as medidas da localização dos mesmos. Após essa identificação de pontos de

referência então será possível calcular a localização do veículo baseado na distância e posição a esses pontos. Ao conseguir a posição do veículo em relação a os pontos de referência, temos a possibilidade de mapear a área e construir um mapa com o conjunto dos pontos.

O acúmulo de erros ainda é algo a se considerar, visto que o método SLAM pode fazer uso de um ou mais tipos de sensores. Isso gera incertezas ao confiar nos dados gerados após sessões longas de mapeamento, mas o algoritmo tenta minimizar esses problemas utilizando medições de uma etapa anterior, aumentando assim a precisão da etapa atual.

2.2 Redes Neurais

2.2.1 Neurônios

Um dos objetivos no campo da computação sempre foi o de imitar funções do cérebro humano, sonhando assim ter um pedaço da capacidade que essa máquina computacional natural possui. O cérebro tem como seu elemento de processamento de informação a célula do neurônio mostrado na figura 1, onde podemos ver o seu meio de comunicação (os dendritos), e o seu núcleo de processamento de dados (núcleo).

Desta forma, podemos analisar o comportamento de um neurônio, a base comum para o funcionamento do cérebro, e entender que ele trabalha criando conexões com outros neurônios através dos dendritos, fortalecendo essa conexão quanto mais se usa. Essas conexões podem ser feitas entre 10.000 a 20.000 outros neurônios, formando assim um conglomerado de células capazes de processar as informações que chegam a elas. Assim, temos camadas e camadas de neurônios interconectados, cada um com uma potência de conexão diferente, gerando informações úteis como saída, para o que eles conseguiram se adaptar naturalmente.

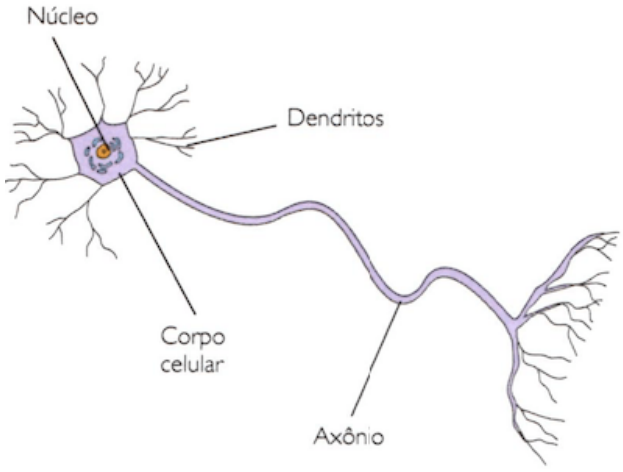

Figura 1: Neurônio humano, inspiração para a criação de redes neurais.

Para trazer esse sistema para o computador, primeiro é preciso entender de forma matemática como funciona. Dessa forma, podemos considerar as conexões que chegam ao neurônio como dados de entrada n. Após a entrada desses dados faremos uso de alguma função, seja ela multiplicativa, somatória ou equivalente, que fará o processamento deles, fazendo essa área ser equivalente ao núcleo do neurônio.

Seguindo a comparativa, temos que os dados irão sair para se comunicar com os próximos neurônios através dos dendritos que, da mesma forma que o cérebro, pode se conectar com até n outros neurônios artificiais. Essas conexões vão ter sua potência variante, ou seja, cada meio de comunicação entre os dendritos possui um elo com sinal mais forte que outras possuindo assim pesos diferentes. No nosso neurônio artificial esses pesos também precisam ser considerado, pois as informações contidas neles caracterizam o objetivo do complexo neural inteiro, podendo assim ser replicado em arquiteturas similares. Para isso, cada peso w possuirá um valor que será salvo para aplicações futuras.

No final disso temos a geração de um sinal de saída y, podendo ser a saída de um único neurônio ou um grupo deles, que será usada como geração de dados que precisamos ou entrada para um próximo neurônio ou grupo dele. A imagem 2 mostra um diagrama onde é possível ver as n entradas, a função e a saída Y.

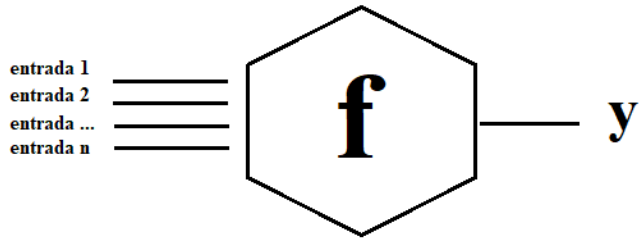

Figura 2: Neurônio artificial, tendo suas n entradas, funções de processamento f e saída Y.

2.2.2 Redes Neurais Profundas

Apesar de os neurônios artificiais serem capazes de gerar resultados promissores, algumas aplicações precisam de um nível mais complexo de processamento de dados. Pensando em mitigar esse problema, algumas técnicas de melhoria de capacidade de processamento foram desenvolvidas, como a interligação de neurônios artificiais, dividindo-os por camadas. Ao interligar neurônios artificiais podemos agora fazer uso de entradas de dados com mais informação, como por exemplo enviar todos os pixeis de uma imagem ao mesmo tempo, assim como obter saídas mais complexas.

Essa nova arquitetura pode ser dividida em três camadas para melhor entendi-

mento:

1. Camada de entrada: Na camada de entrada temos os dados que a rede irá usar como base para gerar sua saída, podendo ser imagens, textos ou qualquer dado que seja organizado de forma a se encaixar na arquitetura de entrada.

2. Camada oculta: A camada oculta é onde todo o processamento acontece, aqui temos acúmulos de resultados de funções sendo passadas adiante, gerando dados diversificados na tentativa de extrair características que serão usadas posteriormente.

3. Camada de Saída: Finalmente na camada de saída temos o resultado dos dados após todo o processamento sequencial gerado pela etapa anterior, podendo ser um dado de classificação ou um valor aproximado, em caso de problemas de regressão.

Na figura 3 podemos ver a camada de entrada em verde, seguido por suas conexões com todos os nós da próxima camada, a oculta, em laranja. Após o processamento interconectado sequencialmente teremos a saída, em vermelho, gerando nosso resultado.

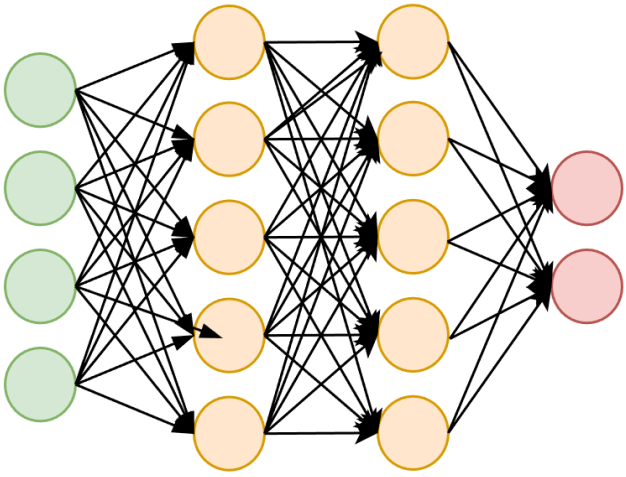

Figura 3: Rede Neural Profunda. Camada de entrada em verde, camada oculta em laranja e camada de saída em vermelho.

2.2.3 Redes Neurais Convolucionais

Para que seja possível criar sistemas que obtenham informações cada vez mais importantes, é preciso que redes neurais possuam formas de extrair informações providas pelos dados de entrada. Dito isso, modificações e diferentes aplicações arquiteturais são feitas, modificando a forma que redes neurais funcionam, adaptando elas para o objetivo necessário. Assim, uma das formas arquiteturais mais relevantes para extração de dados é a conhecida como Redes Neurais Convolucionais.

Tais redes, assim como as técnicas de convolução, possuem uma máscara convolucional que desliza por entre os dados de entrada, extraindo informações desejadas em áreas específicas dos mesmos que servirão nos próximos passos como um destaque dos dados relevantes.

Na figura 4 é possível ver um exemplo onde uma máscara já conhecida, a máscara de Sobel – que é responsável por identificar e extrair características de contornos em imagens – é aplicada em uma imagem de exemplo do dataset do LaSER. Após a aplicação da máscara podemos ver que temos como resultado a mesma imagem, porém com os contornos destacados, ficando claro a posição deles na imagem RGB, facilitando caso a necessidade seja identificar e destacar bordas.

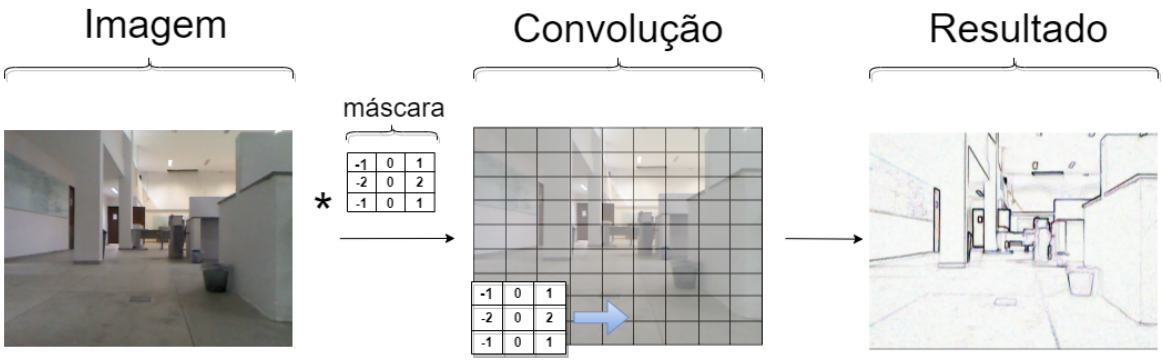

Figura 4: Etapas simplificadas da convolução, onde a máscara desliza pela imagem fazendo cáculos e extrai resultados esperados para qual foi feita.

Em resumo temos a convolução agindo como um tipo de filtro, que deixa passar apenas os dados que lhe foi programado. Essa etapa em uma rede neural é conhecida como "extração de características". E a cada passagem da máscara por grupo de píxel do mesmo tamanho, temos um novo valor gerado que será salvo em uma nova matriz de informações. Esses novos dados gerados serão então alimentados aos próximos neurônios.

2.2.4 PoseNet

Com o sucesso do método SLAM e aproveitando as peculiaridades dos sistemas criados por rede neurais, Kendall, Grimes e Cipoola criaram a Rede PoseNet [4]. A PoseNet tinha como objetivo criar um framework de aprendizagem de máquina para utilizar o SLAM, ou seja, uma rede neural que geraria dados equivalentes aos resultados do SLAM porém utilizando como entrada imagens geradas por um sistema monocular, podendo aplicar uma regressão para assim conseguir informações sobre o posicionamento, sendo coordenadas e angulação da câmera.

Assim, temos um sistema facilitado, possuindo como entrada uma imagem RGB vinda de uma câmera monocular e uma saída contendo informações de coordenadas X,

Y e Z, tal qual dados de orientação descritos por quatérnio W, P, Q e R, podendo assim fornecer dados de localização para robôs em ambientes de tamanho médio. O algoritmo funciona bem em ambientes internos e externos, levando apenas 5ms para retornar um valor esperado.

2.3 Considerações Finais

Neste capitulo falamos sobre a base que inspirou a criação da solução que a SpaceYNet propõe. A inspiração do método SLAM e a tentativa de simplificação dele para aplicações com sensores mais acessíveis foi visto como uma possibilidade, principalmente contando com os avanços na área de redes neurais mostrados.

Com essa carga informativa podemos assim aplicar o que temos para criar um sistema que, com o uso de apenas uma unidade de computação e uma câmera, pode ser aplicada em qualquer unidade móvel robótica, auxiliando o mesmo a se identificar no meio, exibir posicionamento no ambiente e possivelmente lidar com alterações no mesmo.

3 Metodologia

3.1 Laboratório LaSER

O laboratório LaSER (Laboratório de Sistemas Embarcados e Robótica) se encontra dentro do Centro de Informática da Universidade Federal da Paraíba (UFPB) e tem como objetivo prover um espaço dedicado a pesquisas nas áreas de robótica, automação, inteligência artificial e microcontroladores no geral.

O espaço contém uma área de 30x40 metros, um total de 1200 metros quadrados (m^2), contendo bancadas, entradas para salas, cadeiras e outros obstáculos comuns. Esse espaço foi utilizado para criação do Dataset LaSER, baseando-se em um caminho pré-definido no qual um robô de modelo Turtlebot 2 acompanhado de câmera seguiria pelo mesmo, gravando imagens de formato RGB, tal qual imagens com informações sobre a profundidade.

O caminho definido pode ser visto na figura 5, onde podemos ver os indicativos de início no marcador de entrada apontando a porta de entrada de uma das salas de pesquisa onde, após todo o circuito ser completado, também é o ponto de fim da rota do robô. Essa rota foi criada pensada no melhor caminho que pudesse extrair a maior quantidade de informações para imagens sequenciais, auxiliando assim ao chegar na etapa de criação de rótulos para elas contendo os dados de posicionamento no ambiente através do programa VisualSFM.

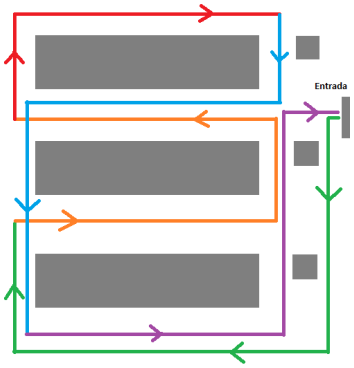

Figura 5: Rota definida para que o robô de modelo Turtlebot 2 conseguisse extrair a maior quantidade de informações sobre o ambiente, auxiliando assim na extração das informações do posicionamento.

Dessa forma temos que o ambiente do laboratório pode se considerar um onde obstáculos e modificações humanas acontecem no dia a dia, tornando-o um bom candidato para testes após treinamento dos modelos. Apesar disso, a criação do dataset foi feita com um certo nível de controle do ambiente, evitando que houvessem grandes modificações, como cadeiras em diferentes locais ou mesas excluídas antes do fim do ciclo, visando obter um treinamento mais limpo para os testes da rede neural.

3.2 Turtlebot 2

Uma necessidade implícita para criação do dataset é a câmera, assim como o robô que irá guiá-la pelo caminho e área que serão aprendidos pela rede neural. Para isso foi disponibilizado pelo laboratório um robô modelo TurtleBot 2, constituído de rodas para movimentação, uma câmera Kinect da Microsoft para captura de imagens e suportes para controle de posicionamento da câmera acima do robô, afim de manter uma angulação estável da mesma. O robô pode ser visto na figura 6, assim com os itens citados anteriormente.

A câmera kinect foi escolhida pois sozinha pode proporcionar informações de profundidade assim como imagens RGB, pois nela está incluso sensores de proximidade próximos aos sensores padrões de imagens. O robô Turtlebot 2 fornecido funciona com o sistema operacional ROS (Robot Operating System).Esse sistema nos fornece acesso a rotinas de movimentação, capturas de imagens em câmeras conectadas e, no geral, controle sobre periféricos conectados no mesmo.

Os scripts para controle, comunicação e criação foram feitos utilizando a linguagem Python, o que facilitou a implementação por ser uma linguagem com um bom suporte. A comunicação com o robô foi feita através de protocolo SSH, acessando o sistema por um cabo USB de conexão direta fazendo uso de um notebook ou tendo a possibilidade de se conectar pela internet, e o controle do mesmo foi feito por módulos já presentes no sistema responsável pela movimentação das rodas, já abstraindo os detalhes para movimentos como giro e ré.

Figura 6: Robô terrestre de modelo Turtlebot 2, fornecido pelo laboratório para criação do dataset LaSER. O robô originalmente não vem com a câmera kinect, porém foi adaptada para solucionar nossas necessidades.

3.3 Criação do dataset

Para se obter bons resultados as redes neurais convolucionais precisam de uma grande quantidade de imagens, de preferência com as características bem evidentes. Se inspirando nesta necessidade foi indicado que as imagens do local fossem criadas seguindo um padrão de rotas predefinidas pelo professor orientador, essa rota pode ser vista na figura 5. Assim, utilizando-se desta abordagem podemos suprir a primeira necessidade para uma boa acurácia da rede neural convolucional.

A captura das imagens foi feita através de um script em Python que gera um vídeo na frequência de 30 imagens por segundo nas câmeras de profundidade e RGB, considerando a necessidade de sincronizar o tempo de captura entre as câmeras, visto que poderia acontecer de o sensor RGB processar informações mais rápido que o sensor de profundidade.

O objetivo era seguir um caminho específico por entre os balcões do laboratório completando um circuito (Figura 5) que gerasse imagens com informações distintas e complementares do ambiente. O circuito teve uma duração entre 15 a 25 minutos e gravações de video foram feitas do caminho. A seguir foram extraído frames do video (figura 7 e 8), na frequência de 30 para cada segundo passado, gerando mais de 15 mil imagens para treinamento e teste da rede neural.

Figura 7: Frame retirado do video RGB

Figura 8: Frame retirado do video de profundidade

3.4 Características do Dataset

Para o treinamento da rede neural houve a necessidade de adaptar o formato e tamanho das imagens que seriam usadas. Para isso as imagens foram redimensionadas para tamanhos de 256x256 pixeis, controlando assim o tamanho dos dados de entrada para treinamento da rede neural.

O dataset LaSER foi, como dito anteriormente no capítulo 3.3, criado fazendo uso de um robô turtlebot 2, que seguia um caminho definido previamente visando obter a melhor qualidade de sequencias de imagens onde seria possível criar uma conexão entre a imagem anterior e a atual. Este caminho foi feito dentro do laboratório LaSER, laboratório este que contém uma área de 30x40 metros quadrados.

No primeiro caminho feito pelo robô Turtlebot 2, seguindo a rota vista na imagem 5 e capturando imagens RGB-D (cores e profundidade) pelos sensores da câmera kinect obtivemos 17483 imagens RGB e, de forma similar, a mesma quantidade de imagens com informações sobre a profundidade da cena. Um exemplo dessas imagens e de suas equivalentes pode ser visto na figura 9, onde temos a imagem RGB na esquerda e suas informações de profundidade na direita.

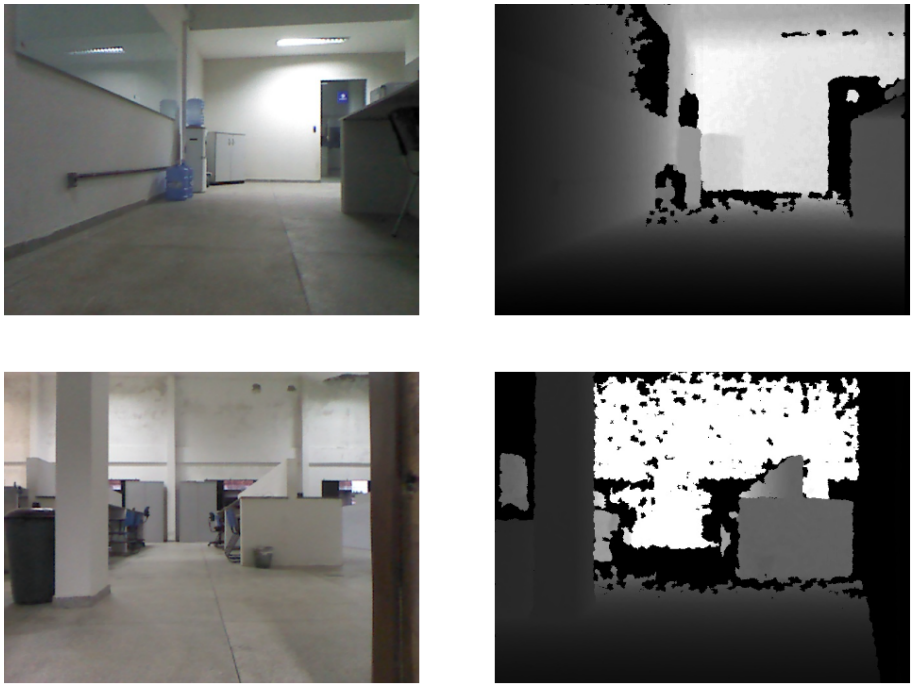

Figura 9: Exemplo de imagens do dataset. A esquerda imagens RGB e a direita exemplos de imagens de profundidade.

Enquanto a captura de imagens acontecia, o sistema de odometria do robô capturava dados informando a localização relativa que a imagem foi capturada. É possível assim então assimilar uma certa imagem e uma certa informação de profundidade a uma

coordenada dentro do laboratório, criando assim um rótulo para as imagens que seriam ingeridas como parte do treinamento da rede neural.

Sendo assim, temos que cada imagem RGB tem um par contendo informações de profundidade do local, assim como dados sobre o posicionamento (coordenadas cartesianas e ângulos quaterniões) atrelados a ela. Como o robô era terrestre, o eixo Z foi então ignorado pois foi considerado que ele estaria em um ambiente reto e sem tamanhas modificações de relevo.

Temos então que o dataset LaSER contém informações de uma grande rota, fundindo informações de imagens com localização no ambiente que pode assim ser utilizado para treinamento de redes neurais.

3.5 A rede SpaceYNet

A rede SpaceYNet surgiu da ideia do mestrando Dunfrey Pires de Aragão, ao notar que aspectos da rede GoogLeNet e PoseNet poderiam ser melhorados, podendo gerar novos dados de saídas de forma a unir sistemas de localização em um único Framework.

Citada anteriormente, a rede GoogLeNet 1v1 [7] foi uma das redes pioneiras ao usar convoluções em imagens bi-dimensionais para obter informações que seriam usadas para tarefas de classificação. Com ela foi possível classificar objetos e obter aprendizado em estruturas de dados irregulares, tornando dela uma inspiração primordial ao se considerar montar outra rede neural na tentativa de obter feitos similares.

A SpaceYNet teve também como modelo de incentivo a rede PoseNet, rede esta que foi inspirada na anteriormente citada GoogLeNet, porém com uma característica a mais: a possibilidade de obter a informação de posicionamento do robô, usando uma imagem como entrada, montando assim uma rede que serve como framework de aprendizado profundo do método SLAM de posicionamento.

A modificação da rede PoseNet que habilitou a saída da mesma de gerar dados no espaço cartesiano, em vez de dados de classificação, como geralmente é esperado, foi uma das características principais usada pela SpaceYNet, visto que a PoseNet obteve ótimos resultados com essa técnica [4].

A SpaceYNet também faz uso de técnicas desenvolvidas na criação da rede U-Net. Esta rede tem como características tanto a possibilidade de poder treinar uma rede com poucos dados, fazendo uso de técnicas de data augmentation, quanto analisar, extrair e gerar uma nova imagem com as informações obtidas. Assim, a SpaceYNet tem a U-Net como base na primeira metade de sua arquitetura, devido a sua baixa complexidade agregada com seus bons resultados.

Na figura 10 é possível ver como todas essas inspirações serviram como base para

a montagem da arquitetura da rede neural SpaceYNet. Nela temos as camadas similares a U-Net, que irão regredir as informações de profundidade da imagem, que logo a seguir serão aplicadas a camadas inception, similares a PoseNet. Essas camadas então serão responsáveis por gerar dados de posicionamento, tendo então coordenadas no plano cartesiano X, Y e Z, assim como dados do quaternião W, P, Q e R obtendo assim posicionamento e orientação da câmera.

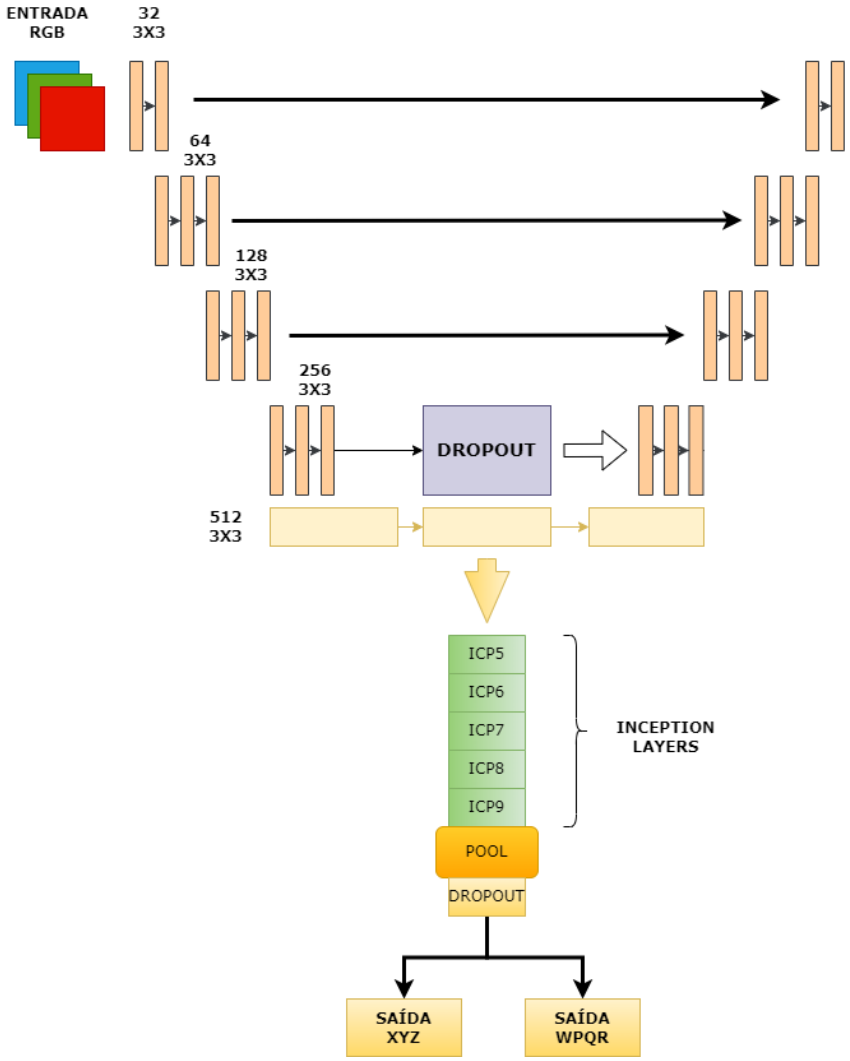

Figura 10: Arquitetura da SpaceYNet. Entrada RGB, passando por camadas inspiradas pela U-Net que irão obter imagens de profundidade para, logo a seguir, passarem por camadas inception para extrair informações de pose do robô.

4 Resultados

Com o dataset LaSER criado, contendo inclusive os rótulos para cada imagem, é possível começar a etapa de experimentos. Esses experimentos começam com o treinamento da rede, para a seguir analisar as possibilidades de overfitting e underfitting, assim como erro acumulado. A seguir temos então os testes que nos dizem quanto de acerto aconteceu para cada imagem da rede, ou seja, no nosso caso o quão próximo os dados de pose e angulação ficaram do real.

4.1 Treinamento da Rede

Para treinamento da rede é preciso um grande poder computacional, pois ele é uma série de cálculos e comparações numéricas que podem se estender por mais de 100 épocas de treinamento, o que dificulta o processo de criação da rede.

Para o treinamento inicial da SpaceYNet foi utilizado um processador intel i9-8750H com 6 núcleos e 12 threads e frequência de funcionamento máxima de 4.10 GHz, combinado com a placa de vídeo Nvidia GeForce RTX 2070 com 8GB de memória interna e 2304 núcleos de processamento CUDA que são diretamente usados e otimizados para os cálculos de treinamento de redes neurais. O Computador também possuía 32GB de RAM e rodava o sistema operacional Ubuntu versão 19.04 64 bits.

O treinamento foi configurado para rodar por 500 épocas ou parar de forma automática caso a métrica de validação não venha a diminuir após 30 épocas com uma taxa de aprendizagem de 0.00146.

O dataset LaSER foi dividido em uma proporção de 70% para treinamento e 30% para o processo de testes. Essa divisão é feita de forma aleatória e a comparação dos dados é feita de forma automática pelo algoritmo onde é feito uma regressão dos dados para obter os resultados.

Nessa etapa de treinamento temos a criação da rede respeitando sua arquitetura, que vimos na figura 10. Nesse processo de treinamento será calculado a "força"ou seja, o peso das conexões entre neurônios, que define o cálculo inteiro que acontece dentro das redes neurais. Esses pesos são então salvos em um arquivo para serem reutilizados posteriormente como desejar. Após isso a rede pode também ser exportada como arquivo para então ser analisada visualmente, gerando dados sobre perda, acurácia entre outros.

4.2 Resultados da Rede

4.2.1 SpaceYNet vs PoseNet

Para uma base de comparação, foi decidido utilizar dados da rede neural PoseNet, por ser uma rede estado da arte e inspiração, para obter uma noção das melhorias que a SpaceYNet trouxe com suas modificações.

Como dito anteriormente, o dataset para este processo foi dividido em uma proporção de 70% para treinamento e 30% para testes. Para a parte de testes os resultados foram aplicados em um algoritmo que calcula a média de acerto gerado pela regressão em relação aos dados reais. Assim, de acordo com a figura 11 podemos ter uma boa noção comparativa entre a SpaceYNet e a PoseNet.

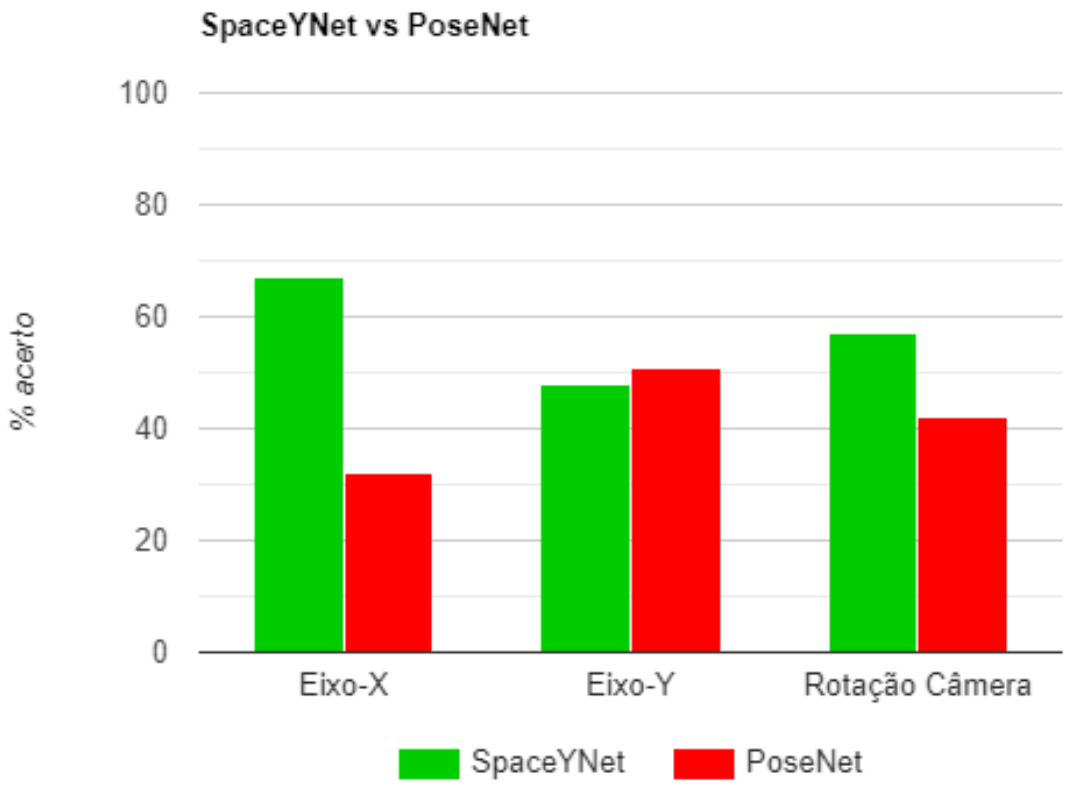

Figura 11: Resultados da SpaceYNet e PoseNet em comparação com os dados reais.

Na figura 11 notamos que a SpaceYNet obteve um melhor resultado em prever o eixo X e a rotação da câmera do robô, enquanto que a PoseNet teve um melhor resultado ao prever os dados do eixo Y, porém não tão diferentes dos resultados gerados pela SpaceYNet. Também foi possível notar que a PoseNet gerada mais ruídos nos seus dados

que a SpaceYNet ao prever as posições no laboratório LaSER.

Sendo assim, podemos dizer que a SpaceYNet obteve uma maior tolerância ruídos, conseguindo seguir uma rota devido a possibilidade de identificar falso-positivos no caminho.

Figura 12: Dados gerados pela saída da predição de profundidade da SpaceYNet. Dados gerados estão embaixo.

Os resultados gerados ao prever os dados de profundidade podem ser vistos na figura 12. Ficando claro que a rede mantém informações importantes do ambiente.

5 Conclusão

Neste trabalho foi apresentado um processo de criação de dataset para treinamento de uma nova rede neural capaz de servir como framework de localização completa através de técnicas de aprendizagem de máquina.

Aqui mostramos que técnicas de localização mais consolidadas podem ser utilizadas como complemento para gerar dados, podendo assim criar uma relação entre eles e imagens, tanto RGB quanto de profundidade, para obter uma base de dados capaz de facilitar a implementação de novas soluções.

O dataset criado serve como uma base detalhada com imagens RGB, de profundidade e dados de odometria rotulados a cada imagem, podendo ser utilizado em trabalhos futuros na área ou em análise de dados.

A rede neural citada neste trabalho, criada pelo mestrando Dunfrey Pires de Aragão, se mostrou capaz de gerar imagens e prever a posição do robô de forma superior a PoseNet, no meio que foi testado: o LaSER - Laboratório de Sistemas Embarcados e Robótica, na UFPB - Universidade Federal da Paraíba.

REFERÊNCIAS

[1] Dunfrey Aragao, Tiago Nascimento, and Adriano Mondini. Spaceynet: A novel approach to pose and depth-scene regression simultaneously. pages 217–222, 07 2020.

[2] Anderson Frazão. Kinect-Depth-and-Rgb-sync, 12 2022.

[3] Lingyan Ran, Yanning Zhang, Qilin Zhang, and Tao Yang. Convolutional neural network-based robot navigation using uncalibrated spherical images. *Sensors*, 17:1341, 06 2017.

[4] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Convolutional networks for real-time 6-dof camera relocalization. *CoRR*, abs/1505.07427, 2015.

[5] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[6] Hugh Durrant-whyte and Tim Bailey. Simultaneous localization and mapping: Part i. *Robotics Automation Magazine, IEEE*, 13:99 – 110, 07 2006.

[7] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions, 2014.

[8] Alex Kendall and Roberto Cipolla. Modelling uncertainty in deep learning for camera relocalization, 2016.

[9] Kevin Lai, Liefeng Bo, and Dieter Fox. Unsupervised feature learning for 3d scene labeling. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3050–3057, 2014.

[10] Changchang Wu. Towards linear-time incremental structure from motion. In *2013 International Conference on 3D Vision - 3DV 2013*, pages 127–134, 2013.

[11] Laser dataset, oct 2021. [online]. available: https://www.kaggle.com/dunfrey/laser-dataset.