**UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS**

Nícolas dos Santos Rosa

**Estimação monocular de profundidade por aprendizagem
profunda para veículos autônomos: influência da
esparsidade dos mapas de profundidade no treinamento
supervisionado**

São Carlos

2019

Nícolas dos Santos Rosa

Estimação monocular de profundidade por aprendizagem profunda para veículos autônomos: influência da esparsidade dos mapas de profundidade no treinamento supervisionado

Dissertação apresentada à Escola de Engenharia de São Carlos da Universidade de São Paulo, para obtenção do título de Mestre em Ciências - Programa de Pós-Graduação em Engenharia Elétrica.

Área de concentração: Sistemas Dinâmicos

Orientador: Prof. Dr. Valdir Grassi Junior

São Carlos

2019

AUTORIZO A REPRODUÇÃO TOTAL OU PARCIAL DESTE TRABALHO,
POR QUALQUER MEIO CONVENCIONAL OU ELETRÔNICO, PARA FINS
DE ESTUDO E PESQUISA, DESDE QUE CITADA A FONTE.

Ficha catalográfica elaborada pela Biblioteca Prof. Dr. Sérgio Rodrigues Fontes da
EESC/USP com os dados inseridos pelo(a) autor(a).

D788e dos Santos Rosa, Nicolas
 Estimação monocular de profundidade por
aprendizagem profunda para veículos autônomos:
influência da esparsidade dos mapas de profundidade no
treinamento supervisionado / Nicolas dos Santos Rosa;
orientador Valdir Grassi Junior. São Carlos, 2019.

 Dissertação (Mestrado) - Programa de Pós-Graduação
em Engenharia Elétrica e Área de Concentração em
Sistemas Dinâmicos -- Escola de Engenharia de São
Carlos da Universidade de São Paulo, 2019.

 1. Estimação de Profundidade. 2. Visão Monocular.
3. Aprendizado Profundo. 4. Aprendizado Supervisionado.
5. Aprendizado de Máquina. I. Título.

FOLHA DE JULGAMENTO

Candidato: Engenheiro **NÍCOLAS DOS SANTOS ROSA**.

Título da dissertação: "Estimação monocular de profundidade por aprendizagem profunda para veículos autônomos: influência da esparsidade dos mapas de profundidade no treinamento supervisionado".

Data da defesa: 24/06/2019.

Comissão Julgadora:

Resultado:

Prof. Dr. **Valdir Grassi Junior**
(Orientador)
(Escola de Engenharia de São Carlos/EESC)

Aprovado

Prof. Associado **Denis Fernando Wolf**
(Instituto de Ciências Matemáticas e de Computação/ICMC-USP)

APROVADO

Prof. Dr. **Carlos Henrique Quartucci Forster**
(Instituto Tecnológico de Aeronáutica/ITA)

Aprovado

Coordenador do Programa de Pós-Graduação em Engenharia Elétrica:
Prof. Associado **Ivan Nunes da Silva**

Presidente da Comissão de Pós-Graduação:
Prof. Titular **Murilo Araujo Romero**

AGRADECIMENTOS

Primeiramente, agradeço a Deus por propiciar saúde e felicidade a todos aqueles que me rodeiam.

À minha família: aos meus pais, Valdenilce e Francisco, pela ferrenha dedicação em mostrar a mim e a minha irmã, Natália, a importância da educação em nossa formação pessoal; e por fazer o possível, e o impossível, para possibilitar condições para que atingíssemos nossos objetivos.

Ao meu orientador, Prof. Dr. Valdir Grassi Junior, pela oportunidade, apoio e orientação ao longo dos últimos anos no desenvolvimento deste trabalho e por despertar em mim a curiosidade por veículos autônomos, reafirmando ainda mais minha paixão pela engenharia.

Ao pesquisador Dr. Vitor Campanholo Guizilini, que durante seu Pós-Doutorado pacientemente me apresentou diretrizes e conceitos essenciais quanto ao desenvolvimento de redes neurais profundas.

Aos membros do Laboratório de Sistemas Inteligentes (LASI), Anderson, Gustavo, Eduardo, Rafael, Filipe, Raul, Kaio, Elizandra, Lucas, Josi, João, Kenny, Daiane, Maurício e Rayza, pela convivência amigável e companheirismo.

A todos aqueles que contribuíram, direta ou indiretamente, para a realização desta dissertação, o meu sincero agradecimento.

Por fim, gostaria de agradecer às instituições que ampararam o desenvolvimento deste trabalho. Esta pesquisa foi apoiada financeiramente pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), sob concessão 130463/2017-5 e 465755/2014-3, financiada em parte pela Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil - CAPES (Código de Financiamento 001 e 88887.136349/2017-00), pela Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) 2014/50851-0 e pela Faculdade de Engenharia e Tecnologias da Informação (*University of Sydney*). Também gostaria de agradecer o apoio da *NVIDIA Corporation* com a doação das GPUs Titan Xp usadas neste projeto.

RESUMO

Rosa, NS **Estimação monocular de profundidade por aprendizagem profunda para veículos autônomos: influência da esparsidade dos mapas de profundidade no treinamento supervisionado.** 2019. 129p. Dissertação (Mestrado) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2019.

Este trabalho aborda o problema da estimação de profundidade a partir de imagens monoculares (SIDE), com foco em melhorar a qualidade das previsões de redes neurais profundas. Em um cenário de aprendizado supervisionado, a qualidade das previsões está intrinsecamente relacionada aos rótulos de treinamento, que orientam o processo de otimização. Para cenas internas, sensores de profundidade baseados em escaneamento por luz estruturada (Ex.: *Kinect*) são capazes de fornecer mapas de profundidade densos, embora de curto alcance. Enquanto que para cenas externas, consideram-se LiDARs como sensor de referência, que comparativamente fornece medições mais esparsas, especialmente em regiões mais distantes. Em vez de modificar a arquitetura de redes neurais para lidar com mapas de profundidade esparsa, este trabalho introduz um novo método de densificação para mapas de profundidade, usando o *framework* de Mapas de Hilbert. Um mapa de ocupação contínuo é produzido com base nos pontos 3D das varreduras do LiDAR, e a superfície reconstruída resultante é projetada em um mapa de profundidade 2D com resolução arbitrária. Experimentos conduzidos com diferentes subconjuntos do conjunto de dados do KITTI mostram uma melhora significativa produzida pela técnica proposta (esparso-para-contínuo), sem necessitar inserir informações extras durante a etapa de treinamento.

Palavras-chave: Estimação de Profundidade. Visão Monocular. Aprendizado Profundo. Aprendizado Supervisionado. Aprendizado de Máquina.

ABSTRACT

Rosa, NS **Monocular depth estimation by deep learning for autonomous vehicles: influence of depth maps sparsity in supervised training.** 2019. 129p.
Dissertação (Mestrado) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2019.

This work addresses the problem of single image depth estimation (SIDE), focusing on improving the quality of deep neural network predictions. In a supervised learning scenario, the quality of predictions is intrinsically related to the training labels, which guide the optimization process. For indoor scenes, structured-light-based depth sensors (e.g. Kinect) are able to provide dense, albeit short-range, depth maps. While for outdoor scenes, LiDARs are considered the standard sensor, which comparatively provide much sparser measurements, especially in areas further away. Rather than modifying the neural network architecture to deal with sparse depth maps, this work introduces a novel densification method for depth maps using the Hilbert Maps framework. A continuous occupancy map is produced based on 3D points from LiDAR scans, and the resulting reconstructed surface is projected into a 2D depth map with arbitrary resolution. Experiments conducted with various subsets of the KITTI dataset show a significant improvement produced by the proposed Sparse-to-Continuous technique, without the introduction of extra information into the training stage.

Keywords: Depth Estimation. Monocular Vision. Deep Learning. Supervised Learning. Machine Learning.

LISTA DE FIGURAS

Figura 1	– Comparação qualitativa entre o mapa de profundidade obtido por visão estéreo contra o obtido por método monocular. Cores mais quentes e mais frias representam distâncias maiores e menores, respectivamente. .	24
Figura 2	– Comparativo em nível de esparsidade entre os <i>datasets</i> (a) <i>KITTI Discrete</i> (esparso), (b) <i>KITTI Depth</i> (semi-denso) e (c) <i>KITTI Continuous</i> (denso, proposto). Cores mais quentes e mais frias representam distâncias maiores e menores, respectivamente.	25
Figura 3	– Filtros obtidos pelo algoritmo de aprendizagem de características invariantes aplicado ao conjunto de caracteres MNIST (imagem à esquerda). Filtros seletivos obtidos pela modelagem linear das representações visuais da visão humana (imagem à direita).	31
Figura 4	– Modelo canônico de um perceptron multicamadas (MLP) e estrutura interna de um neurônio artificial	33
Figura 5	– Neste contexto, um método computacional, por meio de muitos dados (aprendizagem profunda), aprende características mais eficientes (aprendizado de representação), que é, por sua vez, um tipo de algoritmo em aprendizado de máquina, e que pode ser usado para muitas, mas não todas, abordagens em IA.	35
Figura 6	– Ilustração de arquiteturas arbitrárias de uma MLP e uma CNN (tamanho do <i>kernel</i> = 3)	38
Figura 7	– Topologia convencional de uma CNN-FC (<i>CNN with Fully-Connected layers</i>)	39
Figura 8	– Topologia convencional de uma FCN (<i>Fully Convolutional Network</i>) . .	40
Figura 9	– Ilustração comparativa dos diferentes tipos de <i>Padding</i> com <i>stride</i> unitário ($s = 1$).	41
Figura 10	– Ilustração de um Filtro Convolutivo.	43
Figura 11	– Exemplificação de filtros de <i>pooling</i> (à esquerda) e <i>unpooling</i> (à direita). .	44
Figura 12	– Exemplificação de filtros convolucionais (à esquerda) e deconvolucionais (à direita).	45
Figura 13	– Modelo de rede neural com <i>Dropout</i> . Esquerda: Rede neural padrão com duas camadas escondidas. Direita: Exemplo de rede produzida pela aplicação de <i>Dropout</i> na rede à esquerda. Unidades cruzadas foram desligadas.	50
Figura 14	– Arquitetura da rede neural convolutiva <i>AlexNet</i>	51
Figura 15	– Estrutura interna dos módulos <i>Inception</i>	51
Figura 16	– Arquitetura da rede neural convolutiva <i>VGGNet</i>	52

Figura 17 – Conexões internas do Bloco Residual, módulo fundamental para construção de redes residuais.	52
Figura 18 – Arquiteturas das redes neurais convolucionais <i>GoogLeNet</i> (à esquerda) e <i>ResNet</i> (à direita)	53
Figura 19 – Principais Deep Learning Frameworks de 2017 e seus mantenedores. . .	55
Figura 20 – Algoritmo de estimação de modelo 3D proposto por Hoiem, Efros e Hebert (2005). Cores não apresentam informação significativa, apenas diferenciam as regiões segmentadas.	61
Figura 21 – Resultados qualitativos obtidos pelas topologias do tipo MRF. Cores mais quentes indicam regiões mais próximas do observador.	62
Figura 22 – Ilustração exemplificando os procedimentos realizados para identificação de relações geométricas e estimação da estrutura da cena com base nos objetos presentes nela. Objeto âncora: Sofá	63
Figura 23 – Ilustração da topologia <i>Coarse-to-Fine</i> para estimação de profundidade. Tons mais claros indicam maiores valores de profundidades.	65
Figura 24 – Resultado obtido por uma topologia do tipo CNN+CRF com superpixelização. Cores indicam profundidades (tons vermelhos correspondem a regiões mais distantes e tons azuis, a regiões mais próximas).	66
Figura 25 – Modelo desenvolvido por Cao, Wu e Shen (2018), no qual utiliza-se classes de intervalos de profundidade e cada píxel é classificado em um destes intervalos (<i>Pixel-wise Classification</i>), cuja saída é posteriormente processada por um CRF.	67
Figura 26 – Modelo desenvolvido por Garg et al. (2016), cuja topologia é um rede do tipo <i>autoencoder</i> . (1) <i>Encoder</i> da rede, responsável por mapear a imagem de entrada I_1 (imagem esquerda do par estéreo) ao seu mapa de profundidade D . (2) <i>Decoder</i> da rede, o qual estima a imagem distorcida I_w , esta que é resultante de distorções impostas pelo mapa de profundidade estimado ao aplicá-lo na imagem direita do par estéreo I_2 . (3) O treinamento do método minimiza o erro de reconstrução entre I_w e I_1	69
Figura 27 – <i>Monodepth</i> : Módulo desenvolvido por Godard, Aodha e Brostow (2017) para cálculo das penalizações do método não-supervisionado proposto. A partir da combinação dos três termos da função de custo gera-se os mapas de disparidades d_l e d_r . Legenda: <i>Convolution</i> (C), <i>Up-Convolution</i> (UC), <i>Bilinear Sampling</i> (S), <i>Up Sampling</i> (US).	69
Figura 28 – Topologia ilustrativa da abordagem <i>non-Guided Depth Upsampling</i> . . .	78
Figura 29 – Topologia ilustrativa da abordagem <i>Image Guided Depth Completion</i> , na qual utilize-se diferentes tipos de informações auxiliares para aprimorar a qualidade das predições de profundidade da CNN.	78

Figura 30 – Exemplo de projeção da nuvem de pontos reconstruída usando o <i>framework</i> de mapas de Hilbert.	79
Figura 31 – Ilustração comparativa entre topologias clássicas e a topologia proposta para a tarefa de <i>Estimação de Profundidade</i>	84
Figura 32 – Representação ilustrativa da geração de mapas de profundidade (esparso) – Projeção da nuvem de pontos no plano de imagem 2D da câmera. . .	85
Figura 33 – A arquitetura de rede utilizada foi proposta por Laina et al. (2016), que é inspirada na ResNet-50, porém suas camadas totalmente conectadas foram substituídas por blocos de <i>upsampling</i> . Diferentemente dos autores originais, alterou-se a estrutura de treinamento para fazer com que a rede proposta estime metros em vez de distâncias em escala logarítmica.	88
Figura 34 – Diagrama de blocos da arquitetura implementada no ROS.	90
Figura 35 – Comparação qualitativa entre estimativas de profundidade quando treinadas nos conjuntos de dados propostos. (a) imagem RGB de entrada. (b) <i>KITTI Discrete</i> (Esparso, 1 Varredura). (c) <i>KITTI Depth</i> (Semi-Denso, 11 Varreduras). (d) <i>KITTI Continuous</i> (Denso, Mapas de ocupação contínuos). (e) Imagem de profundidade de referência (Contínuo, <i>Ground Truth</i>).	100
Figura 36 – Comparação qualitativa entre as estimativas do método proposto (FCRN + <i>KITTI Continuous</i>) e outros trabalhos no estado da arte.	101
Figura 37 – Predições da rede (FCRN + <i>KITTI Continuous</i>) para cenas genéricas em diferentes cidades ao redor do mundo. Todas as cenas foram retiradas da plataforma de compartilhamento de vídeos <i>Youtube</i> . Da esquerda para direita: Imagem RGB de entrada, Predição (Jet), Predição (HSV), Sobreposição.	102
Figura 38 – Predições da rede (FCRN + <i>KITTI Continuous</i>) para cenas colhidas pelas plataformas robóticas desenvolvidas do Laboratório de Robótica Móvel (LRM) do ICMC/USP. Da esquerda para direita: Imagem RGB de entrada, Predição (Jet), Predição (HSV), Sobreposição.	103
Figura 39 – <i>RViz</i> – Visualização das projeções 3D das predições da rede. Topo à esquerda: Imagem RGB, Entrada da Rede. Topo à direita: Imagem de Profundidade, Predição da rede. Branco: Nuvem de pontos do LiDAR. Vermelho: Nuvem de pontos predita pela rede. Cenas retiradas da <i>ROSBag</i> “kitti_2011_09_26_drive_0001_synced.bag”, cuja sequência de quadros foi originalmente disponibilizada pelo KITTI.	105

LISTA DE TABELAS

Tabela 1	– Marcos importantes na história do aprendizado de máquinas, abordando exclusivamente acontecimentos referentes a redes neurais clássicas e profundas.	36
Tabela 2	– Visão comparativa sobre o treinamento dos trabalhos estado da arte. Legenda: (M) Make3D, (N) NYU Depth V2, (K) KITTI, (CS) CityScapes.	73
Tabela 3	– Resultados no estado da arte para o <i>dataset</i> KITTI. Desempenho avaliado nos dados brutos do LiDAR segundo o <i>Eigen Split</i> (697 imagens de teste) (EIGEN; PUHRSCH; FERGUS, 2014). Os resultados relatados são aqueles apresentados pelos próprios autores em seus respectivos artigos originais.	74
Tabela 4	– Número de pares de imagens usados nos subconjuntos de treinamento e validação para cada <i>dataset</i>	86
Tabela 5	– Especificações do <i>Hardware</i> e <i>Softwares</i> utilizados.	94
Tabela 6	– Resultados da avaliação de desempenho da técnica de mapas de Hilbert para a tarefa de densificação de mapas.	96
Tabela 7	– Lista de Hiperparâmetros	97
Tabela 8	– Tabela resumida. Resultados obtidos das melhores configurações da FCRN quando treinadas nos conjuntos de dados propostos, os quais possuem diferentes níveis de esparsidade nas imagens de profundidade de referência, e avaliadas no <i>Eigen Split para KITTI Depth</i>	97
Tabela 9	– Tabela Completa. Resultados de todas as configurações da FCRN quando treinadas nos conjuntos de dados propostos e avaliadas no <i>Eigen Split para KITTI Depth</i> considerando o intervalo de 0 – 50 <i>m</i>	98
Tabela 10	– Tabela Completa. Resultados de todas as configurações da FCRN quando treinadas nos conjuntos de dados propostos e avaliadas no <i>Eigen Split para KITTI Depth</i> considerando o intervalo de 0 – 80 <i>m</i>	99

LISTA DE ABREVIATURAS E SIGLAS

AR	<i>Augmented Reality</i> – Realidade Aumentada
BN	<i>Batch Normalization</i> – Normalização por Lote
CIFAR	<i>Canadian Institute for Advanced Research</i> – Instituto Canadense para Pesquisas Avançadas
CLI	<i>Command-Line Interface</i> – Interface de linha de comandos
CNN	<i>Convolutional Neural Network</i> – Rede Neural Convolucional
CRF	<i>Conditional Random Field</i> – Campo Aleatório Condicional
DCNF	<i>Deep Convolutional neural field</i>
EESC	Escola de Engenharia de São Carlos
FC	<i>Fully Connected</i> – Totalmente Conectada
FCRN	<i>Fully Convolutional Residual Network</i>
FPS	<i>Frames per Second</i> – Quadros por Segundo
GAN	<i>Generative Adversarial Network</i>
GPS	<i>Global Positioning System</i> – Sistema de Posicionamento Global
GPU	<i>Graphics Processing Unit</i> – Unidade de processamento gráfico
HCRF	<i>Hierarchical Conditional Random Field</i> – Campo Aleatório Condicional Hierárquico
HDD	<i>Hard Disk Drive</i> – Unidade de disco rígido
HS	<i>Hilbert Space</i> – Espaço de Hilbert
IA	Inteligência Artificial
ICMC	Instituto de Ciências Matemáticas e de Computação
ILSVRC	<i>ImageNet Large Scale Visual Recognition Challenge</i>
KIT	<i>Karlsruher Institut für Technologie</i> – Instituto tecnológico de Karlsruhe
LBP	<i>Local Binary Patterns</i>

LIDAR	<i>Light Detection And Ranging</i>
LRM	Laboratório de Robótica Móvel
LSTM	<i>Long Short-Term Memory</i>
MAP	<i>Maximum a Posteriori Probability</i> – Máxima Probabilidade a Posteriori
MCP	McCulloch-Pitts
MLP	Perceptron Multicamadas – Multilayer Perceptron
MR	<i>Mixed Reality</i> – Realidade Mista
MRF	<i>Markov Random Field</i> – Campo Aleatório de Markov
MSE	<i>Mean Squared Error</i> – Erro Quadrático Médio
NRF	<i>Neural Regression Forest</i>
RADAR	<i>Radio Detection And Ranging</i>
REL	<i>Mean Relative Error</i> – Erro Relativo Médio
ReLU	<i>Rectified Linear Unit</i> – Unidade Linear Retificada
RGB	<i>Red, Green and Blue</i> – Vermelho, Verde e Azul
RKHS	<i>Reproducing Kernel Hilbert Space</i>
RL	<i>Reinforcement Learning</i> – Aprendizado por reforço
RMSE	<i>Root Mean Squared Error</i> – Raiz do Erro Quadrático Médio
RNA	Rede Neural Artificial
RNN	<i>Recurrent Neural Network</i> – Rede Neural Recorrente
ROS	<i>Robot Operating System</i>
SfM	<i>Structure from Motion</i>
SIDE	<i>Single Image Depth Estimation</i> – Estimação de Profundidade a partir de Imagens Monoculares
SLAM	<i>Simultaneous Localization and Mapping</i>
SVM	<i>Support Vector Machines</i> – Máquinas de Vetor de Suporte)
SONAR	<i>Sound Navigation And Ranging</i>

SSD	<i>Solid State Drive</i> – Unidade de Estado Sólido
SVM	<i>Support Vector Machine</i> - Máquina de vetores de suporte
USP	Universidade de São Paulo
USPSC	Campus USP de São Carlos
VR	<i>Virtual Reality</i> – Realidade Virtual

SUMÁRIO

1	INTRODUÇÃO	23
2	REDES NEURAIS PROFUNDAS	29
2.1	Aprendizado de Máquina	29
2.2	Aprendizagem de Representação	30
2.2.1	Relevância da Representação	30
2.3	Redes Neurais Artificiais	32
2.4	Aprendizagem Profunda (<i>Deep Learning</i>)	34
2.4.1	Histórico	34
2.4.2	Redes Neurais Convolucionais	37
2.4.3	Topologias – Processamento de Imagens	38
2.4.3.1	Preenchimento (<i>Padding</i>)	40
2.4.3.2	Camada Convolucional	42
2.4.3.3	Camadas de <i>SubSampling</i> e <i>UpSampling</i>	43
2.4.3.4	Camada Deconvolucional	45
2.4.3.5	<i>Rectified Linear Unit</i> (ReLU)	45
2.4.4	<i>Algoritmos de Otimização</i>	46
2.4.5	Normalização de Lote (<i>Batch Normalization</i>)	47
2.4.6	Técnicas de Regularização	48
2.4.7	Arquiteturas de Redes Convolucionais de Referência	49
2.4.8	<i>Datasets</i>	54
2.4.8.1	<i>Dataset Augmentation</i>	54
2.4.9	<i>Frameworks</i> de desenvolvimento	54
2.5	Considerações Finais	55
3	PERCEPÇÃO VISUAL DE PROFUNDIDADE	57
3.1	Pistas Visuais de Profundidade	57
3.1.1	Pistas Proprioceptivas	58
3.1.2	Pistas Binoculares de Profundidade	58
3.1.3	Pistas Monoculares de Profundidade	59
3.1.4	Pistas Dinâmicas	60
3.2	Trabalho Correlatos	60
3.2.1	Métodos Supervisionados	64
3.2.2	Métodos Não-Supervisionados	68
3.2.3	<i>Datasets</i>	70
3.2.4	Aplicações	71

3.3	Considerações Finais	72
4	DENSIFICAÇÃO DE MAPAS DE PROFUNDIDADE	77
4.1	Mapas de Ocupação	77
4.2	Considerações Finais	82
5	MATERIAIS E MÉTODOS	83
5.1	Estimação de Profundidade – Pré-processamento	83
5.2	Imagens Contínuas de Profundidade	83
5.3	<i>Datasets</i>	84
5.3.1	<i>Data Augmentation</i>	86
5.4	Arquitetura da Rede	87
5.5	Funções de Custo	88
5.5.1	<i>Mean Squared Error (mse)</i>	88
5.5.2	<i>Scale-Invariant Mean Squared Error (eigen)</i>	89
5.5.3	<i>Adaptive BerHu Penalty (berhu)</i>	89
5.6	Nuvem de pontos a partir da estimação SIDE	89
6	RESULTADOS EXPERIMENTAIS	93
6.1	Descrição do Experimento	93
6.2	Detalhes da Implementação	93
6.3	Avaliação de Desempenho (<i>Benchmark</i>)	93
6.4	Métricas de Avaliação	95
6.5	Resultados	95
6.5.1	Densificação de Mapas (<i>Depth Completion Evaluation</i>)	95
6.5.2	Estimação de Profundidade (<i>Depth Estimation Evaluation</i>)	96
6.6	Predição em tempo real	101
6.6.1	Resultados do <i>framework</i> desenvolvido em ROS	104
7	CONCLUSÃO	107
	REFERÊNCIAS	111
	APÊNDICES	127
	APÊNDICE A – REPOSITÓRIO	129

1 INTRODUÇÃO

Plataformas robóticas estão cada vez mais presentes em nossa sociedade, visto que progressivamente têm realizado atividades de maior complexidade, o que diretamente implica na diversificação dos ambientes em que podem atuar. Um dos fatores determinantes por trás dessa inovação é o desenvolvimento de melhores sistemas de percepção, permitindo que entendam o ambiente ao seu redor, tão bem quanto, ou melhor que, humanos.

Para que um veículo navegue autonomamente com segurança é imprescindível que a plataforma mapeie o ambiente em sua volta, detectando potenciais obstáculos e superfícies navegáveis durante o percurso. Deste modo, estas plataformas devem, de alguma forma, ser capazes de extrair informações quanto à profundidade dos ambientes nos quais estão inseridas. Não somente a tarefa de detecção de obstáculos, como também segmentação semântica e compreensão de cenas podem se beneficiar consideravelmente quando medições de distância são disponibilizadas (CAO; WU; SHEN, 2018).

Atualmente, existem uma série de sensores (LiDAR, SONAR, RADAR, câmeras monoculares/estéreo) que permitem que isto seja feito, e por explorarem diferentes tecnologias estes dispositivos apresentam vantagens e desvantagens particulares de acordo com a luminosidade e as condições meteorológicas do ambiente. No entanto, diferentes aspectos podem restringir seu uso, principalmente quando avaliados sob critérios como preço, alcance, visibilidade, acurácia e confiabilidade das informações medidas (YAN; XU; LIU, 2016).

Diferentemente da Tesla¹ (KING, 2019), a maior parte das companhias que desenvolvem carros autônomos acredita que LiDARs sejam os sensores mais indicados para o mapeamento 3D de ambientes, dados o elevado nível de precisão e a quantidade de dados fornecidos. Entretanto, esses dispositivos apresentam medições esparsas, até mesmo ruidosas, do ambiente em questão, além de serem extremamente caros dependendo do alcance e do nível de detalhes exigidos pela aplicação (SCHWARZ, 2010). Normalmente, radares e sonares são aparatos auxiliares ao sistema de percepção, tendo como funções medição de movimento e detecção de obstáculos próximos, respectivamente. Câmeras também podem ser utilizadas como principal sensor, além de serem dispositivos mais baratos que os sensores mencionados anteriormente, possibilitam a captura de cores, o que as torna bastante indicadas para interpretação de cenas, segmentação e detecção de objetos, por exemplo. Adicionalmente, também permitem a extração de profundidade e reconstrução de estruturas 3D a partir de observações 2D da cena (REZENDE et al., 2016).

¹ **Tesla Motors, Inc.:** Empresa automotiva americana especializada na manufatura de carros elétricos e de equipamentos para de armazenamento de energia.

Em contrapartida, os *softwares* que realizam o processamento das imagens exigem elevado custo computacional, dado o número de operações envolvidas para extração das informações de interesse, que nem sempre são prontamente disponibilizadas pelas câmeras. Como é o caso da profundidade, estimá-la a partir de imagens 2D é uma tarefa computacionalmente desafiadora, sendo descrita como um problema mal-posto, uma vez que as imagens observadas podem ser resultantes de várias projeções possíveis da cena real no mundo (EIGEN; PUHRSCHE; FERGUS, 2014). Tecnicamente, este tipo de informação pode ser recuperada pela correspondência de duas ou mais perspectivas (Visão Estéreo) (CHEN et al., 2015; ŽBONTAR; LECUN, 2015a; LUO; SCHWING; URTASUN, 2016) ou por estimação de profundidade a partir de imagens monoculares (SIDE, do inglês *Single Image Depth Estimation*) (EIGEN; PUHRSCHE; FERGUS, 2014; LIU; SHEN; LIN, 2015; GODARD; AODHA; BROSTOW, 2017; FU et al., 2018).

Em aplicações robóticas, esta tarefa é comumente realizada por métodos estéreo, os quais se fundamentam na percepção da estrutura 3D do ambiente por meio da pista visual Estereopsia (*Stereopsis*) (HOWARD; ROGERS, 1995). Entretanto, nem todas aplicações permitem que sejam utilizados, como é o caso de robôs de pequeno porte, que normalmente apresentam sistemas embarcados com poder computacional e autonomia limitados (POGGI et al., 2018); e de aplicações médicas em que os instrumentos de investigação (endoscópios, por exemplo) apresentam restrições físicas (FORSTER; TOZZI, 2000; MAHMOOD; DURR, 2017). No contexto de carros autônomos, a limitação mais preocupante de métodos estéreo é o fato de apresentarem dificuldades em computar correspondências em regiões refletoras ou com baixa textura. Métodos SIDE são capazes de inferir os valores de profundidades destas regiões, porém apresentam certa imprecisão ao longo das bordas dos objetos, como ilustrado na figura 1.

Figura 1: Comparação qualitativa entre o mapa de profundidade obtido por visão estéreo contra o obtido por método monocular. Cores mais quentes e mais frias representam distâncias maiores e menores, respectivamente.

Fonte: Imagem adaptada de Laina et al. (2016).

Este trabalho concentra-se no segundo tipo de abordagem, que, de modo geral,

Figura 2: Comparativo em nível de esparsidade entre os *datasets* (a) *KITTI Discrete* (esparso), (b) *KITTI Depth* (semi-denso) e (c) *KITTI Continuous* (denso, proposto). Cores mais quentes e mais frias representam distâncias maiores e menores, respectivamente.

Fonte: Elaborada pelo autor.

busca desenvolver soluções capazes de superar algumas das limitações presentes em métodos estéreo, ao passo que se almeja obter nível de detalhes semelhante ao de mapas de profundidades² gerados por correspondência estéreo.

Os métodos SIDE exploram, de certa forma, as pistas visuais monoculares para recuperar a informação de profundidade (SAXENA; SCHULTE; NG, 2007), e mais recentemente, métodos de aprendizado de máquina para diretamente relacionar as intensidades dos píxeis aos mapas de profundidade (*cor* $\rightarrow$ *profundidade*). Assim como em outras aplicações de visão computacional, aprendizagem profunda (*Deep Learning*) por redes convolucionais (CNNs, do inglês *Convolutional Neural Networks*) também é uma das técnicas exploradas, as quais impactaram positivamente em aplicações SIDE, no caso, vêm apresentando melhorias significativas na precisão das estimativas e no nível de detalhes presentes nos mapas de profundidade. Muitos desses métodos modelam a tarefa SIDE como

² **Mapa de profundidade:** Imagem em nível de cinza, onde píxeis com tons mais claros representam valores de distância mais próximos ao observador (câmera), e tons mais escuros, valores mais distantes.

um problema de regressão, sendo em sua maioria supervisionados, os quais frequentemente utilizam mapas de profundidade esparsos como referência, uma vez que são prontamente disponibilizados pelas medições de outros sensores (Ex.: LiDAR).

No entanto, o grau de esparsidade presente nesses mapas é muito alto. Por exemplo, em uma imagem (375×1242 píxeis) do conjunto de dados do *KITTI Depth*, 84,78 % de seus píxeis não contêm informações válidas. Para esses píxeis, o LiDAR não consegue adquirir dados devido a feixes de luz que não refletem de volta ao sensor (por exemplo, céu, superfícies refletoras), que não poderiam ser projetados para o sensor de imagem da câmera, ou talvez, o sensor seja apenas esparsos e os valores medidos não caíam nesses píxeis. Para o treinamento efetivo de redes neurais profundas, um dos paliativos encontrados para superar esta falta de informação é usar conjuntos de dados que forneçam um grande número de exemplos. Estritamente para a tarefa SIDE em ambientes externos, os *datasets KITTI Raw Data* (GEIGER et al., 2013) e *KITTI Depth* (UHRIG et al., 2017), satisfazem o requisito mencionado acima, no entanto, também apresentam o problema de alta esparsidade em seus mapas de profundidade. Recentemente, alguns trabalhos também propõem o uso de informações secundárias (mapas de profundidade de baixa resolução, superfícies normais, mapas semânticos), associando-as às imagens RGB como entradas adicionais (MA; KARAMAN, 2018; LIAO et al., 2017; ZHANG; FUNKHOUSER, 2018), ou então focam no desenvolvimento de arquiteturas de rede que são mais adequadas para o processamento de informações esparsas (UHRIG et al., 2017; HUA; GONG, 2018). Trabalhos similares também propuseram o uso de imagens de profundidade renderizadas de *datasets* sintéticos, que são contínuas (MANCINI et al., 2017; MANCINI et al., 2018). No entanto, o poder de generalização das redes treinadas usando esse tipo de conjunto de dados ainda é questionado (UHRIG et al., 2017), principalmente devido à lacuna existente em termos do grau de realismo entre ambientes virtuais e cenas do mundo real.

Dentro do contexto de percepção visual para veículos autônomos, este trabalho apresenta uma técnica de pré-processamento que beneficia o treinamento de redes convolucionais profundas usadas para recuperar a informação de profundidade de cenas externas (ruas, rodovias) a partir de imagens monoculares (SIDE). Mais especificamente, propôs-se o uso de um método de densificação para interpolar regiões com informações válidas (provenientes das medições esparsas do LiDAR) e gerar mapas contínuos, que então servem para treinar de maneira supervisionada uma rede convolucional profunda. No caso, mapas de ocupação foram utilizados como técnica de densificação e uma rede baseada na arquitetura *ResNet* foi adotada para mapear imagens RGB aos mapas contínuos de profundidade. O mesmo conceito também pode ser aplicado a outras técnicas de densificação e arquiteturas de rede. A principal contribuição deste trabalho é mostrar que, ao explorar esses mapas contínuos, pode-se tornar a convergência de rede mais rápida e fácil, enquanto utiliza-se menos pares de imagens para treiná-la, e – mais importante – melhora a qualidade de suas predições na etapa de teste. Finalmente, para demonstrar o benefício de treinar

redes convolucionais profundas usando o método proposto, comparou-se as estimativas obtidas quando treinadas em três conjuntos de dados diferentes com diferentes níveis de esparsidade, como ilustrado na Figura 2.

Para produzir os modelos de ocupação necessários para projeções contínuas, utilizou-se o framework para Mapas de Hilbert, devido às suas propriedades eficientes de treino e consulta, e escalabilidade para grandes conjuntos de dados. Anteriormente, os mapas de Hilbert eram estritamente relacionados aos problemas de planejamento de trajetórias (*Motion Planning*) como uma forma de melhorar as tarefas de reconhecimento de objetos e de aprendizado por reforço (*Reinforcement Learning*) (RAMOS; OTT, 2016). Em vez disso, este trabalho submeteu a técnica ao contexto de estimação de profundidade, onde foi usada como um método de densificação de mapas, provando ser capaz de beneficiar diretamente a qualidade das predições da rede neural profunda proposta.

De modo geral, o método SIDE proposto restringe-se à aplicação de veículos inteligentes, o que implica na utilização de *datasets* dentro deste contexto (Ex.: KITTI, *CityScapes*, etc), os quais também precisam disponibilizar milhares de imagens RGB (entrada), predominantemente contendo cenas de tráfego urbano, e mapas de profundidade correspondentes (saída) para o treinamento das redes convolucionais. Além disso, as redes desenvolvidas devem ser suficientemente profundas para que sejam capazes de inferir a informação de profundidade, mas também precisam possibilitar a execução do método em tempo real (tempo de inferência curto).

2 REDES NEURAIS PROFUNDAS

Neste capítulo estão descritas as técnicas utilizadas no desenvolvimento do método proposto, majoritariamente apresentando os procedimentos presentes na literatura para o desenvolvimento uma rede neural profunda para a tarefa SIDE.

2.1 Aprendizado de Máquina

Aprendizado de Máquina (*Machine Learning*) é uma ramificação da ciência da computação, dentro do campo de inteligência artificial e que está diretamente relacionada com reconhecimento de padrões e a teoria do aprendizado computacional, sendo área especializada no estudo e desenvolvimento de algoritmos inteligentes, isto é, algoritmos capazes de extrair conhecimento. A primeira descrição deste conceito foi apresentada no final dos anos 60:

“*Machine Learning*, campo de estudo que dá aos computadores a habilidade de aprender sem serem explicitamente programados.” — Arthur Lee Samuel, 1959

Mitchell (1997) também definiu o conceito de forma bastante sucinta: “É dito que um programa de computador aprende com a experiência E em relação a um certo conjunto de tarefas T e medida de desempenho P, se o seu desempenho na tarefa T, medido por P, melhora com a experiência E”. Além disso, também pode ser referido como um conjunto de técnicas computacionais capazes de automaticamente identificar padrões e regras nas amostras (conjunto de treinamento) para posteriormente classificá-las, prever comportamentos de dados futuros, ou então, tomar decisões em situações de incerteza (MURPHY, 2012). A descrição deste conceito foi atualizada ao passar do tempo, tornando-se cada vez mais abrangente, e remodelada devido ao surgimento de técnicas que também podem realizar as tarefas descritas anteriormente. Em seu livro, François Chollet descreve o conceito como:

“Um modelo de aprendizado de máquina transforma seus dados de entrada em resultados significativos, um processo que é “aprendido” a partir de sua exposição a exemplos conhecidos de entradas e saídas.” — Chollet (2017)

Geralmente, estas técnicas podem ser classificadas de acordo com o modo de aprendizado: supervisionado, não-supervisionado, dentre outros.

O **aprendizado supervisionado** utiliza um conjunto de amostras corretamente rotuladas ($\mathcal{D}$), onde cada amostra (x_i, y_i) consiste no dado de entrada (x_i) e seu correspondente dado de saída (rótulo, y_i) para treinamento da tarefa desejada. Matematicamente, o

conjunto de treinamento para este tipo de aprendizado poder ser expresso pela equação 2.1:

$$\mathcal{D} = \left\{ (x_i, y_i)_{i=1}^N \right\}, \quad (2.1)$$

onde cada amostras é indexada por i e N é número total de amostras.

No **aprendizado não-supervisionado**, o conjunto de treinamento é composto apenas por amostras contendo os dados de entradas (x_i) , como apresentado pela equação 2.2, cujos métodos buscam encontrar similaridades entre as instâncias de treinamento. Essencialmente, os problemas que trabalham com este último tipo de conjunto de dados são mais complexos, uma vez que não possuem funções de custo diretas, isto é, não minimizam o erro das saídas e os valores desejados diretamente, além de precisarem determinar as relações/padrões existentes apenas por meio de observações dos dados disponibilizados.

$$\mathcal{D} = \left\{ (x_i)_{i=1}^N \right\} \quad (2.2)$$

Em suma, aprendizado de máquina engloba diferentes técnicas de aprendizado como árvores de decisão, redes neurais artificiais, avaliação de hipóteses, aprendizado Bayesiano, aprendizagem baseada em instâncias, dentre outros. Mesmo aplicando diferentes formas de aprendizados, um ponto em comum é que estes métodos buscam encontrar representações matematicamente e computacionalmente convenientes para a tarefa a ser aprendida.

2.2 Aprendizagem de Representação

Como apresentado anteriormente, no contexto de aprendizado de máquina, encontra-se uma variedade de métodos de aprendizado, os quais processam diferentes tipos de dados e proporcionam um número ainda maior de aplicações, uma vez que segmentam dados, reconhecem padrões, realizam tomadas de decisão, inferem ações, dentre outros. Entretanto, um questionamento vigente é quanto à informação extraída dos conjuntos de dados. Uma vez induzidas pelos algoritmos, o que são cada uma destas representações extraídas dos dados? A indagação vai mais além. O que faz uma representação dos dados melhor que outra? De modo geral, a melhor representação dependerá da escolha da tarefa subsequente, uma vez que a melhor solução será aquela que tornar o aprendizado da tarefa posterior mais fácil (GOODFELLOW; BENGIO; COURVILLE, 2016).

2.2.1 Relevância da Representação

O campo de pesquisa chamado aprendizagem de representação (*Representation Learning*) busca conhecer representações dos dados de modo que possibilitem a extração mais fácil de informações úteis ao treinar classificadores e preditores

(BENGIO; COURVILLE; VINCENT, 2013). Em outras palavras, a aprendizagem de características (*feature learning*) é um conjunto de técnicas aplicadas sobre os dados brutos, permitindo que seja possível um sistema descobrir automaticamente as representações mais indicadas para a detecção ou classificação de características.

A utilização destas técnicas substituem o chamado engenheiramento de características manuais (*Feature Engineering*), uma vez que permite que os algoritmos aprendam as características por si só, aprendendo abstrações desde o nível mais baixo aos conceitos de mais alto nível. Como pontuado por Bengio (2009), os pesquisadores desta área têm almejado desenvolver algoritmos de aprendizagem capazes de descobrir tais representações com tão pouco esforço como os humanos fazem, isto é, sem ao menos precisar definir manualmente todas as abstrações necessárias para a concepção de um conceito ou ter que providenciar um grande conjunto de exemplos rotulados a mão (*hand-labeled samples*) e heurísticas que insiram algum tipo de conhecimento prévio.

O estudo das representações visuais aprendidas pode até mesmo apresentar as mesmas representações encontradas no contexto biológico. Alguns trabalhos desenvolveram arquiteturas de redes neurais capazes de mimetizar propriedades semelhantes às dos neurônios da região V1 do córtex visual. Mais especificamente, os mapas de características (*feature maps*) obtidos pelas redes neurais são semelhantes aos campos receptivos destas células, os quais operam como filtros de Gabor, isto é, agem como filtros de passagem de banda do espectro luminoso. Estes modelos reforçam a hipótese de codificação eficiente, a qual postula que estes neurônios são responsáveis por codificar a informação visual da forma mais eficiente possível (LEE; EKANADHAM; NG, 2008; LEE et al., 2009).

Figura 3: Filtros obtidos pelo algoritmo de aprendizagem de características invariantes aplicado ao conjunto de caracteres MNIST (imagem à esquerda). Filtros seletivos obtidos pela modelagem linear das representações visuais da visão humana (imagem à direita).

Fonte: Imagem adaptada de Ranzato et al. (2007) e Bruce, Rahman e Carrier (2016).

Alguns trabalhos propuseram arquiteturas que aprendessem hierarquias de características invariantes (*invariant features hierarchies*), cujas camadas escondidas são estudadas principalmente para compreender melhor como os mapas de características gerados contribuem para extração de informações visuais (RANZATO et al., 2007; LECUN, 2012). Como pode-se observar na Figura 3, os filtros aprendidos para processamento

das imagens, mesmo pertencendo a diferentes contextos – computacional e biológico – compartilham semelhanças.

Em visão computacional, tem-se como objetivo encontrar vetores de características que descrevam informações presentes na imagem (ou parte dela). No caso de extratores clássicos, isto é realizado utilizando descritores de imagem, que se baseiam em informações presentes nos histogramas, nos momentos invariantes de Hu ou Zernike, nas características de Haralick ou *Local Binary Patterns* (LBP), dentre outros, para extrair informações quanto a cor, forma e textura, respectivamente (GONZALEZ; WOODS, 2002). Mais recentemente, técnicas como redes neurais artificiais e redes neurais profundas têm sido empregadas para aprender automaticamente características mais adequadas para uma dada aplicação.

2.3 Redes Neurais Artificiais

Redes neurais artificiais (RNAs) são modelos computacionais inspirados no sistema nervoso central de animais, de modo a mimetizar os processos de aprendizado do neurônio biológico (SILVA; SPATTI; FLAUZINO, 2010). Dentre as topologias de RNA que podem ser construídas, o Perceptron Multicamadas (MLP do inglês *Multilayer Perceptron*) é a arquitetura mais conhecida, uma vez que permite que uma série de problemas, especialmente em reconhecimento de padrões e regressão numérica, sejam resolvidos, sendo isto decorrente da capacidade de generalização deste tipo de topologia. A estrutura básica utilizada é o neurônio artificial, cujo primeiro modelo matemático foi inspirado na geração e propagação do potencial de ação gerado por um neurônio biológico, proposto por McCulloch e Pitts (1943). Com o passar dos anos, o modelo foi constantemente aprimorado, onde muitos outros também foram propostos, como o modelo elétrico de Hodgkin e Huxley (1952), cujo estudo tinha como objetivo investigar a capacidade computacional e emular a funcionalidade dos axônios biológicos.

Como ilustrado à esquerda da Figura 4, a saída do neurônio artificial (y_i) é resultado da função de ativação (f) sobre o nível de ativação (a_i) que é a ponderação de suas r entradas ($x_1, x_2, \dots, x_r$) pelos pesos sinápticos ($w_{i,1}, w_{i,2}, \dots, w_{i,r}$) menos o valor de limiar de disparo (b_i). A equação 2.3 explicita matematicamente como esses valores se relacionam.

$$a_i = w_{i,1}x_1 + w_{i,2}x_2 + \dots + w_{i,r}x_r + b_i \quad (2.3)$$

onde:

x_j = j -ésima entrada do neurônio,

w_{ij} = (i, j)-ésimo valor de peso sináptico,

b_i = Limiar de disparo do i -ésimo neurônio,

a_i = Nível de ativação resultante da ponderação das entradas do i -ésimo neurônio.

Figura 4: Modelo canônico de um perceptron multicamadas (MLP) e estrutura interna de um neurônio artificial

Fonte: Imagem adaptada de Tanikić e Despotovic (2012).

O nível de ativação a_i é então submetido a função de ativação, a qual define o sinal sináptico de saída do neurônio, isto é, avalia se a combinação ponderada das entradas é suficiente para dispará-lo. Como apresentado na equação 2.4:

$$y_i = f(a_i) = f\left(\sum_j^r x_j w_{i,j} + b_i\right), \quad (2.4)$$

onde $f(a_i)$ é equivalente ao sinal sináptico do i -ésimo neurônio, y_i .

Diferentes funções de ativação podem ser empregadas, desde que sejam monótonas crescentes e diferenciáveis. Estes requisitos são impostos pelos algoritmos de otimização, uma vez que normalmente baseiam-se na estratégia por Descida de Gradiente Estocástico (*Stochastic Gradient Descent*), sendo usados para iterativamente minimizar uma dada função de custo. Para isso, utiliza-se o algoritmo de retropropagação dos erros, mais conhecido como (*Backpropagation*) para reajustar os pesos sinápticos dos neurônios das camadas escondidas, camadas mais internas da rede (FELDMAN; ROJAS, 2013). As funções mais utilizadas são: linear, limiar, logística, tangente hiperbólica e *softmax* (BISHOP, 2006), mostradas nas equações 2.5 a 2.9 a seguir:

Também é possível observar na Figura 4 que as redes neurais utilizam vários neurônios, os quais são densamente interconectados, recebendo sinais de entrada tanto das entradas da rede quanto de neurônios intermediários. Estruturalmente, os neurônios são classificados em camadas de entrada, escondidas e de saída, tendo em consideração onde suas entradas e saídas estão conectadas. A rede neural em si recebe um vetor de entrada $I = [i_1, i_2, \dots, i_n]$, cujas relações são processadas por k camadas escondidas, onde os neurônios da última camada geram m saídas $O = [o_1, o_2, \dots, o_m]$, apresentando assim as previsões para qual a topologia de rede foi treinada. Sob outra perspectiva, redes neurais comportam-se como “máquinas de mapeamento” uma vez que são capazes de modelar funções $F: \mathbb{R}^n \mapsto \mathbb{R}^m$ (FELDMAN; ROJAS, 2013).

2.4 Aprendizagem Profunda (*Deep Learning*)

Aprendizagem profunda é uma das subáreas de aprendizado de máquina (Figura 5) que recentemente tem recebido maior notoriedade, isto graças aos seus resultados bastante promissores no campo de inteligência artificial e visão computacional, o que tem contribuído significativamente no desenvolvimento de sistemas inteligentes e sistemas de percepção para plataformas robóticas. Outros nomes como aprendizado de representações em camadas (*layered representations learning*) ou hierárquicas (*hierarchical representations learning*) também são apropriados para referir-se à técnica, uma vez que é uma abordagem que enfatiza o aprendizado de camadas sucessivas a partir dos dados de entrada, gerando mapas de características (representações) cada vez mais significativos (CHOLLET, 2017).

2.4.1 Histórico

O termo *Deep learning* em si é bastante antigo, sendo introduzido por Rina Dechter em 1986, porém apresentando uma conotação diferente da que vem sendo

popularizada desde 2006. O termo também tem sido utilizado para demarcar um novo período em aprendizado de máquina, distinguindo esta nova abordagem de técnicas anteriores a ela, como as desenvolvidas durante os períodos do *cybernetics* (1940-1960) e do *connectionism/neural networks* (1980-1990). O histórico das redes neurais pode ser subdividido em três momentos bastantes característicos, sendo eles referentes a sua criação, interrupção e reascensão (SCHMIDHUBER, 2015; GOODFELLOW; BENGIO; COURVILLE, 2016).

Figura 5: Neste contexto, um método computacional, por meio de muitos dados (aprendizagem profunda), aprende características mais eficientes (aprendizado de representação), que é, por sua vez, um tipo de algoritmo em aprendizado de máquina, e que pode ser usado para muitas, mas não todas, abordagens em IA.

Fonte: Imagem adaptada de Goodfellow, Bengio e Courville (2016).

O primeiro momento é referente ao período de aproximação entre redes neurais e neurociência, no qual desenvolveu-se modelos computacionais bioinspirados em neurônios (*Perceptron*, *Neocognitron* (FUKUSHIMA, 1980)). Alguns anos depois, modelos lineares passaram a ser questionados, uma vez que apresentavam dificuldades em solucionar o problema do ou-exclusivo (XOR) que consistia em utilizar redes neurais na tentativa de prever corretamente as saídas desta porta lógica dadas duas entradas binárias. Diferentemente das funções booleanas “AND” e “OR”, esta função lógica não é linearmente separável. Deste modo, o problema não poderia ser solucionado por um perceptron de camada única, demandando pelo menos duas retas discriminatórias para realizar a separação de suas saídas. Este episódio agravou severamente a popularidade de redes neurais, cuja solução foi a adição de mais uma unidade de processamento à camada escondida (MINSKY;

PAPERT, 1969). Este primeiro momento deixou como legado a representação distribuída e o algoritmo de retropropagação (*Backpropagation*) (RUMELHART; HINTON; WILLIAMS, 1986).

O segundo momento é referente ao hiato entre meio-1990s até 2007, período no qual a técnica ficou desacreditada, esta fase também é referida como a queda do movimento conexionista. Os motivos que levaram a esta situação foram as promessas ambiciosas a investidores, que não se cumpriram, e o sucesso de outras técnicas (*Kernel machines* e *Graphical models*). Neste meio tempo, a linha de pesquisa manteve-se ativa graças aos pesquisadores Geoffrey Hinton, Yoshua Bengio e Yann LeCun, e ao *Canadian Institute for Advanced Research* (CIFAR) (GOODFELLOW; BENGIO; COURVILLE, 2016).

O terceiro momento é referente ao período da aprendizagem profunda (2006-Hoje). A credibilidade da técnica retornou com criação de um novo método de pré-treinamento (*greedy layer-wise*) que possibilitava a aprendizagem de redes neurais mais profundas, como a *Deep Belief Network* desenvolvida por Hinton, Osindero e Teh (2006). Desta vez, as arquiteturas de rede desenvolvidas apresentavam desempenho superior aos sistemas concorrentes baseados em outras tecnologias de aprendizagem de máquinas.

Tabela 1: Marcos importantes na história do aprendizado de máquinas, abordando exclusivamente acontecimentos referentes a redes neurais clássicas e profundas.

Milestone, Contribuição	Contribuidor, Ano
Modelo MCP, considerado o antepassado de Redes Neurais Artificiais	McCulloch & Pitts, 1943
Regra de aprendizado hebbiano (<i>Hebbian Learning Rule</i>)	Hebb, 1949
Primeiro Perceptron	Rosenblatt, 1958
Retropropagação (<i>Backpropagation</i>)	Werbos, 1974
<i>Neocognitron</i> , considerado o antepassado de Redes Neurais Convolucionais	Fukushima, 1980
Máquina de Boltzmann (<i>Boltzmann Machine</i>)	Ackley, Hinton & Sejnowski, 1985
Máquina de Boltzmann restrita (<i>Restricted Boltzmann Machine</i>), ou também <i>Harmonium</i>	Smolensky, 1986
Redes Neurais Recorrentes (<i>Recurrent Neural Networks</i>)	Jordan, 1986
Autoencoders	Rumelhart, Hinton & Williams, 1986
Aprendizado por Reforço (<i>Reinforcement learning</i>)	Ballard, 1987
<i>LeNet</i> , início da era das redes neurais convolucionais	Watkins, 1989
LSTM (<i>Long-Short Term Memory</i>)	LeCun, 1990
Redes de Crença Profunda (<i>Deep Belief Networks</i>), inaugurando a “era da aprendizagem profunda”	Hochreiter & Schmidhuber, 1997
Máquina de Boltzmann Profunda (<i>Deep Boltzmann Machine</i>)	Hinton, 2006
<i>Kaggle</i> , lançamento da plataforma para competições em aprendizado de máquina	Salakhutdinov & Hinton, 2009
<i>AlexNet</i> , começando a era de CNNs serem usadas para a classificação no ImageNet ¹	Kaggle, 2010
<i>AlphaGo</i> , superando humanos em Go	Krizhevsky, Sutskever & Hinton, 2012
<i>OpenAI Five</i> , superando humanos em DotA 2	DeepMind, 2016
	OpenAI, 2018

Fonte: Tabela adaptada de Voulodimos et al. (2018).

¹ **ImageNet:** é um grande banco de imagens projetado para pesquisa e desenvolvimento de

A Tabela 1 apresenta resumidamente os marcos mais relevantes e as principais contribuições para o assunto.

Redes neurais clássicas podem ser utilizadas em diferentes aplicações, tais como: classificação de padrões, agrupamento de dados, controle de processos, sistemas de previsões, dentre outros (SILVA et al., 2016). Analogamente, redes neurais profundas também são capazes de desempenhar as tarefas mencionadas em diferentes aplicações, e por conta disso, vêm sendo amplamente empregadas em:

1. Mineração de dados (WITTEN et al., 2016)
2. Robótica
 - a) Visão computacional – Identificação e rastreamento de objetos (Pedestres, automóveis, etc.) (HELD; THRUN; SAVARESE, 2016; BRESILLA et al., 2018)
 - b) Robôs e veículos autônomos (LENZ; LEE; SAXENA, 2015; CHEN et al., 2015)
 - c) Identificação de modelos dinâmicos (CUTLER; HOW, 2016; PUNJANI; ABBEEL, 2015)
3. Finanças – Predição de movimentações financeiras (DING et al., 2015; SINGH; SRIVASTAVA, 2016)
4. Médicas, farmacêuticas e genéticas – Análise de imagens médicas (RAJPURKAR et al., 2017; FRID-ADAR et al., 2018)
5. Segurança – Identificação de faces, gestos, etc. (SUN; WANG; TANG, 2014; YUAN et al., 2014)

2.4.2 Redes Neurais Convolucionais

Antes do advento de redes neurais convolucionais, o caminho encontrado para resolver problemas mais complexos era o de incrementar o número de camadas escondidas das MLPs. Entretanto, esta ação dificulta ainda mais o treinamento destas arquiteturas, uma vez que o número de parâmetros treináveis aumenta consideravelmente. Em outras palavras, o espaço Vapnik–Chervonenkis de características que precisam ser aprendidas pelo regressor neural apresenta uma maior dimensão. Consequentemente, exige-se um alto custo computacional e uma numerosa base de dados para o ajuste dos pesos sinápticos sem que ocorra sobreajuste (*overfitting*) (LUXBURG; SCHOELKOPF, 2008), mas não necessariamente representará um aumento em sua capacidade de generalização.

Diante deste problema, LeCun et al. (1989) propuseram substituir as camadas intermediárias altamente conectadas por filtros convolucionais, o que permite o compartilhamento dos valores dos pesos/parâmetros treináveis para todos os canais de uma mesma *softwares* para reconhecimento visual de objetos.

camada, ao mesmo tempo que conserva-se a relação espacial das entradas. A inspiração originou-se em estudos do modelo biológico do sistema visual de gatos (FUKUSHIMA, 1980). E a técnica empregada é conhecida como *weight sharing*, primeiramente descrita por Rumelhart, Hinton e Williams (1986), e que além de facilitar a detecção de características particulares em qualquer localização de uma imagem, também reduz consideravelmente o número de parâmetros livres.

As modificações anteriores definiram um novo tipo de topologia, que mais recentemente tem sido denominada como rede neural convolucional (CNN do inglês *Convolutional Neural Network*). CNNs diferem-se em virtude de utilizarem máscaras convolucionais para extrair características (*features*) automaticamente durante o treinamento, isto é realizado aplicando o filtro convolucional sobre o campo receptivo em análise e deslizando-o ao longo das dimensões da entrada. Já RNAs clássicas possuem muito mais conexões sinápticas e exigem que o tamanho da entrada seja fixo, em razão de interligarem individualmente um único neurônio a todos os outros neurônios da próxima camada. Resumidamente, estes dois tipos de redes diferenciam-se principalmente na forma que os neurônios se relacionam entre camadas adjacentes, como ilustrado na Figura 6.

Figura 6: Ilustração de arquiteturas arbitrárias de uma MLP e uma CNN (tamanho do *kernel* = 3)

Fonte: Imagem adaptada de Theano Development Team (2018).

2.4.3 Topologias – Processamento de Imagens

CNNs podem possuir tipos de topologias bastante peculiares, uma vez que podem combinar diferentes camadas/módulos para comporem sua arquitetura, sendo esta subdivi-

dida em duas partes: *encoder* e *decoder*. *Encoder* é o nome designado ao conjunto das primeiras camadas da arquitetura que são responsáveis por extrair características que sejam mais representativas dos dados de entrada. Para recuperar as informações contidas nos mapas de características, comprimidas pelo *encoder*, pode-se acoplar um conjunto de camadas de saída, *decoder*, para decodificá-las. Especificamente para visão computacional, as topologias podem diferir no modo que as camadas do *decoder* estão dispostas, o que diretamente implica na forma que as previsões de saída são computadas. Normalmente, as arquiteturas básicas mais utilizadas são: CNN-FC e FCN.

A CNN com camadas totalmente conectadas (CNN-FC do inglês *CNN with Fully-Connected layers*), como nome sugere, possui um conjunto de neurônios totalmente conectados, bem como a arquitetura de MLPs. A Figura 7 ilustra uma rede com este tipo de arquitetura, onde, especificamente para este exemplo, cada saída é responsável por classificar a imagem de entrada em uma das classes do *dataset* MNIST (VERNWAL, 2017). Esta topologia também pode ser facilmente modificada de modo que o número de saídas corresponda ao número de píxeis e de canais, ou seja, às dimensões de uma imagem de saída, fazendo com que cada saída estime o valor de um único píxel desta imagem (*pixel-wise prediction*) (EIGEN; PUHRSCHE; FERGUS, 2014).

Já a rede totalmente convolucional (FCN do inglês *Fully Convolutional Network*), ilustrada na Figura 8, emprega filtros “deconvolucionais”², os quais realizam a sobreamostragem dos mapas de características de saída no intuito de recuperar a resolução original da imagem de entrada (LONG; SHELHAMER; DARRELL, 2015).

Figura 7: Topologia convencional de uma CNN-FC (*CNN with Fully-Connected layers*)

Fonte: Imagem adaptada de Vernwal (2017).

² Filtro “deconvolucional” é o termo mais comumente utilizado para se referir a este tipo de camada, porém tecnicamente é incorreto e não deveria ser assim denominado. Mais informações estão apresentadas na subseção 2.4.3.4.

Figura 8: Topologia convencional de uma FCN (*Fully Convolutional Network*)

Fonte: Imagem adaptada de Noh, Hong e Han (2015).

Diferentemente da primeira topologia, as FCNs são puramente convolucionais, cuja arquitetura codifica e decodifica os mapas de características. Neste tipo de topologia, o *encoder* exerce a mesma função que na CNN-FC, porém as camadas FC do conjunto de camadas de saída (*decoder*) são substituídas por camadas deconvolucionais, as quais assumem a função de mapear, de forma densa, as características encontradas da entrada em análise às predições de saída. Devido à forma que este processamento é realizado, uma das vantagens de se utilizar este tipo de rede é a possibilidade de aplicar entradas de diferentes dimensões espaciais (LONG; SHELHAMER; DARRELL, 2015).

Uma vez descritas algumas das topologias de redes convolucionais possíveis, estão apresentados a seguir os elementos estruturais que as compõem. Mais precisamente, os próximos tópicos esclarecem a funcionalidade e as operações matemáticas das camadas convolucionais, deconvolucionais, de subamostragem (*Pooling*), normalização de lote, dentre outras.

2.4.3.1 Preenchimento (*Padding*)

Normalmente, esta técnica é empregada para evitar que a resolução espacial da imagem de entrada ou dos mapas de características seja reduzida quando alguma das operações realizadas nas camadas da CNN (convolução, *pooling*, dentre outras) são aplicadas. Além disso, este recurso permite que as dimensões e a profundidade dos mapas de características sejam alteradas de forma independente.

A seguir estão apresentados diferentes tipos de preenchimentos que podem ser utilizados. Entretanto, para simplificar sua análise, assumiu-se que tanto as dimensões das entradas ($i_1 = i_2$) e dos *kernels* ($k_1 = k_2$) quanto os *strides* ($s_1 = s_2$) e o *padding* ($p_1 = p_2$) utilizados são iguais ao longo dos eixos (DUMOULIN; VISIN, 2016).

Onde:

i = Tamanho da entrada,

k = Tamanho do *kernel*,

s = *Stride*, distância entre duas posições consecutivas do *kernel*,

p = *Zero padding*, número de zeros concatenados no início e no final de um eixo.

1. **No Padding**, $s \neq 1$

Para qualquer i , k e s , e para $p = 0$,

$$o = \left\lceil \frac{(i - k)}{s} \right\rceil + 1. \quad (2.10)$$

2. **Zero padding**, $s \neq 1$

Para qualquer i , k , p e s ,

$$o = \left\lceil \frac{(i + 2p - k)}{s} \right\rceil + 1. \quad (2.11)$$

3. **Half (same) Padding**

Para qualquer i , e para k ímpar ($k = 2n + 1, n \in \mathbb{N}$), $s = 1$ e $p = \left\lceil \frac{k}{2} \right\rceil = n$,

$$o = i + 2 \left\lceil \frac{k}{2} \right\rceil - (k - 1) = i + 2n - 2n = i. \quad (2.12)$$

4. **Full Padding**

Para qualquer i e k , e para $p = k - 1$ e $s = 1$,

$$o = i + 2(k - 1) - (k - 1) = i + (k - 1). \quad (2.13)$$

Figura 9: Ilustração comparativa dos diferentes tipos de *Padding* com *stride* unitário ($s = 1$).

Fonte: Imagem adaptada de Dumoulin e Visin (2016).

Entretanto, deve-se atentar ao uso excessivo da técnica, uma vez que informações contidas nas bordas das imagens, ou dos mapas de características, podem ser perdidas com o processo (GOODFELLOW; BENGIO; COURVILLE, 2016).

2.4.3.2 Camada Convolutacional

As camadas convolucionais são elementos essenciais na composição de CNNs, as quais utilizam diferentes máscaras para convoluir, não somente a imagem em análise, mas também mapas de características intermediários (GUO et al., 2016). Tecnicamente, estas máscaras nada mais são que matrizes utilizadas para ponderar características peculiares em uma imagem, sendo também denominadas como filtros ou *kernels*. Como já discutido na seção 2.2, durante o treinamento, os valores destes filtros são ajustados de modo que os mapas resultantes sejam representações mais eficientes e extraiam informações relevantes para aplicação. Algumas destas máscaras podem operar como detectores de borda, filtros derivativos, dentre outros, extraindo assim particularidades primárias da imagem.

Existem três principais vantagens para utilizar filtros convolucionais (ZEILER, 2014):

1. **Compartilhamento de Pesos (*Weight Sharing*)**: o compartilhamento de pesos em um mesmo mapa de características, o que reduz o número de parâmetros livres da rede.
2. **Conectividade Local**: filtros aprendem correlações entre píxeis vizinhos.
3. **Invariância de localização**: rede tornar-se robusta a translações do objeto na imagem.

Ao construir uma *convolutional layer*, deve-se determinar o tamanho do *kernel* (k), o passo/salto/*stride* (s), os valores iniciais dos pesos e o tipo de *padding* utilizado. Os dois primeiros hiperparâmetros são responsáveis por determinar como a operação de convolução será realizada espacialmente, uma vez que definem quais serão os píxeis considerados, ou não, no cálculo. Normalmente, no início do treinamento, estas camadas são preenchidas com valores escolhidos aleatoriamente dentro de um determinado intervalo. Isto pode ser feito empregando diferentes heurísticas na inicialização dos pesos (*weight initialization*), tais como: distribuições uniforme, $X \sim \mathcal{U}(a, b)$, e normal, $X \sim N(\mu, \sigma^2)$; inicializadores de Xavier (GLOROT; BENGIO, 2010) ou He (HE et al., 2015), por exemplo. Por fim, o *padding* determina a forma que os píxeis da borda devam ser tratados durante a operação de convolução. Em conjunto com os dois primeiros, estes parâmetros definem qual será o tamanho do mapa resultante, que pode preservar, ou não, a dimensão do mapa original.

Ao convoluir uma matriz de entrada $\mathbf{X}$ de dimensão $m \times n$ com um *kernel* $\mathbf{k}$ de dimensão $k \times k$ e deslizando-o a passos unitários ($s = 1$), o mapa de características resultante $\mathbf{Y}$ apresentará dimensão $(m - k + 1) \times (n - k + 1)$ (MENDES, 2017). Matematicamente,

esta operação pode ser expressa pela equação:

$$\mathbf{Y}_{(i, j)} = \sum_{i'=1}^k \sum_{j'=1}^k \mathbf{k}_{(i', j')} \mathbf{X}_{(i - \frac{k+1}{2} + i', j - \frac{k+1}{2} + j')}, \quad k, s \in \mathbb{N}. \quad (2.14)$$

onde:

$\mathbf{Y}_{ij}$ = (i, j) -ésimo elemento do mapa de características resultante $\mathbf{Y}$,

$\mathbf{k}_{i'j'}$ = (i', j') -ésimo elemento da máscara convolucional $\mathbf{k}$,

k = Tamanho da máscara convolucional,

$\mathbf{X}$ = Matriz de entrada $(x_{i''j''}) \in \mathbb{R}^{m \times n} : i'' = (i - \frac{k+1}{2} + i')$, $j'' = (j - \frac{k+1}{2} + j')$

m = Comprimento da matriz de entrada $\mathbf{X}$,

n = Largura da matriz de entrada $\mathbf{X}$.

Figura 10: Ilustração de um Filtro Convolucional.

Fonte: Imagem adaptada de Apple (2016).

O elemento central da máscara é posicionado sobre o píxel principal, cujo valor é substituído pela soma ponderada de si mesmo e seus vizinhos (*receptive field*), como ilustrado pela Figura 10.

2.4.3.3 Camadas de *SubSampling* e *UpSampling*

As camadas de subamostragem (*pooling layer*) e sobreamostragem (*unpooling layer*) são responsáveis por comprimir e expandir, respectivamente, as informações mais relevantes

presentes em um mapa de característica. Em outras palavras, são formas não-lineares de amostragem capazes de reduzir/aumentar espacialmente sua escala (*subsampling/upsampling*). No caso da primeira camada, este redimensionamento pode utilizar diferentes operações aritméticas como *max*, média, mediana, dentre outros. A *max pooling* é a mais comum delas (KRIZHEVSKY; SUTSKEVER; HINTON, 2012; SIMONYAN; ZISSERMAN, 2015), e um mapa $\mathbf{L}_{k+1}$ pode ser obtido deslizando o *kernel* de tamanho $n \times n$ sobre um mapa $\mathbf{L}_k$, utilizando a relação:

$$\mathbf{L}_{k+1}(i', j') = \max_{n \times n} \mathbf{L}_k(i, j), \quad (2.15)$$

onde $\mathbf{L}_k(i, j)$ é (i, j) -ésimo valor do k -ésimo mapa de características e $\mathbf{L}_{k+1}(i', j')$ é (i', j') -ésimo valor do $(k + 1)$ -ésimo mapa de características.

Figura 11: Exemplificação de filtros de *pooling* (à esquerda) e *unpooling* (à direita).

Fonte: Imagem adaptada de Noh, Hong e Han (2015).

A operação de *pooling* encontra-se ilustrada na Figura 11, a qual identifica qual o máximo valor presente dentre os neurônios avaliados (campo receptivo). A operação de *unpooling* realiza a operação inversa, e necessita de *switch variables* para identificar a posição original das ativações e restaurar os valores do *feature map* (NOH; HONG; HAN, 2015).

A utilização de camadas de *pooling* em arquiteturas *multilayers* é justificada pelo fato que mapas de características subamostrados apresentam *features* mais robustas que os mapas que os geraram, uma vez que são menos sensíveis a transformações (translação, rotação, mudanças no brilho e contraste) na imagem de entrada. Na prática, transformações como estas podem impedir que um objeto seja identificado corretamente (ZEILER; FERGUS, 2014).

2.4.3.4 Camada Deconvolucional

As camadas deconvolucionais são responsáveis por realizar a sobreamostragem dos *features maps* mais internos da rede, não como as camadas de *UpSampling*, que somente recuperam a resolução espacial (*unpooling*), mas também decodificando as informações neles comprimidas (*unpooling + convs*). O empilhamento de várias destas camadas permite que diferentes níveis de detalhes sejam recuperados (Figura 12), sendo capaz de modelar até mesmo um processo de sobreamostragem não-linear (*nonlinear upsampling*), onde as camadas mais internas capturam o escopo global de informações e as camadas mais próximas a saída aprendem os detalhes mais finos quanto a reconstrução dos objetos (LONG; SHELHAMER; DARRELL, 2015; NOH; HONG; HAN, 2015). Por mais disseminado que o termo deconvolução seja, a nomenclatura mais indicada é convolução transposta, uma vez que aritmeticamente os filtros deconvolucionais também realizam convoluções, recuperam a resolução original das características, mas não reconstituem exatamente os valores antes do processo de convolução (DUMOULIN; VISIN, 2016).

Figura 12: Exemplificação de filtros convolucionais (à esquerda) e deconvolucionais (à direita).

Fonte: Imagem adaptada de Noh, Hong e Han (2015).

2.4.3.5 Rectified Linear Unit (ReLU)

Bengio, Simard e Frasconi (1994) ao estudarem redes neurais recorrentes deparou-se com a dificuldade em treinar este tipo de rede por métodos de aprendizado baseado em gradientes, e previram que redes neurais mais profundas também seriam acometidas pelo mesmo problema. Atualmente, essa adversidade é bastante conhecida na área de aprendizado de máquina é o chamado problema do desaparecimento do gradiente (*Vanishing gradient problem*), nome designado à situação na qual existe uma complicação em realizar o ajuste dos pesos sinápticos da rede neural uma vez que os valores dos gradientes ficam demasiadamente pequenos.

Hahnloser et al. (2000) propuseram utilizar a equação 2.16 como função de ativação, que mais tarde provou ser mais adequada do que a função logística para o redes mais profundas, uma vez que introduz um efeito de esparsidade na topologia da rede. Consequentemente, possibilitando a obtenção de melhores valores de mínimos locais durante o treinamento, como demonstrado em (GLOROT; BORDES; BENGIO, 2011).

ReLU:

$$f(a_i) = a_i^+ = \max(0, a_i), \quad \text{onde } a_i \text{ é o nível de ativação resultante da ponderação das entradas do } i\text{-ésimo neurônio.}$$

(2.16)

2.4.4 Algoritmos de Otimização

Algoritmos de otimização são mecanismos matemáticos utilizados para minimizar os valores de erros apontados pela função de custo e ajustar os parâmetros de uma arquitetura de rede. Em conjunto com a técnica de *backpropagation*, a otimização por Descida de Gradiente é um dos métodos mais simples e mais comumente adotados para calcular o gradiente dos parâmetros a serem ajustados. Devido ao número elevado de parâmetros livres presentes em CNNs e a grande quantidade de instâncias que precisam ser aprendidas, a utilização desta técnica de otimização fica inviabilizada, dado o custo computacional requerido para o cálculo dos gradientes (PONTI; COSTA, 2018). Desta forma, algoritmos de otimização mais adequados a redes convolucionais foram elaborados.

Gradiente Descendente Estocástico (*Stochastic Gradient Descent, SGD*) sugere utilizar *mini-batches*, amostras aleatórias dos dados (Estocástico), para acelerar o treinamento. Isto fornece aproximações do Gradiente Descendente e dispensa a necessidade de processar todas as instâncias existentes de uma única vez. O ajuste dos pesos é definido pela seguinte expressão:

$$W_{t+1} = W_t - \eta \sum_{j=1}^B \nabla \mathcal{L}(W_t, x_j^B), \quad (2.17)$$

onde $\nabla \mathcal{L}(W, x_j^B)$ é o gradiente da função de custo segundo os pesos atuais da rede W_t e a j -ésima instância do *mini-batch* x_j^B , η é a taxa de aprendizado, e W_{t+1} são os novos valores de peso.

A convergência de um método otimizado por SGD não é suave, dada a aleatoriedade das amostras usadas para cálculo dos gradientes. Para contornar este problema, *momentum* e normalização podem ser utilizados, tais conceitos estão presentes nos seguintes algoritmos de otimização: *AdaGrad* (*Adaptive Gradient*) (DUCHI; HAZAN; SINGER, 2011), *AdaDelta* (*Adaptive learning rate*) (ZEILER, 2012) e *Adam* (*Adaptive moment estimation*) (KINGMA; BA, 2014).

Momentum permite ajustar a taxa de atualização (velocidade) dos pesos da rede. Mais especificamente, esta técnica controla o efeito das mudanças anteriores dos pesos para determinar a atual direção do gradiente no espaço de parâmetros, evitando que os novos valores de parâmetros W_{t+1} tenham um desvio muito abrupto da direção atual, como explicitado abaixo:

$$W_{t+1} = W_t + \alpha(W_t - W_{t-1}) + (1 - \alpha) [-\eta \nabla \mathcal{L}(W_t)], \quad (2.18)$$

onde α é a constante de ajuste do efeito de momento.

Adaptive Moment Estimation (Adam), calcula taxas adaptativas de aprendizado para cada parâmetro. No caso, a técnica armazena as médias tanto dos gradientes passados quanto os gradientes de segunda ordem, onde as influências em W e g decaem exponencialmente e são controladas, respectivamente, por α e γ (KINGMA; BA, 2014; PONTI; COSTA, 2018).

$$\hat{m}_{t+1} = \frac{m_{t+1}}{1 - \alpha_{t+1}}, \quad m_{t+1} = \alpha_{t+1}g_t + (1 - \alpha_{t+1})\nabla \mathcal{L}(W_t), \quad (2.19)$$

$$\hat{g}_{t+1} = \frac{g_{t+1}}{1 - \gamma_{t+1}}, \quad g_{t+1} = \gamma_{t+1}g_t + (1 - \gamma_{t+1})\nabla \mathcal{L}(W_t)^2, \quad (2.20)$$

onde m e g são os momentos de primeira e segunda ordem de $\nabla \mathcal{L}(W_t)$, sendo $\hat{m}$ e $\hat{g}$ os valores após a aplicação do fator de decaimento. Os parâmetros são então normalizados e atualizados segundo a seguinte expressão:

$$W_{t+1} = W_t - \frac{\eta \hat{m}_{t+1}}{\sqrt{\hat{g}_{t+1} + \epsilon}}, \quad (2.21)$$

onde ϵ é uma constante para estabilidade numérica.

2.4.5 Normalização de Lote (*Batch Normalization*)

Os dados de entrada da rede comumente são normalizados para evitar que as funções de ativação não-lineares (sigmoide, logística) sejam saturadas precocemente. Esta prática assegura que os valores de entradas estejam na mesma escala. Entretanto, o mesmo não ocorre com as camadas intermediárias, onde a distribuição das ativações muda constantemente durante o treinamento da rede. Este problema é conhecido como mudança de covariável interna (*internal covariate shift*) e desacelera o processo de aprendizado uma vez que as camadas precisam se adaptar a novas distribuições a cada passo de treinamento.

Ioffe e Szegedy (2015) propuseram técnica *Batch Normalization* (BN) como solução para o problema mencionado acima, nela as entradas de cada camada são forçadas a

Algoritmo 1: Normalização em lote, aplicada a um sinal de ativação x em um mini-lote $\mathcal{B}$ de tamanho m .

Entrada: Valores de x de um mini-lote: $\mathcal{B} = \{x_{1...m}\}$;
Parâmetros a serem aprendidos: γ, β

Saída: $y_i = \text{BN}_{\gamma, \beta}(x_i)$

1. Calcule a média e a variância das entradas da camada.

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_{i=1}^m x_i \qquad \sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_{\mathcal{B}})^2$$

2. Normalize as entradas da camada usando as estatísticas de lote calculadas anteriormente.

$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}$$

ϵ : constante para estabilidade numérica

3. Escale e desloque para obter a saída da camada.

$$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma, \beta}(x_i)$$

Fonte: Algoritmo adaptado de Ioffe e Szegedy (2015).

apresentarem a mesma distribuição a cada passo de treinamento. Para isso, a camada BN normaliza cada uma das ativações de entrada de cada canal, subtraindo a média do mini-lote $\mu_{\mathcal{B}}$ e dividindo pelo desvio padrão do mini-lote $\sigma_{\mathcal{B}}^2$. Em seguida, a camada desloca a entrada por um deslocamento β e a redimensiona por um fator de escala γ , os quais são aprendidos durante o treinamento junto com os parâmetros originais da rede, como descrito no algoritmo 1.

Como resultado, a técnica acelera o treinamento da rede convolucional e reduz sua sensibilidade à inicialização dos pesos. Além disso, recomenda-se utilizar as camadas BN entre camadas convolucionais e operações que insiram não-linearidades, como as camadas de ReLU. Nos últimos anos, tornou-se uma importante técnica de regularização, em alguns casos, até mesmo dispensando o uso de *Dropout* (IOFFE; SZEGEDY, 2015).

2.4.6 Técnicas de Regularização

Decorrente do elevado número de parâmetros em uma rede, diversos conjuntos de pesos podem ser possíveis soluções para o problema analisado. Como há múltiplas soluções, o treinamento pode estender-se por várias horas até que o modelo convirja e encontre uma solução global (melhor conjunto de parâmetros). Esta situação permite que ocorra o *overfitting* da rede, prejudicando assim a capacidade de generalização da mesma. Para diminuir o espaço de busca dos parâmetros e evitar situação descrita, uma série de técnicas de regularização podem ser aplicadas.

1. $\mathcal{L}_2$ *Regularization*:

Para desencorajar o surgimento de altos valores de pesos, este tipo de regularização, também denominada como *weight decay*, incorpora um termo de penalização à função de custo original, $L_o(\mathbf{w})$. No caso, minimiza-se a função em questão juntamente com a norma $\mathcal{L}_2$, termo referente à soma dos quadrados dos pesos,

$$L(\mathbf{w}) = L_o(\mathbf{w}) + \frac{1}{2}\lambda \sum_i \mathbf{w}_i^2, \quad (2.22)$$

onde $\mathbf{w}$ é o vetor de pesos (contêm todos os parâmetros livres da rede), λ é a constante que controla a influência do termo de regularização, e $L(\mathbf{w})$ é a função de custo modificada. Deste modo, limita-se o crescimento dos parâmetros, o que reduz a complexidade da rede (KROGH; HERTZ, 1992).

2. Taxa de aprendizado com decaimento (*Learning Rate Decay*):

A taxa de aprendizado influencia diretamente o processo de convergência da rede, uma vez que um valor muito alto realiza modificações mais agressivas dos pesos, o que causa uma convergência mais rápida, porém inadequada, visto que é incapaz de realizar pequenos incrementos de gradiente em direção à melhor solução. O mais indicado seria utilizar uma taxa mais baixa, porém sujeito a um aumento considerável no tempo de treinamento. Esta técnica de regularização explora o melhor das duas situações. A taxa de aprendizado pode decair com base no tempo (*Time-based Decay*), no número de passos (*Step Decay*) ou segundo uma função exponencial (*Exponential Decay*) (BENGIO, 2012; BERGSTRA; BENGIO, 2012).

3. Desligamento (*Dropout*):

Ao contrário da regularização $\mathcal{L}_1$ e $\mathcal{L}_2$ que alteram diretamente a função de custo, o *Dropout* evita o *overfitting* das redes alterando estruturalmente suas arquiteturas. Durante o treinamento algumas de suas unidades são aleatoriamente desligadas, isto é, zera-se o valor de ativação daquela unidade, como ilustrado na Figura 13. Este procedimento, comumente, faz com que a rede generalize melhor, além de torná-la mais robusta, uma vez que os neurônios são forçados a não dependerem de contribuições de neurônios vizinhos, o que força-os a aprenderem características mais robustas (SRIVASTAVA et al., 2014a).

2.4.7 Arquiteturas de Redes Convolucionais de Referência

Os principais fatores que impulsionaram a acurácia de redes neurais profundas foram o aumento do tamanho dos modelos desenvolvidos e do número de exemplos rotulados/anotados disponibilizados para treiná-los. Com relação ao desempenho, os fatores que contribuíram para seu desenvolvimento foram a disponibilidade de CPUs e conexões de rede ainda mais rápidas, mas principalmente pelo advento de unidades de

Figura 13: Modelo de rede neural com *Dropout*. **Esquerda:** Rede neural padrão com duas camadas escondidas. **Direita:** Exemplo de rede produzida pela aplicação de *Dropout* na rede à esquerda. Unidades cruzadas foram desligadas.

Fonte: Imagem adaptada de Srivastava et al. (2014b).

processamento gráfico (GPUs) de propósito geral e *frameworks* de *Deep Learning*, os quais possibilitaram que redes neurais pudessem ser aceleradas por este tipo de *hardware* (RAINA; MADHAVAN; NG, 2009; GOODFELLOW; BENGIO; COURVILLE, 2016).

Dentre as arquiteturas desenvolvidas, algumas delas são reconhecidas por sua capacidade de abstração de conceitos e de extração de características. Dada a dificuldade em desenvolver uma arquitetura do princípio, tornou-se bastante comum utilizá-las como referência, sendo recorrentemente utilizadas em diferentes aplicações, as quais ocasionalmente apresentam pequenas alterações na arquitetura original. Algumas delas estão apresentadas a seguir.

AlexNet. A arquitetura desenvolvida por Krizhevsky, Sutskever e Hinton (2012) (Figura 14) foi responsável por popularizar CNNs para aplicações em visão computacional. *AlexNet* tornou-se conhecida por ser a primeira CNN a vencer o *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC) de 2012, superando significativamente as outras técnicas desenvolvidas (diminuiu a taxa de erro top-5 de 26.1% para 15.3%). Estruturalmente, esta rede é bastante similar à *LeNet*, diferindo pelo fato de ser maior e mais profunda, empilhando, pela primeira vez, várias camadas convolucionais uma sobre a outra. Deste então, esta competição tem sido consistentemente vencida por redes convolucionais (GOODFELLOW; BENGIO; COURVILLE, 2016).

Figura 14: Arquitetura da rede neural convolucional *AlexNet*.

Fonte: Imagem adaptada de Krizhevsky, Sutskever e Hinton (2012).

GoogLeNet/Inception. A arquitetura campeã da ILSVRC 2014 (Figura 18), que também é uma CNN, foi desenvolvida por Szegedy et al. (2015) na Google. A arquitetura introduziu o módulo *Inception* e a camada de *Average Pooling*, os quais reduziram drasticamente o número de parâmetros livres. A Figura 15 apresenta a disposição em que as camadas convolucionais e de *pooling* foram organizadas para a constituição do módulo, este que em sua versão mais simples (*naïve*) não utiliza convoluções 1×1 .

Figura 15: Estrutura interna dos módulos *Inception*

Fonte: Imagem adaptada de Szegedy et al. (2015).

VGGNet. No mesmo ano, a arquitetura desenvolvida por Simonyan e Zisserman (2015) ficou com o segundo lugar. Sua principal contribuição foi demonstrar que modelos mais profundos realmente apresentam um melhor desempenho. Estruturalmente, a arquitetura possui 16 camadas ao total, alternando entre camadas convolucionais, *max pooling* e FCs (Figura 16), o que resulta em aproximadamente 138 milhões de parâmetros treináveis.

Dado seu poder extrativo e a disponibilidade de seus pesos já treinados em outros desafios, a VGG tornou-se a escolha preferida da comunidade como extrator de características para diferentes aplicações, tendo desempenho desta arquitetura como referência (linha de base).

Figura 16: Arquitetura da rede neural convolucional *VGGNet*.

Fonte: Imagem adaptada de Frossard (2016).

ResNet. He et al. (2016) com um novo tipo de rede foram os vencedores da ILSVRC 2015. No caso, introduziu-se os chamados blocos residuais, elementos estruturais com conexões salteadas (*skip connections*) que são utilizados para evitar o problema de desaparecimento de gradiente. Isto é feito reutilizando ativações de camadas anteriores até que camadas adjacentes aprendam seus pesos, como ilustrado na Figura 17. Além disso, a arquitetura também utiliza camadas responsáveis por realizar normalização de lote (*batch normalization*) e dispensa camadas FC no fim da rede (Figura 18).

Figura 17: Conexões internas do Bloco Residual, módulo fundamental para construção de redes residuais.

Fonte: Imagem adaptada de He et al. (2016).

Figura 18: Arquiteturas das redes neurais convolucionais *GoogLeNet* (à esquerda) e *ResNet* (à direita)

Fonte: Szegedy et al. (2015) e He et al. (2016), respectivamente.

2.4.8 *Datasets*

De acordo com Goodfellow, Bengio e Courville (2016), uma informação que se tornou bastante comum a partir de 2016 era que treinar um algoritmo de *Deep Learning* de aprendizado supervisionado com pelo menos 5000 exemplos rotulados (*samples, labels*) por classe geralmente alcançaria desempenho aceitável. Caso o conjunto de dados tenha pelo menos 10 milhões de exemplos, a precisão deste sistema pode vir a exceder o desempenho humano. No caso de aplicações em visão computacional, isto significa que um elevado número de imagens é necessário para treinamento das redes. Por mais audaciosa que a informação seja, uma vez que não especifica a aplicação e o que seria um desempenho razoável para uma rede neural profunda, a quantidade de exemplos anotados necessária é preocupante, visto que nem sempre o processo de rotulação é automatizado.

2.4.8.1 *Dataset Augmentation*

Nem todas aplicações apresentam conjuntos de dados volumosos o suficiente para treinar redes convolucionais. Uma solução encontrada para contornar esta situação é a adição de cópias das amostras de treinamento que sofreram algum tipo de transformação ao conjunto de treinamento. Em problemas de classificação, as transformações podem ser utilizadas desde que não alterem a classe da amostra original (GOODFELLOW et al., 2013). Em problemas de regressão envolvendo imagens, alguns trabalhos realizam diversas transformações randômicas: redimensionam a escala, realizam rotações e translações, alteram as cores (brilho, contraste, saturação), invertem horizontalmente/verticalmente as imagens (*flips*), dentre outras (EIGEN; PUHRSCHE; FERGUS, 2014). Usualmente, a aplicação de *data augmentation* aumenta o poder de generalização dos métodos, uma vez que as redes tornam-se mais robustas a pequenas mudanças geométricas ou na iluminação, estas que poderiam prejudicar seriamente o processo de inferência da rede. Além das transformações mencionadas, pode-se inserir distorções geométricas não-lineares (LECUN et al., 1998), ou então, perturbações randômicas nos canais RGB da imagem (KRIZHEVSKY; SUTSKEVER; HINTON, 2012).

2.4.9 *Frameworks* de desenvolvimento

Esta seção enumera alguns dos *frameworks* disponíveis para implementação computacional de métodos de aprendizagem profunda. Originalmente, algumas das bibliotecas foram desenvolvidas para otimização de métodos numéricos, mas posteriormente foram modificadas de modo que também permitissem a construção e o treinamento de redes neurais profundas. A análise comparativa destes sistemas pode ser realizada contrapondo-os sob critérios como: linguagem de programação utilizada, modo que os dados são estruturados, se há suporte à aceleração em GPU, se há suporte a processamento distribuído, dentre outros.

1. Theano – Python (BERGSTRA et al., 2010)
2. Caffe – C, C++, Python, MATLAB, CLI (JIA et al., 2014)
3. CNTK – Python, C++, C#, CLI (SEIDE; AGARWAL, 2016)
4. **TensorFlow** – C++, Python (ABADI et al., 2016)
5. Torch – C, C++, Lua (COLLOBERT; KAVUKCUOGLU; FARABET, 2011)
6. MXNet – Python, R, C++, Julia (CHEN et al., 2015)
7. Keras – Biblioteca em Python, roda sobre TensorFlow, CNTK ou Theano (CHOLLET et al., 2015)
8. PyLearn2 – Biblioteca em Python, roda sobre Theano (GOODFELLOW et al., 2013)

Figura 19: Principais Deep Learning Frameworks de 2017 e seus mantenedores.

Fonte: Indra den Bakker (2017).

A Figura 19 ilustra o infográfico apresentando os principais *frameworks* utilizados para o desenvolvimento de redes neurais profundas no ano de 2017, além de informar as companhias mais interessadas em manter o desenvolvimento destas ferramentas.

2.5 Considerações Finais

Devido à sua habilidade de extrair de características cada vez mais representativas e a sua capacidade de interpretar conceitos cada vez mais complexos, redes neurais profundas

vêm tornando-se uma das técnicas de aprendizado de máquina mais promissoras. Isto pode ser comprovado observando resultados e avaliações em *benchmarks* obtidos por esta nova abordagem em diferentes áreas de pesquisa e aplicações, curiosamente, vindo a superar até mesmo técnicas mais clássicas no estado da arte.

No contexto de estimação de profundidade, de alguma forma, elas são aptas a extrair diferentes informações visuais, identificando padrões e características capazes de relacionar as imagens de entrada com os mapas de profundidade. Entretanto, ao mesmo tempo que estas redes são adequadas para aplicações nunca antes exploradas, ainda são computacionalmente custosas, requerem uma elevada quantidade de dados para serem treinadas e o ajuste de seus hiperparâmetros é uma tarefa árdua.

3 PERCEPÇÃO VISUAL DE PROFUNDIDADE

Biologicamente, humanos e outros animais aprendem a estimar distâncias ao estabelecer relações entre as imagens projetadas nas retinas, sendo uma destas relações a informação de disparidade. Analogamente, diferentes técnicas de visão computacional também buscam extrair a mesma informação do ambiente utilizando duas ou mais câmeras. Entretanto, a percepção de profundidade não se restringe apenas à visão binocular, visto que também é possível extrair outras pistas visuais acerca da profundidade a partir de apenas uma imagem (*Monocular Cues*), isto é, a partir da imagem capturada de uma única perspectiva (HOWARD; ROGERS, 2012). Esta capacidade pode ser justificada com base na forma que o mundo é estruturado, o qual apresenta repetibilidade nos fenômenos físicos e cuja taxa de mudança dos ambientes geralmente é mais lenta que nossa capacidade de aprendizado. Ela também pode ser procedente dos conhecimentos obtidos previamente por humanos e animais, uma vez que percebendo variações nas texturas dos objetos e no tamanho de objetos conhecidos, ou então, algum tipo de desfocagem ou neblina é possível predizer se um objeto está longe ou perto (LOOMIS, 2001; SAXENA; SCHULTE; NG, 2007).

Até o presente momento, tem-se buscado aprimorar o desempenho dos sistemas computacionais por meio de melhorias na precisão e na velocidade de processamento para algoritmos de correspondência estéreo. O desafio encontra-se em desenvolver técnicas computacionais capazes de explorar não somente as pistas binoculares, mas também as monoculares para extrair a informação de profundidade de cenas dinâmicas. Visto que recuperar esta informação é um problema primário em visão computacional, entendê-lo é crucial para o desenvolvimento de aplicações posteriores em robótica, como em compreensão estrutural de cenas e reconstrução 3D de ambientes (SAXENA; CHUNG; NG, 2006).

3.1 Pistas Visuais de Profundidade

Como já mencionado, a percepção de profundidade em uma imagem pode ser assimilada por diferentes pistas visuais, as quais podem ser classificadas em pistas proprioceptivas, binoculares, monoculares e dinâmicas. Os seguintes tópicos apresentam algumas dessas pistas e descrevem os conceitos envolvidos na extração de profundidade (OKOSHI, 2012; HOWARD; ROGERS, 2012). Recentemente, Jamiy e Marsh (2019) realizaram uma revisão bibliográfica sobre a percepção de profundidade em aplicações de *virtual reality* (VR), *augmented reality* (AR) e *mixed reality* (MR), reunindo informações bem detalhadas de pistas visuais para percepção de profundidade.

3.1.1 Pistas Proprioceptivas

Pistas proprioceptivas atuam ativamente na acomodação do cristalino e da convergência dos olhos, permitindo a percepção de profundidade de objetos a curto alcance (até 2 metros).

Acomodação (*Accommodation*)

A acomodação também é uma pista oculomotor e está timidamente relacionada com percepção de profundidade. No caso, ao invés de ter uma relação com os músculos que movimentam os globos oculares, possuem uma relação com os músculos responsáveis por realizar a acomodação do cristalino.

Convergência

Dada a distância entre os olhos (6,3 cm, em média), os raios de luz que chegam a cada retina apresentam uma pequena diferença em si. Em outras palavras, para uma mesma cena, as retinas percebem imagens ligeiramente diferentes. Esta pista também é referida como disparidade retinal, ou também paralaxe binocular, e a partir desta pequena diferença o cérebro é capaz de extrair informação de profundidade.

3.1.2 Pistas Binoculares de Profundidade

Dentre as pistas binoculares, tem-se:

Estereopsia (*Stereopsis*)

A estereopsia, ou também disparidade retinal ou *binocular parallax*, é o principal mecanismo de percepção de profundidade que humanos e animais possuem, o qual é capaz de extrair informações de uma cena a partir de diferentes perspectivas (campos visuais devem se sobrepor).

Convergência Binocular

A convergência é uma pista binocular relacionada a estimulação e às contrações dos músculos extraoculares, os quais são responsáveis por movimentar os globos oculares, fazendo-os focar (convergir) em objetos observados. Ao fazê-los confluírem a um ponto, os músculos são movimentados e estas sensações cinestésicas (sugestões oculomotoras) também ajudam na percepção de distância (OKOSHI, 2012).

Estereopsia de Sombra (*Shadow Stereopsis*)

Sombras são resultantes da projeção de luz estruturada e interceptação de seus raios por um corpo opaco, porém não correspondem as bordas do objeto iluminado, além de terem suas posições alteradas ao variar-se a posição da fonte de luz. Essa última funciona como um outro ponto de vista, permitindo que as sombras forneçam características confiáveis acerca das superfícies de objetos para percepção de disparidades, sugerindo informações quanto a forma 3D dos objetos (PUERTA, 1989).

3.1.3 Pistas Monoculares de Profundidade

As pistas monoculares são informações que podem ser obtidas a partir de uma única perspectiva, sendo subdivididas em pistas oculomotores, pistas pictóricas (extrai informações a partir de imagens bidimensionais) e pistas baseadas em movimento (GOLDSTEIN, 2009; COON; MITTERER, 2015). Dentre as elas, tem-se:

Tamanho Relativo (*Relative Size*)

Considerando dois objetos de mesmo tamanho, caso sejam dispostos em diferentes distâncias do observador um deles aparentará ser maior que outro. Nesta situação, provavelmente o observador julgará que o maior objeto se encontra mais próximo.

Tamanho Absoluto ou Familiar (*Absolute or Familiar Size*)

A familiaridade com o tamanho absoluto, tamanho real, de objetos conhecidos também é uma das formas de perceber profundidade. Uma situação que exemplifica esta pista visual é: considere dois amigos andando na mesma rua e que estejam prestes a se encontrarem. Supondo que saibam mais ou menos a altura de cada um, eles podem perceber o quão longe estão um do outro apenas comparando o tamanho observado com o tamanho conhecido.

Elevação ou Altura relativa (*Elevation*)

O horizonte também pode ser utilizado como referência para percepção de profundidade. No caso, esta pista monocular baseia-se na altura relativa de um objeto com a linha do horizonte, sendo que objetos mais próximos a ela podem estar mais distantes, enquanto que objetos mais afastados dela, encontram-se mais próximos ao observador.

Gradiente de Texturas (*Texture Gradient*)

O gradiente de textura pode ser percebido devido ao encolhimento do tamanho de elementos uniformes, ou então, do espaçamento entre eles, presentes na cena observada (BAJCSY; LIEBERMAN, 1976). Este gradiente facilita a percepção de variações em outras dicas visuais, como o tamanho relativo entre os objetos (TOZAWA, 2010), por exemplo.

Perspectiva Aérea (*Aerial Perspective*)

Visto que o ar pode apresentar diferentes partículas suspensas (poeira, vapor de água), objetos a distâncias terão mais partículas obscurecendo sua visualização, o que deixa a cena com um aspecto mais borrado, com uma espécie de neblina a seu redor.

Perspectiva Linear (*Linear Perspective*)

Desenhar espaços tridimensionais em pinturas ou em folhas de papel (espaço 2D) requer que sejam representados em perspectiva. Linhas paralelas no ambiente real tocam-se em pontos de fuga na representação, os quais são os responsáveis por inserir a sensação de profundidade espacial em desenhos.

Sobreposição (*Overlap or Interposition*)

Ao observar uma cena onde um objeto parcialmente encobre outro, o observador provavelmente julgará que o objeto obscurecido se encontra atrás do primeiro, logo, estará mais distante. Este tipo de informação permite que observador estime a relação em profundidade relativa existente entre eles.

Sombreamento e Iluminação (*Shading and Lighting*)

O jogo de luz e sombra sobre os objetos observados também dá importantes sugestões quanto à profundidade. Por exemplo, objetos melhores iluminados podem parecer mais próximos que objetos sombreados, os quais podem parecer mais distantes do que realmente estão.

3.1.4 Pistas Dinâmicas

As pistas dinâmicas estão diretamente relacionadas à percepção de movimento pelo sistema visual. Dependendo da forma que o observador varia sua disposição na cena, seja transladando ou rotacionando, a informação de movimento permite observá-la de diferentes perspectivas, o que aumenta a quantidade de informação para ser interpretada (JAMIY; MARSH, 2019).

Paralaxe de Movimento (*Motion Parallax*)

Movimento é um importante fator para a percepção espacial em cenas monoculares. A movimentação tanto do observador quanto de objetos, permite perceber objetos mais próximos se movendo mais rapidamente que objetos localizados no plano de fundo.

3.2 Trabalho Correlatos

Estimação de Profundidade (Depth Estimation) a partir de imagens monoculares é um problema mal-posto, uma vez que a imagem observada pode ser gerada a partir de várias projeções possíveis a partir da cena real (EIGEN; PUHRSCHE; FERGUS, 2014). Além disso, é inerentemente ambíguo, pois os métodos propostos buscam recuperar a informação de profundidade diretamente das intensidades de cor da imagem de entrada (LAINA et al., 2016). A evolução histórica do tema encontra-se descrita nos parágrafos que se sucedem.

De acordo com a literatura, o problema de estimação de profundidade em imagens monoculares foi primeiramente abordado por métodos de alto nível que se baseavam em *Line Drawing*, técnica responsável por extrair representações 2D (planar) dos objetos (ROBERTS, 1963). Uma década mais tarde, os métodos desenvolvidos permitiam interpretar objetos com base em protótipos 3D, gerados a partir de representações bidimensionais, no caso, também sendo capazes de lidar com linhas imperfeitas e objetos oclusos (FALK, 1972; GRAPE, 1973).

No campo de inteligência artificial, em contrapartida, Guzman-Arenas (1968) pesquisou como poderia interpretar objetos a partir de suas linhas, porém definindo suas bordas sem a utilização de protótipos 3D. Para decompor a cena em diferentes objetos, os autores atentaram-se às junções das linhas para realizar a decomposição das regiões. A teoria desta abordagem foi reforçada com o estudo de restrições para construção de poliedros (HUFFMAN, 1971), a utilização de linguagens descritivas para interpretação de padrões (CLOWES, 1971), e o desenvolvimento de dicionários para classificação de combinações das linhas em junções (WALTZ; WINSTON, 1975). Anos depois, a tarefa em questão foi modelada em termos algébricos para recuperar formas tridimensionais de objetos poliédricos (SUGIHARA, 1984), e foi estendida para também lidar com linhas curvas (MALIK, 1987), oclusões de superfícies (*border ownership*) e objetos de linha fina (*thinline objects*) (SAUND, 2006).

Uma descrição mais completa dos métodos mencionados anteriormente foi realizada por Dimiccoli (2009). Além disso, as técnicas desenvolvidas até então baseavam-se em representações de alto nível da imagem, estas que não eram prontamente disponibilizadas e que eram arduamente extraídas dos dados da imagem (DIMICCOLI, 2009).

Nitzberg e Mumford (1990) apresentaram uma abordagem de baixo nível, *2.1-D sketch*, cujo modelo era capaz de extrair representações em camadas diretamente dos dados da imagem. Mais especificamente, a técnica segmentava a imagem em análise de acordo com as bordas dos objetos gerando um conjunto de regiões ordenadas por profundidade.

Hoiem, Efros e Hebert (2005) desenvolveram um método automático com base na extração de características e descritores geométricos para a composição de um modelo 3D da cena analisada. Mais precisamente, a imagem a ser analisada é segmentada em regiões com características similares (superpíxeis), os quais são posteriormente agrupados em constelações para que características mais complexas possam ser calculadas. Com base nas informações extraídas, as regiões segmentadas são classificadas em estruturas geométricas (chão, céu, vertical). Por fim, o modelo 3D é estimado com base nas rotulações realizadas, como pode ser observado na Figura 20.

Figura 20: Algoritmo de estimação de modelo 3D proposto por Hoiem, Efros e Hebert (2005). Cores não apresentam informação significativa, apenas diferenciam as regiões segmentadas.

Fonte: Imagem adaptada de Hoiem, Efros e Hebert (2005).

Michels, Saxena e Ng (2005) identificaram uma circunstância em que explorar pistas monoculares foi incisivo para que os requisitos que seu projeto impunha fossem satisfeitos. No caso, o problema estudado foi o de navegação e desvio de obstáculos em altas velocidades de um carrinho de controle remoto em ambiente externo. Neste contexto, o sistema estéreo não obteve um bom resultado. Entretanto, devido ao tamanho reduzido da linha de base (*baseline*) e às altas vibrações, o alcance detectável ficou limitado e as imagens ficaram borradas, o que prejudicou a qualidade da correspondência estéreo.

Saxena, Chung e Ng (2006) propuseram utilizar campos aleatórios de Markov (MRF do inglês *Markov Random Fields*) para inferir distâncias a partir de imagens monoculares. No caso, a profundidade era estimada a partir de pequenos trechos da imagem (*patches*) e filtros convolucionais para o processamento de texturas e a extração de informações. Na abordagem proposta pelos autores, os campos aleatórios de Markov Gaussiano e Laplaciano definem os modelos probabilísticos das relações entre os imagens e mapas de profundidade analisados de acordo com as distribuições gaussiana e de Laplace, respectivamente (como ilustrado na Figura 21).

Figura 21: Resultados qualitativos obtidos pelas topologias do tipo MRF. Cores mais quentes indicam regiões mais próximas do observador.

Fonte: Imagem adaptada de Saxena, Chung e Ng (2006).

Diferentemente dos humanos que utilizam tanto as pistas monoculares quanto as binoculares para extrair informação de profundidade, os trabalhos científicos da época focavam quase que exclusivamente em visão estéreo. Isto estava relacionado à dificuldade

em realizar estimativas de profundidade, uma vez que os métodos desenvolvidos até então baseavam-se fortemente nas estruturas globais e informações a priori das cenas analisadas.

Saxena, Schulte e Ng (2007) investigaram de que forma podia-se explorar estas pistas monoculares e integrá-las aos sistemas estéreo existentes. No caso, foi demonstrado que a incorporação de ambos tipos de pistas possibilita obter melhores estimativas de profundidade do que as abordagens utilizando pistas monoculares e binoculares unicamente.

Alguns anos mais tarde, outros métodos tentaram solucionar o problema realizando fortes suposições geométricas. Uma das técnicas utilizadas consistia em procurar relações geométricas entre modelos na forma de caixa (*box-shaped models*) e os móveis presentes na cena analisada tais como camas e mesas (HEDAU; HOIEM; FORSYTH, 2010; LEE et al., 2010), como ilustrado na Figura 22. Entretanto, estes métodos focavam na representação paramétrica de objetos 3D, objetivando a incorporação de restrições volumétricas do mundo físico, uma vez que aprimoravam o desempenho das predições (HEDAU; HOIEM; FORSYTH, 2010).

Figura 22: Ilustração exemplificando os procedimentos realizados para identificação de relações geométricas e estimação da estrutura da cena com base nos objetos presentes nela.

Objeto âncora: Sofá

Fonte: Imagem adaptada de Lee et al. (2010).

Mais tarde, alguns pesquisadores passaram a investigar se a integração de outras informações adicionais poderia auxiliar na estimação de distâncias da cena como anotações de usuários (RUSSELL; TORRALBA, 2009), rotulação semântica (*semantic labeling*) (LIU; GOULD; KOLLER, 2010; LADICKÝ; SHI; POLLEFEYS, 2014), presença de estruturas repetidas (WU; FRAHM; POLLEFEYS, 2011), dentre outros. Segmentação

semântica provou ser bastante benéfica, pois píxeis pertencentes a um mesmo objeto (rótulo) apresentam uma maior probabilidade de apresentar valores de profundidade próximos. Tecnicamente, a segmentação pode ser utilizada antes (LIU; GOULD; KOLLER, 2010) ou aprendida em conjunto da estimação de profundidade (LADICKÝ; SHI; POLLEFEYS, 2014; WANG et al., 2015; EIGEN; FERGUS, 2015). Entretanto, visto que a rotulação normalmente é realizada manualmente, o número de exemplos anotados necessários pode impedir a ampla exploração desta prática.

Contudo, nem sempre é possível explorar informações adicionais, visto que a linha de desenvolvimento dos métodos propostos busca torná-los cada vez mais robustos, aplicando a técnica em cenas mais genéticas, evitando o uso excessivo de ambientes tratados. Desta forma, alguns trabalhos passaram a formular o problema de estimação como um problema de inferência de Máxima Probabilidade a Posteriori (MAP) em Campos Aleatórios de Markov (MRFs) (SAXENA; CHUNG; NG, 2008; SAXENA; SUN; NG, 2009) ou Campos Aleatórios Condicionais (CRFs) (LIU; SHEN; LIN, 2015; LIU et al., 2016; XU et al., 2017).

Neste mesmo período, redes convolucionais tornaram-se mais populares dada a repercussão gerada pelo desempenho excepcional da rede *AlexNet* na ILSVRC 2012 (KRIZHEVSKY; SUTSKEVER; HINTON, 2012). Deste então, alguns pesquisadores passaram a resolver o problema de estimação de profundidade utilizando técnicas de aprendizagem profunda, tanto em sistemas estéreo quanto em sistemas monoculares.

No caso de sistemas estéreo, as redes profundas foram empregadas no modelamento da relação existente entre pares de imagens esquerda-direita, $I_L(x)$ e $I_R(x)$, onde x é o indexador do píxel em análise. Diferentemente do problema de estimação monocular, o problema de inferência de profundidade em sistemas estéreo consiste em estimar o deslocamento em posição (disparidade, d) que um píxel em I_L deva ser submetido para que se encontre a coordenada do píxel homólogo em I_R (MEMISEVIC; CONRAD, 2011). Deste modo, propunha-se a utilização de métodos de aprendizagem automática para modelar a função de correspondência (*matching function*) (ŽBONTAR; LECUN, 2015b; LADICKÝ; HÄNE; POLLEFEYS, 2015) ou para identificar as transformações, de modo independente, dos píxeis de $I_L(x)$ e $I_R(x)$ (MAYER et al., 2016).

No caso de sistemas monoculares, as redes profundas foram empregadas tanto para mapear diretamente as imagens RGB a mapas de profundidade, quanto para otimizar os hiperparâmetros de outros métodos de aprendizado. Os trabalhos que exploraram esta técnica podem ser subdivididos em supervisionados e não-supervisionados.

3.2.1 Métodos Supervisionados

Eigen, Puhersch e Fergus (2014) foram os pioneiros a empregarem redes convolucionais para a tarefa SIDE, o que impactou positivamente na qualidade das estimativas, e inspirando o uso de aprendizagem profunda para trabalhos posteriores. Mais especificamente,

foi desenvolvido um sistema de aprendizado supervisionado com topologia *Coarse-to-Fine* capaz de estimar mapas de profundidade diretamente de imagens RGB. O treinamento é realizado de modo ponta-a-ponta (*end-to-end fashion*), onde sua topologia multi-escala encarrega-se de aprender as relações necessárias para relacionar as entradas com a saída, o que dispensa o uso de superpixelização e passos intermediários para extração de características. A topologia *Coarse-to-Fine* (Figura 23) consiste na utilização de duas CNNs: uma é responsável por realizar uma análise a nível global da cena e uma predição mais grosseira do mapa de profundidade; a outra é incumbida de refinar as predições de escala global da rede anterior, com o objetivo de tornar mais evidente detalhes locais como contorno dos objetos (quinas dos móveis, cantos das paredes, etc).

Figura 23: Ilustração da topologia *Coarse-to-Fine* para estimação de profundidade. Tons mais claros indicam maiores valores de profundidades.

Fonte: Imagem adaptada de Eigen, Puhrsch e Fergus (2014).

Dado o potencial das CNNs, Liu et al. (2016) propuseram a arquitetura *Deep Convolutional Neural Field* (DCNF) para a aplicação, a qual utiliza redes convolucionais para modelar os potenciais de um campo aleatório condicional (CNN+CRF) e estimar as profundidades de superpíxeis (Figura 24).

O trabalho anterior não foi o único a utilizar superpixelização. Li et al. (2015) propuseram uma CNN capaz de estimar tanto mapas de profundidade quanto mapas de normais (*surface normal estimation*). Além disso, aplicava-se CRFs à saída do *framework* para o refinamento das predições de profundidade dos superpíxeis. Assim como os trabalhos anteriores, Wang et al. (2015) também modelaram o problema de estimação utilizando CRFs, porém exploraram a relação entre estimação de profundidade e segmentação

Figura 24: Resultado obtido por uma topologia do tipo CNN+CRF com superpixelização. Cores indicam profundidades (tons vermelhos correspondem a regiões mais distantes e tons azuis, a regiões mais próximas).

Fonte: Imagem adaptada de Liu et al. (2016).

semântica, no caso, demonstrando o benefício em integrar estas duas informações. Mais precisamente, eles utilizaram duas camadas hierárquicas (predições a nível de píxel e a nível de regiões) de campos aleatórios condicionais (HCRF do inglês *Hierarchical Conditional Random Field*) para inferir os mapas semânticos e de profundidade, cujos potenciais unários são provenientes de predições das redes convolucionais (CNNs).

Todavia, redes convolucionais enfrentam um problema quanto à resolução final dos mapas de profundidade (*upsampling problem*), uma vez que possuem uma resolução muito inferior à entrada. Por exemplo, a predição da rede proposta por Eigen, Puhersch e Fergus (2014) possui $\frac{1}{4}$ do tamanho da imagem de entrada. Durante a avaliação, normalmente o tamanho da predição é recuperado utilizando interpolação bilinear ou *nearest-neighbor* (GONZALEZ; WOODS, 2002), tornando assim possível compará-la diretamente com o *ground truth*. Consequentemente, uma série de diferentes abordagens para solucionar o problema mencionado foram propostas, algumas, assim como no trabalho anterior, permitem que o *upsampling* seja diretamente aprendido pela rede, utilizando camadas deconvolucionais (LONG; SHELHAMER; DARRELL, 2015), camadas de *superpixel pooling* (LIU et al., 2016), ou então, blocos diretamente desenvolvidos para isto, como os *Up-Projection Blocks* propostos por (LAINA et al., 2016). Entretanto, este redimensionamento precisa ser feito com cautela dado que o aumento exagerado da resolução de saída da topologia afeta diretamente a qualidade final da predição.

Esta não foi a única contribuição desse último trabalho mencionado, ele também propôs a utilização de uma nova arquitetura de rede, redes residuais totalmente convolucionais (FCRN do inglês *Fully Convolutional Residual Networks*), para abordar a tarefa SIDE. A rede proposta foi inspirada na topologia ResNet-50, porém as camadas FC foram substituídas pelos já mencionados *Up-Projection Blocks*, os quais permitem que o aprendizado do *upsampling* dos mapas de características seja mais rápido e eficiente (Mais detalhes na seção 5.4). O trabalho também contribuiu com a *reverse Huber loss* (BerHu), função de custo desenvolvida especialmente para aprimorar a qualidade das predições,

cujo diferencial encontra-se em tratar de maneira distinta erros residuais de diferentes ordens (Equação 5.4).

Diferentemente dos trabalhos anteriores que modelam o problema como um tarefa bem estruturada de regressão, dado que valores de profundidade são intrinsecamente contínuos, Cao, Wu e Shen (2018) propuseram tratar o problema como uma tarefa de classificação a nível de píxel (*pixel-wise classification*). Os autores desenvolveram um sistema FCRN+CRF (Figura 25) que ao invés de prever um valor exato de profundidade, ele classifica o píxel a uma certa classe de intervalos de profundidade. A linha de pensamento seguida partiu da ideia que seres humanos dificilmente conseguem prever valores exatos de distâncias, porém são capazes de estimar mais ou menos a distância de um certo ponto. Mesmo utilizando uma abordagem incomum, o método proposto foi capaz de desbancar os melhores resultados no estado da arte da época (EIGEN; PUHRSCH; FERGUS, 2014; LAINA et al., 2016; GODARD; AODHA; BROSTOW, 2017).

Figura 25: Modelo desenvolvido por Cao, Wu e Shen (2018), no qual utiliza-se classes de intervalos de profundidade e cada píxel é classificado em um destes intervalos (*Pixel-wise Classification*), cuja saída é posteriormente processada por um CRF.

Fonte: Imagem adaptada de Cao, Wu e Shen (2018).

Xu et al. (2017) propuseram duas novas topologias para integração de CNNs e CRFs. Mais especificamente, uma das topologias baseia-se no cascadeamento de múltiplos CRFs e a outra na integração direta com CNNs de múltiplas escalas.

Redes neurais recorrentes também foram empregadas para inferir profundidade, no caso, estas arquiteturas buscam explorar não somente as relações espaciais, mas também as relações temporais existentes entre os quadros (GRIGOREV et al., 2017; KUMAR; BHANDARKAR; PRASAD, 2018).

Com relação às melhores arquiteturas no estado da arte, diferentes particularidades foram exploradas para lhe conferirem um alto desempenho. Kuznietsov, Stückler e Leibe (2017) desenvolveram uma arquitetura semi-supervisionada (supervisionado e não-supervisionado), na qual exploraram as vantagens de ambos tipos de aprendizado. Fu et al. (2018) introduziram uma estratégia de discretização de espaçamento crescente para discretizar a profundidade, além de reformularem o aprendizado da rede como um problema de regressão ordinal. Chen et al. (2018) desenvolveram um *framework* que unifica duas

arquiteturas, U-Net e *Generative Adversarial Network* (GAN), sendo então otimizado por uma função de custo adversária.

3.2.2 Métodos Não-Supervisionados

No contexto de estimação de profundidade, o desenvolvimento de métodos não-supervisionados representa a emancipação do uso de mapas de profundidade como supervisão, uma vez que são capazes de aprender as relações existentes desta informação com base no conteúdo das imagens RGB e nas transformações geométricas entre elas. Mais especificamente, os trabalhos mais relevantes deste tipo de aprendizado concentraram-se em minimizar erros de reconstrução fotométrica entre imagens estéreo (GARG et al., 2016; GODARD; AODHA; BROSTOW, 2017) ou imagens sequenciais (ZHOU et al., 2017; ZHAN et al., 2018).

Assim como Flynn et al. (2016), Xie, Girshick e Farhadi (2016) abordaram o problema de síntese de novas perspectivas. O *framework Deep3D* tinha como objetivo prever a perspectiva direita a partir da imagem esquerda (entrada). No caso, o treinamento consistia em minimizar o erro de reconstrução da imagem estimada a partir de pares estéreo extraídos de filmes 3D, os quais forneciam um número elevado de exemplos.

Bem como o trabalho anterior, Garg et al. (2016) desenvolveram uma arquitetura (Figura 26) onde também se minimiza o erro de reconstrução para ajuste da rede, porém abordando o problema de estimação monocular de profundidade. Entretanto, como apontado por Godard, Aodha e Brostow (2017), o modelo proposto para a inferência das imagens não era totalmente diferenciável, uma vez que os autores empregaram aproximações de Taylor para linearizar a função de custo. Dado que isto dificulta o processo de otimização, o ajuste dos parâmetros da arquitetura é diretamente prejudicado.

Para contornar esta adversidade, Godard, Aodha e Brostow (2017) utilizaram interpolação bilinear, o que torna a função de custo diferenciável em toda sua extensão. O modelo *Monodepth* foi inspirada na rede *DispNet* (MAYER et al., 2016) e seu diferencial foi utilizar as perspectivas reconstruídas para geração mapas de disparidades. Os esforços foram direcionados em desenvolver um modelo (Figura 27) que melhorasse as predições das imagens reconstruídas ao minimizar o erro fotométrico e que ao mesmo tempo aprimorasse as predições de profundidade, respeitando a restrição imposta pela consistência esquerda-direita (*left-right consistency*).

Mais recentemente, alguns trabalhos têm explorado o uso de sequências de vídeo, monoculares (ZHOU et al., 2017) e estéreo (ZHAN et al., 2018), para o treinamento de seus métodos. Entretanto, de modo geral, abordagens não-supervisionadas apresentam desempenho inferior aos métodos diretamente supervisionados por mapas de profundidade.

Figura 26: Modelo desenvolvido por Garg et al. (2016), cuja topologia é um rede do tipo *autoencoder*. (1) *Encoder* da rede, responsável por mapear a imagem de entrada I_1 (imagem esquerda do par estéreo) ao seu mapa de profundidade D . (2) *Decoder* da rede, o qual estima a imagem distorcida I_w , esta que é resultante de distorções impostas pelo mapa de profundidade estimado ao aplicá-lo na imagem direita do par estéreo I_2 . (3) O treinamento do método minimiza o erro de reconstrução entre I_w e I_1 .

Fonte: Imagem adaptada de Garg et al. (2016).

Figura 27: *Monodepth*: Módulo desenvolvido por Godard, Aodha e Brostow (2017) para cálculo das penalizações do método não-supervisionado proposto. A partir da combinação dos três termos da função de custo gera-se os mapas de disparidades d_l e d_r . Legenda: *Convolution* (C), *Up-Convolution* (UC), *Bilinear Sampling* (S), *Up Sampling* (US).

Fonte: Imagem adaptada de Godard, Aodha e Brostow (2017).

3.2.3 Datasets

No contexto de estimação monocular de profundidade, as aplicações podem abordar ambientes tanto internos (*indoor*) quanto externos (*outdoor*), e caso utilizem métodos supervisionados, não somente as imagens RGB como também os mapas de profundidade precisam ser fornecidos. Os *datasets* mais adequados para esta tarefa estão apresentados a seguir.

Make3D foi desenvolvido por Saxena, Sun e Ng (2009) e consiste de 400 imagens de treinamento e 134 imagens para teste. As imagens foram coletadas em ambientes externos na cidade de Palo Alto por um scanner 3D personalizado, obtendo imagens RGB de 1704×2272 de resolução e mapas de referência de apenas 305×55 píxeis.

NYU Depth v2 é um dos maiores *datasets* de imagens RGB-D coletadas para reconstrução de cenas em ambientes internos. As imagens foram coletadas utilizando um Microsoft Kinect em 464 ambientes diferentes, cujas 1449 imagens do *split* oficial estão subdivididas em 795 imagens de treinamento e 654 de teste (SILBERMAN et al., 2012).

KITTI Vision Benchmark Suite é uma coletânea de conjuntos de dados (imagens estereoscópicas, varreduras de LiDAR, fluxo óptico) coletados pelo carro autônomo desenvolvido pelo Instituto tecnológico de Karlsruhe (KIT), Alemanha. O *dataset* busca oferecer informações que auxiliem o desenvolvimento de carros autônomos, além de propor *benchmarks* mais recentes e mais desafiadores, de modo que se aprimore técnicas de visão computacional aplicadas ao mundo real. Nele é possível encontrar duas versões (2012 e 2015) de *benchmarks* para avaliação de correspondência estéreo, odometria, detecção e rastreamento de objetos, dentre outros (GEIGER; LENZ; URTASUN, 2012; MENZE; GEIGER, 2015). Além disso, o *dataset* também disponibiliza os dados não processados (*raw data*), isto é, não sincronizados e não retificados (GEIGER et al., 2013). Por fim, o conjunto de dados também propõe como desafio a detecção de áreas navegáveis (*road benchmark*) (FRITSCH; KUEHNL; GEIGER, 2013).

Dentre suas últimas atualizações, foi anunciado um novo conjunto de dados, *KITTI Depth*, contendo mais de 93 mil exemplos de mapas de profundidade com suas imagens RGB e varreduras de LiDAR correspondentes, os quais estão alinhados com os dados já disponibilizados do *KITTI Raw Data*. Este novo *dataset* permite que modelos complexos de *Deep Learning* possam ser treinados e aplicados principalmente às tarefas de preenchimento de profundidade (*Depth Completion*) e estimação de profundidade a partir de imagens monoculares (*Depth Prediction*) (UHRIG et al., 2017).

CityScapes foi desenvolvido com o objetivo de suprir a lacuna de conjuntos de dados em larga escala para desenvolvimento de algoritmos de aprendizagem profunda, visto que os *datasets* até então não capturavam adequadamente a complexidade inerente das cenas urbanas do mundo real. Para treinar e testar estas abordagens, 5000 imagens com rótulos de alta qualidade (*fine annotations*) e 20000 com a rotulação mais grosseira (*coarse annotations*) foram coletadas por câmeras estereoscópicas em diferentes cidades da Alemanha (CORDTS et al., 2016).

Datasets Sintéticos também foram empregados para recuperar informação de profundidade (MANCINI et al., 2017; MANCINI et al., 2018). Esses conjuntos de dados fornecem mapas de profundidade densos de alta qualidade que são extraídos diretamente de ambientes virtuais. Alguns dos conjuntos de dados disponíveis são: Apolloscape (HUANG et al., 2018), SUNCG (SONG et al., 2017), SYNTHIA (ROS et al., 2016) e Virtual KITTI (GAIDON et al., 2016). No entanto, ainda existem discussões em aberto questionando se os níveis de complexidade e realismo das informações em tais conjuntos de dados são suficientes para treinar os algoritmos para a tarefa proposta, de modo que possam ser implantados com sucesso em situações do mundo real (UHRIG et al., 2017).

3.2.4 Aplicações

Como apresentado na seção 3.2, a maior parte dos trabalhos busca desenvolver métodos que melhorem cada vez mais a qualidade dos mapas de profundidade estimados e, normalmente, isto é feito propondo-se arquiteturas de rede com maior poder preditivo. Entretanto, poucos trabalhos aplicaram suas estimativas a algum tipo de aplicação específica. Alguns deles estão apresentados a seguir.

Tateno et al. (2017) propuseram combinar as predições de uma CNN para a reconstrução de ambientes com as características extraídas em métodos de Localização e Mapeamento Simultâneo (SLAM do inglês) monocular. Esta abordagem é capaz de reconstruir mapas muito mais densos, além de conseguir contornar situações em que outros métodos tendem a falhar como rotações puras e superfícies com baixa textura. Adicionalmente, o sistema desenvolvido também permite a segmentação semântica de estruturas ao mesmo tempo que estima os mapas de profundidade.

Bem como o trabalho anterior, Mukasa, Xu e Bjorn (2017) também desenvolveram um sistema unindo características extraídas por SLAM monocular com as predições de profundidade de uma CNN. Tecnicamente, utilizou-se a técnica baseada em características ORB-SLAM (MUR-ARTAL; MONTIEL; TARDOS, 2015) em um dispositivo móvel para a extração de pontos 3D esparsos do ambiente. Os autores utilizaram representação em malhas 3D (*3D Scene Mesh*) e o sistema foi remodelado de modo que os módulos de predição de profundidade (CNN) e rastreamento de características (SLAM) ficassem

separados, sendo que apenas o SLAM é executado localmente no dispositivo.

Além da estimação monocular de profundidade, Zhou et al. (2017) desenvolveram o sistema *SfM-Learner* de modo que também fosse capaz de estimar transformações $(\mathbf{R}, \mathbf{t})$ entre quadros (*Pose estimation*). Mais precisamente, utilizaram-se duas redes convolucionais profundas, treinadas de maneira totalmente não-supervisionada a partir de sequências de vídeo monoculares, onde empregou-se síntese de novas perspectivas como sinal de supervisão.

Mahmood e Durr (2017) desenvolveram um *framework* para estimação de topografia de mucosas (*Imaging tissue topography*) usando uma rede convolucional profunda em conjunto com um campo aleatório condicional (CNN-CRF). O sistema auxilia na identificação e remoção de pólipos (crescimento de tecido anormal em uma membrana mucosa), mitigando assim o risco de câncer colorretal.

3.3 Considerações Finais

As primeiras abordagens para inferir profundidade costumavam ter fortes suposições geométricas e baseavam-se em características artesanais, que eram manualmente selecionadas por possuírem propriedades adequadas à aplicação. Mais recentemente, a tarefa de estimação monocular de profundidade tem sido modelada majoritariamente como um problema de regressão, exceto por Cao, Wu e Shen (2018) e Fu et al. (2018) que a modelaram como sendo um problema de classificação e de regressão ordinal, respectivamente. Além disso, modelos probabilísticos gráficos e redes convolucionais profundas foram amplamente explorados para recuperar informações contextuais e para automaticamente extrair características hierárquicas de múltiplas escalas presentes na cena, características estas capazes de relacionar as intensidades dos píxeis de entrada às estimativas de profundidade. Entretanto, o problema não se restringe às técnicas apresentadas, como é o caso de Roy e Todorovic (2016) que demonstraram com sucesso que também é possível abordar a tarefa SIDE utilizando *Neural Regression Forest* (NRF).

Com relação ao treinamento, pode-se classificar os métodos apresentados quanto aos conjuntos de dados utilizados, ao tipo de supervisão que foi empregada para otimização das funções de custo e ao tipo de aprendizado do método. Os trabalhos presentes na literatura demonstram que o aprendizado dos modelos pode ocorrer tanto de maneira supervisionada, não-supervisionada, e até mesmo semi-supervisionada, como demonstrado por Kuznetsov, Stückler e Leibe (2017). Com relação ao tipo de supervisão, os métodos podem ser classificados por mapas de profundidade, estéreo e monocular.

1. **Mapas de Profundidade:** A primeira classe refere-se aos métodos que utilizam diretamente os mapas de profundidade para ajuste de seus pesos e que, em sua maioria, compreende os métodos supervisionados.

2. **Pares Estéreo:** A classe estéreo agrupa métodos que utilizaram pares imagens estéreo calibrados para o treinamento da rede.
3. **Monocular:** Já a classe monocular, agrupa métodos que utilizam sequências de quadros e as relações de transformação entre eles para minimização de suas funções de custo.

As duas últimas classes referem-se ao tipo de supervisão empregada em métodos que dispensam o uso de mapas de profundidade, sendo estes em sua maioria de aprendizado não-supervisionado. Eles explicitamente não apresentam supervisão direta com algum tipo de rótulo, porém internamente exploram outros recursos para realizar o ajuste de seus parâmetros. A Tabela 2 apresenta as informações de cada trabalho no estado da arte quanto à supervisão e ao tipo de aprendizado.

Tabela 2: Visão comparativa sobre o treinamento dos trabalhos estado da arte. Legenda: (M) Make3D, (N) NYU Depth V2, (K) KITTI, (CS) CityScapes.

Método	Dataset	Supervisão			Aprendizado	
		Profundidade	Estéreo	Mono	Supervisionado	Não-Supervisionado
Saxena, Sun e Ng (2009)	M	✓			✓	
Liu, Salzmann e He (2014)	M+N	✓			✓	
Ladický, Shi e Pollefeys (2014)	N+K	✓			✓	
Eigen, Puhrsch e Fergus (2014)	N+K	✓			✓	
Eigen e Fergus (2015)	N	✓			✓	
Liu, Shen e Lin (2015)	M+N	✓			✓	
Wang et al. (2015)	N	✓			✓	
Roy e Todorovic (2016)	M+N	✓				✓
Laina et al. (2016)	M+N	✓			✓	
Garg et al. (2016)	K		✓			✓
Godard, Aodha e Brostow (2017)	K+CS		✓			✓
Kuznetsov, Stückler e Leibe (2017)	K	✓	✓		✓	✓
Xu et al. (2017)	M+N	✓			✓	
Zhou et al. (2017)	M+K+CS			✓		✓
Zhan et al. (2018)	K		✓			✓
Fu et al. (2018)	M+N+K	✓			✓	
Chen et al. (2018)	M+N+K	✓			✓	

Fonte: Elaborada pelo autor.

Normalmente, os pesquisadores procuram analisar seus métodos segundo uma série de métricas padronizadas (*Abs rel*, *Sqr Rel*, *RMSE*, dentre outras. Mais informações na seção 6.4), estes indicadores avaliam a exatidão de seus métodos de acordo com as imagens de teste dos *splits*¹ dos *datasets* (NYU Depth v2, Make3D, KITTI), o que permite que os resultados obtidos possam ser diretamente comparados com outros trabalhos. A Tabela 3 apresenta os resultados no estado da arte para o *dataset* KITTI quando avaliados segundo as imagens de teste do *Eigen Split*² (EIGEN; PUHRSCHE; FERGUS, 2014).

¹ A divisão (*split*) que define o subconjunto de avaliação que permite a comparação com outros resultados no estado da arte pode não ser a mesma divisão oficial do dataset analisado.

² Este trabalho concentrou-se em cenas externas para aplicação em carros autônomos, por este motivo omitiu-se as tabelas referentes aos *datasets* NYU Depth v2 e Make3D.

Tabela 3: Resultados no estado da arte para o *dataset* KITTI. Desempenho avaliado nos dados brutos do LiDAR segundo o *Eigen Split* (697 imagens de teste) (EIGEN; PUHRSCHE; FERGUS, 2014). Os resultados relatados são aqueles apresentados pelos próprios autores em seus respectivos artigos originais.

Método	Alcance	Abs Rel	Sqr Rel	RMSE	RMSE (log)	$\delta < 1, 25$	$\delta < 1, 25^2$	$\delta < 1, 25^3$
			menor é melhor				maior é melhor	
Mancini et al. (2016)	0 – 40 m	-	-	7,508	0,524	0,318	0,617	0,813
Garg et al. (2016), L12 Aug 8x	1 – 50 m	0,169	1,080	5,104	0,273	0,740	0,904	0,962
Zhou et al. (2017) (sem explicabilidade)	0 – 50 m	0,208	1,551	5,452	0,273	0,695	0,900	0,964
Zhou et al. (2017) (CS+K)	0 – 50 m	0,190	1,436	4,975	0,258	0,735	0,915	0,968
Godard, Aodha e Brostow (2017) (CS+K)	0 – 50 m	0,118	0,932	4,941	0,215	0,858	0,947	0,974
Kuznetsov, Stückler e Leibe (2017)	1 – 50 m	0,108	0,595	3,518	0,179	0,875	0,964	0,988
Fu et al. (2018), DORN (ResNet)	0 – 50 m	0,071	0,268	2,271	0,116	0,936	0,985	0,995
Saxena, Sun e Ng (2009), Make 3D	0 – 80 m	0,280	3,012	8,734	0,361	0,601	0,820	0,926
Liu, Shen e Lin (2015), DCNF-FCSP FT	0 – 80 m	0,217	1,841	6,986	0,289	0,647	0,882	0,961
Eigen, Puhrsch e Fergus (2014), <i>coarse</i> 28×144	0 – 80 m	0,194	1,531	7,216	0,273	0,679	0,897	0,967
Eigen, Puhrsch e Fergus (2014), <i>fine</i> 27×142	0 – 80 m	0,190	1,515	7,156	0,270	0,692	0,899	0,967
Godard, Aodha e Brostow (2017) (CS+K)	0 – 80 m	0,136	1,512	5,763	0,236	0,836	0,935	0,968
Kuznetsov, Stückler e Leibe (2017)	0 – 80 m	0,113	0,741	4,621	0,189	0,862	0,960	0,986
Fu et al. (2018), DORN (ResNet)	0 – 80 m	0,072	0,307	2,727	0,120	0,932	0,984	0,994
Chen et al. (2018), U-Net Adv.	0 – 80 m	0,061	0,282	2,349	0,106	0,943	0,988	0,996
Ma e Karaman (2018), apenas RGB	0 – 100 m	0,208	-	6,266	-	0,591	0,900	0,962
Ma e Karaman (2018), RGBd 500 Amostras	0 – 100 m	0,073	-	3,378	-	0,935	0,976	0,989

Fonte: Elaborada pelo autor.

Observando a Tabela 2 é possível perceber uma tendência de trabalhos que estão investigando abordagens não-supervisionados. Entretanto, como pode-se verificar na Tabela 3, métodos supervisionados ainda apresentam os melhores desempenhos nos *benchmarks*.

A tarefa SIDE ainda enfrenta uma questão quanto à recuperação do fator de escala das predições. Segundo Tateno et al. (2017), esta também é uma das principais limitações de SLAM monocular e, caso os parâmetros intrínsecos das imagens utilizadas para o treinamento da CNN sejam diferentes dos parâmetros do sensor utilizado no SLAM, a escala absoluta da reconstrução 3D não será correta. No caso, os autores recuperaram o fator de escala encontrando a relação existente entre o comprimento focal da câmera atual (f_{cur}) e o comprimento focal da câmera utilizada para coletar as imagens do treinamento (f_{tr}), como descrito pela equação 3.1:

$$D_{k_i}(\mathbf{u}) = \frac{f_{cur}}{f_{tr}} \tilde{D}_{k_i}(\mathbf{u}), \quad \mathbf{u} = (x, y) \quad (3.1)$$

onde:

- f_{cur} = Comprimento focal da câmera atual,
- f_{tr} = Comprimento focal da câmera utilizada durante o treinamento,
- $\mathbf{u}$ = (x, y) -ésimo elemento do mapa de profundidade, $\mathbf{u} \in \mathbb{R}^2$,
- $\tilde{D}_{k_i}(\mathbf{u})$ = Mapa de profundidade estimado pela CNN a um fator de escala,
- $D_{k_i}(\mathbf{u})$ = Mapa de profundidade estimado pela CNN.

Zhou et al. (2017) propuseram recuperar o fator de escala de uma forma diferente, multiplicando os mapas estimados pela constante $\hat{s}$ que faz a mediana dos valores dos

mapas de profundidade estimados (D_{pred}) coincidir com a mediana dos mapas de referência (D_{gt}), como descrito pela equação 3.2:

$$\hat{s} = \frac{\text{mediana}(D_{gt})}{\text{mediana}(D_{pred})}. \quad (3.2)$$

No mais, sintetizando as informações dos resultados experimentais das arquiteturas apresentadas, pode-se afirmar que a estimação de profundidade é beneficiada pelo aumento do número de camadas das redes neurais profundas e pela utilização de topologias do tipo *Coarse-to-Fine* (CAO; WU; SHEN, 2018). Este tipo de topologia permite que se extraia aspectos globais da cena e que posteriormente refine-se as previsões grosseiras, realçando assim detalhes locais.

O alto poder computacional exigido pelos métodos apresentados ainda é um aspecto preocupante. Por este motivo, Peluso et al. (2019) preocuparam-se com a eficiência energética de sua arquitetura. No caso, apresentaram a *PyD-Net* para a tarefa SIDE, modelo simples e leve o suficiente para ser processada por sistemas embarcados, facilitando assim a integração com sistemas de baixa potência.

De modo geral, os métodos supervisionados por mapas de profundidade utilizam mapas prontamente disponibilizados, ou que então precisam ser gerados a partir das nuvens de pontos fornecidas pelos *datasets*, o que normalmente resulta em um conjunto de mapas esparsos. Notadamente, os *benchmarks* criam uma atmosfera competitiva, cenário este que demanda um fluxo constante de contribuições. Diante ao panorama apresentado, pode-se constatar que estas contribuições apresentadas são quase que exclusivamente referentes a melhorias nas topologias de rede e técnicas utilizadas. Este trabalho difere-se de outros trabalhos presentes na literatura por sugerir que a melhoria em desempenho dos métodos seja consequência da diminuição do nível de esparsidade presente nos mapas de profundidade usados para treinamento.

4 DENSIFICAÇÃO DE MAPAS DE PROFUNDIDADE

A densificação de mapas (*Depth Completion task*) tem sido amplamente estudada em visão computacional e processamento de imagens, e tem como objetivo diminuir o nível de esparsidade presente nos mapas de profundidade. A estimação monocular de profundidade difere desta tarefa, pois procura aproximar diretamente as imagens RGB aos mapas de profundidade. Resumidamente, os métodos existentes para preenchimento buscam gerar mapas mais densos, predizendo distâncias para os píxeis que o sensor não foi capaz de extrair informação de profundidade. Atualmente, este método pode ser classificado em duas abordagens: *Non-Guided Depth UpSampling* ou *Image Guided Depth Completion*.

O primeiro, *non-Guided Depth Upsampling*, tem como objetivo gerar mapas mais densos usando apenas mapas esparsos obtidos diretamente dos dados 3D ou das características geradas pelo SLAM, como ilustrado na Figura 28. Esses métodos se assemelham aos algoritmos propostos na tarefa de super-resolução de profundidade (UHRIG et al., 2017), onde o objetivo é recuperar mapas de profundidade mais precisos e de maior resolução. Mais recentemente, redes convolucionais também têm sido empregadas em aplicações de super-resolução tanto para imagens (YANG et al., 2010; DONG et al., 2014) quanto para profundidade (RIEGLER; RÜTHER; BISCHOF, 2016; SONG; DAI; QIN, 2016). Outros trabalhos concentram-se em preencher (*in-painting*) a informação de profundidade que falta, p. ex., Uhrig et al. (2017) empregou camadas convolucionais esparsas para processar os dados 3D de laser que são irregularmente distribuídos. Como apontado por Zhang e Funkhouser (2018), métodos que prevêem profundidade quando treinados somente em informações brutas geralmente não funcionam muito bem.

A segunda abordagem, *Image Guided Depth Completion*, sugere a incorporação de algum tipo de orientação para alcançar um desempenho superior, p. ex. usar mapas esparsos e imagens RGB da cena (dados RGB-D) como entradas. Além de amostras esparsas de baixa resolução obtidas a partir de LiDARs de baixo custo ou características geradas pelo SLAM (MA; KARAMAN, 2018; WEERASEKERA et al., 2018), outras informações auxiliares também podem ser empregadas (Figura 29), tais como: rótulos semânticos (SCHNEIDER et al., 2016), pontos 2D de laser (LIAO et al., 2017), mapas de superfície normais e de oclusão de fronteiras (ZHANG; FUNKHOUSER, 2018).

4.1 Mapas de Ocupação

Uma maneira comum de armazenar dados de sensores de distância é através do uso de nuvens de pontos (*pointclouds*), que podem ser projetadas de volta em um plano 2D para produzir imagens de profundidade, contendo estimativas de distância para todos os píxeis que possuem um ponto correspondente no mundo. O ponto tridimensional $\mathbf{p}$ pode

Figura 28: Topologia ilustrativa da abordagem *non-Guided Depth Upsampling*.

Fonte: Elaborada pelo autor.

Figura 29: Topologia ilustrativa da abordagem *Image Guided Depth Completion*, na qual utilize-se diferentes tipos de informações auxiliares para aprimorar a qualidade das predições de profundidade da CNN.

Fonte: Elaborada pelo autor.

ser projetado no píxel $\mathbf{u}$, tal como:

$$\mathbf{u} = \mathbf{P}_{rect} \mathbf{R}_{rect} \mathbf{T}_{range}^{cam} \mathbf{p}, \quad (4.1)$$

onde $\mathbf{P}_{rect} \in \mathbb{R}^{3 \times 4}$ é a matriz de projeção de uma câmera retificada, $\mathbf{R}_{rect} \in \mathbb{R}^{3 \times 3}$ é a matriz de rotação retificadora e $\mathbf{T}_{sensor}^{cam} \in \mathbb{R}^{4 \times 4}$ é a matriz de transformação de corpo rígido da câmera ao sensor de distância.

Um exemplo dessa projeção pode ser visto na Figura 30c, onde é possível observar a esparsidade gerada pela projeção direta das informações da nuvem de pontos, mais notavelmente em regiões mais distantes do sensor. A modelagem de dependência espacial é um aspecto crucial em visão computacional e a introdução de tais lacunas irregulares pode afetar seriamente o desempenho das redes convolucionais. Por isso, propôs-se não

Figura 30: Exemplo de projeção da nuvem de pontos reconstruída usando o *framework* de mapas de Hilbert.

Fonte: Imagem adaptada de Guizilini e Ramos (2018a).

reprojetar a própria nuvem de pontos, mas sim seu modelo de ocupação (Figura 30e), gerado pelo *framework Hilbert Maps* (HM) (RAMOS; OTT, 2016).

A utilização desta técnica é justificada primeiramente pela extensão da técnica a outro contexto que não o de reconstrução 3D de mapas, mas sim o de estimação de profundidade a partir de imagens monoculares. Além disso, o *framework* utilizado encontra-se publicamente disponibilizado, o que, no aspecto prático, desperta interesse pela utilização da técnica.

Recentemente, *Hilbert Maps* foram aplicados com sucesso à modelagem de ambientes 3D de larga escala (GUIZILINI; RAMOS, 2016), produzindo uma função de ocupação contínua que pode ser consultada em resoluções arbitrárias. A técnica consiste no cálculo de características a partir de uma função *kernel* $\Phi(x)$ responsável por projetar os dados dos sensores no espaço de *Hilbert*, sendo este um espaço de características de maior dimensão. Nele treina-se um modelo que seja capaz de classificar quanto a ocupação de um determinado ponto, no caso, utiliza-se um classificador de regressão logística. Este método foi desenvolvido para aprimorar as técnicas existentes de mapeamento em grades de ocupação, ou também mapas de ocupação, propondo assim alterar a forma de representação do ambiente mapeado.

Nesta técnica um conjunto de dados $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^N$ é coletado por uma plataforma robótica, onde $\mathbf{x}_i \in \mathbb{R}^3$ representa um ponto no espaço 2D ou 3D. Estes dados são submetidos a uma certa função *kernel*, no caso, utilizou-se uma função exponencial obtida

a partir de estimativas da média e da variância ao clusterizar os pontos da nuvem. O classificador prediz a saída $y_i \in \{-1, +1\}$, esta que representa o estado de ocupação correspondente a i -ésima entrada ($\mathbf{x}_i$). A partir dos dados disponíveis, deseja-se treinar os parâmetros do vetor $\mathbf{w}$ de um modelo discriminativo do tipo $p(y|\mathbf{x}, \mathbf{w})$ que é capaz de prever o nível de ocupação de novos pontos $\mathbf{x}_*$ (RAMOS; OTT, 2016). Um dos modelos sugeridos é um simples classificador de regressão logística, onde a probabilidade de uma entrada $\mathbf{x}_*$ não estar ocupado é definido por:

$$p(y_* = -1|\Phi(\mathbf{x}_*), \mathbf{w}) = \frac{1}{1 + \exp(\mathbf{w}^T \Phi(\mathbf{x}_*))}, \quad (4.2)$$

e a probabilidade de estar ocupado é o evento complementar:

$$p(y_* = +1|\Phi(\mathbf{x}_*), \mathbf{w}) = 1 - p(y_* = -1|\Phi(\mathbf{x}_*), \mathbf{w}). \quad (4.3)$$

onde:

- $\mathbf{x}_*$ = Ponto a ser consultado, $\mathbf{x}_* \in \mathbb{R}^3$,
- $\Phi(\mathbf{x}_*)$ = Vetor de características,
- $\mathbf{w}$ = Vetor de pesos, parâmetros do modelo discriminativo $p(y|\mathbf{x}, \mathbf{w})$,
- y_* = Variável de classificação que indica o status de ocupação de $\mathbf{x}_*$.

Mais especificamente, empregou-se o mesmo vetor de características $\Phi(\mathbf{x}, \mathcal{M})$ de (GUIZILINI; RAMOS, 2016), que é definido por uma série de estimativas quadráticas de um *kernel* exponencial $k(\mathbf{x}, \mathcal{M}_i)$ contra um conjunto de pontos de indução $\mathcal{M} = \{\mathcal{M}_i\}_{i=1}^M = \{\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\}_{i=1}^M$, e é obtido a partir do agrupamento dos pontos da nuvem e do cálculo das estimativas da média $\boldsymbol{\mu}$ e da variância $\boldsymbol{\Sigma}$ para cada subconjunto de pontos:

$$\Phi(\mathbf{x}, \mathcal{M}) = \left[k(\mathbf{x}, \mathcal{M}_1), \dots, k(\mathbf{x}, \mathcal{M}_M) \right] \quad (4.4)$$

$$k(\mathbf{x}, \mathcal{M}_i) = \exp \left(-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}_i^{-1} (\mathbf{x} - \boldsymbol{\mu}_i) \right). \quad (4.5)$$

A clusterização foi realizada usando o algoritmo *Quick-Means* proposto por Guizilini e Ramos (2017), devido à sua eficiência computacional e capacidade de produzir agrupamentos consistentes. Visto que a aplicação não requer necessariamente que as localizações dos *clusters* sejam precisas, eles são posicionados regularmente de acordo com uma métrica de distância $d(\cdot, \cdot)$, de modo que fossem alocados dentro da delimitação imposta pelo raio interno r_i . O algoritmo também permite a sobreposição de *clusters*, isto é feito utilizando um limiar de raio externo $r_o > r_i$, este recurso permite que se aumente a complexidade das características e que seja possível modelar estruturas a partir de diferentes perspectivas.

Resumidamente, este algoritmo tem como entradas: a nuvem de pontos a ser clusterizada $\mathcal{X} = \{x_i\}_{i=1}^N$; os parâmetros r_i e r_o , onde r_i e r_o são os raios interno e externo usados para definir o tamanho do *cluster*; k , número mínimo de pontos por *cluster*; e $d(\cdot, \cdot)$, função de distância entre quaisquer dois pontos do espaço de entrada. O algoritmo de clusterização é inicializado com um conjunto de *clusters* vazio $\mathcal{C}$ e itera sobre todos os pontos da nuvem. Uma amostra aleatória x_* é escolhida a partir da lista de pontos disponíveis, caso seus vizinhos respeitem a delimitação imposta por r_o , um novo *cluster* $\mathcal{M}$ é produzido. Caso o cluster $\mathcal{M}$ criado tenha pelo menos k pontos, ele é incorporado à lista de *clusters* $\mathcal{C}$. Em seguida, as distâncias entre cada um dos pontos presentes em $\mathcal{M}$ e o ponto consultado x_* são calculadas, se estes valores forem menores que r_i , este ponto em questão não pode ser mais considerado como uma semente e nem pertencente a um novo *cluster* (GUIZILINI; RAMOS, 2018c). Este processo é repetido até que todos os pontos tenham sido processados, preenchendo assim o conjunto de *clusters* $\mathcal{C}$.

No entanto, esse algoritmo foi modificado pelos autores para considerar diferentes densidades para os *clusters*, cujos valores variam de acordo com uma função de densidade f tendo o valor de profundidade, variável de distância d a partir da origem, como entrada $f(d)$. Isto pode ser alcançado ajustando $r_i = r_o = \tau \cdot f(d)$, onde τ é uma constante de escala. Em outras palavras, quanto mais perto da origem, mais denso os *clusters*, enquanto que as áreas mais distantes do centro terão menos pontos e, portanto, *clusters* de maiores dimensões são necessários para interpolar adequadamente sobre essas estruturas esparsas. O compromisso para este aumento de poder interpolativo é a perda nos detalhes da estrutura, uma vez que um volume maior será modelado pelo mesmo *cluster*. Os valores ideais dos $\mathbf{w}$ são calculados minimizando a seguinte função de custo de verossimilhança negativa:

$$\mathcal{L}(\mathbf{w}) = \sum_{i=1}^N \log \left(1 + \exp \left(-y_i \mathbf{w}^T \Phi(\mathbf{x}_i) \right) \right) + R(\mathbf{w}). \quad (4.6)$$

onde $R(\mathbf{w})$ é uma função de regularização, empregada principalmente para prevenir que ocorra *overfitting*, além de promover uma certa esparsidade em $\mathbf{w}$, agindo similarmente à uma rede elástica (ZOU; HASTIE, 2005). O regularizador bem como o modelo também possui parâmetros a ajustar, no caso, o *framework* permite que esses sejam otimizados juntamente com pesos $\mathbf{w}$ durante o processo de treinamento (GUIZILINI; RAMOS, 2018b).

Uma vez que o modelo de ocupação tenha sido treinado, ele pode ser usado para produzir uma reconstrução do ambiente, e cada píxel é então avaliado quanto à colisão no espaço 3D, produzindo estimativas de profundidade. Caso a avaliação dos pontos seja realizada por uma função linear contínua, as coordenadas espaciais são mapeadas em um vetor de características de maior dimensão pela função *kernel* $\Phi(x)$ para o *Reproducing Kernel Hilbert Space* (RKHS). RKHS é um HS (*Hilbert Space*) com

propriedades particulares, por exemplo, como quanto à continuidade das funções (se $\|f - g\|$, é pequena para as funções f e g , então $|f(x) - g(x)|$ também é pequena para todo x).

Entretanto, assim como em *Support Vector Machines* (SVMs), a dificuldade deste tipo de problema encontra-se em propor funções *kernel* capazes de organizar os *features* para que sejam linearmente separáveis. Para isso, outras técnicas de aprendizado também podem ser utilizadas, por exemplo, Guizilini e Ramos (2016) propuseram a utilização de *autoencoders* convolucionais para a seleção automática de características, as quais posteriormente seriam linearmente separadas, e então, classificadas quanto a ocupação. Por outro lado, a técnica escolhida precisa permitir que a reconstrução do ambiente seja rápida o suficiente para que permita sua utilização em tempo real.

4.2 Considerações Finais

Neste trabalho, propôs-se utilizar mapas de ocupação utilizando a teoria de Hilbert como técnica de densificação. Originalmente, esta técnica é utilizada para modelagem 3D de ambientes de larga escala, e tem como principal objetivo treinar uma função contínua que capaz de prever o nível de ocupação de novos pontos (x_*).

Conjunto de Dados:

$$\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^N$$

Saída:

$$y_i \in \{-1, +1\}, \quad -1 : \text{Não Ocupado}, \quad 1 : \text{Ocupado}$$

Mais especificamente, uma função *kernel* $\Phi(x)$, projeta os dados dos sensores no espaço de Hilbert, sendo este um espaço de características de maior dimensão. Nele treina-se um modelo que seja capaz de classificar quanto a ocupação de um determinado ponto.

Função *kernel*, $\Phi(x)$:

$$\text{Clusterização dos Pontos} \rightarrow \{\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\}_{i=1}^M \rightarrow \text{Função exponencial}$$

Modelo:

$$\text{Classificador de Regressão Logística}$$

Particularmente neste trabalho, a função *kernel* utilizada foi uma função exponencial obtida a partir de estimativas da média e da variância ao clusterizar os pontos da nuvem. Como modelo, utilizou-se um classificador de regressão logística, que uma vez treinado, é então utilizado para produzir uma reconstrução do ambiente.

5 MATERIAIS E MÉTODOS

Este capítulo explicita as atividades que foram realizadas para que as redes convolucionais pudessem ser treinadas em mapas contínuos, e posteriormente avaliadas. Mais especificamente, apresenta-se as principais modificações e informações acerca dos *datasets*, da arquitetura e das funções de custo utilizadas. Por fim, o procedimento para obtenção da nuvem de pontos a partir das imagens estimadas é demonstrado.

5.1 Estimação de Profundidade – Pré-processamento

Em uma comparação singela entre os diferentes métodos existentes na literatura (Topologia clássica), os principais elementos que mudam de uma abordagem para outra estão diretamente relacionados à arquitetura da técnica que relaciona as imagens RGB e os mapas de profundidade, porém pouco se altera quanto a qualidade dos dados usados para treinamento e teste. No caso, questionou-se o que poderia ser aprimorado com relação a estas etapas. Em ambos estágios, identificou-se um elevado grau de esparsidade presente nos mapas utilizados. No treinamento, isto impacta diretamente no poder preditivo dos métodos, e no teste, diretamente no processo de avaliação.

A principal contribuição deste trabalho é propor que seja realizada uma etapa de pré-processamento para densificação dos mapas de profundidade que são utilizados no treinamento supervisionado das redes convolucionais (Topologia Proposta). Deste modo, o sistema de predição passa a preocupar-se não somente com a definição da arquitetura do método de *Depth Estimation*, mas também a escolha do método de densificação (*Depth Completion*). Neste trabalho, utilizou-se mapas de Hilbert como sendo a técnica de densificação, dado à disponibilidade do *framework* e ao interesse em testar a técnica em um novo contexto, Estimação de profundidade, diferindo de seu contexto original, onde eram utilizados para representação 3D de ambientes.

A Figura 31 apresenta uma comparação ilustrativa das duas topologias mencionadas.

5.2 Imagens Contínuas de Profundidade

Nem sempre os conjuntos de dados disponibilizam diretamente os mapas de profundidade que podem ser usados como referência (*ground truth*), porém, ainda assim, é possível obtê-los se as varreduras 3D do LiDAR e os parâmetros extrínsecos/intrínsecos das câmeras RGB forem fornecidos. Neste caso, uma imagem esparsa de profundidade pode ser gerada projetando diretamente a nuvem de pontos da cena no plano de imagem do sensor visual (EIGEN; PUHRSCHE; FERGUS, 2014; GODARD; AODHA; BROSTOW, 2017).

Figura 31: Ilustração comparativa entre topologias clássicas e a topologia proposta para a tarefa de *Estimação de Profundidade*.

Fonte: Elaborada pelo autor.

Imagens contínuas de profundidade, por sua vez, podem ser obtidas interpolando os pontos medidos em superfícies contínuas geradas anteriormente à projeção. Neste trabalho, utilizou-se o framework para mapas de Hilbert sobre os dados do LiDAR para gerar estas superfícies. Após restringir o mapa contínuo a apenas a região sob o campo de visão da câmera, projetam-se os pontos remanescentes no plano de imagem. Um exemplo de imagem de profundidade reconstruída é representado na Figura 30f, onde pode-se observar que praticamente todas as áreas anteriormente vazias foram preenchidas pelo modelo de ocupação, mantendo as dependências espaciais intactas (obedecendo os limites das capacidades reconstitutivas do *framework*).

5.3 Datasets

Três conjuntos de dados diferentes são considerados neste trabalho: *KITTI Discrete* (esparso), *KITTI Depth* (semi-denso) e *KITTI Continuous* (denso), incluindo quadros das sequências “city”, “residential”, “road”, “campus” e “person”. Os *datasets* esparso e denso foram gerados a partir do *KITTI Raw Data*. Normalmente, a resolução das imagens RGB e de profundidade usadas é de 375×1242 píxeis.

KITTI Discrete: Antes do lançamento do *KITTI Depth*, imagens de profundidade de referência não eram fornecidas diretamente, como já mencionado em seção 5.2. Portanto, foi necessário gerá-las a partir dos dados da nuvem de pontos do *KITTI Raw Data* dataset. No entanto, os mapas de profundidade gerados são muito esparsos, pois foram construídos a partir de uma única varredura do LiDAR. Este conjunto de dados foi

construído paralelamente ao *KITTI Continuous* e consiste em imagens de profundidade que são projeções diretas das nuvens de pontos no plano de imagem 2D da câmera, como ilustrado pela Figura 32.

Figura 32: Representação ilustrativa da geração de mapas de profundidade (esparso) – Projeção da nuvem de pontos no plano de imagem 2D da câmera.

Fonte: Imagem adaptada de Qi et al. (2018).

KITTI Depth: Devido à sua complexidade, o treinamento e a avaliação de redes convolucionais profundas exigem um grande número de pares de imagens anotadas. O *KITTI Depth* é um conjunto de dados de larga escala criado para permitir que fossem treinadas de ponta a ponta e de maneira supervisionada, já que outros conjuntos de dados como *Middlebury* (SCHARSTEIN; SZELISKI, 2002), *Make3D* (SAXENA; SUN; NG, 2009) e *KITTI* (GEIGER; LENZ; URTASUN, 2012; MENZE; GEIGER, 2015) não possuem dados suficientes para ajustar todos os parâmetros internos deste tipo de rede (UHRIG et al., 2017). O conjunto de dados é alinhado com cenas apresentadas no *KITTI Raw Data* dataset (GEIGER et al., 2013) e consiste em 92750 mapas de profundidade semi-densos, subdivididos em 85898 e 6852 pares para os subconjuntos de treinamento e validação, respectivamente. As imagens de profundidade foram obtidas por meio da acumulação de 11 varreduras do *laser*, cujos *outliers* foram removidos ao forçar a consistência entre os pontos do LiDAR e mapas de profundidade reconstruídos, gerados pelo algoritmo de correspondência estéreo semi-global (SGM) (UHRIG et al., 2017).

KITTI Continuous: Diferentemente do procedimento realizado no *KITTI Depth* para tornar as imagens de profundidade menos esparsas, que consistia em acumular diferentes varreduras do *laser*, neste trabalho foi gerado um *dataset* usando um modelo de ocupação para tornar os mapas de profundidade mais densos. Em outras palavras, o objetivo era aumentar o número de píxeis válidos disponíveis nas imagens de profundidade para treinamento. Nesse contexto, essa alternativa requer um número menor de imagens

de treinamento do que outras técnicas que usam conjuntos de dados com informações esparsas/semi-densas. O *KITTI Continuous* também é baseado em nuvens de pontos 3D do *Velodyne*, mas primeiro interpolou-se suas medidas com superfícies para gerar imagens contínuas de profundidade (mais detalhes na seção 5.2).

Devido a limitações no *framework* de pré-processamento, gerou-se um número menor de imagens que o *KITTI Depth* para os conjuntos de dados *KITTI Discrete* e *KITTI Continuous*. Mais especificamente, 27817 e 2911 amostras (Imagem RGB + Imagem de Profundidade) foram criadas para os subconjuntos de treinamento e validação, respectivamente. Mesmo com menos imagens, a divisão oficial de treinamento/validação do *KITTI Depth* foi usada como referência, mantendo estritamente a separação original dos pares, ou seja, os pares que faziam parte do subconjunto treinamento/validação continuaram a pertencer a ele.

O número resultante de pares para cada conjunto de dados está apresentado na Tabela 4. Em média, o número de píxeis válidos disponíveis nos *datasets KITTI Discrete* e *KITTI Depth* representa apenas 6,63 % e 24,33 % que o número de pontos disponíveis no *KITTI Continuous*, ou seja, respectivamente 15 e 4 vezes menores.

Tabela 4: Número de pares de imagens usados nos subconjuntos de treinamento e validação para cada *dataset*.

<i>Dataset</i>	Treinamento	Validação	Total	Porcentagem média de píxeis válidos ¹
KITTI Depth	85898	6852	92750	15,22%
KITTI Discrete	27817	2911	30728	4,15%
KITTI Continuous	27817	2911	30728	62,58%

5.3.1 Data Augmentation

Uma vez obtidos os pares de imagens RGB-D, é possível treinar as redes convolucionais, porém, por apresentarem milhares de parâmetros treináveis, existe a possibilidade do número de pares disponíveis não ser suficiente. Neste trabalho, dois tipos de transformações foram executadas *online*, isto é, durante o treinamento da rede, e de forma aleatória, aumentando artificialmente o número de amostras disponíveis.

1. **Espelhamento (*Flips*)**: A imagem de entrada e seu correspondente mapa de profundidade são invertidos horizontalmente com probabilidade de 50%.
2. **Distorção de Cor**: Altera a intensidade das componentes de cor em uma imagem RGB, as quais são ajustadas por valores aleatórios das seguintes transformações:

¹ Tamanho típico das imagens fornecidas pelos *KITTI datasets*, 375×1242 , usado como referência.

- *Brilho* por um valor aleatório $b \in [-32/255; 32/255]$
- *Saturação* por um valor aleatório $s \in [0, 5; 1, 5]$
- *Matiz* por um valor aleatório $h \in [-0, 2; 0, 2]$
- *Contraste* por um valor aleatório $c \in [0, 5; 1, 5]$

A ordem na qual as transformações acima são executadas também é escolhida aleatoriamente.

Como apontado por Eigen, Puhersch e Fergus (2014), a geometria espacial em uma cena do mundo não é preservada por transformações de escala e translações na imagem. Portanto, optou-se por não utilizar essas transformações e rotações, mesmo que esta última seja capaz de preservar a geometria. Além disso, acredita-se que distorções agressivas nas cores impedem que a rede se torne tendenciosa ao relacionar intensidades dos píxeis a valores de profundidade, concentrando-se, assim, em aprender as relações geométricas da cena.

5.4 Arquitetura da Rede

Neste trabalho, utilizou-se a arquitetura de rede *Fully Convolutional Residual Network* (FCRN) proposta por Laina et al. (2016). Essa rede foi selecionada por apresentar um menor número de parâmetros treináveis que algumas topologias clássicas (AlexNet, VGG), o que implica em requisitar um menor número de imagens para o treinamento, sem prejudicar a qualidade das predições. Além disso, os blocos residuais presentes na arquitetura permitem a construção de um modelo mais profundo capaz de estimar mapas de saída mais precisos e de maior resolução. Mais especificamente, a FCRN (Figura 33) é baseada na topologia ResNet-50, mas as camadas totalmente conectadas foram substituídas por um conjunto de blocos residuais de expansão (*residual upsampling blocks*), também conhecidos como *up-projections*, que são camadas responsáveis por deconvoluir e recuperar a resolução espacial dos mapas de características. Os fundamentos teóricos necessários para compreender cada uma das camadas presentes na arquitetura foram apresentados na seção 2.4.

Essa rede foi treinada de ponta a ponta de forma supervisionada, mas diferentemente dos autores que a propuseram, modificou-se sua saída para predizer distâncias em metros (d), em vez de distâncias em espaço logarítmico, $\log(d)$. A rede usa imagens RGB de 228×304 píxeis como entradas para treinar os 63M parâmetros treináveis e fornece um mapa de saída com tamanho de 128×160 píxeis.

Figura 33: A arquitetura de rede utilizada foi proposta por Laina et al. (2016), que é inspirada na ResNet-50, porém suas camadas totalmente conectadas foram substituídas por blocos de *upsampling*. Diferentemente dos autores originais, alterou-se a estrutura de treinamento para fazer com que a rede proposta estime metros em vez de distâncias em escala logarítmica.

Fonte: Imagem adaptada de Laina et al. (2016).

5.5 Funções de Custo

Apresentada a arquitetura da rede, as imagens RGB e os mapas de referência precisam ser redimensionados para as dimensões, respectivamente, de entrada e saída da rede. Para isso, o método de interpolação por área foi utilizado, uma vez que é o algoritmo mais indicado para encolhimento de imagens (WONG; HERLEY, 1999). Neste ponto, deve-se definir uma expressão matemática que melhor aproxime as saídas (y_i) aos valores de referência (y_i^*) para o i -ésimo píxel, e que melhor suavize as previsões da rede. Estas expressões são as chamadas funções de custo e estabelecem o modo que os pesos e limiares precisam ser ajustados, uma vez que sua lei matemática ditam os parâmetros precisam ser reforçados ou penalizados. A seguir estão apresentadas as funções de custo que foram utilizadas para aprendizado supervisionado, no caso a supervisão é feita por mapas de profundidade. Mais especificamente, empregou-se as funções de custo *mse*, *eigen* e *berhu* para ajustar os parâmetros internos da rede neural profunda apresentada (CARVALHO et al., 2018). A motivação por trás disso foi simplesmente determinar qual seria a função mais adequada. As expressões matemáticas para cada uma delas estão apresentadas a seguir:

5.5.1 Mean Squared Error (*mse*)

É a função de custo mais comumente utilizada para otimização de redes neurais, na qual calcula-se o erro quadrático médio (Equação 5.1) entre as previsões (y) e os

ground-truths (y^*) (LAINA et al., 2016).

$$\mathcal{L}_2(y - y^*) = \|y - y^*\|_2^2 = \frac{1}{n} \sum_i^n (y_i - y_i^*)^2 \quad (5.1)$$

5.5.2 Scale-Invariant Mean Squared Error (*eigen*)

No contexto de predição de profundidade, descobrir a escala global de uma cena é uma tarefa naturalmente ambígua. Eigen, Puhrsch e Fergus (2014) constataram que subtrair a escala média das cenas avaliadas (2º termo da equação 5.2) resulta em uma melhora significativa na forma em que a rede realiza as predições. Posteriormente, constataram que a adição dos gradientes horizontais e verticais ($\nabla_x d_i$, $\nabla_y d_i$) da imagem de diferença (3º termo da equação 5.3), força as estruturas locais serem semelhantes (EIGEN; FERGUS, 2015).

$$\mathcal{L}_{eigen}(y, y^*) = \frac{1}{n} \sum_i d_i^2 - \frac{\lambda}{n^2} \left(\sum_i d_i \right)^2, \quad d_i = \log y_i - \log y_i^*, \quad \lambda = \frac{1}{2} \quad (5.2)$$

$$\mathcal{L}_{eigen_{grads}}(y, y^*) = \frac{1}{n} \sum_i d_i^2 - \frac{\lambda}{n^2} \left(\sum_i d_i \right)^2 + \frac{1}{n} \sum_i [(\nabla_x d_i)^2 + (\nabla_y d_i)^2] \quad (5.3)$$

5.5.3 Adaptive BerHu Penalty (*berhu*)

Também conhecida como função reversa de Huber, *Reverse Huber's Function*, foi proposta a fim de obter estimativas mais robustas, penalizando diferentemente as predições que estão próximas da fonte (isto é, câmera) do que aquelas que estão distantes (LAINA et al., 2016; OWEN, 2007; ZWALD; LAMBERT-LACROIX, 2012).

$$\mathcal{L}_{berhu}(x) = \begin{cases} |x| & |x| \leq c, \\ \frac{x^2 + c^2}{2c} & |x| > c \end{cases}, \quad x = y - y^*, \quad c = \frac{1}{5} \max_i (|y_i - y_i^*|) \quad (5.4)$$

Como apresentado na Equação 5.4, a penalização se adapta de acordo com o quão distantes as estimativas estão das profundidades de referência, onde menores valores de erro estão sujeitos à norma $\mathcal{L}_1$, enquanto que maiores valores, à norma $\mathcal{L}_2$.

5.6 Nuvem de pontos a partir da estimação SIDE

Como mencionado anteriormente, os mapas estimados são capazes de beneficiar outras tarefas como reconhecimento de objetos, segmentação semântica, compreensão de cenas, dentre outras (CAO; WU; SHEN, 2018). Além disso, caso os parâmetros intrínsecos

da câmera sejam fornecidos, também é possível gerar as nuvens de pontos correspondentes. Desta forma, as estimativas da rede podem auxiliar outras aplicações em robótica móvel e realidade aumentada, porém desta vez fornecendo informações tridimensionais. Em navegação autônoma, podem ser integradas ao SLAM, sendo utilizadas como características durante a etapa de mapeamento, o que resulta na reconstrução densa do ambiente. Esta nova vertente da técnica é chamada de *Deep Monocular SLAM* (ZHAO et al., 2018; ZHOU; UMMENHOFER; BROX, 2018).

Os modelos desenvolvidos podem ser encapsulados de modo que possam ser integrados a sistemas de percepção mais complexos para que, então, sejam implantados em plataformas robóticas. Operacionalmente, uma forma prática para fazer isto é utilizando ROS (*Robot Operating System*), sistema meta-operacional essencialmente desenhado para facilitar o desenvolvimento de robôs (QUIGLEY et al., 2009). Mais precisamente, sua arquitetura de comunicação foi estruturada na forma de grafos, permitindo que o processamento seja realizado em nós que podem receber e enviar mensagens, além de também possuir outras funcionalidades de um sistema operacional, tais como: abstração de hardware, controle de dispositivos de baixo nível, gerenciamento de pacotes, etc.

Neste trabalho, a arquitetura desenvolvida em ROS (Figura 34) procura cobrir parte do *pipeline* de processamento para reconstrução de ambientes, porém não apresenta o módulo responsável pela Localização e Mapeamento Simultâneo (SLAM).

Figura 34: Diagrama de blocos da arquitetura implementada no ROS.

Fonte: Elaborada pelo autor.

O nó referente à *ROSBAG* das cenas do KITTI, além de fornecer informações do LiDAR e as imagens capturadas (coloridas e em nível de cinza), também fornece informações acerca da calibração das câmeras, contendo dados como: resolução das imagens, vetor e modelo de distorção das lentes (D), matriz de calibração (K), matriz de retificação (R), matriz de projeção (P), dentre outras.

$$D = [k_1, k_2, t_1, t_2, k_3] \quad (5.5)$$

$$= [-0.3691481, 0.1968681, 0.001353473, 0.0005677587, -0.06770705] \quad (5.6)$$

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 959.791 & 0.0 & 696.0217 \\ 0.0 & 956.9251 & 224.1806 \\ 0.0 & 0.0 & 1.0 \end{bmatrix} \quad (5.7)$$

$$R = \begin{bmatrix} 0.9998817 & 0.01511453 & -0.002841595 \\ -0.01511724 & 0.9998853 & -0.000933851 \\ 0.002827154 & 0.0009766976 & 0.9999955 \end{bmatrix} \quad (5.8)$$

$$P = \begin{bmatrix} f'_x & 0 & c'_x & T_x \\ 0 & f'_y & c'_y & T_y \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 721.5377 & 0.0 & 609.5593 & 44.85728 \\ 0.0 & 721.5377 & 172.854 & 0.2163791 \\ 0.0 & 0.0 & 1.0 & 0.002745884 \end{bmatrix} \quad (5.9)$$

Com as informações disponibilizadas é possível projetar as imagens capturadas no sensor de imagem da câmera para o mundo ($2D \rightarrow 3D$). No caso, utilizou-se o tópico referente às imagens RGB da câmera esquerda para gerar as predições da rede, e assumiu-se que as imagens de profundidade geradas apresentariam as mesmas informações de calibração desta câmera, que fora utilizada como entrada.

A seguir, utilizou-se o pacote *depth_image_proc* do ROS, o qual fornece *nodelets* para o processamento de imagens de disparidade e de profundidade, por exemplo, para gerar nuvem de pontos. O *nodelet point_cloud_xyz* foi utilizado para converter a imagem de profundidade predita para uma nuvem XYZ, o qual requer a imagem de profundidade e suas informações de calibração como entradas. Para projetar um píxel 2D para um ponto 3D, é necessário conhecer os comprimentos focais (f_x, f_y) e o ponto principal da câmera (c_x, c_y). Desta forma, um píxel $p(u, v)$, conhecendo-se o valor de distância a partir da imagem de profundidade estimada $D(u, v)$ e um fator de correção de escala s , é possível encontrar as coordenadas no mundo (X,Y,Z), segundo:

$$X = (u - c_x) * D(u, v) * \frac{s}{f_x}, \quad s = 1 \quad (5.10)$$

$$Y = (v - c_y) * D(u, v) * \frac{s}{f_y} \quad (5.11)$$

$$Z = D(u, v) \quad (5.12)$$

6 RESULTADOS EXPERIMENTAIS

6.1 Descrição do Experimento

Para demonstrar a eficácia do método proposto, desenvolveu-se um ambiente de experimentação para testes onde apenas o grau de esparsidade fosse avaliado, no qual a mesma arquitetura de rede e os mesmos hiperparâmetros de treinamento foram utilizados. Assim, os resultados obtidos demonstram como os diferentes níveis de esparsidade (esparsos, semi-densos e densos) nos conjuntos de dados aqui considerados influenciam a qualidade das predições finais. No caso, este experimento investiga se a utilização de mapas de profundidade densificados melhora a qualidade das estimativas de profundidade de redes neurais profundas. Para isso, tomou-se os modelos treinados no *KITTI Discrete* (esparso, sem HM) como referencial de comparação – linha de base (*baseline*).

6.2 Detalhes da Implementação

Neste trabalho, utilizou-se o *TensorFlow* (ABADI et al., 2016) como *framework* de desenvolvimento, visto que permite que todo processamento das redes possa ser executado em GPU, além de permitir que sejam implementadas em *Python*. Mais especificamente, os modelos foram treinados nos conjuntos de dados propostos (seção 5.3), cujos valores dos filtros foram inicialmente determinados pelo algoritmo de Xavier (inicializador) (GLOROT; BENGIO, 2010) e posteriormente atualizados pelo algoritmo ADAM (otimizador) (KINGMA; BA, 2014).

Os valores de tamanho de lote (*batch size*) e passos de treinamento (*training steps*) utilizados foram iguais a 4 e a 300.000, respectivamente. O valor inicial da taxa de aprendizado foi de 0,0001, reduzindo 5% a cada 1000 passos. Além de decaimento de taxa de aprendizado (*learning decay*), também empregou-se desligamento (*dropout*) de 50% e normalização $\mathcal{L}_2$ como técnicas de regularização (SRIVASTAVA et al., 2014c; GOODFELLOW; BENGIO; COURVILLE, 2015). Informações mais detalhadas do equipamento utilizado estão apresentadas na Tabela 5.

Mesmo utilizando 300 mil passos como critério de parada, em média, a convergência dos modelos ocorria entre 150~180 mil passos, o que, no *hardware* especificado, representava aproximadamente 2 dias de treinamento para os modelos treinados no *KITTI Discrete* e *Continuous*, e 4 dias, no *KITTI Depth*.

6.3 Avaliação de Desempenho (*Benchmark*)

Nesta seção foi realizada a avaliação de desempenho (*benchmark evaluation*) das redes treinadas. Trabalhos no estado da arte costumam ser avaliados utilizando o *dataset*

Tabela 5: Especificações do *Hardware* e *Softwares* utilizados.

Máquina	Olorin
Sistema Operacional	Ubuntu 16.04 LTS 64-bits
Processador	Intel Core i7-7700K CPU @ 4.20 GHz x 8
Memória RAM	16 Gb
Armazenamento	256 Gb (SSD) 2 Tb (HDD)
Placa Gráfica	GPU Titan Xp/PCIe/SSE2 (12 Gb VRAM)
CUDA	v9.0.176
cuDNN	v7.2.1
TensorFlow	v1.10.1
Python	v3.5

Fonte: Elaborada pelo autor.

KITTI Raw Data, e por não possuir uma divisão oficial de treinamento/teste, Eigen, Puhersch e Fergus (2014) subdividiram as imagens disponíveis em 33131 para treinamento e 697 para teste, esta separação é mais comumente conhecida por *Eigen Split*. Como esse subconjunto de teste é muito esparsos para uma medição precisa da qualidade preditiva das estimativas de profundidade, descartou-se o método de avaliação utilizando as 697 imagens esparsas do *KITTI Raw Data* e utilizou-se as 652 imagens de teste correspondentes do *KITTI Depth* como alternativa para avaliação dos modelos. Este último subconjunto de teste foi denominado *Eigen Split para KITTI Depth* e utilizado para avaliação dos três *datasets* explorados neste trabalho, *KITTI Discrete*, *KITTI Depth* e *KITTI Continuous*. Em outras palavras, foram avaliados usando as mesmas 652 imagens de teste (semi-densas) do *KITTI Depth*, não necessariamente seguindo o *split* oficial do *dataset*, mas sim a subdivisão proposta no *Eigen Split*. Por fim, as predições das redes foram redimensionadas para o tamanho do mapa de profundidade original usando interpolação bilinear e, em seguida, comparadas com os mapas de profundidade de referência correspondentes.

Eigen Split para KITTI Depth (semi-denso): Para uma avaliação justa dos métodos aqui considerados, escolheu-se um subconjunto de teste que fornecesse um número maior de pontos de avaliação, já que a técnica proposta melhora a qualidade de predição não apenas para pontos esparsos dos mapas de profundidade originais – que são contabilizados nas métricas – mas também para a cena como um todo. Essa modificação torna possível destacar ainda mais os benefícios da densificação da nuvem de pontos ao analisar os impactos do uso de mapas de profundidade esparsos e contínuos. Em resumo, este *subset* de teste preocupa-se em ser alinhado com o *Eigen Split* (*KITTI Raw data*), porém utiliza as imagens de profundidade correspondentes do conjunto de dados *KITTI Depth*. No entanto, este *dataset* fornece apenas 652 mapas precisos de referência para avaliação, ou seja, 93,5 % da subdivisão original do *Eigen Split* é coberta. Além dessa pequena redução

nas amostras disponíveis, o uso dessas imagens propostas, que são mais confiáveis do que as imagens esparsas usadas no *Eigen Split*, permite melhor avaliar o desempenho dos métodos de estimação de profundidade (ALEOTTI et al., 2018). Todos os resultados apresentados neste trabalho foram obtidos através do subconjunto de testes mencionado acima.

6.4 Métricas de Avaliação

Visto que os resultados finais geralmente são um conjunto de predições das imagens do subconjunto de teste, a análise qualitativa (visual) pode ser tendenciosa e não é suficiente para dizer se uma abordagem é melhor que outra. Dessa forma, vários trabalhos utilizam as seguintes métricas para avaliar as saídas estimadas (y_i) de métodos com os mapas de referência correspondentes (y_i^*) e, assim, possibilitando compará-los com outras técnicas presentes na literatura (EIGEN; PUHRSCHE; FERGUS, 2014; LI et al., 2015; CAO; WU; SHEN, 2018):

Erro Relativo Médio (Abs Rel): $\frac{1}{|T|} \sum_{i=1}^T \frac{|y_i - y_i^*|}{y_i^*}$

Erro Quadrático Médio (Sqr Rel): $\frac{1}{|T|} \sum_{i=1}^T \frac{\|y_i - y_i^*\|^2}{y_i^*}$

Raiz do Erro Quadrático Médio (RMSE): $\sqrt{\frac{1}{|T|} \sum_{i=1}^T \|y_i - y_i^*\|^2}$

Raiz do Erro Quadrático Médio (RMSE – log): $\sqrt{\frac{1}{|T|} \sum_{i=1}^T \|\log y_i - \log y_i^*\|^2}$

Taxa de acerto com Limiar (δ): % de y_i sujeito a $\max\left(\frac{y_i}{y_i^*}, \frac{y_i^*}{y_i}\right) = \delta < thr$, $thr \in \{1, 25; 1, 25^2; 1, 25^3\}$

Erro Logarítmico Médio (log10): $\frac{1}{|T|} \sum_{i=1}^T |\log_{10} y_i - \log_{10} y_i^*|$

onde T é o número de píxeis válidos de todas as imagens avaliadas. Além disso, a fim de comparar os resultados obtidos com outros trabalhos, também utilizou-se o protocolo de avaliação de restringir os valores de profundidade de referência e as predições a certos intervalos, mais precisamente os intervalos de $0 - 50\text{ m}$ e $0 - 80\text{ m}$. Em outras palavras, restringe-se as predições aos intervalos em questão, quando necessário, satura-se as distâncias estimadas aos valores mínimos e máximos. Alguns trabalhos (GARG et al., 2016; GODARD; AODHA; BROSTOW, 2017) requerem intervalos diferentes para serem comparados de forma justa.

6.5 Resultados

6.5.1 Densificação de Mapas (*Depth Completion Evaluation*)

Assim como os métodos de estimação de profundidade, as técnicas de densificação de mapas também podem ser avaliadas segundo as estratégias de avaliação apresentadas na seção 6.3. No caso, como a técnica de mapas de Hilbert gera imagens contínuas de

profundidade, optou-se por também avaliar o seu desempenho em imagens semi-densas do *Eigen Split para KITTI Depth*. Como já mencionado, este subconjunto de teste especifica 652 imagens para avaliação, porém o *KITTI Continuous* possui apenas 594 imagens em comum. O cálculo das métricas pode ser realizado mesmo assim, porém os resultados aqui apresentados (Tabela 6) não devem ser diretamente comparados com outros métodos de densificação.

Tabela 6: Resultados da avaliação de desempenho da técnica de mapas de Hilbert para a tarefa de densificação de mapas.

Método	Intervalo	Abs Rel	Sqr Rel	RMSE	RMSE (log)	$\delta < 1, 25$	$\delta < 1, 25^2$	$\delta < 1, 25^3$
			menor é melhor			mais alto é melhor		
Mapas de Hilbert (Denso)	0 – 50 m	0,111	0,622	4,606	0,615	0,924	0,965	0,979
	0 – 80 m	0,107	0,566	4,179	0,614	0,933	0,965	0,978

Fonte: Elaborada pelo autor.

Os resultados obtidos nesta avaliação são os limites superiores de desempenho da tarefa de estimação de profundidade, uma vez que a rede é treinada nas imagens de profundidade geradas pelos mapas de Hilbert, as mesmas que compõe o dataset *KITTI Continuous*. Em outras palavras, os melhores modelos da rede, avaliados na seção a seguir, dificilmente superariam os valores aqui apresentados, estes que em seu maior potencial desempenhariam, no máximo, comportamento semelhante à técnica que gerou suas imagens de referência.

6.5.2 Estimação de Profundidade (*Depth Estimation Evaluation*)

Além de avaliar a arquitetura FCRN usando diferentes versões dos conjuntos de dados do KITTI, alguns testes exploratórios foram realizados para identificar a melhor combinação de treinamento. Mais especificamente, testou-se diferentes funções de custo e estudou-se a influência do uso de todos os píxeis (px_{all}), que incluem o céu e superfícies refletoras, ou apenas píxeis válidos (px_{valid}), que mandatoriamente devem possuir informação de profundidade correspondente.

A bateria de testes foi composta pela combinação de três *datasets*; duas máscaras de píxeis, que definem quais contribuíram, ou não, para o cálculo da função de custo; e quatro funções de custo diferentes, como apresentado na Tabela 7. A variação do primeiro hiperparâmetro demonstra a sensibilidade dos resultados quanto ao grau de esparsidade presente nos mapas, a do segundo indica como os píxeis que não contêm informação de profundidade correspondente influenciam no aprendizado da rede, e a do último hiperparâmetro, corresponde à busca pela função de custo que melhor contribua para otimização da rede, ou seja, aquela que pondere de forma mais eficiente aspectos da predição que devem ser reforçados ou penalizados para que isso seja atingido.

Ao menos, 24 casos distintos precisavam ser executados. Na prática, algumas configurações foram testadas mais vezes que outras, sendo realizados na medida que se encontravam resultados mais promissores.

Tabela 7: Lista de Hiperparâmetros

<i>Datasets</i>	Configurações de Píxel	Funções de Custo
<i>KITTI Discrete</i>		<i>berhu</i>
<i>KITTI Depth</i>	px_{all}	<i>mse</i>
<i>KITTI Continuous</i>	px_{valid}	<i>eigen</i>
		<i>eigen_{grads}</i>

Fonte: Elaborada pelo autor.

Tabela 8: Tabela resumida. Resultados obtidos das melhores configurações da FCRN quando treinadas nos conjuntos de dados propostos, os quais possuem diferentes níveis de esparsidade nas imagens de profundidade de referência, e avaliadas no *Eigen Split para KITTI Depth*.

Dataset	Intervalo	Abs Rel	Sqr Rel	RMSE	RMSE (log)	$\delta < 1,25$	$\delta < 1,25^2$	$\delta < 1,25^3$
			menor é melhor			mais alto é melhor		
KITTI Discrete (esparso)	0 – 50 <i>m</i>	0,904	12,835	17,643	2,434	0,000	0,000	0,000
KITTI Depth (semi-denso)	0 – 50 <i>m</i>	0,619	6,268	12,448	1,199	0,004	0,075	0,246
KITTI Continuous (denso)	0 – 50 <i>m</i>	0,119	0,653	4,705	0,211	0,894	0,966	0,985
KITTI Discrete (esparso)	0 – 80 <i>m</i>	0,904	12,835	17,643	2,434	0,000	0,000	0,000
KITTI Depth (semi-denso)	0 – 80 <i>m</i>	0,619	6,268	12,448	1,199	0,004	0,075	0,246
KITTI Continuous (denso)	0 – 80 <i>m</i>	0,118	0,630	4,520	0,209	0,898	0,966	0,985

Fonte: Elaborada pelo autor.

Na tabela 8, apresenta-se os resultados obtidos pelas melhores configurações da FCRN quando treinada nos conjuntos de dados propostos e avaliada no *Eigen Split para KITTI Depth*. Já as tabelas 9 e 10 apresentam os resultados obtidos de todas as configurações treinadas, no caso os dados foram separados conforme o intervalo no qual foram avaliados, 0 – 50 *m* e 0 – 80 *m*. Como pode-se observar, os resultados das tabelas estão agrupados de acordo com o *dataset* utilizado e ordenadas de acordo com a métrica *Abs Rel*. Desta forma, é possível identificar as configurações que apresentaram os melhores resultados no subconjunto testado, sendo aqueles realçados em negrito.

Nos experimentos realizados, os modelos treinados no *KITTI Continuous* apresentaram os melhores desempenhos, como esperado inicialmente, bem como superaram os modelos treinados nos conjuntos de dados esparsos e semi-densos por uma margem considerável. Além disso, os melhores modelos foram coincidentemente obtidos ao utilizar a função de custo de *BerHu*, o que reafirma as conclusões apresentadas por Laina et al. (2016), e ao utilizar apenas informações válidas (px_{valid}).

Como todos os modelos têm a mesma arquitetura de rede, esses ganhos de desempenho demonstram como o processo de densificação produzido pelo *framework*

Tabela 9: Tabela Completa. Resultados de todas as configurações da FCRN quando treinadas nos conjuntos de dados propostos e avaliadas no *Eigen Split para KITTI Depth* considerando o intervalo de 0 – 50 m.

Modelo	Dataset	Intervalo	Abs Rel	Sqr Rel	RMSE	RMSE (log)	$\delta < 1, 25$ $\delta < 1, 25^2$ $\delta < 1, 25^3$		
							menor é melhor mais alto é melhor		
FCRN, px_{valid}, <i>berhu</i>	KITTI Continuous (denso)	0 – 50 m	0,119	0,653	4,705	0,211	0,894	0,966	0,985
FCRN, px_{valid} , <i>berhu</i>	KITTI Continuous (denso)	0 – 50 m	0,119	0,656	4,734	0,204	0,891	0,966	0,985
FCRN, px_{valid} , <i>berhu</i>	KITTI Continuous (denso)	0 – 50 m	0,120	0,658	4,727	0,204	0,891	0,965	0,984
FCRN, px_{valid} , <i>berhu</i>	KITTI Continuous (denso)	0 – 50 m	0,120	0,653	4,713	0,207	0,892	0,966	0,985
FCRN, px_{valid} , <i>berhu</i>	KITTI Continuous (denso)	0 – 50 m	0,120	0,651	4,711	0,204	0,892	0,966	0,985
FCRN, px_{valid} , <i>berhu</i>	KITTI Continuous (denso)	0 – 50 m	0,121	0,659	4,724	0,205	0,890	0,965	0,985
FCRN, px_{valid} , <i>mse</i>	KITTI Continuous (denso)	0 – 50 m	0,125	0,674	4,786	0,201	0,877	0,966	0,986
FCRN, px_{valid} , <i>mse</i>	KITTI Continuous (denso)	0 – 50 m	0,125	0,674	4,790	0,204	0,878	0,966	0,986
FCRN, px_{all} , <i>berhu</i>	KITTI Continuous (denso)	0 – 50 m	0,126	0,806	5,383	0,477	0,881	0,962	0,981
FCRN, px_{all} , <i>mse</i>	KITTI Continuous (denso)	0 – 50 m	0,131	0,772	5,150	0,350	0,866	0,964	0,984
FCRN, px_{valid} , <i>berhu</i>	KITTI Continuous (denso)	0 – 50 m	0,141	0,927	5,433	0,248	0,828	0,930	0,965
FCRN, px_{valid} , <i>berhu</i>	KITTI Continuous (denso)	0 – 50 m	0,141	0,926	5,429	0,248	0,828	0,931	0,965
FCRN, px_{valid} , <i>berhu</i>	KITTI Continuous (denso)	0 – 50 m	0,141	0,929	5,450	0,248	0,827	0,930	0,965
FCRN, px_{valid} , <i>berhu</i>	KITTI Continuous (denso)	0 – 50 m	0,141	0,933	5,429	0,249	0,828	0,930	0,965
FCRN, px_{valid} , <i>berhu</i>	KITTI Continuous (denso)	0 – 50 m	0,142	0,937	5,450	0,249	0,826	0,929	0,964
FCRN, px_{valid} , <i>berhu</i>	KITTI Continuous (denso)	0 – 50 m	0,142	0,933	5,441	0,249	0,825	0,929	0,964
FCRN, px_{valid} , <i>eigen_grads</i>	KITTI Continuous (denso)	0 – 50 m	0,143	0,912	5,780	0,234	0,820	0,949	0,980
FCRN, px_{valid} , <i>eigen</i>	KITTI Continuous (denso)	0 – 50 m	0,145	0,918	5,807	0,229	0,816	0,947	0,980
FCRN, px_{valid} , <i>mse</i>	KITTI Continuous (denso)	0 – 50 m	0,146	0,928	5,422	0,245	0,812	0,932	0,969
FCRN, px_{valid} , <i>mse</i>	KITTI Continuous (denso)	0 – 50 m	0,146	0,932	5,432	0,246	0,813	0,932	0,969
FCRN, px_{all} , <i>berhu</i>	KITTI Continuous (denso)	0 – 50 m	0,147	1,052	5,914	0,423	0,817	0,925	0,962
FCRN, px_{all} , <i>eigen</i>	KITTI Continuous (denso)	0 – 50 m	0,150	1,133	6,617	0,367	0,808	0,941	0,973
FCRN, px_{all} , <i>mse</i>	KITTI Continuous (denso)	0 – 50 m	0,151	0,994	5,640	0,310	0,803	0,929	0,967
FCRN, px_{all} , <i>eigen_grads</i>	KITTI Continuous (denso)	0 – 50 m	0,152	1,156	6,660	0,375	0,800	0,939	0,973
FCRN, px_{valid} , <i>eigen_grads</i>	KITTI Continuous (denso)	0 – 50 m	0,161	1,132	6,231	0,268	0,765	0,916	0,962
FCRN, px_{valid} , <i>eigen</i>	KITTI Continuous (denso)	0 – 50 m	0,163	1,138	6,269	0,268	0,762	0,914	0,962
FCRN, px_{all} , <i>eigen</i>	KITTI Continuous (denso)	0 – 50 m	0,168	1,323	6,903	0,332	0,752	0,906	0,956
FCRN, px_{all} , <i>eigen_grads</i>	KITTI Continuous (denso)	0 – 50 m	0,170	1,340	6,941	0,338	0,745	0,904	0,955
<hr/>									
FCRN, px_{valid}, <i>berhu</i>	KITTI Depth (semi-denso)	0 – 50 m	0,619	6,268	12,448	1,199	0,004	0,075	0,246
FCRN, px_{valid} , <i>berhu</i>	KITTI Depth (semi-denso)	0 – 50 m	0,623	6,324	12,485	1,217	0,004	0,075	0,241
FCRN, px_{valid} , <i>mse</i>	KITTI Depth (semi-denso)	0 – 50 m	0,624	6,232	12,313	1,163	0,003	0,053	0,204
FCRN, px_{valid} , <i>mse</i>	KITTI Depth (semi-denso)	0 – 50 m	0,625	6,237	12,313	1,166	0,003	0,052	0,202
FCRN, px_{all} , <i>mse</i>	KITTI Depth (semi-denso)	0 – 50 m	0,664	7,028	13,013	1,483	0,002	0,047	0,178
FCRN, px_{valid} , <i>mse</i>	KITTI Depth (semi-denso)	0 – 50 m	0,666	7,117	13,223	1,339	0,003	0,043	0,157
FCRN, px_{valid} , <i>mse</i>	KITTI Depth (semi-denso)	0 – 50 m	0,667	7,125	13,219	1,345	0,003	0,043	0,156
FCRN, px_{all} , <i>berhu</i>	KITTI Depth (semi-denso)	0 – 50 m	0,667	7,340	13,486	2,585	0,004	0,066	0,218
FCRN, px_{valid} , <i>eigen_grads</i>	KITTI Depth (semi-denso)	0 – 50 m	0,668	7,319	13,595	1,289	0,001	0,028	0,128
FCRN, px_{valid} , <i>eigen</i>	KITTI Depth (semi-denso)	0 – 50 m	0,670	7,374	13,664	1,292	0,001	0,031	0,126
FCRN, px_{valid} , <i>berhu</i>	KITTI Depth (semi-denso)	0 – 50 m	0,671	7,296	13,415	1,442	0,003	0,061	0,188
FCRN, px_{valid} , <i>berhu</i>	KITTI Depth (semi-denso)	0 – 50 m	0,674	7,339	13,436	1,458	0,003	0,061	0,185
FCRN, px_{valid} , <i>eigen</i>	KITTI Depth (semi-denso)	0 – 50 m	0,707	8,117	14,285	1,471	0,001	0,026	0,101
FCRN, px_{valid} , <i>eigen_grads</i>	KITTI Depth (semi-denso)	0 – 50 m	0,708	8,108	14,251	1,505	0,001	0,024	0,101
FCRN, px_{all} , <i>mse</i>	KITTI Depth (semi-denso)	0 – 50 m	0,720	8,232	14,114	1,946	0,002	0,038	0,135
FCRN, px_{all} , <i>eigen_grads</i>	KITTI Depth (semi-denso)	0 – 50 m	0,722	8,624	14,791	1,715	0,001	0,022	0,101
FCRN, px_{all} , <i>eigen</i>	KITTI Depth (semi-denso)	0 – 50 m	0,726	8,698	14,864	1,745	0,001	0,022	0,098
FCRN, px_{all} , <i>berhu</i>	KITTI Depth (semi-denso)	0 – 50 m	0,734	8,686	14,557	3,862	0,003	0,054	0,166
FCRN, px_{all} , <i>eigen_grads</i>	KITTI Depth (semi-denso)	0 – 50 m	0,772	9,653	15,541	2,216	0,001	0,018	0,078
FCRN, px_{all} , <i>eigen</i>	KITTI Depth (semi-denso)	0 – 50 m	0,774	9,692	15,581	2,287	0,001	0,018	0,077
<hr/>									
FCRN, px_{valid}, <i>mse</i>	KITTI Discrete (esparso)	0 – 50 m	0,904	12,835	17,643	2,434	0,000	0,000	0,000
FCRN, px_{valid} , <i>berhu</i>	KITTI Discrete (esparso)	0 – 50 m	0,905	12,847	17,655	2,455	0,000	0,000	0,000
FCRN, px_{valid} , <i>mse</i>	KITTI Discrete (esparso)	0 – 50 m	0,907	12,873	17,628	2,474	0,000	0,000	0,000
FCRN, px_{valid} , <i>berhu</i>	KITTI Discrete (esparso)	0 – 50 m	0,908	12,890	17,638	2,519	0,000	0,000	0,000
FCRN, px_{valid} , <i>eigen</i>	KITTI Discrete (esparso)	0 – 50 m	0,909	13,050	17,852	2,475	0,000	0,000	0,000
FCRN, px_{valid} , <i>eigen</i>	KITTI Discrete (esparso)	0 – 50 m	0,909	13,034	17,796	2,491	0,000	0,000	0,000
FCRN, px_{valid} , <i>eigen_grads</i>	KITTI Discrete (esparso)	0 – 50 m	0,911	13,057	17,829	2,524	0,000	0,000	0,000
FCRN, px_{valid} , <i>eigen_grads</i>	KITTI Discrete (esparso)	0 – 50 m	0,915	13,129	17,817	2,699	0,000	0,000	0,000
FCRN, px_{all} , <i>mse</i>	KITTI Discrete (esparso)	0 – 50 m	0,918	13,156	17,843	2,769	0,000	0,000	0,000
FCRN, px_{all} , <i>berhu</i>	KITTI Discrete (esparso)	0 – 50 m	0,919	13,183	17,878	3,226	0,000	0,000	0,000
FCRN, px_{all} , <i>mse</i>	KITTI Discrete (esparso)	0 – 50 m	0,922	13,223	17,845	2,969	0,000	0,000	0,000
FCRN, px_{all} , <i>berhu</i>	KITTI Discrete (esparso)	0 – 50 m	0,922	13,252	17,882	3,416	0,000	0,000	0,000
FCRN, px_{all} , <i>eigen_grads</i>	KITTI Discrete (esparso)	0 – 50 m	0,926	13,476	18,130	2,845	0,000	0,000	0,000
FCRN, px_{all} , <i>eigen</i>	KITTI Discrete (esparso)	0 – 50 m	0,927	13,526	18,177	2,847	0,000	0,000	0,000
FCRN, px_{all} , <i>eigen_grads</i>	KITTI Discrete (esparso)	0 – 50 m	0,928	13,512	18,109	3,038	0,000	0,000	0,000
FCRN, px_{all} , <i>eigen</i>	KITTI Discrete (esparso)	0 – 50 m	0,929	13,541	18,139	2,989	0,000	0,000	0,000

Fonte: Elaborada pelo autor.

Tabela 10: Tabela Completa. Resultados de todas as configurações da FCRN quando treinadas nos conjuntos de dados propostos e avaliadas no *Eigen Split para KITTI Depth* considerando o intervalo de 0 – 80 *m*.

Modelo	Dataset	Intervalo	Abs Rel	Sqr Rel	RMSE	RMSE (log)	$\delta < 1, 25$	$\delta < 1, 25^2$	$\delta < 1, 25^3$
				menor é melhor				mais alto é melhor	
FCRN, px_{valid}, <i>berhu</i>	KITTI Continuous (denso)	0 – 80 <i>m</i>	0,118	0,630	4,520	0,209	0,898	0,966	0,985
FCRN, px_{valid} , <i>berhu</i>	KITTI Continuous (denso)	0 – 80 <i>m</i>	0,118	0,633	4,540	0,203	0,895	0,965	0,984
FCRN, px_{valid} , <i>berhu</i>	KITTI Continuous (denso)	0 – 80 <i>m</i>	0,118	0,634	4,557	0,203	0,895	0,966	0,985
FCRN, px_{valid} , <i>berhu</i>	KITTI Continuous (denso)	0 – 80 <i>m</i>	0,118	0,630	4,528	0,206	0,896	0,966	0,985
FCRN, px_{valid} , <i>berhu</i>	KITTI Continuous (denso)	0 – 80 <i>m</i>	0,119	0,628	4,530	0,202	0,896	0,966	0,985
FCRN, px_{valid} , <i>berhu</i>	KITTI Continuous (denso)	0 – 80 <i>m</i>	0,120	0,635	4,536	0,204	0,894	0,965	0,985
FCRN, px_{valid} , <i>mse</i>	KITTI Continuous (denso)	0 – 80 <i>m</i>	0,123	0,649	4,592	0,200	0,881	0,966	0,986
FCRN, px_{valid} , <i>mse</i>	KITTI Continuous (denso)	0 – 80 <i>m</i>	0,124	0,648	4,596	0,202	0,882	0,967	0,986
FCRN, px_{all} , <i>berhu</i>	KITTI Continuous (denso)	0 – 80 <i>m</i>	0,125	0,785	5,243	0,477	0,885	0,962	0,981
FCRN, px_{all} , <i>mse</i>	KITTI Continuous (denso)	0 – 80 <i>m</i>	0,129	0,747	4,968	0,349	0,871	0,964	0,984
FCRN, px_{valid} , <i>berhu</i>	KITTI Continuous (denso)	0 – 80 <i>m</i>	0,140	0,911	5,313	0,247	0,831	0,931	0,965
FCRN, px_{valid} , <i>berhu</i>	KITTI Continuous (denso)	0 – 80 <i>m</i>	0,140	0,909	5,303	0,247	0,831	0,931	0,965
FCRN, px_{valid} , <i>berhu</i>	KITTI Continuous (denso)	0 – 80 <i>m</i>	0,140	0,914	5,332	0,248	0,830	0,930	0,965
FCRN, px_{valid} , <i>berhu</i>	KITTI Continuous (denso)	0 – 80 <i>m</i>	0,141	0,917	5,305	0,248	0,831	0,930	0,965
FCRN, px_{valid} , <i>berhu</i>	KITTI Continuous (denso)	0 – 80 <i>m</i>	0,141	0,920	5,326	0,248	0,829	0,930	0,964
FCRN, px_{valid} , <i>berhu</i>	KITTI Continuous (denso)	0 – 80 <i>m</i>	0,142	0,916	5,315	0,248	0,828	0,930	0,964
FCRN, px_{valid} , <i>eigen_{grabs}</i>	KITTI Continuous (denso)	0 – 80 <i>m</i>	0,143	0,909	5,760	0,234	0,821	0,950	0,980
FCRN, px_{valid} , <i>eigen</i>	KITTI Continuous (denso)	0 – 80 <i>m</i>	0,145	0,917	5,801	0,229	0,816	0,947	0,980
FCRN, px_{valid} , <i>mse</i>	KITTI Continuous (denso)	0 – 80 <i>m</i>	0,145	0,909	5,289	0,244	0,816	0,932	0,969
FCRN, px_{valid} , <i>mse</i>	KITTI Continuous (denso)	0 – 80 <i>m</i>	0,145	0,914	5,299	0,245	0,816	0,932	0,969
FCRN, px_{all} , <i>berhu</i>	KITTI Continuous (denso)	0 – 80 <i>m</i>	0,146	1,038	5,815	0,423	0,819	0,925	0,962
FCRN, px_{all} , <i>eigen</i>	KITTI Continuous (denso)	0 – 80 <i>m</i>	0,150	1,129	6,598	0,367	0,809	0,941	0,973
FCRN, px_{all} , <i>mse</i>	KITTI Continuous (denso)	0 – 80 <i>m</i>	0,150	0,976	5,509	0,309	0,806	0,929	0,967
FCRN, px_{all} , <i>eigen_{grabs}</i>	KITTI Continuous (denso)	0 – 80 <i>m</i>	0,152	1,152	6,641	0,375	0,800	0,939	0,973
FCRN, px_{valid} , <i>eigen_{grabs}</i>	KITTI Continuous (denso)	0 – 80 <i>m</i>	0,161	1,129	6,215	0,268	0,766	0,916	0,962
FCRN, px_{valid} , <i>eigen</i>	KITTI Continuous (denso)	0 – 80 <i>m</i>	0,163	1,137	6,265	0,268	0,762	0,914	0,962
FCRN, px_{all} , <i>eigen</i>	KITTI Continuous (denso)	0 – 80 <i>m</i>	0,168	1,320	6,888	0,331	0,753	0,906	0,956
FCRN, px_{all} , <i>eigen_{grabs}</i>	KITTI Continuous (denso)	0 – 80 <i>m</i>	0,170	1,337	6,927	0,338	0,745	0,904	0,955
FCRN, px_{valid}, <i>berhu</i>	KITTI Depth (semi-denso)	0 – 80 <i>m</i>	0,619	6,268	12,448	1,199	0,004	0,075	0,246
FCRN, px_{valid} , <i>berhu</i>	KITTI Depth (semi-denso)	0 – 80 <i>m</i>	0,623	6,324	12,485	1,217	0,004	0,075	0,241
FCRN, px_{valid} , <i>mse</i>	KITTI Depth (semi-denso)	0 – 80 <i>m</i>	0,624	6,232	12,313	1,163	0,003	0,053	0,204
FCRN, px_{valid} , <i>mse</i>	KITTI Depth (semi-denso)	0 – 80 <i>m</i>	0,625	6,237	12,312	1,166	0,003	0,052	0,202
FCRN, px_{all} , <i>mse</i>	KITTI Depth (semi-denso)	0 – 80 <i>m</i>	0,664	7,028	13,012	1,483	0,002	0,047	0,178
FCRN, px_{valid} , <i>mse</i>	KITTI Depth (semi-denso)	0 – 80 <i>m</i>	0,666	7,117	13,223	1,339	0,003	0,043	0,157
FCRN, px_{valid} , <i>mse</i>	KITTI Depth (semi-denso)	0 – 80 <i>m</i>	0,667	7,125	13,219	1,345	0,003	0,043	0,156
FCRN, px_{all} , <i>berhu</i>	KITTI Depth (semi-denso)	0 – 80 <i>m</i>	0,667	7,340	13,486	2,585	0,004	0,066	0,218
FCRN, px_{valid} , <i>eigen_{grabs}</i>	KITTI Depth (semi-denso)	0 – 80 <i>m</i>	0,668	7,319	13,595	1,289	0,001	0,028	0,128
FCRN, px_{valid} , <i>eigen</i>	KITTI Depth (semi-denso)	0 – 80 <i>m</i>	0,670	7,374	13,664	1,292	0,001	0,031	0,126
FCRN, px_{valid} , <i>berhu</i>	KITTI Depth (semi-denso)	0 – 80 <i>m</i>	0,671	7,296	13,415	1,442	0,003	0,061	0,188
FCRN, px_{valid} , <i>berhu</i>	KITTI Depth (semi-denso)	0 – 80 <i>m</i>	0,674	7,339	13,436	1,458	0,003	0,061	0,185
FCRN, px_{valid} , <i>eigen</i>	KITTI Depth (semi-denso)	0 – 80 <i>m</i>	0,707	8,117	14,285	1,471	0,001	0,026	0,101
FCRN, px_{valid} , <i>eigen_{grabs}</i>	KITTI Depth (semi-denso)	0 – 80 <i>m</i>	0,708	8,108	14,251	1,505	0,001	0,024	0,101
FCRN, px_{all} , <i>mse</i>	KITTI Depth (semi-denso)	0 – 80 <i>m</i>	0,720	8,232	14,114	1,946	0,002	0,038	0,135
FCRN, px_{all} , <i>eigen_{grabs}</i>	KITTI Depth (semi-denso)	0 – 80 <i>m</i>	0,722	8,624	14,791	1,715	0,001	0,022	0,101
FCRN, px_{all} , <i>eigen</i>	KITTI Depth (semi-denso)	0 – 80 <i>m</i>	0,726	8,698	14,864	1,745	0,001	0,022	0,098
FCRN, px_{all} , <i>berhu</i>	KITTI Depth (semi-denso)	0 – 80 <i>m</i>	0,734	8,686	14,557	3,862	0,003	0,054	0,166
FCRN, px_{all} , <i>eigen_{grabs}</i>	KITTI Depth (semi-denso)	0 – 80 <i>m</i>	0,772	9,653	15,541	2,216	0,001	0,018	0,078
FCRN, px_{all} , <i>eigen</i>	KITTI Depth (semi-denso)	0 – 80 <i>m</i>	0,774	9,692	15,581	2,287	0,001	0,018	0,077
FCRN, px_{valid}, <i>mse</i>	KITTI Discrete (esparso)	0 – 80 <i>m</i>	0,904	12,835	17,643	2,434	0,000	0,000	0,000
FCRN, px_{valid} , <i>berhu</i>	KITTI Discrete (esparso)	0 – 80 <i>m</i>	0,905	12,847	17,655	2,455	0,000	0,000	0,000
FCRN, px_{valid} , <i>mse</i>	KITTI Discrete (esparso)	0 – 80 <i>m</i>	0,907	12,873	17,628	2,474	0,000	0,000	0,000
FCRN, px_{valid} , <i>berhu</i>	KITTI Discrete (esparso)	0 – 80 <i>m</i>	0,908	12,890	17,638	2,519	0,000	0,000	0,000
FCRN, px_{valid} , <i>eigen</i>	KITTI Discrete (esparso)	0 – 80 <i>m</i>	0,909	13,050	17,852	2,475	0,000	0,000	0,000
FCRN, px_{valid} , <i>eigen</i>	KITTI Discrete (esparso)	0 – 80 <i>m</i>	0,909	13,034	17,796	2,491	0,000	0,000	0,000
FCRN, px_{valid} , <i>eigen_{grabs}</i>	KITTI Discrete (esparso)	0 – 80 <i>m</i>	0,911	13,057	17,829	2,524	0,000	0,000	0,000
FCRN, px_{valid} , <i>eigen_{grabs}</i>	KITTI Discrete (esparso)	0 – 80 <i>m</i>	0,915	13,129	17,817	2,699	0,000	0,000	0,000
FCRN, px_{all} , <i>mse</i>	KITTI Discrete (esparso)	0 – 80 <i>m</i>	0,918	13,156	17,843	2,769	0,000	0,000	0,000
FCRN, px_{all} , <i>berhu</i>	KITTI Discrete (esparso)	0 – 80 <i>m</i>	0,919	13,183	17,878	3,226	0,000	0,000	0,000
FCRN, px_{all} , <i>mse</i>	KITTI Discrete (esparso)	0 – 80 <i>m</i>	0,922	13,223	17,845	2,969	0,000	0,000	0,000
FCRN, px_{all} , <i>berhu</i>	KITTI Discrete (esparso)	0 – 80 <i>m</i>	0,922	13,252	17,882	3,416	0,000	0,000	0,000
FCRN, px_{all} , <i>eigen_{grabs}</i>	KITTI Discrete (esparso)	0 – 80 <i>m</i>	0,926	13,476	18,130	2,845	0,000	0,000	0,000
FCRN, px_{all} , <i>eigen</i>	KITTI Discrete (esparso)	0 – 80 <i>m</i>	0,927	13,526	18,177	2,847	0,000	0,000	0,000
FCRN, px_{all} , <i>eigen_{grabs}</i>	KITTI Discrete (esparso)	0 – 80 <i>m</i>	0,928	13,512	18,109	3,038	0,000	0,000	0,000
FCRN, px_{all} , <i>eigen</i>	KITTI Discrete (esparso)	0 – 80 <i>m</i>	0,929	13,541	18,139	2,989	0,000	0,000	0,000

Fonte: Elaborada pelo autor.

de mapas de Hilbert pode melhorar a qualidade das predições quando comparado a versões de linha de base (sem HM). Esta mesma técnica pode ser aplicada a outras arquiteturas de rede para produzir ganhos de desempenho semelhantes, o que é capaz de impulsionar ainda mais o desempenho de modelos mais atuais no estado da arte. Além disso, os resultados obtidos demonstram o quão prejudicial é treinar redes convolucionais a partir de dados esparsos, isso ficou claramente explicitado ao submetê-las a *benchmarks* que utilizam imagens semi-densas (mais pontos de avaliação).

Figura 35: Comparação qualitativa entre estimativas de profundidade quando treinadas nos conjuntos de dados propostos. (a) imagem RGB de entrada. (b) *KITTI Discrete* (Esparsos, 1 Varredura). (c) *KITTI Depth* (Semi-Denso, 11 Varreduras). (d) *KITTI Continuous* (Denso, Mapas de ocupação contínuos). (e) Imagem de profundidade de referência (Contínuo, *Ground Truth*).

Fonte: Elaborada pelo autor.

A Figura 35 ilustra a comparação qualitativa entre as predições ao treinar a rede nos conjuntos de dados propostos. Como pode ser observado, as imagens contínuas aumentaram a qualidade das estimativas de distância. Em outras palavras, elas tornaram as imagens estimadas muito menos borradas, ou seja, elas têm uma definição melhor das bordas dos objetos, além de terem medições mais precisas de acordo com os mapas de referência. A principal causa das predições de conjuntos de dados esparsos serem borradas é o uso de filtros convolucionais 2D em regiões amplamente esparsas e a ocasionalidade de informações de profundidade, já que o valor da distância em um dado píxel é intermitente e isso depende de onde os pontos do *laser* serão reprojados.

Uma comparação qualitativa entre resultados obtidos e alguns dos métodos no estado da arte está apresentada na Figura 36. Como DORN (FU et al., 2018), o método proposto (FCRN + *KITTI Continuous*) também detecta bem obstáculos presentes nas cenas, com a notável diferença de que as predições da rede apresentam uma certa margem de segurança em torno dos obstáculos, devido às propriedades reconstrutivas do *framework* de mapas de Hilbert, como mostrado na seção 4.1, além de obter desempenho semelhante usando uma arquitetura mais simples.

Figura 36: Comparação qualitativa entre as estimativas do método proposto (FCRN + *KITTI Continuous*) e outros trabalhos no estado da arte.

Fonte: Imagem adaptada de Fu et al. (2018).

6.6 Predição em tempo real

Uma vez treinados, os modelos podem ser empregados em um seleto nicho de aplicações, os quais são submetidos a realizar inferências em tempo real de cenas genéricas. Deste modo, o modelo desenvolvido pode ser incorporado a sistemas ainda mais complexos de percepção, estes que não necessariamente restringem-se a percepção de profundidade unicamente, mas também de pessoas, sinalização de trânsito, vias navegáveis, dentre outros. Em outras palavras, ele é responsável por processar as imagens RGB de entrada, estimando os valores de distâncias para cada uma destas cenas, desde que os quadros do vídeo ou filmagem estejam dentro do contexto no qual a rede foi treinada (*indoor*, *outdoor*, etc.).

Na Figura 37, apresenta-se alguns exemplos de predições da rede quando submetida a cenas genéricas. Em uma análise visual, pode-se observar que a rede, mesmo sendo treinada usando o *dataset* KITTI (Alemanha), foi capaz de generalizar cenas de trânsitos em diferentes localidades do mundo, como Londres, Nova Iorque e Dubai. É válido enfatizar que estas cenas foram capturadas por câmeras diferentes das utilizadas pelo *dataset*, logo, apresentam diferentes parâmetros intrínsecos, tais como: distância focal, ponto principal, distorção das lentes, etc. Devido a esta última informação, as estimativas de distância precisam ser corrigidas por um fator de escala para que coincidam com a escala real (metros). Os comentários da Figura 37 valem para a Figura 38.

Uma vez que estas plataformas são sistemas críticos e possuem os equipamentos necessários para a geração de um *dataset*, sugere-se capturar as cenas de treinamento e teste nas mesmas condições nas quais serão submetidas durante a etapa de inferência em tempo real. Isto dispensa a necessidade de estimar fatores de correção para o ajuste de escala das predições da rede devido a diferenças nos comprimentos focais das câmeras usadas durante o treinamento e das câmeras usadas para a inferência. Além disso, uma grande falha de segurança é confiar integralmente no poder de generalização de uma rede treinada em outros cenários (as imagens do KITTI, por exemplo, são de estradas da Alemanha) para estimar cenas de outros países. Em outras palavras, é recomendado construir *datasets*

Figura 37: Predições da rede (FCRN + *KITTI Continuous*) para cenas genéricas em diferentes cidades ao redor do mundo. Todas as cenas foram retiradas da plataforma de compartilhamento de vídeos *Youtube*. **Da esquerda para direita:** Imagem RGB de entrada, Predição (Jet), Predição (HSV), Sobreposição.

(a) Londres (MUROKI, 2016)

(b) Nova Iorque (J UTAH, 2016)

(c) Dubai (VEDATALI, 2013)

Fonte: Elaborada pelo autor.

Figura 38: Predições da rede (FCRN + *KITTI Continuous*) para cenas colhidas pelas plataformas robóticas desenvolvidas do Laboratório de Robótica Móvel (LRM) do ICMC/USP. **Da esquerda para direita:** Imagem RGB de entrada, Predição (Jet), Predição (HSV), Sobreposição.

(a) Caminhão autônomo

(b) Carro autônomo – CaRINA

Fonte: Elaborada pelo autor.

exclusivos para cada uma das plataformas, assim como foi feito originalmente no KITTI.

Especificamente em aplicações robóticas, a incorporação deste tipo de técnica tem como exigência principal o tempo de resposta que se pode realizar as predições de distância, cujo requisito mínimo varia de plataforma para plataforma (carros autônomos, VANTs, humanoides, etc.), o qual é comumente relacionado ao comportamento dinâmico do sistema. Mais precisamente, este valor dependerá do poder computacional disponível, cuja especificação está estreitamente relacionada a autonomia e a restrições físicas da plataforma. Neste trabalho, o tempo de inferência da arquitetura de rede proposta, no *hardware* utilizado, foi de 25 *ms*, o que representa uma taxa de atualização de aproximadamente 40 FPS.

Apesar das dificuldades técnicas, as redes desenvolvidas estreitam a distância entre o tema abordado e aplicações reais, uma vez que suas predições podem ser utilizadas na reconstrução de mapas tridimensionais, cumprindo o papel de mapeamento no problema computacional de localização e mapeamento simultâneo (SLAM), como já demonstrado em (LAINA et al., 2016; TATENO et al., 2017).

6.6.1 Resultados do *framework* desenvolvido em ROS

Como descrito na seção 5.6, caso os parâmetros intrínsecos da câmera sejam conhecidos, é possível gerar a nuvem de pontos dos mapas de profundidade estimados pela rede. Para isso, foi desenvolvido um *framework* em ROS para integrar a rede proposta e os módulos de geração e visualização da nuvem estimada. A Figura 39 demonstra os resultados obtidos no *framework* em questão, mais precisamente, apresenta as visualizações das nuvens de pontos de dois quadros diferentes de uma sequência disponibilizada pelo KITTI.

Devido ao processo de projeção da nuvem a um plano 2D, o campo de visão observado nas imagens de profundidade usadas como referência é restringido ao campo visual da câmera RGB, logo a nuvem estimada (nuvem de pontos em vermelho) também está restrita a este campo. Além disso, é possível observar que os objetos delineados pela nuvem de pontos do LiDAR (nuvem de pontos em branco), também foram percebidos, de forma suavizada, pela nuvem em vermelho, como pode ser visualizado nas deformidades próximas aos carros da segunda cena. Visualmente, a escala de predição da rede parece coincidir com a escala da nuvem real. Neste caso em específico, as cenas testadas também são do KITTI e foram coletadas com os mesmos equipamentos e nas mesmas condições das cenas utilizadas para o treinamento da rede. Desta forma, os valores de distâncias estimados poderiam ser considerados como absolutos na escala de metros, sendo este o principal fator para que as nuvens terem se alinhado. Assim, qualquer erro destas estimativas estaria diretamente relacionado ao poder preditivo da rede.

Por outro lado, se as cenas de teste fossem genéricas e que não se conhecesse os parâmetros de calibração, os valores estimados deveriam ser utilizados apenas como medidas relativas de distância. Neste caso, qualquer erro destas estimativas poderia estar relacionado tanto ao poder preditivo da rede quanto à incerteza no fator de escala das predições. Algumas formas de como estimar este fator de escala foram apresentadas ao fim da Capítulo 3.

Figura 39: *RViz* – Visualização das projeções 3D das previsões da rede. **Topo à esquerda:** Imagem RGB, Entrada da Rede. **Topo à direita:** Imagem de Profundidade, Predição da rede. **Branco:** Nuvem de pontos do LiDAR. **Vermelho:** Nuvem de pontos predita pela rede. Cenas retiradas da *ROSBag* “kitti_2011_09_26_drive_0001_synced.bag”, cuja sequência de quadros foi originalmente disponibilizada pelo KITTI.

Fonte: Elaborada pelo autor.

7 CONCLUSÃO

Este trabalho aborda o tema “Percepção visual de profundidade”. Assim como os sistemas visuais de humanos/animais permitem que se orientem e percebam objetos e obstáculos dentro de um espaço tridimensional, deseja-se reproduzir esta habilidade em plataformas robóticas mediante métodos computacionais. Biologicamente, isto é realizado por diferentes pistas visuais (paralaxe de movimento, tamanho relativo, acomodação, etc.). Na literatura, diversos métodos computacionais, estéreos e monoculares, foram sugeridos para a extração deste tipo de informação. No caso dos métodos estéreo, a profundidade pode ser obtida por meio de diferenças nas disparidade existentes entre duas, ou mais, perspectivas. No caso dos métodos monoculares, as primeiras abordagens apresentavam forte embasamento matemático e exploravam extrair esta informação por meio de relações geométricas, enquanto que abordagens mais recentes, recorrem a métodos de aprendizado de máquina.

O interesse no problema de *Estimação de Profundidade de Imagens Monoculares* (*Single Image Depth Estimation*) é decorrente a algumas limitações existentes nos métodos estéreo. Dentre elas podem ser citadas: a obrigatoriedade do processo de pré-calibração do sistema visual, as falhas de correspondência em superfícies refletoras ou de baixa textura, e o elevado custo computacional. Além disso, algumas das aplicações requerem que a linha de base seja suficientemente grande para que as estimativas em distância cubram o intervalo de medição desejado, o que pode ser impraticável em sistemas de pequeno porte, como por exemplo em *smartphones*. Dentre os métodos monoculares, métodos clássicos modelam o problema utilizando MRFs e CRFs, enquanto que abordagens mais atuais exploram o potencial de redes neurais profundas (*Deep Learning*). No caso, as redes desenvolvidas podem aprender diretamente dos mapas de profundidade da cena (Supervisionado), ou a partir da geração de novas perspectivas e da informação de movimento presente entre sequências de quadros (Não-supervisionado).

De modo geral, a maior parte dos trabalhos é voltada ao desenvolvimento da técnica de otimização (diferentes algoritmos de aprendizado, topologia de redes), e pouco preocupava-se com o elevado nível de esparsidade presente nos dados de treinamento. Dessa forma, neste trabalho foi apresentada uma rede convolucional profunda treinada, de forma supervisionada, por imagens contínuas de profundidade oriundas de projeções dos mapas de Hilbert, ou seja, sugeriu-se utilizar o modelo de ocupação como etapa de pré-processamento para a tarefa SIDE. Em resumo, neste tipo de tarefa as técnicas de otimização precisam ser capazes de aprender as relações existentes entre as imagens monoculares e seus mapas correspondentes, permitindo que seja possível realizar inferências quanto à profundidade a partir dos valores de intensidade dos píxeis. Estes modelos de ocupação geram mapas

contínuos de profundidade para o treinamento de uma rede residual (FCRN), diferindo de típicas abordagens supervisionadas que usam mapas esparsos. Mais especificamente, avaliou-se seu desempenho quando treinada em três *datasets* que possuíam diferentes níveis de esparsidade nos mapas de profundidade das sequências externas (*outdoor*) disponibilizadas pelo KITTI. Este trabalho é o primeiro a alavancar uma ferramenta de reconstrução 3D como técnica de densificação para melhorar as estimativas de profundidade. Além disso, a ideia proposta não se restringe unicamente a esta arquitetura de rede e técnica de densificação, e não requer nenhum outro tipo de sensor ou informação auxiliar, apenas imagens RGB como entrada e mapas de profundidade contínuos como supervisão, o que melhorou significativamente a qualidade das predições de rede em relação a típicos mapas esparsos. Além disso, a metodologia proposta apresentou desempenho superior mesmo usando 67,6% menos exemplos do que aqueles treinados no conjunto de dados *KITTI Depth*, o que é consequência do aumento de 15,2% para 62,6% da informação válida presente nos mapas de referência.

As principais limitações do método proposto para a etapa de pré-processamento são o custo computacional necessário para calcular cada mapa contínuo de profundidade usado no treinamento, além de ser limitado pelas capacidades reconstruídas do *framework* de mapas de Hilbert. No caso da rede convolucional, as principais limitações são em relação ao número elevado de exemplos para treinamento e o problema quanto à recuperação da escala absoluta dos mapas estimados. Como trabalho futuro, sugere-se focar na otimização do método em si, abordando principalmente as limitações mencionadas acima, e aprimorando a topologia da rede ao incorporar novos elementos em sua arquitetura que sejam mais adequados à tarefa SIDE. Também pode-se realizar futuras investigações quanto a outras variantes (somente as que transformam diretamente profundidade para profundidade) de técnicas de densificação (*in-painting*) (DORIA; RADKE, 2012; GONG et al., 2013; CHEN et al., 2014; LU; FORSYTH, 2015; UHRIG et al., 2017) para o passo de pré-processamento proposto, possibilitando, assim, melhor avaliar como os mapas de Hilbert comparativamente desempenham à elas.

Como já mencionado, redes convolucionais tem demonstrado resultados expressivos em várias aplicações em visão computacional. Entretanto, pouco se sabe sobre as representações extraídas por elas. Quais seriam as características mais relevantes para a rede e por que tem apresentado tal desempenho? Estas foram algumas das questões abordadas por Zeiler e Fergus (2014) ao analisarem de forma mais profunda as características extraídas de uma rede convolucional para segmentação semântica. Similarmente, como extensão deste trabalho, pode-se investigar as características extraídas por modelos que abordam a tarefa SIDE.

Por fim, este trabalho atingiu três grandes metas. A primeira era desenvolver um amplo *framework* capaz de permitir a realização do treinamento, de testes e de predições em

tempo real. A segunda era avaliar o desempenho da rede proposta ao treiná-la utilizando mapas contínuos como supervisão. E a última era garantir que as técnicas implementadas estivessem de acordo com as métricas e desempenho comparável aos métodos presentes na literatura.

REFERÊNCIAS

- ABADI, M.; BARHAM, P.; CHEN, J.; CHEN, Z.; DAVIS, A.; DEAN, J.; DEVIN, M.; GHEMAWAT, S.; IRVING, G.; ISARD, M. et al. TensorFlow: A System for Large-Scale Machine Learning. In: **Proc. of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI)**. [S.l.: s.n.], 2016. v. 16, p. 265–283.
- ALEOTTI, F.; TOSI, F.; POGGI, M.; MATTOCCIA, S. Generative adversarial networks for unsupervised monocular depth prediction. In: SPRINGER. **Proc. of the European Conference on Computer Vision (ECCV)**. [S.l.], 2018. p. 337–354.
- APPLE. **Performing Convolution Operations**. 2016. Disponível em: <<https://developer.apple.com/library/content/documentation/Performance/Conceptual/vImage/ConvolutionOperations/ConvolutionOperations.html>>.
- BAJCSY, R.; LIEBERMAN, L. Texture gradient as a depth cue. **Computer Graphics and Image Processing**, Elsevier, v. 5, n. 1, p. 52–67, 1976.
- BENGIO, Y. Learning Deep Architectures for AI. **Foundations and Trends® in Machine Learning**, v. 2, n. 1, p. 1–127, 2009. ISSN 1935-8237. Disponível em: <<http://www.nowpublishers.com/article/Details/MAL-006>>.
- _____. Practical recommendations for gradient-based training of deep architectures. In: **Neural networks: Tricks of the trade**. [S.l.]: Springer, 2012. p. 437–478.
- BENGIO, Y.; COURVILLE, A.; VINCENT, P. Representation learning: A review and new perspectives. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 35, n. 8, p. 1798–1828, 2013. ISSN 01628828.
- BENGIO, Y.; SIMARD, P.; FRASCONI, P. Learning Long Term Dependencies with Gradient Descent is Difficult. **IEEE Transactions on Neural Networks**, v. 5, n. 2, p. 157–166, 1994. ISSN 1045-9227.
- BERGSTRA, J.; BENGIO, Y. Random search for hyper-parameter optimization. **Journal of Machine Learning Research**, v. 13, n. Feb, p. 281–305, 2012.
- BERGSTRA, J.; BREULEUX, O.; BASTIEN, F. F.; LAMBLIN, P.; PASCANU, R.; DESJARDINS, G.; TURIAN, J.; WARDE-FARLEY, D.; BENGIO, Y. Theano: a CPU and GPU math compiler in Python. **Proceedings of the Python for Scientific Computing Conference (SciPy)**, n. SciPy, p. 1–7, 2010. Disponível em: <http://www-etud.iro.umontreal.ca/~wardefar/publications/theano_scipy2010.pdf>.
- BISHOP, C. M. **Pattern Recognition and Machine Learning**. [s.n.], 2006. v. 4. 738 p. ISSN 10179909. ISBN 9780387310732. Disponível em: <<http://www.springer.com/gb/book/9780387310732>>.
- BRESILLA, K.; MANFRINI, L.; MORANDI, B.; BOINI, A.; PERULLI, G.; GRAPPADELLI, L. C. Using deep learning real-time object detection convolution neural networks for fast fruit recognition in the tree. **World Academy of Science, Engineering and Technology**, v. 12, n. 8, p. 1588, 2018. ISSN eISSN:1307-6892. Disponível em: <<http://waset.org/abstracts/Agricultural-and-Biosystems-Engineering>>.

BRUCE, N. D.; RAHMAN, S.; CARRIER, D. Sparse coding in early visual representation: From specific properties to general principles. **Neurocomputing**, v. 171, p. 1085–1098, 2016. ISSN 18728286.

CAO, Y.; WU, Z.; SHEN, C. Estimating depth from monocular images as classification using deep fully convolutional residual networks. **Proc. of the IEEE Transactions on Circuits and Systems for Video Technology**, IEEE, v. 28, n. 11, p. 3174–3182, 2018.

CARVALHO, M.; SAUX, B. L.; TROUVÉ-PELOUX, P.; ALMANSA, A.; CHAMPAGNAT, F. On regression losses for deep depth estimation. In: **2018 25th IEEE International Conference on Image Processing (ICIP)**. [S.l.: s.n.], 2018. p. 2915–2919. ISSN 2381-8549.

CHEN, C.; SEFF, A.; KORNHAUSER, A.; XIAO, J. DeepDriving: Learning affordance for direct perception in autonomous driving. In: **Proceedings of the IEEE International Conference on Computer Vision**. [S.l.: s.n.], 2015. v. 2015 Inter, p. 2722–2730. ISBN 9781467383912. ISSN 15505499.

CHEN, R.; MAHMOOD, F.; YUILLE, A.; DURR, N. J. Rethinking monocular depth estimation with adversarial training. **arXiv preprint arXiv:1808.07528**, 2018.

CHEN, T.; LI, M.; LI, Y.; LIN, M.; WANG, N.; WANG, M.; XIAO, T.; XU, B.; ZHANG, C.; ZHANG, Z. MXNet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems. **eprint arXiv:1512.01274**, p. 1–6, 2015. ISSN 03600300. Disponível em: <<http://arxiv.org/abs/1512.01274>>.

CHEN, W.; YUE, H.; WANG, J.; WU, X. An improved edge detection algorithm for depth map inpainting. **Optics and Lasers in Engineering**, v. 55, p. 69 – 77, 2014. ISSN 0143-8166.

CHEN, Z.; SUN, X.; WANG, L.; YU, Y.; HUANG, C. A Deep Visual Correspondence Embedding Model for Stereo Matching Costs. In: **Proc. of the 2015 IEEE International Conference on Computer Vision (ICCV)**. [S.l.: s.n.], 2015. p. 972–980. ISBN 978-1-4673-8391-2. ISSN 15505499.

CHOLLET, F. **Deep Learning with Python**. Manning Publications Company, 2017. ISBN 9781617294433. Disponível em: <<https://books.google.com.br/books?id=Y03CAQAACAAJ>>.

CHOLLET, F. et al. **Keras**. 2015.

CLOWES, M. B. On seeing things. **Artificial intelligence**, Elsevier, v. 2, n. 1, p. 79–116, 1971.

COLLOBERT, R.; KAVUKCUOGLU, K.; FARABET, C. Torch7: A matlab-like environment for machine learning. **BigLearn, NIPS Workshop**, p. 1–6, 2011. Disponível em: <http://infoscience.epfl.ch/record/192376/files/Collobert_NIPSWORKSHOP_2011.pdf>.

COON, D.; MITTERER, J. O. **Introduction to Psychology: Gateways to Mind and Behavior**. Cengage Learning, 2015. ISBN 9781305545007. Disponível em: <<https://books.google.com.br/books?id=-4jCBAAAQBAJ>>.

CORDTS, M.; OMRAN, M.; RAMOS, S.; REHFELD, T.; ENZWEILER, M.; BENENSON, R.; FRANKE, U.; ROTH, S.; SCHIELE, B. The Cityscapes Dataset for Semantic Urban Scene Understanding. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2016.

CUTLER, M.; HOW, J. P. Autonomous drifting using simulation-aided reinforcement learning. In: **2016 IEEE International Conference on Robotics and Automation (ICRA)**. [S.l.: s.n.], 2016. p. 5442–5448.

DIMICCOLI, M. Monocular depth estimation for image segmentation and filtering. **Dept. Signal Theory Commun., Ph. D. dissertation, Univ. Politecnica de Catalunya, Barcelona, Spain**, 2009.

DING, X.; ZHANG, Y.; LIU, T.; DUAN, J. Deep learning for event-driven stock prediction. In: **IJCAI International Joint Conference on Artificial Intelligence**. [S.l.: s.n.], 2015. v. 2015-Janua, p. 2327–2333. ISBN 9781577357384. ISSN 10450823.

DONG, C.; LOY, C. C.; HE, K.; TANG, X. Learning a deep convolutional network for image super-resolution. In: SPRINGER. **Proc. of the European conference on computer vision (ECCV)**. [S.l.], 2014. p. 184–199.

DORIA, D.; RADKE, R. J. Filling large holes in lidar data by inpainting depth gradients. In: IEEE. **Proc. of the 2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPR)**. [S.l.], 2012. p. 65–72.

DUCHI, J.; HAZAN, E.; SINGER, Y. Adaptive subgradient methods for online learning and stochastic optimization. **Journal of Machine Learning Research**, v. 12, n. Jul, p. 2121–2159, 2011.

DUMOULIN, V.; VISIN, F. A guide to convolution arithmetic for deep learning. **Arxiv**, p. 1–31, 2016. ISSN 16113349. Disponível em: <<http://arxiv.org/abs/1603.07285>>.

EIGEN, D.; FERGUS, R. Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-scale Convolutional Architecture. In: **Proc. of the 2015 IEEE International Conference on Computer Vision (ICCV)**. [S.l.: s.n.], 2015. p. 2650–2658. ISBN 978-1-4673-8391-2. ISSN 15505499.

EIGEN, D.; PUHRSCHE, C.; FERGUS, R. Depth map prediction from a single image using a multi-scale deep network. In: **Proc. of the 27th International Conference on Neural Information Processing Systems (NIPS)**. [S.l.: s.n.], 2014. p. 2366–2374.

FALK, G. Interpretation of imperfect line data as a three-dimensional scene. **Artificial intelligence**, Elsevier, v. 3, p. 101–144, 1972.

FELDMAN, J.; ROJAS, R. **Neural Networks: A Systematic Introduction**. Springer Berlin Heidelberg, 2013. ISBN 9783642610684. Disponível em: <<https://books.google.com.br/books?id=4rESBwAAQBAJ>>.

FLYNN, J.; NEULANDER, I.; PHILBIN, J.; SNAVELY, N. Deep Stereo: Learning to Predict New Views from the World's Imagery. In: **2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. IEEE, 2016. p. 5515–5524. ISBN 978-1-4673-8851-1. ISSN 10636919. Disponível em: <<http://ieeexplore.ieee.org/document/7780964/>>.

FORSTER, C. H. Q.; TOZZI, C. L. Towards 3d reconstruction of endoscope images using shape from shading. In: **Proceedings 13th Brazilian Symposium on Computer Graphics and Image Processing (Cat. No.PR00878)**. [S.l.: s.n.], 2000. p. 90–96. ISSN 1530-1834.

FRID-ADAR, M.; KLANG, E.; AMITAI, M.; GOLDBERGER, J.; GREENSPAN, H. Synthetic Data Augmentation using GAN for Improved Liver Lesion Classification. **arXiv preprint arXiv:1801.02385**, 2018.

FRITSCH, J.; KUEHNLE, T.; GEIGER, A. A new performance measure and evaluation benchmark for road detection algorithms. In: **International Conference on Intelligent Transportation Systems (ITSC)**. [S.l.: s.n.], 2013.

FROSSARD, D. **VGG in TensorFlow - Model and pre-trained parameters for VGG16 in TensorFlow**. 2016. Disponível em: <<http://www.cs.toronto.edu/~frossard/post/vgg16/>>.

FU, H.; GONG, M.; WANG, C.; BATMANGHELICH, K.; TAO, D. Deep Ordinal Regression Network for Monocular Depth Estimation. In: **Proc. of the 2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2018.

FUKUSHIMA, K. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. **Biological Cybernetics**, v. 36, n. 4, p. 193–202, 1980. ISSN 03401200.

GAIDON, A.; WANG, Q.; CABON, Y.; VIG, E. Virtual Worlds as Proxy for Multi-object Tracking Analysis. In: **Proc. of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2016. p. 4340–4349. ISBN 978-1-4673-8851-1. ISSN 10636919.

GARG, R.; Vijay Kumar, B. G.; CARNEIRO, G.; REID, I. Unsupervised CNN for single view depth estimation: Geometry to the rescue. In: **Proc. of the European Conference on Computer Vision (ECCV)**. [S.l.: s.n.], 2016. p. 740–756. ISBN 9783319464831. ISSN 16113349.

GEIGER, A.; LENZ, P.; STILLER, C.; URTASUN, R. Vision meets robotics: The kitti dataset. **The International Journal of Robotics Research (IJRR)**, Sage Publications Sage UK: London, England, v. 32, n. 11, p. 1231–1237, 2013.

GEIGER, A.; LENZ, P.; URTASUN, R. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In: **Proc. of the Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2012.

GLOROT, X.; BENGIO, Y. Understanding the difficulty of training deep feedforward neural networks. In: **Proceedings of the thirteenth international conference on artificial intelligence and statistics**. [S.l.: s.n.], 2010. p. 249–256.

GLOROT, X.; BORDES, A.; BENGIO, Y. Deep sparse rectifier neural networks. **AISTATS '11: Proceedings of the 14th International Conference on Artificial Intelligence and Statistics**, v. 15, p. 315–323, 2011. ISSN 15324435.

GODARD, C.; AODHA, O. M.; BROSTOW, G. J. Unsupervised Monocular Depth Estimation with Left-Right Consistency. In: **Proc. of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2017. p. 6602–6611. ISBN 978-1-5386-0457-1.

GOLDSTEIN, E. B. **Sensation and Perception**. Cengage Learning, 2009. ISBN 9780495601494. Disponível em: <<https://books.google.com.br/books?id=2tW91BWeNq4C>>.

GONG, X.; LIU, J.; ZHOU, W.; LIU, J. Guided depth enhancement via a fast marching method. **Image and Vision Computing**, v. 31, n. 10, p. 695–703, 2013. ISSN 0262-8856.

GONZALEZ, R.; WOODS, R. **Digital image processing**. [s.n.], 2002. 190 p. ISSN 0024094X. ISBN 0201180758. Disponível em: <http://web.ipac.caltech.edu/staff/fmasci/home/astro_refs/Digital_Image_Processing_3rdEd_truncated.pdf>.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. Deep Learning. **Nature Methods**, v. 13, n. 1, p. 35–35, 2015. ISSN 1548-7091.

_____. **Deep Learning**. [S.l.]: MIT Press, 2016. <<http://www.deeplearningbook.org>>.

GOODFELLOW, I. J.; WARDE-FARLEY, D.; LAMBLIN, P.; DUMOULIN, V.; MIRZA, M.; PASCANU, R.; BERGSTRA, J.; BASTIEN, F.; BENGIO, Y. Pylearn2: a machine learning research library. p. 1–9, 2013. Disponível em: <<http://arxiv.org/abs/1308.4214>>.

GRAPE, G. R. **Model based (intermediate-level) computer vision**. [S.l.], 1973.

GRIGOREV, A.; JIANG, F.; RHO, S.; SORI, W. J.; LIU, S.; SAI, S. Depth estimation from single monocular images using deep hybrid network. **Multimedia Tools and Applications**, Multimedia Tools and Applications, v. 76, n. 18, p. 18585–18604, sep 2017. ISSN 1380-7501. Disponível em: <<http://link.springer.com/10.1007/s11042-016-4200-x>>.

GUIZILINI, V.; RAMOS, F. Large-scale 3D scene reconstruction with Hilbert Maps. **IEEE International Conference on Intelligent Robots and Systems (IROS)**, v. 2016-November, n. October, p. 3247–3254, 2016. ISSN 21530866.

_____. Learning to reconstruct 3d structures for occupancy mapping. In: **Proceedings of Robotics: Science and Systems (RSS)**. [S.l.: s.n.], 2017.

_____. Fast 3d modeling with approximated convolutional kernels. In: BILLARD, A.; DRAGAN, A.; PETERS, J.; MORIMOTO, J. (Ed.). **Proceedings of The 2nd Conference on Robot Learning**. PMLR, 2018. (Proceedings of Machine Learning Research, v. 87), p. 190–199. Disponível em: <<http://proceedings.mlr.press/v87/guizilini18a.html>>.

_____. Fast 3d modeling with approximated convolutional kernels. In: BILLARD, A.; DRAGAN, A.; PETERS, J.; MORIMOTO, J. (Ed.). **Proceedings of The 2nd Conference on Robot Learning**. PMLR, 2018. (Proceedings of Machine Learning Research, v. 87), p. 190–199. Disponível em: <<http://proceedings.mlr.press/v87/guizilini18a.html>>.

_____. Towards real-time 3d continuous occupancy mapping using hilbert maps. **International Journal of Robotics Research (IJRR)**, v. 37, n. 6, p. 566–584, 2018.

GUO, Y.; LIU, Y.; OERLEMANS, A.; LAO, S.; WU, S.; LEW, M. S. Deep learning for visual understanding: A review. **Neurocomputing**, v. 187, p. 27–48, 2016. ISSN 18728286.

GUZMAN-ARENAS, A. **Computer recognition of three-dimensional objects in a visual scene**. [S.l.], 1968.

HAHNIOSE, R. H.; SARPESHKAR, R.; MAHOWALD, M. A.; DOUGLAS, R. J.; SEUNG, H. S. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. **Nature**, v. 405, n. 6789, p. 947–951, 2000. ISSN 00280836.

HE, K.; ZHANG, X.; REN, S.; SUN, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: **Proceedings of the IEEE international conference on computer vision**. [S.l.: s.n.], 2015. p. 1026–1034.

_____. Deep Residual Learning for Image Recognition. In: **2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [s.n.], 2016. p. 770–778. ISBN 978-1-4673-8851-1. ISSN 1664-1078. Disponível em: <<http://ieeexplore.ieee.org/document/7780459/>>.

HEDAU, V.; HOIEM, D.; FORSYTH, D. Thinking inside the box: Using appearance models and context based on room geometry. **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**, v. 6316 LNCS, n. PART 6, p. 224–237, 2010. ISSN 03029743.

HELD, D.; THRUN, S.; SAVARESE, S. Learning to Track at 100 {FPS} with Deep Regression Networks. **CoRR**, abs/1604.0, 2016. Disponível em: <<http://arxiv.org/abs/1604.01802>>.

HINTON, G. E.; OSINDERO, S.; TEH, Y.-W. A fast learning algorithm for deep belief nets. **Neural computation**, MIT Press, v. 18, n. 7, p. 1527–1554, 2006.

HODGKIN, A. L.; HUXLEY, A. F. A quantitative description of membrane current and its application to conduction and excitation in nerve. **The Journal of Physiology**, v. 117, n. 4, p. 500–544, 1952. ISSN 1469-7793. Disponível em: <<http://dx.doi.org/10.1113/jphysiol.1952.sp004764>>.

HOIEM, D.; EFROS, A. A.; HEBERT, M. Automatic photo pop-up. **ACM Transactions on Graphics**, v. 24, n. 3, p. 577, 2005. ISSN 07300301.

HOWARD, I. P.; ROGERS, B. J. **Binocular vision and stereopsis**. [S.l.]: Oxford University Press, USA, 1995.

_____. **Perceiving in Depth, Volume 3: Other Mechanisms of Depth Perception**. Oxford University Press, USA, 2012. (Oxford Psychology Series). ISBN 9780199764167. Disponível em: <https://books.google.com.br/books?id=SFgfgFrdB_oC>.

HUA, J.; GONG, X. A normalized convolutional neural network for guided sparse depth upsampling. In: **Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)**. [S.l.: s.n.], 2018. p. 2283–2290.

HUANG, X.; CHENG, X.; GENG, Q.; CAO, B.; ZHOU, D.; WANG, P.; LIN, Y.; YANG, R. The ApolloScape Dataset for Autonomous Driving. **arXiv preprint arXiv:1803.06184**, 2018. Disponível em: <<http://arxiv.org/abs/1803.06184>>.

HUFFMAN, D. A. Impossible object as nonsense sentences. **Machine intelligence**, Edinburgh Univ. Press, v. 6, p. 295–324, 1971.

Indra den Bakker. **Battle of the Deep Learning frameworks — Part I: 2017, even more frameworks and interfaces**. 2017. Disponível em: <<https://towardsdatascience.com/battle-of-the-deep-learning-frameworks-part-i-cff0e3841750>>.

IOFFE, S.; SZEGEDY, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. **arXiv preprint arXiv:1502.03167**, 2015.

J UTAH. **Driving Downtown - Americas Avenue 4K - New York City USA**. 2016. <<https://www.youtube.com/watch?v=6uu9EwoWxo4&t=1054s>>.

JAMIY, F. E.; MARSH, R. Survey on depth perception in head mounted displays: distance estimation in virtual reality, augmented reality, and mixed reality. **IET Image Processing**, IET, v. 13, n. 5, p. 707–712, 2019.

JIA, Y.; SHELHAMER, E.; DONAHUE, J.; KARAYEV, S.; LONG, J.; GIRSHICK, R.; GUADARRAMA, S.; DARRELL, T. Caffe. **Proceedings of the ACM International Conference on Multimedia - MM '14**, p. 675–678, 2014. ISSN 9781450330633. Disponível em: <<http://dl.acm.org/citation.cfm?doid=2647868.2654889>>.

KING, I. **Elon Musk's Chipmaking Claims Don't Match the Reality**. Bloomberg, 2019. Disponível em: <<https://www.bloomberg.com/news/articles/2019-04-23/musk-s-chipmaking-boasts-clash-with-autonomous-car-reality>>.

KINGMA, D. P.; BA, J. Adam: A Method for Stochastic Optimization. p. 1–15, 2014. ISSN 09252312. Disponível em: <<http://arxiv.org/abs/1412.6980>>.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. ImageNet Classification with Deep Convolutional Neural Networks. **Advances In Neural Information Processing Systems**, p. 1–9, 2012. ISSN 10495258.

KROGH, A.; HERTZ, J. A. A simple weight decay can improve generalization. In: **Advances in neural information processing systems**. [S.l.: s.n.], 1992. p. 950–957.

KUMAR, A. C.; BHANDARKAR, S. M.; PRASAD, M. Depthnet: A recurrent neural network architecture for monocular depth prediction. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops**. [S.l.: s.n.], 2018. p. 283–291.

KUZNIETSOV, Y.; STÜCKLER, J.; LEIBE, B. Semi-supervised deep learning for monocular depth map prediction. In: **Proc. of the 30th IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2017. p. 6647–6655. ISBN 9781538604571. ISSN 1063-6919.

LADICKÝ, L.; HÄNE, C.; POLLEFEYS, M. Learning the Matching Function. 2015. Disponível em: <<http://arxiv.org/abs/1502.00652>>.

LADICKÝ, L.; SHI, J.; POLLEFEYS, M. Pulling things out of perspective. **Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)**, p. 89–96, 2014. ISSN 10636919.

LAINA, I.; RUPPRECHT, C.; BELAGIANNIS, V.; TOMBARI, F.; NAVAB, N. Deeper depth prediction with fully convolutional residual networks. In: **IEEE. 2016 Fourth International Conference on 3D Vision (3DV)**. [S.l.], 2016. p. 239–248.

LECUN, Y. Learning invariant feature hierarchies. In: **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**. [S.l.: s.n.], 2012. v. 7583 LNCS, n. PART 1, p. 496–505. ISBN 9783642338625. ISSN 03029743.

LECUN, Y.; BOSER, B.; DENKER, J. S.; HENDERSON, D.; HOWARD, R. E.; HUBBARD, W.; JACKEL, L. D. Backpropagation Applied to Handwritten Zip Code Recognition. **Neural Computation**, v. 1, n. 4, p. 541–551, 1989. ISSN 0899-7667. Disponível em: <<http://www.mitpressjournals.org/doi/10.1162/neco.1989.1.4.541>>.

LECUN, Y.; BOTTOU, L.; BENGIO, Y.; HAFFNER, P. Gradient-based learning applied to document recognition. **Proceedings of the IEEE**, v. 86, n. 11, p. 2278–2323, 1998. ISSN 00189219.

LEE, D.; GUPTA, A.; HEBERT, M.; KANADE, T. Estimating Spatial Layout of Rooms using Volumetric Reasoning about Objects and Surfaces. In **Proc. NIPS**, v. 1, p. 1–9, 2010. Disponível em: <<https://papers.nips.cc/paper/4120-estimating-spatial-layout-of-rooms-using-volumetric-reasoning-about-objects-and-surfaces.pdf>>.

LEE, H.; EKANADHAM, C.; NG, A. Y. Sparse deep belief net model for visual area V2. In: **Advances in Neural Information Processing Systems 20**. [S.l.: s.n.], 2008. p. 873–880. ISBN 160560352X | 9781605603520.

LEE, H.; GROSSE, R.; RANGANATH, R.; NG, A. Y. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In: **Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09**. [s.n.], 2009. p. 1–8. ISBN 9781605585161. ISSN 02643294. Disponível em: <<http://portal.acm.org/citation.cfm?doid=1553374.1553453>>.

LENZ, I.; LEE, H.; SAXENA, A. Deep learning for detecting robotic grasps. **International Journal of Robotics Research**, v. 34, n. 4-5, p. 705–724, 2015. ISSN 17413176.

LI, B.; SHEN, C.; DAI, Y.; Van Den Hengel, A.; HE, M. Depth and surface normal estimation from monocular images using regression on deep features and hierarchical CRFs. In: **Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2015. v. 07-12-June, p. 1119–1127. ISBN 9781467369640. ISSN 10636919.

LIAO, Y.; HUANG, L.; WANG, Y.; KODAGODA, S.; YU, Y.; LIU, Y. Parse geometry from a line: Monocular depth estimation with partial laser observation. In: **Proc. of the IEEE International Conference on Robotics and Automation (ICRA)**. [S.l.: s.n.], 2017. p. 5059–5066. ISBN 9781509046331. ISSN 10504729.

LIU, B.; GOULD, S.; KOLLER, D. Single image depth estimation from predicted semantic labels. In: . [S.l.: s.n.], 2010. p. 1253–1260. ISBN 9781424469840. ISSN 10636919.

LIU, F.; SHEN, C.; LIN, G. Deep convolutional neural fields for depth estimation from a single image. In: **Proc. of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [s.n.], 2015. p. 5162–5170. ISBN 978-1-4673-6964-0. ISSN 10636919. Disponível em: <<http://ieeexplore.ieee.org/document/7299152/>>.

LIU, F.; SHEN, C.; LIN, G.; REID, I. D. Learning Depth from Single Monocular Images Using Deep Convolutional Neural Fields. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 38, n. 10, p. 2024–2039, 2016.

LIU, M.; SALZMANN, M.; HE, X. Discrete-Continuous Depth Estimation from a Single Image. In: **Proc. of the 2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2014. p. 716–723. ISBN 978-1-4799-5118-5. ISSN 10636919.

LONG, J.; SHELHAMER, E.; DARRELL, T. Fully convolutional networks for semantic segmentation. **Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition**, v. 07-12-June, p. 3431–3440, 2015. ISSN 10636919.

LOOMIS, J. M. Looking down is looking up. **Nature**, v. 414, n. 6860, p. 155–156, nov 2001. ISSN 0028-0836. Disponível em: <<http://www.nature.com/doifinder/10.1038/35102648>>.

LU, J.; FORSYTH, D. Sparse depth super resolution. In: **Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2015.

LUO, W.; SCHWING, A. G.; URTASUN, R. Efficient Deep Learning for Stereo Matching. In: **Proc. of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2016. p. 5695–5703. ISBN 978-1-4673-8851-1. ISSN 10636919.

LUXBURG, U. von; SCHOELKOPF, B. Statistical Learning Theory: Models, Concepts, and Results. p. 1–40, 2008. ISSN 18745857. Disponível em: <<http://arxiv.org/abs/0810.4752>>.

MA, F.; KARAMAN, S. Sparse-to-Dense: Depth Prediction from Sparse Depth Samples and a Single Image. In: **Proc. of the 2018 IEEE International Conference on Robotics and Automation (ICRA)**. [S.l.: s.n.], 2018.

MAHMOOD, F.; DURR, N. J. Deep Learning and Conditional Random Fields-based Depth Estimation and Topographical Reconstruction from Conventional Endoscopy. **CoRR**, abs/1710.1, 2017.

MALIK, J. Interpreting line drawings of curved objects. **International Journal of Computer Vision**, Springer, v. 1, n. 1, p. 73–103, 1987.

MANCINI, M.; COSTANTE, G.; VALIGI, P.; CIARFUGLIA, T. A. Fast Robust Monocular Depth Estimation for Obstacle Detection with Fully Convolutional Networks. In: **International Conference on Intelligent Robots and Systems (IROS)**. [S.l.: s.n.], 2016. p. 4296–4303. ISBN 9781509037629. ISSN 21530866.

MANCINI, M.; COSTANTE, G.; VALIGI, P.; CIARFUGLIA, T. A.; DELMERICO, J.; SCARAMUZZA, D. Toward Domain Independence for Learning-Based Monocular Depth Estimation. **IEEE Robotics and Automation Letters**, v. 2, n. 3, p. 1778–1785, 2017. ISSN 2377-3766.

MANCINI, M.; COSTANTE, G.; VALIGI, P.; CIARFUGLIA, T. A. J-mod 2: Joint monocular obstacle detection and depth estimation. **IEEE Robotics and Automation Letters**, IEEE, v. 3, n. 3, p. 1490–1497, 2018.

MAYER, N.; ILG, E.; HAUSSER, P.; FISCHER, P.; CREMERS, D.; DOSOVITSKIY, A.; BROX, T. A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation. In: **Proc. of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2016. p. 4040–4048. ISBN 978-1-4673-8851-1. ISSN 10636919.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. **The Bulletin of Mathematical Biophysics**, v. 5, n. 4, p. 115–133, 1943. ISSN 00074985.

MEMISEVIC, R.; CONRAD, C. Stereopsis via deep learning. **Neural Information Processing Systems workshop on Deep Learning and Unsupervised Feature Learning**, p. 1–9, 2011.

MENDES, C. C. T. **Navigability estimation for autonomous vehicles using machine learning**. 2017. 96 p. Tese (Tese de Doutorado) — Universidade de São Paulo, 2017. Disponível em: <<http://www.teses.usp.br/teses/disponiveis/55/55134/tde-25092017-102021/pt-br.php>>.

MENZE, M.; GEIGER, A. Object scene flow for autonomous vehicles. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2015. p. 3061–3070.

MICHELS, J.; SAXENA, A.; NG, A. Y. High speed obstacle avoidance using monocular vision and reinforcement learning. **Proceedings of the 22nd international conference on Machine learning - ICML '05**, p. 593–600, 2005. ISSN <null>. Disponível em: <<http://portal.acm.org/citation.cfm?doid=1102351.1102426>>.

MINSKY, M.; PAPERT, S. **Perceptrons**. [S.l.]: MIT press, 1969.

MITCHELL, T. M. **Machine Learning**. [s.n.], 1997. v. 1. 432 p. ISSN 08856125. ISBN 0070428077. Disponível em: <<https://books.google.com.br/books?id=EoYBngEACAAJ>>.

MUKASA, T.; XU, J.; BJORN, S. 3D Scene Mesh from CNN Depth Predictions and Sparse Monocular SLAM. In: **2017 IEEE International Conference on Computer Vision Workshops (ICCVW)**. [S.l.: s.n.], 2017. p. 912–919.

MUR-ARTAL, R.; MONTIEL, J. M. M.; TARDOS, J. D. Orb-slam: a versatile and accurate monocular slam system. **IEEE Transactions on**, v. 31, n. 5, p. 1147–1163, 2015. ISSN 15523098.

MUROKI, B. **Driving on London roads**. 2016. <<https://www.youtube.com/watch?v=E0k92wHArFE>>.

MURPHY, K. P. **Machine Learning: A Probabilistic Perspective**. [S.l.: s.n.], 2012. 25 p. ISSN 0036-8733. ISBN 0070428077.

NITZBERG, M.; MUMFORD, D. The 2.1-d sketch. In: IEEE. **[1990] Proceedings Third International Conference on Computer Vision**. [S.l.], 1990. p. 138–144.

NOH, H.; HONG, S.; HAN, B. Learning deconvolution network for semantic segmentation. **Proceedings of the IEEE International Conference on Computer Vision**, v. 2015 Inter, p. 1520–1528, may 2015. ISSN 15505499. Disponível em: <<http://arxiv.org/abs/1505.04366>>.

OKOSHI, T. **Three-dimensional imaging techniques**. [S.l.]: Elsevier, 2012.

OWEN, A. B. A robust hybrid of lasso and ridge regression. **Contemporary Mathematics**, Providence, RI: American Mathematical Society, v. 443, n. 7, p. 59–72, 2007.

PELUSO, V.; CIPOLLETTA, A.; CALIMERA, A.; POGGI, M.; TOSI, F.; MATTOCCIA, S. Enabling energy-efficient unsupervised monocular depth estimation on armv7-based platforms. In: IEEE. **2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)**. [S.l.], 2019. p. 1703–1708.

POGGI, M.; ALEOTTI, F.; TOSI, F.; MATTOCCIA, S. Towards real-time unsupervised monocular depth estimation on cpu. In: IEEE. **2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. [S.l.], 2018. p. 5848–5854.

PONTI, M. A.; COSTA, G. B. P. da. Como funciona o deep learning. **arXiv preprint arXiv:1806.07908**, 2018.

PUERTA, A. M. The power of shadows: shadow stereopsis. **JOSA A**, Optical Society of America, v. 6, n. 2, p. 309–311, 1989.

PUNJANI, A.; ABBEEL, P. Deep learning helicopter dynamics models. **Proceedings - IEEE International Conference on Robotics and Automation**, v. 2015-June, n. June, p. 3223–3230, 2015. ISSN 10504729.

QI, C. R.; LIU, W.; WU, C.; SU, H.; GUIBAS, L. J. Frustum pointnets for 3d object detection from rgb-d data. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2018. p. 918–927.

QUIGLEY, M.; CONLEY, K.; GERKEY, B.; FAUST, J.; FOOTE, T.; LEIBS, J.; WHEELER, R.; NG, A. Y. Ros: an open-source robot operating system. In: KOBE, JAPAN. **ICRA workshop on open source software**. [S.l.], 2009. v. 3, n. 3.2, p. 5.

RAINA, R.; MADHAVAN, A.; NG, A. Y. Large-scale deep unsupervised learning using graphics processors. In: ACM. **Proceedings of the 26th annual international conference on machine learning**. [S.l.], 2009. p. 873–880.

RAJPURKAR, P.; IRVIN, J.; ZHU, K.; YANG, B.; MEHTA, H.; DUAN, T.; DING, D.; BAGUL, A.; LANGLOTZ, C.; SHPANSKAYA, K.; OTHERS. CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning. **arXiv preprint arXiv:1711.05225**, 2017.

RAMOS, F.; OTT, L. Hilbert maps: Scalable continuous occupancy mapping with stochastic gradient descent. **International Journal of Robotics Research (IJRR)**, v. 35, n. 14, p. 1717–1730, 2016.

RANZATO, M.; HUANG, F. J.; BOUREAU, Y. L.; LECUN, Y. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In: **Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2007. ISBN 1424411807. ISSN 10636919.

REZENDE, D. J.; ESLAMI, S. M. A.; MOHAMED, S.; BATTAGLIA, P.; JADERBERG, M.; HEES, N. Unsupervised Learning of 3D Structure from Images. In: **Proc. of the 30th International Conference on Neural Information Processing Systems (NIPS)**. USA: Curran Associates Inc., 2016. p. 5003–5011. ISBN 978-1-5108-3881-9.

RIEGLER, G.; RÜTHER, M.; BISCHOF, H. Atgv-net: Accurate depth super-resolution. In: SPRINGER. **Proc. of the European Conference on Computer Vision (ECCV)**. [S.l.], 2016. p. 268–284.

ROBERTS, L. G. **Machine perception of three-dimensional solids**. 1963. Tese (Doutorado) — Massachusetts Institute of Technology, 1963.

ROS, G.; SELLART, L.; MATERZYNSKA, J.; VAZQUEZ, D.; LOPEZ, A. M. The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes. In: **2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2016. p. 3234–3243. ISBN 978-1-4673-8851-1. ISSN 10636919.

ROY, A.; TODOROVIC, S. Monocular Depth Estimation Using Neural Regression Forest. In: **Proc. of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2016. p. 5506–5514. ISBN 9781467388511. ISSN 10636919.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. **Nature**, v. 323, n. 6088, p. 533–536, 1986. ISSN 00280836.

RUSSELL, B. C.; TORRALBA, A. Building a database of 3D scenes from user annotations. **2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2009**, p. 2711–2718, 2009. ISSN 1063-6919.

SAUND, E. Logic and mrf circuitry for labeling occluding and thinline visual contours. In: **Advances in Neural Information Processing Systems**. [S.l.: s.n.], 2006. p. 1153–1159.

SAXENA, A.; CHUNG, S. H.; NG, A. Y. Learning Depth from Single Monocular Images. **Advances in Neural Information Processing Systems**, v. 18, p. 1161–1168, 2006. ISSN 0920-5691.

_____. 3-D depth reconstruction from a single still image. **International Journal of Computer Vision**, v. 76, n. 1, p. 53–69, 2008. ISSN 09205691.

SAXENA, A.; SCHULTE, J.; NG, A. Y. Depth Estimation using Monocular and Stereo Cues. **International Joint Conference on Artificial Intelligence**, 2007. Disponível em: <<http://www.cs.cornell.edu/~asaxena/learningdepth/>>.

SAXENA, A.; SUN, M.; NG, A. Y. Make3D: Learning 3D scene structure from a single still image. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 31, n. 5, p. 824–840, 2009. ISSN 01628828.

SCHARSTEIN, D.; SZELISKI, R. A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms. **International Journal of Computer Vision**, v. 47, n. 1-3, p. 7–42, 2002. ISSN 0920-5691.

SCHMIDHUBER, J. Deep learning in neural networks: An overview. **Neural networks**, Elsevier, v. 61, p. 85–117, 2015.

SCHNEIDER, N.; SCHNEIDER, L.; PINGGERA, P.; FRANKE, U.; POLLEFEYS, M.; STILLER, C. Semantically guided depth upsampling. **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**, v. 9796 LNCS, p. 37–48, 2016. ISSN 16113349.

SCHWARZ, B. Lidar: Mapping the world in 3d. **Nature Photonics**, Nature Publishing Group, v. 4, n. 7, p. 429, 2010.

SEIDE, F.; AGARWAL, A. CNTK: Microsoft’s open-source deep-learning toolkit. In: **Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)**. ACM Press, 2016. p. 2135–2135. ISBN 9781450342322. Disponível em: <<http://dl.acm.org/citation.cfm?doid=2939672.2945397>>.

SILBERMAN, N.; HOIEM, D.; KOHLI, P.; and Rob Fergus. Indoor segmentation and support inference from rgbd images. **European Conference on Computer Vision (ECCV)**, p. 1–14, 2012.

SILVA, I. N. da; SPATTI, D. H.; FLAUZINO, R. A. **Redes Neurais Artificiais para Engenharia e ciências aplicadas**. [S.l.]: ARTLIBER, 2010. ISBN 9788588098534.

SILVA, I. N. da; SPATTI, D. H.; FLAUZINO, R. A.; LIBONI, L. H. B.; dos Reis Alves, S. F. **Artificial Neural Networks: A Practical Course**. Springer International Publishing, 2016. ISBN 9783319431628. Disponível em: <https://books.google.com.br/books?id=DL{_}mDAAAQ>.

SIMONYAN, K.; ZISSERMAN, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. **International Conference on Learning Representations (ICRL)**, p. 1–14, sep 2015. ISSN 09505849. Disponível em: <<http://arxiv.org/abs/1409.1556>>.

SINGH, R.; SRIVASTAVA, S. Stock prediction using deep learning. **Multimedia Tools and Applications**, p. 1–16, 2016. ISSN 15737721. Disponível em: <<http://dx.doi.org/10.1007/s11042-016-4159-7>>.

SONG, S.; YU, F.; ZENG, A.; CHANG, A. X.; SAVVA, M.; FUNKHOUSER, T. Semantic Scene Completion from a Single Depth Image. In: **Proc. of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2017. p. 190–198. ISBN 978-1-5386-0457-1. ISSN 1611.08974.

SONG, X.; DAI, Y.; QIN, X. Deep depth super-resolution: Learning depth super-resolution using deep convolutional neural network. In: SPRINGER. **Asian Conference on Computer Vision**. [S.l.], 2016. p. 360–376.

SRIVASTAVA, N.; HINTON, G.; KRIZHEVSKY, A.; SUTSKEVER, I.; SALAKHUTDINOV, R. Dropout: A simple way to prevent neural networks from overfitting. **Journal of Machine Learning Research (JMLR)**, v. 15, p. 1929–1958, 2014.

_____. Dropout: a simple way to prevent neural networks from overfitting. **The Journal of Machine Learning Research**, JMLR. org, v. 15, n. 1, p. 1929–1958, 2014.

_____. Dropout: A simple way to prevent neural networks from overfitting. **Journal of Machine Learning Research**, v. 15, p. 1929–1958, 2014.

SUGIHARA, K. An algebraic approach to shape-from-image problems. **Artificial Intelligence**, v. 23, n. 1, p. 59 – 95, 1984. ISSN 0004-3702. Disponível em: <<http://www.sciencedirect.com/science/article/pii/0004370284900055>>.

SUN, Y.; WANG, X.; TANG, X. Deep learning face representation from predicting 10,000 classes. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2014. p. 1891–1898.

SZEGEDY, C.; LIU, W.; JIA, Y.; SERMANET, P.; REED, S.; ANGUELOV, D.; ERHAN, D.; VANHOUCKE, V.; RABINOVICH, A. Going deeper with convolutions. In: **Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2015. v. 07-12-June, p. 1–9. ISBN 9781467369640. ISSN 10636919.

TANIKIĆ, D.; DESPOTOVIC, V. Artificial intelligence techniques for modelling of temperature in the metal cutting process. In: **Metallurgy-Advances in Materials and Processes**. [S.l.]: InTech, 2012.

TATENO, K.; TOMBARI, F.; LAINA, I.; NAVAB, N. CNN-SLAM: Real-time dense monocular SLAM with learned depth prediction. **Computer Vision and Pattern Recognition (CVPR)**, 2017. Disponível em: <<http://arxiv.org/abs/1704.03489>>.

Theano Development Team. **Convolutional Neural Networks (LeNet) — DeepLearning 0.1 documentation**. 2018. Disponível em: <<http://deeplearning.net/tutorial/lenet.html>>.

TOZAWA, J. Role of a texture gradient in the perception of relative size. **Perception**, SAGE Publications Sage UK: London, England, v. 39, n. 5, p. 641–660, 2010.

UHRIG, J.; SCHNEIDER, N.; SCHNEIDER, L.; FRANKE, U.; BROX, T.; GEIGER, A. Sparsity invariant cnns. In: **2017 International Conference on 3D Vision (3DV)**. [S.l.: s.n.], 2017. p. 11–20. ISSN 2475-7888.

VEDATALI. **Dubai Drive**. 2013. <<https://www.youtube.com/watch?v=K0EoRwGOAVA>>.

VERNWAL, R. **How should I start with CNN? – Becoming Human: Artificial Intelligence Magazine**. 2017. Disponível em: <<https://becominghuman.ai/how-should-i-start-with-cnn-c62a3a89493b>>.

VOULODIMOS, A.; DOULAMIS, N.; DOULAMIS, A.; PROTOPAPADAKIS, E. Deep Learning for Computer Vision: A Brief Review. **Computational Intelligence and Neuroscience**, Hindawi, v. 2018, 2018.

WALTZ, D.; WINSTON, P. The psychology of computer vision. 1975.

WANG, P.; SHEN, X.; LIN, Z.; COHEN, S.; PRICE, B.; YUILLE, A. Towards unified depth and semantic prediction from a single image. In: **Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2015. p. 2800–2809. ISBN 9781467369640. ISSN 10636919.

WEERASEKERA, C. S.; DHARMASIRI, T.; GARG, R.; DRUMMOND, T.; REID, I. Just-in-Time Reconstruction: Inpainting Sparse Maps using Single View Depth Predictors as Priors. In: **Proc. of the 2018 IEEE International Conference on Robotics and Automation (ICRA)**. [S.l.: s.n.], 2018. ISBN 9781538630808.

WITTEN, I. H.; FRANK, E.; HALL, M. A.; PAL, C. J. **Data Mining: Practical machine learning tools and techniques**. [S.l.]: Morgan Kaufmann, 2016.

WONG, P. W.; HERLEY, C. **Area based interpolation for image scaling**. [S.l.]: Google Patents, 1999. US Patent 5,889,895.

WU, C.; FRAHM, J. M.; POLLEFEYS, M. Repetition-based dense single-view reconstruction. In: **Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)**. IEEE, 2011. p. 3113–3120. ISBN 9781457703942. ISSN 10636919. Disponível em: <<http://ieeexplore.ieee.org/document/5995551/>>.

XIE, J.; GIRSHICK, R.; FARHADI, A. Deep3D: Fully automatic 2D-to-3D video conversion with deep convolutional neural networks. In: **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**. [S.l.: s.n.], 2016. v. 9908 LNCS, p. 842–857. ISBN 9783319464923. ISSN 16113349.

XU, D.; RICCI, E.; OUYANG, W.; WANG, X.; SEBE, N. Multi-Scale Continuous CRFs as Sequential Deep Networks for Monocular Depth Estimation. **CVPR**, 2017. Disponível em: <<http://arxiv.org/abs/1704.02157>>.

YAN, C.; XU, W.; LIU, J. Can you trust autonomous vehicles: Contactless attacks against sensors of self-driving vehicle. **DEF CON**, v. 24, 2016.

YANG, J.; WRIGHT, J.; HUANG, T. S.; MA, Y. Image super-resolution via sparse representation. **IEEE Transactions on Image Processing**, v. 19, n. 11, p. 2861–2873, 2010. ISSN 10577149.

YUAN, Z.; LU, Y.; WANG, Z.; XUE, Y. Droid-Sec : Deep Learning in Android Malware Detection. **Sigcomm 2014**, p. 371–372, 2014. ISSN 19435819.

ŽBONTAR, J.; LECUN, Y. Computing the stereo matching cost with a convolutional neural network. In: **Proc. of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2015. p. 1592–1599. ISBN 978-1-4673-6964-0. ISSN 10636919.

_____. Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches. **CoRR**, abs/1510.0, 2015. Disponível em: <<http://arxiv.org/abs/1510.05970>>.

ZEILER, M. D. Adadelta: an adaptive learning rate method. **arXiv preprint arXiv:1212.5701**, 2012.

ZEILER, M. D. **Hierarchical Convolutional Deep Learning in Computer Vision**. 2014. Tese (Doutorado) — New York University, 2014.

ZEILER, M. D.; FERGUS, R. Visualizing and understanding convolutional networks. In: **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**. [S.l.: s.n.], 2014. v. 8689 LNCS, n. PART 1, p. 818–833. ISBN 9783319105895. ISSN 16113349.

ZHAN, H.; GARG, R.; WEERASEKERA, C. S.; LI, K.; AGARWAL, H.; REID, I. Unsupervised Learning of Monocular Depth Estimation and Visual Odometry with Deep Feature Reconstruction. 2018. Disponível em: <<http://arxiv.org/abs/1803.03893>>.

ZHANG, Y.; FUNKHOUSER, T. Deep Depth Completion of a Single RGB-D Image. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2018. p. 175–185.

ZHAO, C.; SUN, L.; PURKAIT, P.; DUCKETT, T.; STOLKIN, R. Learning monocular visual odometry with dense 3d mapping from dense 3d flow. In: IEEE. **2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. [S.l.], 2018. p. 6864–6871.

ZHOU, H.; UMMENHOFER, B.; BROX, T. Deeptam: Deep tracking and mapping. In: **The European Conference on Computer Vision (ECCV)**. [S.l.: s.n.], 2018.

ZHOU, T.; BROWN, M.; SNAVELY, N.; LOWE, D. G. Unsupervised Learning of Depth and Ego-Motion from Video. In: **Proc. of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2017. v. 2, n. 6, p. 7.

ZOU, H.; HASTIE, T. Regularization and variable selection via the elastic net. **Journal of the Royal Statistical Society, Series B**, v. 67, p. 301–320, 2005.

ZWALD, L.; LAMBERT-LACROIX, S. The BerHu penalty and the grouped effect. **arXiv preprint arXiv:1207.6868**, jul 2012. Disponível em: <<http://arxiv.org/abs/1207.6868>>.

Apêndices

APÊNDICE A – REPOSITÓRIO

Os softwares desenvolvidos para este trabalho estão hospedados no endereço <<https://github.com/nicolasrosa/FCRN-DepthPrediction>>. Os comandos necessários para executá-los estão descritos em detalhes no arquivo “README.md”.