A Comparison Study of Depth Map Estimation in Indoor Environments Using Pix2Pix and CycleGAN

Ricardo S. Casado, and Emerson C. Pedrino

***Abstract*—This article presents a Deep Learning-based approach for comparing automatic depth map estimation in indoor environments, with the aim of using them in navigation aid systems for visually impaired individuals. Depth map estimation is a laborious process, as most high-precision systems consist of complex stereo vision systems. The methodology utilizes Generative Adversarial Networks (GANs) techniques for generating depth maps from single RGB images. The study introduces methods for generating depth maps using pix2pix and CycleGAN. The major challenges still lie in the need to use large datasets, which are coupled with long training times. Additionally, a comparison of L1 Loss with a variation of the MonoDepth2 and DenseDepth systems was performed, using ResNet50 and ResNet18 as encoders, which are mentioned in this work, for comparison and validation of the presented method. The results demonstrate that CycleGAN is capable of generating more reliable maps compared to pix2pix and DepthNet_ResNet50, with an L1 Loss approximately 2.5 times smaller than pix2pix, approximately 2.4 times smaller than DepthNet_ResNet50, and approximately 14 times smaller than DepthNet_ResNet18.**

Link to graphical and video abstracts, and to code: https://latamt.ieeer9.org/index.php/transactions/article/view/8429

***Index Terms*—depth map, navigation for visually impaired individuals, generative adversarial networks (GAN), pix2pix, CycleGAN, MonoDepth2, DenseDepth, DepthNet_Resnet50, DepthNet_Resnet18.**

I. Introdução

A estimativa de mapas de profundidade a partir de imagens monoculares é uma alternativa rápida e eficiente que utiliza técnicas de Aprendizado Profundo. Esses métodos, baseados em aprendizado profundo supervisionado, são considerados de última geração para essa tarefa. No entanto, ainda apresentam um desempenho inferior em comparação com os métodos de estimativa de profundidade estéreo, que utilizam pares de imagens. Devido à complexidade de configurar sistemas estéreo, os métodos monoculares de estimativa de profundidade ainda são preferidos, apesar de seu desempenho inferior [1].

A detecção de profundidade é uma tecnologia fundamental para a segurança e autonomia em veículos autônomos e drones. Além do setor automotivo, há também interesse em utilizar mapas de profundidade em outras áreas, como auxílio na navegação para deficientes visuais. A capacidade de analisar detalhadamente uma cena proporciona recursos aprimorados para a tomada de decisões [2].

Realizar a estimativa de profundidade a partir de imagens monoculares é um desafio devido à ambiguidade de escala e à natureza 2D das imagens. É necessário considerar o contexto global da cena 3D para obter estimativas confiáveis. Modelos anteriores têm conseguido recuperar a profundidade de uma única imagem com base nesse contexto [3]. No entanto, os seres humanos têm a capacidade de inferir profundidade plausível para novas cenas devido à experiência de interação com o mundo real e ao sistema de visão [4].

Estimar a profundidade é uma abordagem poderosa para o pré-treinamento de redes profundas utilizando conjuntos de dados de imagens não rotuladas [5]. No entanto, coletar grandes conjuntos de dados de treinamento diversificados com ground truth preciso é uma tarefa desafiadora [6], [7].

Para a estimativa de mapas de profundidade a partir de uma única imagem, foram desenvolvidos modelos avançados de reconhecimento de imagem [7], [8], [9]. As redes neurais convolucionais profundas (ConvNets) atualmente utilizadas são capazes de compreender as relações globais entre os pixels da imagem e codificar informações prévias. No entanto, essas redes são treinadas apenas com perdas por pixel [3].

As GAN's (*Generative Adversarial Networks*) [10], [11], [12] são capazes de gerar imagens realistas e coerentes globalmente. Elas utilizam uma rede neural discriminadora para penalizar a saída gerada que difere dos dados reais, resultando em uma perda adaptável. Nos últimos anos, as GANs têm recebido muita atenção na pesquisa, com melhorias na qualidade dos dados gerados [13]. Elas têm sido bem-sucedidas em várias tarefas, como reconstrução de imagens, segmentação de imagens, estimativa de ponto de vista e estimativa de profundidade [12], [14], [15], [16].

Portanto, o escopo primordial da presente pesquisa visa à estimativa de mapas de profundidade destinados a aplicações de auxílio à navegação para indivíduos com deficiência visual a partir de imagens monoculares. Foi explorada a capacidade das GANs em lidar com tarefas de visão computacional. Em particular, a análise comparativa é direcionada às redes pix2pix e CycleGAN, notoriamente empregadas na tradução de imagens. A motivação para o estudo é fundamentada na ausência de comparações entre essas duas arquiteturas para tal finalidade. Identificada essa lacuna após uma revisão bibliográfica não exaustiva, que evidencia a escassez de investigações que confrontem as duas no contexto da estimativa de mapas de profundidade associados a sistemas de navegação desenvolvidos para atender às necessidades específicas de deficientes visuais.

Ricardo S. Casado is with Universidade Federal de São Carlos - UFSCar and the Instituto Federal de São Paulo - IFSP, São Carlos, Brazil (e-mail: rscasado@ifsp.edu.br).

Emerson C. Pedrino is with Universidade Federal de São Carlos - UFSCar, São Carlos, Brazil (e-mail: emerson@dc.ufscar.br).

Técnicas de *Deep Learning* apresentam soluções rápidas e eficientes, sendo alternativas aos métodos que utilizam pares de imagens estéreo. Isso porque eliminam a complexidade sistêmica, como a necessidade de realizar a calibração das câmeras para a obtenção das imagens das cenas, visando realizar os cálculos que determinam a profundidade. Após o treinamento do modelo, torna-se possível efetuar previsões dos mapas de uma determinada cena em questão de milissegundos.

II. Artigos Relacionados

No artigo [17], é apresentado um método inovador para estimar a profundidade monocular utilizando uma rede generativa adversarial CycleGAN e segmentação. O método proposto combina informações de segmentação para estimar a profundidade e consiste em três etapas: segmentação e estimativa de profundidade, cálculos de perda adversarial e cálculos de consistência de ciclo. O processo de cálculo de consistência de ciclo avalia a similaridade entre duas imagens quando são restauradas em suas formas originais após serem estimadas separadamente a partir de duas perdas adversariais.

No artigo [18], é proposto um método para estimar a profundidade de uma imagem capturada por uma câmera monocular, visando evitar colisões durante o voo autônomo de um drone. O método utiliza a pix2pix para gerar imagens de profundidade a partir de uma câmera monocular. Além disso, incorpora o fluxo óptico para aprimorar a precisão da estimativa de profundidade. Segundo os autores, o trabalho apresenta um método de estimativa de profundidade altamente preciso, que eficazmente integra um mapa de fluxo óptico em uma imagem monocular. Os modelos são treinados utilizando o AirSim, um dos simuladores de voo disponíveis. O AirSim é capaz de capturar tanto imagens monoculares quanto de profundidade em um ambiente virtual, a uma distância de mais de cem metros. O modelo gera uma imagem de profundidade que fornece informações de longa distância superiores às imagens capturadas por uma câmera de profundidade comum.

No artigo [19], é apresentada uma rede generativa adversarial subaquática (UW-GAN) para estimar a profundidade a partir de uma única imagem subaquática. A rede utiliza uma abordagem em dois níveis, estimando inicialmente um mapa de profundidade grosseiro com a UWC-Net e, em seguida, refinando-o com a UWF-Net. A UWF-Net emprega um bloco de compressão e excitação espacial e por canal para estimar a profundidade em nível fino. Além disso, é proposta uma abordagem de geração de imagens subaquáticas sintéticas para construir um banco de dados em larga escala. A rede é testada em conjuntos de dados reais e sintéticos para avaliar seu desempenho.

III. Metodologia

Esta seção aborda os procedimentos adotados para a estimação dos mapas de profundidade, levando em consideração a capacidade das GANs na tradução de imagens. A ideia principal deste trabalho é explorar essa característica para gerar automaticamente mapas de profundidade a serem utilizados em sistemas de auxílio à navegação para deficientes visuais em ambientes internos.

As GANs, conforme propostas por Goodfellow et al. [10], representam um notável avanço no campo da aprendizagem não supervisionada. Essas redes consistem em dois módulos principais: um gerador e um discriminador, ambos implementados como redes neurais. O gerador tem a função de capturar a distribuição dos dados, enquanto o discriminador trabalha na estimativa da probabilidade de uma amostra ser classificada como real ou sintética [2].

O gerador G e o discriminador D são treinados simultaneamente em um jogo minimax. O gerador usa a descida do gradiente e a retropropagação do erro para minimizar a função de perda $-\log(1-D(G(z)))$, enquanto o discriminador minimiza $-\log D(X)$. Essa otimização é realizada como um jogo entre os dois modelos, com uma função de valor $V(G,D)$, representada pela Equação 1 [2].

$$\begin{matrix} min & max \\ G & D \end{matrix} V(D,G) = E_{x P data(x)} log[D(x)] + E_{z px(z)}[1 - D(G(z))] \quad (1)$$

A. *Arquitetura e Funcionamento da Rede pix2pix*

A pix2pix utiliza uma rede generativa adversarial condicional (cGAN) para aprender a mapear uma imagem de entrada para uma imagem de saída. Em outras palavras, o gerador é condicionado a uma imagem de entrada.

A rede consiste em um gerador e um discriminador. O gerador transforma a imagem de entrada para gerar a imagem de saída, enquanto o discriminador compara a imagem de entrada com uma imagem desconhecida para determinar se foi produzida pelo gerador [12].

Nesse caso, o gerador está tentando aprender como traduzir uma imagem capturada por uma câmera em uma imagem que representa o mapa de profundidade esperado (real), conforme previamente apresentado à rede.

O discriminador analisa as tentativas de tradução do gerador e procura aprender a distinguir entre a tradução fornecida e a imagem verdadeira (esperada) contida no conjunto de dados.

O gerador transforma uma imagem de entrada RGB (*Red Green Blue*) na tentativa de reproduzir a imagem desejada, que, neste caso, é o mapa de profundidade. Sua estrutura é nada mais que um codificador-decodificador.

A entrada é uma imagem de 256x256 pixels com 3 canais de cores (RGB), enquanto a saída é apresentada em níveis de cinza (256x256x1). A escolha dessa resolução para as imagens de entrada se deve apenas à limitação do hardware utilizado para o treinamento da rede.

O gerador utiliza codificadores que possuem camadas de convolução seguidas por uma função de ativação para extrair características importantes da imagem e reduzi-la a uma representação menor. Isso permite obter uma abstração de nível mais alto dos dados após a camada final de codificação. As camadas de decodificação fazem o oposto, ou seja, realizam a deconvolução também acompanhadas de uma função de ativação. Basicamente, o decodificador desfaz as transformações realizadas pelas camadas de codificação.

O discriminador recebe duas imagens: a imagem do mapa de profundidade desejado e uma imagem desconhecida gerada pelo gerador. Sua tarefa é determinar se a imagem gerada é

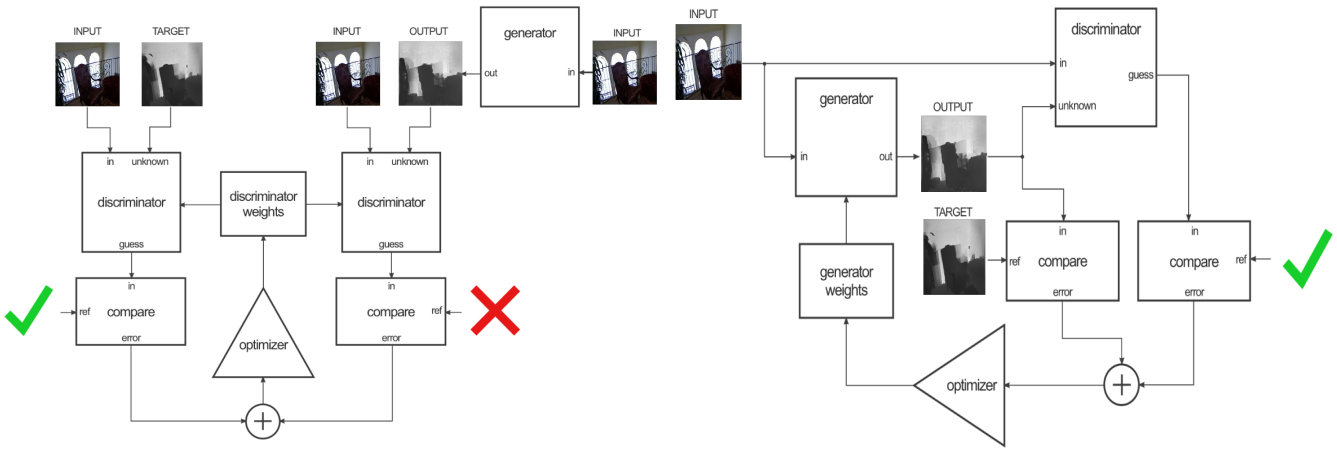


Fig. 1. Estrutura geral da rede pix2pix. Em (a) mostra o discriminador recebendo como entrada as imagens RGB e fazendo a comparação entre o mapa esperado e o mapa gerado para atualizar os pesos. Em (b) mostra o discriminador realizando a comparação entre o mapa esperado e o gerado. Fonte: adaptado de [20].

suficientemente semelhante à imagem esperada, a ponto de não conseguir distinguir qual é real e qual é falsa.

A estrutura do discriminador se assemelha à seção de codificação do gerador, mas opera de maneira diferente. A saída é uma imagem de 30x30 pixels, onde cada valor de pixel representa a plausibilidade da seção correspondente da imagem desconhecida. Na implementação da pix2pix, cada pixel dessa imagem de 30x30 representa a plausibilidade de um *patch* de 70x70 pixels da imagem de entrada, com uma sobreposição considerável, uma vez que as imagens de entrada têm tamanho 256x256. Essa arquitetura é conhecida como "PatchGAN".

As Figs. 1b e 1a ilustram em alto nível a estrutura geral da pix2pix, onde o gerador é condicionado a uma imagem (RGB) de entrada e tenta recriar uma imagem qualquer, que é apresentada ao discriminador. A função do discriminador é pegar pares de imagens, uma imagem de entrada (RGB) e uma imagem desconhecida (que pode ser uma imagem desejada do conjunto de dados ou uma imagem de saída do gerador), e decidir se a segunda imagem foi produzida pelo gerador ou não. O discriminador analisa as tentativas de reprodução do gerador e tenta aprender a distinguir entre as reproduções fornecidas pelo gerador e a verdadeira imagem alvo fornecida no conjunto de dados.

O algoritmo de treinamento simplificado da rede pix2pix observado em Algoritmo [1], é o seguinte:

Início do Procedimento: o procedimento é iniciado, levando como entrada os conjuntos de dados condicionados X e Y, o número de épocas de treinamento (n_epoch) e a taxa de aprendizado (l_rate).

Carregar Pesos do Gerador e Discriminador: os pesos iniciais para o gerador (wG) e o discriminador (wD) são carregados. Esses pesos são os parâmetros ajustáveis da rede que serão otimizados durante o treinamento.

Número de Épocas de Treinamento: define-se o número total de épocas de treinamento (n_epoch), que representa quantas vezes o modelo percorrerá todo o conjunto de treinamento.

Algoritmo 1 Treinamento Pix2Pix

procedure TRAINPIX2PIX(X, Y, n_epoch, l_rate)
  LoadWGen(w_G)
  LoadWDisc(w_D)
  n_epoch=200
  for epoch $\leftarrow 1$ **up to** n_epoch **do**
    ShuffleTrainData: X, Y
    for ForEachPair (x, y) in (X, Y) **do**
      $\hat{y} \leftarrow$ GenImage(x, w_G)
      $\mathcal{L}_{\text{adv}}(D) \leftarrow$ LossAdvDisc$(y, \hat{y}, w_D)$
      $\mathcal{L}_{\text{adv}}(G) \leftarrow$ LossAdvGen$(\hat{y}, w_D)$
      $\mathcal{L}_{\text{rec}}(G) \leftarrow$ LossRec$(y, \hat{y})$
      $\mathcal{L}_{\text{total}}(G) \leftarrow$ WeighLosses$(\mathcal{L}_{\text{adv}}(G), \mathcal{L}_{\text{rec}}(G))$
      $w_D \leftarrow$ UpWDisc$(\mathcal{L}_{\text{adv}}(D)$, l_rate$)$
      $w_G \leftarrow$ UpWGen$(\mathcal{L}_{\text{total}}(G)$, l_rate$)$
    end for
  end for
  return w_G
end procedure

Laço de Treinamento por Época: inicia-se um laço de treinamento que percorre cada época, do 1 até o número total de épocas definido.

Embaralhar Dados de Treinamento: os conjuntos de dados de entrada e saída condicionados (X e Y) são embaralhados, garantindo variedade nas amostras de treinamento a cada época.

Laço de Treinamento para Pares de Entrada e Saída: inicia-se um segundo laço que percorre cada par de entrada e saída condicionada (x, y) nos conjuntos X e Y, respectivamente.

Gerar Imagem Condicionada pelo Gerador: utilizando o gerador (GenImage) e seus pesos atuais (wG), uma imagem condicionada $\hat{y}$ é gerada a partir da imagem de entrada (x).

Calcular Perda Adversarial para Discriminador

(Ladv(D)): calcula-se a perda adversarial para o discriminador (Ladv(D)) comparando a imagem real (y) e a imagem gerada $\hat{y}$ utilizando os pesos atuais do discriminador (wD).

Calcular Perda Adversarial para Gerador (Ladv(G)): calcula-se a perda adversarial para o gerador (Ladv(G)) comparando a imagem gerada $\hat{y}$ com os pesos atuais do discriminador (wD).

Calcular Perda Total para o Gerador (Ltotal(G)): combina as perdas adversariais e de reconstrução ponderadas para calcular a perda total para o gerador (Ltotal(G)).

Atualizar Pesos do Discriminador (wD): utiliza-se a perda adversarial para o discriminador (Ladv(D)) para atualizar os pesos do discriminador (wD) usando uma técnica de otimização (UpWDisc).

Atualizar Pesos do Gerador (wG): utiliza-se a perda total para o gerador (Ltotal(G)) para atualizar os pesos do gerador (wG) usando uma técnica de otimização (UpWGen).

Final do Laço de Treinamento para Pares de Entrada e Saída: o laço interno de treinamento é concluído.

Final do Laço de Treinamento por Época: o laço externo de treinamento por época é concluído.

Retornar Pesos do Gerador Atualizados: após o treinamento completo, os pesos finais atualizados do gerador (wG) são retornados.

Fim do Procedimento: o procedimento de treinamento pix2pix é concluído.

B. Arquitetura e Funcionamento da Rede CycleGAN

A CycleGAN é capaz de aprender transformações entre domínios de origem e destino sem a necessidade de exemplos em pares de treinamento. Diferentemente de métodos anteriores, como o pix2pix, que exigem exemplos com dados correspondentes em ambos os domínios, ela é capaz de aprender essas transformações de maneira mais flexível. Isso é alcançado por meio de um processo em dois passos, no qual a imagem do domínio de origem é mapeada para o domínio de destino e, em seguida, reconstruída para a imagem original. A qualidade da transformação é aprimorada por meio do confronto entre uma rede geradora e um discriminador [21].

O modelo possui dois geradores (um para cada direção da tradução) e dois discriminadores (um para cada domínio). Os geradores G e F tentam traduzir imagens do domínio A para o B e vice-versa, enquanto os discriminadores D_B e D_A avaliam a autenticidade das imagens em seus respectivos domínios.

O treinamento ocorre de maneira adversarial, semelhante a uma GAN convencional. O gerador G traduz imagens de A para B, e o discriminador D_B avalia a autenticidade dessas imagens. O gerador F realiza a tradução reversa de B para A, sendo avaliado pelo discriminador D_A. O ciclo é fechado quando G traduz a imagem de A para B e F a traduz de volta para A. Esses ciclos ajudam a preservar as características da imagem original.

Uma função de perda de ciclo é utilizada para penalizar a diferença entre a imagem original e a imagem reconstruída após o ciclo. Isso garante consistência na tradução nos dois sentidos.

Algoritmo 2 Treinamento CycleGAN

procedure TRAINCYCLEGAN(X, Y, n_epoch, l_rate)
  LoadWGen(w_{G_X}, w_{G_Y})
  LoadWDisc(w_{D_X}, w_{D_Y})
  n_epoch=100
  for epoch $\leftarrow 1$ **up to** n_epoch **do**
    ShuffleTrainData: X, Y
    for ForEachPair (x, y) in (X, Y) **do**
      $UpdateDiscriminators$
      $\hat{y} \leftarrow$ GenImage(x, w_{G_X})
      $\hat{x} \leftarrow$ GenImage(y, w_{G_Y})
      $\mathcal{L}_{\text{adv}}(D_X) \leftarrow$ LossAdvDisc$(x, \hat{x}, w_{D_X})$
      $\mathcal{L}_{\text{adv}}(D_Y) \leftarrow$ LossAdvDisc$(y, \hat{y}, w_{D_Y})$
      $w_{D_X} \leftarrow$ UpWDisc$(x, \hat{x}$, l_rate$)$
      $w_{D_Y} \leftarrow$ UpWDisc$(y, \hat{y}$, l_rate$)$
      $UpdateGenerators$
      $\hat{y} \leftarrow$ GenImage(x, w_{G_X})
      $\hat{x} \leftarrow$ GenImage(y, w_{G_Y})
      $\mathcal{L}_{\text{adv}}(G_X) \leftarrow$ LossAdvGen$(\hat{y}, w_{D_Y})$
      $\mathcal{L}_{\text{adv}}(G_Y) \leftarrow$ LossAdvGen$(\hat{x}, w_{D_X})$
      $\mathcal{L}_{\text{ciclo}}(G_X, G_Y) \leftarrow$ LossCycle$(x, \hat{x}, y, \hat{y})$
      $\mathcal{L}_{\text{idt}}(G_X, G_Y) \leftarrow$ LossIdt$(x, \hat{x}, y, \hat{y})$
      $w_{G_X} \leftarrow$ UpWGen$(\mathcal{L}_{\text{adv}}(G_X), \mathcal{L}_{\text{cycle}}(G_X, G_Y),$
$\mathcal{L}_{\text{idt}}(G_X, G_Y)$, l_rate$)$
      $w_{G_Y} \leftarrow$ UpWGen$(\mathcal{L}_{\text{adv}}(G_Y), \mathcal{L}_{\text{cycle}}(G_X, G_Y),$
$\mathcal{L}_{\text{idt}}(G_X, G_Y)$, l_rate$)$
    end for
  end for
  return w_{G_X}, w_{G_Y}
end procedure

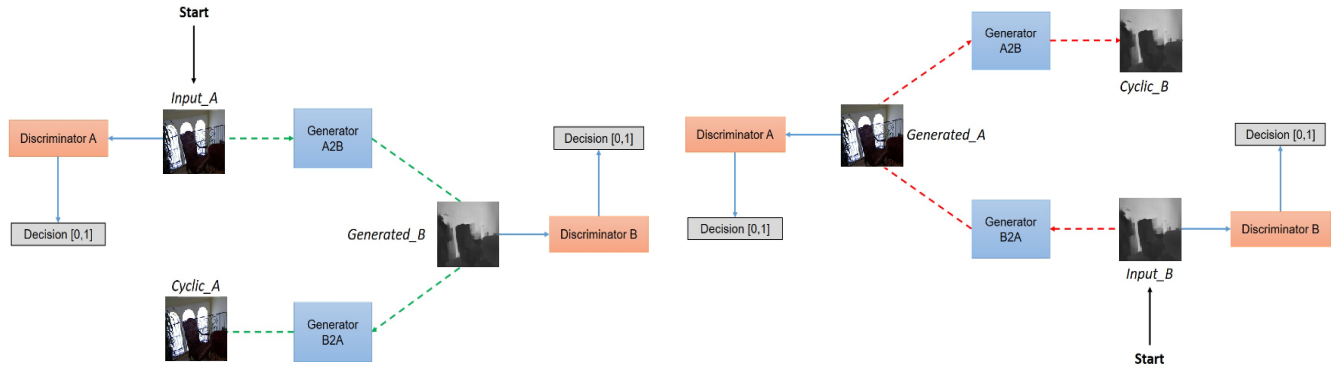
O modelo é treinado de maneira conjunta, otimizando as funções de perda adversarial para os discriminadores e as funções de perda ciclo para os geradores. A otimização visa encontrar pesos que minimizem a discrepância entre as distribuições das imagens nos dois domínios.

Após o treinamento, os geradores G e F podem ser usados para traduzir imagens entre os dois domínios sem a necessidade de correspondências diretas nos conjuntos de dados. Isso permite, por exemplo, a tradução de imagens de cenas em RGB para imagens de mapas de profundidade em níveis de cinza e vice-versa, mesmo na ausência de pares específicos de treinamento correspondentes.

As Figs. 2a e 2b mostram que cada discriminador recebe duas entradas: uma é a imagem original do respectivo domínio, e a outra é a imagem gerada pelo gerador correspondente. O objetivo do discriminador é distinguir entre essas duas imagens, desafiando o gerador e rejeitando as imagens geradas. Por outro lado, o gerador deseja gerar imagens que sejam suficientemente semelhantes às imagens originais para serem aceitas pelo discriminador na classe D_B [21].

O treinamento da CycleGAN descrito no Algoritmo [2] é o seguinte:

Início do Procedimento: o procedimento é iniciado, levando como entrada os conjuntos de dados X e Y, o número de épocas de treinamento (n_epoch) e a taxa de aprendizado



(a) (b)

Fig. 2. Arquitetura da rede CycleGAN. Em (a) o gerador A2B recebe a imagem RGB e gera o mapa estimado, e o gerador B2A reconstrói a imagem de entrada. Em (b) o gerador B2A tem como entrada o mapa desejado e gera a imagem RGB, e o gerador A2B reconstrói a imagem de entrada. Fonte: adaptado de [21].

(l_rate).

Carregar Pesos dos Geradores e Discriminadores: os pesos iniciais para os geradores (wGX, wGY) e discriminadores (wDX, wDY) são carregados. Esses pesos são os parâmetros ajustáveis da rede que serão otimizados durante o treinamento.

Número de Épocas de Treinamento: define-se o número total de épocas de treinamento (n_epoch), que representa quantas vezes o modelo percorrerá todo o conjunto de treinamento.

Laço de Treinamento por Época: inicia-se um laço de treinamento que percorre cada época, do 1 até o número total de épocas definido.

Embaralhar Dados de Treinamento: os conjuntos de dados de entrada e saída condicionada (X e Y) são embaralhados, garantindo variedade nas amostras de treinamento a cada época.

Laço de Treinamento para Pares de Entrada e Saída: inicia-se um segundo laço que percorre cada par de entrada e saída condicionada (x, y) nos conjuntos X e Y, respectivamente.

Atualizar Discriminadores (UpdateDiscriminators): realiza-se uma etapa para atualizar os pesos dos discriminadores (wDX, wDY) usando as perdas adversariais (Ladv) entre as imagens reais e as imagens geradas pelos geradores.

Gerar Imagens Condicionadas pelos Geradores (GenImage): utilizando os geradores (GenImage) e seus pesos atuais (wGX, wGY), geram-se imagens condicionadas $\hat{x}$ e $\hat{y}$ a partir das imagens originais (x e y).

Calcular Perda Adversarial para Discriminadores (Ladv(DX), Ladv(DY)): calcula-se as perdas adversariais para os discriminadores (Ladv(DX), Ladv(DY)) comparando as imagens reais (x e y) com as imagens geradas $\hat{x}$ e $\hat{y}$ usando os pesos atuais dos discriminadores (wDX, wDY).

Atualizar Pesos dos Discriminadores (UpWDisc): utiliza-se as perdas adversariais para os discriminadores para atualizar seus pesos (wDX, wDY) usando uma técnica de otimização.

Atualizar Geradores (UpdateGenerators): realiza-se uma etapa para atualizar os pesos dos geradores (wGX, wGY) usando as perdas adversariais, de ciclo (Lciclo) e de identidade (Lidt).

Calcular Perdas para Geradores (Ladv(GX), Ladv(GY), Lciclo(GX, GY), Lidt(GX, GY)): calcula-se as perdas adversariais (Ladv(GX), Ladv(GY)), de ciclo (Lciclo(GX, GY)) e de identidade (Lidt(GX, GY)) comparando as imagens geradas $\hat{y}$, $\hat{x}$ com as imagens reais (y, x) usando os pesos atuais dos discriminadores (wDY, wDX).

Atualizar Pesos dos Geradores (UpWGen): utiliza-se as perdas calculadas para os geradores para atualizar seus pesos (wGX, wGY) usando uma técnica de otimização.

Final do Laço de Treinamento para Pares de Entrada e Saída: o laço interno de treinamento é concluído.

Final do Laço de Treinamento por Época: o laço externo de treinamento por época é concluído.

Retornar Pesos dos Geradores Atualizados: após o treinamento completo, os pesos finais atualizados dos geradores (wGX, wGY) são retornados.

Fim do Procedimento: o procedimento de treinamento CycleGAN é concluído.

IV. Experimentos Realizados

Nesta seção, serão apresentados os experimentos realizados com as redes pix2pix e CycleGAN para a estimação de mapas de profundidade.

A. *Parâmetros de Treinamento Usados para a Rede pix2pix*

Os principais parâmetros de configuração usados para o treinamento da rede pix2pix foram: tamanho do lote (batch_size): 1, termo de momento de Adam (beta1): 0,5. As imagens usadas no treinamento foram redimensionadas para 256x256 pixels, com tamanho de recorte (crop_size): 256. Em dataset_mode é definido como os conjuntos de dados são carregados, entre as opções [unaligned | aligned | single | colorization]. Para este experimento, a opção selecionada foi dataset_mode: aligned, conforme orientado em [10]. A opção direction: AtoB define a direção da tradução da imagem. No caso deste experimento, as imagens das cenas estão na pasta

"A" e deseja-se traduzi-las para as imagens contendo o mapa de profundidade na pasta "B". Como a pix2pix não usa buffer de imagem, o objetivo do treinamento é: perda da GAN + lambda_L1 * ||G(A)-B||_1. Por padrão, é usada a perda da GAN vanilla, gan_mode: vanilla, dentre as opções [vanilla | lsgan | wgangp]. O peso da perda L1 é lambda_L1: 100,0. A taxa de aprendizado inicial para Adam foi definida como lr: 0,0002. A política da taxa de aprendizagem foi definida como lr_policy: linear, dentre as seguintes opções [linear | step | plateau | cosine]. O número de épocas com a taxa de aprendizagem inicial foi definido como n_epochs: 100 e o número de épocas para reduzir linearmente a taxa de aprendizagem para zero foi definido como n_epochs_decay: 100. A arquitetura do discriminador foi definida como netD: basic, dentre as opções [basic | n_layers | pixel]. O modelo basic é uma PatchGAN 70x70, e n_layers permite que o usuário especifique as camadas no discriminador. Para a arquitetura do gerador, foi selecionada a opção netG: unet_256, dentre as opções [unet_256 | unet_128].

B. Parâmetros de Treinamento Usados para a Rede CycleGAN

Os principais parâmetros de configuração uados para o treinamento da rede CycleGAN foram: tamanho do lote (batch_size): 1, termo de momento de Adam (beta1): 0,5. As imagens usadas no treinamento foram redimensionadas para 256x256 pixels, com tamanho de recorte (crop_size): 256. A opção dataset_mode define como os conjuntos de dados são carregados, entre as opções [unaligned | aligned | single | colorization]. Para este experimento, a opção selecionada foi dataset_mode: unaligned. A opção direction: AtoB define a direção da tradução da imagem. No caso deste experimento, as imagens das cenas estão na pasta "trainA" e deseja-se traduzi-las para as imagens contendo o mapa de profundidade na pasta "trainB". Para a CycleGAN, são introduzidas as perdas lambda_A, lambda_B e lambda_identity para as seguintes perdas: A (domínio de origem), B (domínio de destino). Perda do ciclo direto: lambda_A * ||G_B(G_A(A)) - A||. Perda do ciclo reverso: lambda_B * ||G_A(G_B(B)) - B|| e Perda de identidade (opcional): lambda_identity * (||G_A(B) - B|| * lambda_B + ||G_B(A) - A|| * lambda_A), (Eq. (2) do artigo [22]). Por padrão, é usada a perda de GAN lsgan, gan_mode: lsgan, dentre as opções [vanilla | lsgan | wgangp]. Aqui, o peso da perda L1 é lambda_A: 10,0, lambda_B: 10,0 e lambda_identity: 0,5. A taxa de aprendizado inicial para Adam foi definida como lr: 0,0002. A política da taxa de aprendizagem foi definida como lr_policy: linear, dentre as seguintes opções [linear | step | plateau | cosine]. O número de épocas com a taxa de aprendizagem inicial foi definido como n_epochs: 100 e o número de épocas para reduzir linearmente a taxa de aprendizagem para zero foi definido como n_epochs_decay: 100. A arquitetura do discriminador foi definida como netD: basic, dentre as opções [basic | n_layers | pixel]. O modelo basic é um PatchGAN 70x70, e n_layers permite que o usuário especifique as camadas no discriminador. Para a arquitetura do gerador, foi selecionada a opção netG: resnet_9blocks, dentre as opções [resnet_9blocks | resnet_6blocks].

C. Conjunto de Dados Utilizados no Treinamento e Teste

As imagens utilizadas no treinamento das redes pix2pix e CycleGAN foram obtidas da base de dados NYU-Depth V2 [23]. O conjunto de dados NYU-Depth V2 é composto por sequências de vídeo em diversos ambientes internos, registradas pelas câmeras RGB e de profundidade do Microsoft Kinect, e possui um total de 120 mil imagens RGB e os respectivos mapas deprofundidade.

Para o processo de treinamento, foi selecionado um conjunto aleatório de imagens de cada classe, totalizando 7,031 imagens para treinamento e 416 imagens para testes. As imagens foram divididas em duas subpastas, sendo uma destinada às imagens reais da cena capturadas pela câmera RGB, e a outra contendo as respectivas imagens dos mapas de profundidade correspondentes às imagens capturadas pela câmera RGB. Ambos os conjuntos possuem os mesmos rótulos para as imagens, conforme recomendado em [12] e [22].

V. Resultados Obtidos e Estudos de Ablação

O principal objetivo foi fazer a comparação qualitativa entre os mapas gerados pela pix2pix e a CycleGAN. Os resultados dos testes realizados com a pix2pix estão disponíveis na Tabela I, que apresenta as perdas da pix2pix. Nessa tabela, G_GAN representa a perda do gerador, G_L1 representa a perda L1, D_Real representa a perda do discriminador para as imagens reais, e D_Falsa representa a perda do discriminador para as imagens geradas (mapas de profundidade).

A Fig. 3 apresenta os resultados obtidos pelos métodos utilizados neste trabalho. Foram selecionadas imagens aleatórias para comparação, não necessariamente os melhores resultados obtidos, e o conjunto de testes é composto por 416 imagens. Na figura, a imagem obtida pela câmera RGB está representada em (a), o *Ground Truth* disponível na base NYU-Depth V2 é representado por (b), a imagem do mapa estimada pela CycleGAN é representada em (c), e a estimada pela pix2pix é representada em (d).

Fazendo uma análise qualitativa das imagens, observa-se que a CycleGAN produziu melhores resultados que a pix2pix na estimativa de mapas de profundidade. Mesmo sendo treinada com 100 épocas, enquanto a pix2pix fora treinada com 200 épocas sobre a mesma base de dados. Porém, devido a sua arquitetura a CycleGAN demanda maior processamento e tempo ao ser treinada em comparação à pix2pix.

Os melhores resultados foram obtidos no 'Test 1', produzidos com as seguintes configurações: Generator=unet_256, GAN_mode=vanilla, lr_rate_policy=linear, e net_initialization=normal. Todos os outros parâmetros de treinamento configurados estão detalhados na Seção IV, Subseção IV-A.

Na segunda bateria de testes, 'Test 2', as principais configurações foram: Generator=unet_128, GAN_mode=vanilla, lr_rate_policy=plateau, e net_initialization=xavier.

E para a terceira rodada de testes, denominada 'Test 3', as principais configurações foram: Generator=unet_256, GAN_mode=lsgan, lr_rate_policy=step e net_initialization=kaiming.

O resultado final da geração do mapa de profundidade pela pix2pix e pela CycleGAN podem ser observados na Fig. 3, comparados aos respectivos *ground truth*.

Os resultados obtidos com a CycleGAN para a estimação dos mapas de profundidade estão disponíveis na Tabela II, onde as colunas Cycle_A e Cycle_B representam os resultados da perda L1. Os melhores resultados obtidos estão destacados no 'Test 1', produzidos com as seguintes configurações: Generator=resnet_9blocks, GAN_mode=lsgan, lr_rate_policy=linear, e net_initialization=normal.

Na segunda etapa de testes, denominada 'Test 2', as principais configurações utilizadas foram as seguintes: Generator=resnet_6blocks, GAN_mode=vanilla, lr_rate_policy=plateau, e net_initialization=xavier.

E por fim, na terceira rodada de testes, denominada 'Test 3', a CycleGAN foi configurada com os seguintes parâmetros: Generator=resnet_9blocks, GAN_mode=lsgan, lr_rate_policy=step, e net_inicialization=kaiming.

Os tempos de treinamento da pix2pix para a geração dos mapas de profundidade para o conjunto de dados mencionado em IV-C foram de aproximadamente 24 horas para 200 épocas. Já para a CycleGAN, esse tempo foi de aproximadamente 96 horas para 100 épocas de treinamento em um PC com processador Intel i7 7700HQ, 16GB de memória RAM e placa gráfica Nvidia GTX 1080 com 8gb de VRAM.

A Tabela III apresenta os resultados da perda L1 para a pix2pix e a CycleGAN, comparados aos modelos DepthNet_Resnet50 e DepthNet_Resnet18. Esses modelos são variações da MonoDepth2 [4] e da DenseDepth [24], utilizando a ResNet50 e a ResNet18 como codificadores. Eles foram implementados com o objetivo de fazer comparações com o método proposto neste trabalho.

Para a estimativa da profundidade em metros dos mapas gerados pela pix2pix e CycleGAN, foi utilizado o modelo GLPN fine-tuned on NYUv2, conforme descrito em [25]. O modelo pode ser encontrado em [26]. O *pipeline* retorna um dicionário com duas entradas. Uma delas é chamada de "profundidade prevista", que consiste em um tensor contendo os valores de profundidade em metros para cada pixel. Exemplos dessas profundidades em metros para mapas gerados pela pix2pix e pela CycleGAN podem ser observados nas tabelas IV e V, respectivamente. Verificou-se que a CycleGAN foi capaz de gerar mapas mais confiáveis em comparação com a pix2pix e a DepthNet_ResNet50, com uma perda L1 aproximadamente 2,5 vezes menor que a pix2pix, cerca de 2,4 vezes menor que a DepthNet_ResNet50 e aproximadamente 14 vezes menor que a DepthNet_ResNet18.

TABELA I
PERDAS DA REDE PIX2PIX NA GERAÇÃO DE MAPAS DE PROFUNDIDADE

Média perda pix2pix				
	G_GAN	G_L1	D_Real	D_Falsa
Test 1	7,4346	1,3791	0,0720	0,0706
Test 2	1,6901	2,7822	0,3325	0,3107
Test 3	2,8852	1,9262	0,1815	0,2880

TABELA II
PERDAS DA REDE CYCLEGAN NA GERAÇÃO DE MAPAS DE PROFUNDIDADE

Média perda CycleGAN						
	G_A	G_B	Cycle_A	D_A	D_B	Cycle_B
Test 1	0,0824	0,6911	1.1282	0,0826	0,6928	0.7936
Test 2	0,5366	1,0726	1,3590	0,5919	0,8801	0.7755
Test 3	0,1138	0,7319	1.3433	0,1044	0,7362	0.9363

TABELA III
MÉDIA DAS PERDAS DAS REDES CYCLEGAN E PIX2PIX COMPARADAS COM A DEPTHNET

Média Perda L1 para classes de objetos do NYUv2				
Classes	Basement	Office	Printer	Study_room
CycleGAN	0,0183	0,0825	0,0524	0,0947
Pix2Pix	0,1021	0,1954	0,1578	0,1885
DepthNet_ResNet50	0,2324	0,1778	0,1486	0,1167
DepthNet_ResNet18	1,0942	1,0920	0,8664	0,7557

TABELA IV
PROFUNDIDADES EXPRESSAS EM METROS/PIXEL DE UM MAPA GERADO COM A PIX2PIX

pix2pix: imagem basement_0001a_out_1_fake.png						
3,9574	3,6548	3,8026	...	3,8800	4,0018	4,2734
2,7726	2,2706	2,1473	...	2,2810	2,4023	2,8460
2,8320	2,7123	2,7083	...	2,8483	2,8988	2,8324
			...			
1,3512	1,3721	1,3699	...	1,6469	1,6045	1,6023
1,3910	1,3158	1,3813	...	1,6264	1,6804	1,5753
1,4817	1.4258	1,3231	...	1,6163	1,6166	1,7077

TABELA V
PROFUNDIDADES EXPRESSAS EM METROS/PIXEL DE UM MAPA GERADO COM A CYCLEGAN

CycleGAN: imagem basement_0001a_out_1_rec_B.png						
3,9769	3,8597	3,9669	...	3,7247	3,8243	4,2347
3,0098	2,5377	2,3453	...	2,1812	2,2881	2,8232
3,2025	3,1834	3,0880	...	2,7151	2,7877	2,9051
			...			
2,5631	2,5518	2,5531	...	1,7281	1,7286	1,7461
2,6307	2,5359	2,5820	...	1,7282	1,8107	1,7049
2.7068	2,6468	2.5422	...	1,7305	1,7311	1,8132

VI. CONCLUSÕES

A estimativa automática de mapas de profundidade em ambientes internos, utilizando técnicas de *Deep Learning* a partir de imagens únicas, apresenta um potencial significativo para auxiliar deficientes visuais na navegação. Os resultados demonstraram que a CycleGAN proposta em [22], é capaz de gerar mapas de profundidade mais confiáveis em comparação à pix2pix [12] e DepthNet_ResNet50. Essa melhoria na precisão dos mapas de profundidade é especialmente relevante para aprimorar a percepção espacial e a segurança dos sistemas de auxílio à navegação. No entanto, é importante ressaltar que um dos desafios enfrentados nessa abordagem foi o longo tempo de treinamento. Apesar disso, a utilização de GANs na geração de mapas de profundidade mostra-se como uma solução promissora na estimativa de mapas de profundidade.

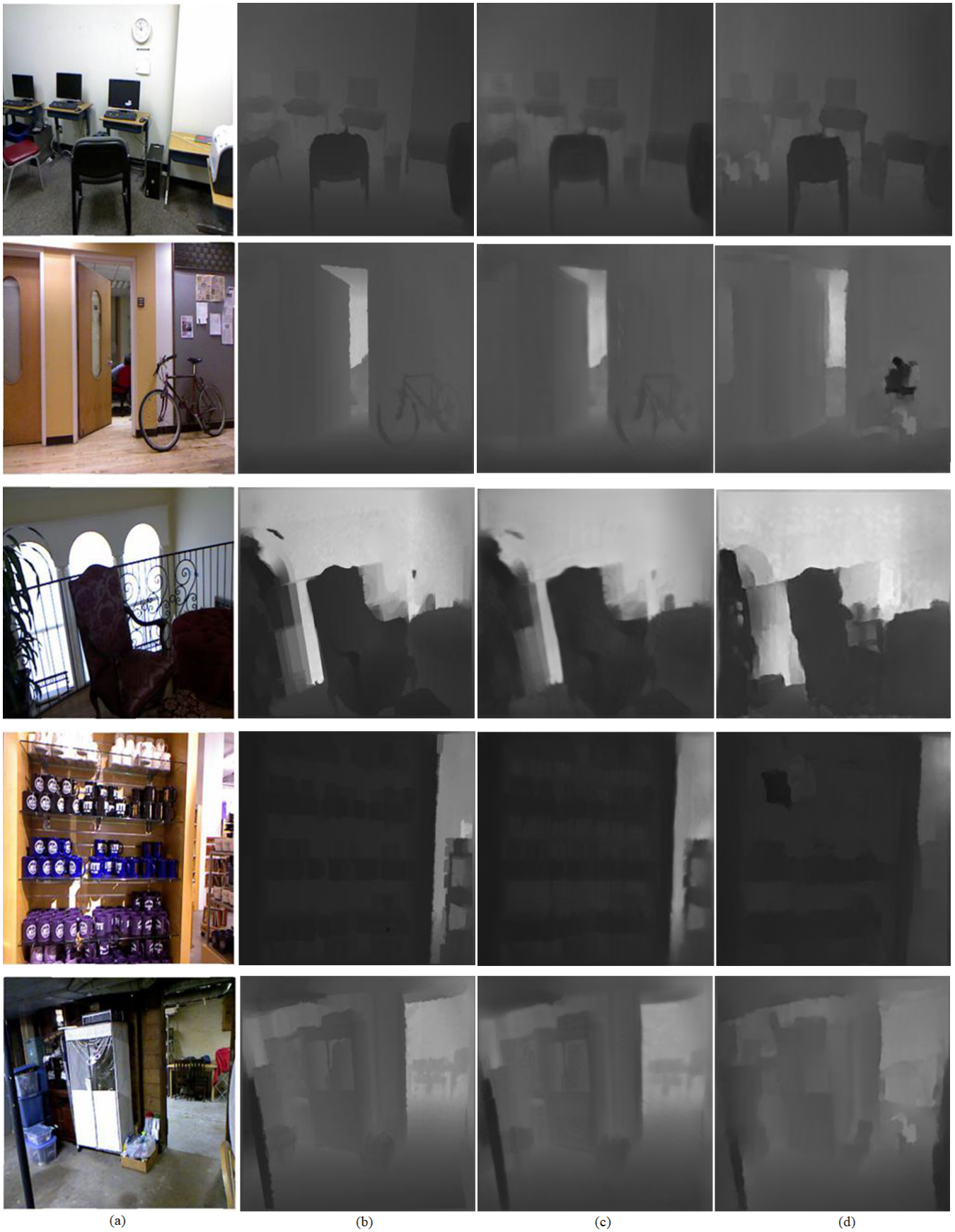


Fig. 3. Conjunto de Imagens resultantes pelos métodos estudados neste trabalho. (a) Imagem da cena RGB, (b) Ground Truth, (c) CycleGAN e (d) pix2pix.

REFERENCES

[1] S. R. N. P. N. Zaman, "Single-image stereo depth estimation using gans," aug 2023. Accessed: Aug. 09, 2023. [Online] Available: https://sharanramjee.github.io/files/projects/cs231a.pdf.

[2] K. G. Lore, K. Reddy, M. Giering, and E. A. Bernal, "Generative adversarial networks for depth map estimation from rgb video," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 1258–12588, IEEE, 2018.

[3] R. Groenendijk, S. Karaoglu, T. Gevers, and T. Mensink, "On the benefit of adversarial training for monocular depth estimation," *Computer Vision and Image Understanding*, vol. 190, p. 102848, 2020.

[4] C. Godard, O. Mac Aodha, M. Firman, and G. J. Brostow, "Digging into self-supervised monocular depth estimation," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 3828–3838, 2019.

[5] H. Jiang, G. Larsson, M. M. G. Shakhnarovich, and E. Learned-Miller, "Self-supervised relative depth learning for urban scene understanding," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 19–35, 2018.

[6] A. Saxena, M. Sun, and A. Y. Ng, "Make3d: Learning 3d scene structure from a single still image," *IEEE transactions on pattern analysis and machine intelligence*, vol. 31, no. 5, pp. 824–840, 2008.

[7] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," in *Advances in Neural Information Processing Systems* (Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, eds.), vol. 27, Curran Associates, Inc., 2014.

[8] C. Godard, O. Mac Aodha, and G. J. Brostow, "Unsupervised monocular depth estimation with left-right consistency," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 270–279, 2017.

[9] A. Saxena, S. Chung, and A. Ng, "Learning depth from single monocular images," *Advances in neural information processing systems*, vol. 18, 2005.

[10] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets in advances in neural information processing systems (nips)," *Curran Associates, Inc. Red Hook, NY, USA*, pp. 2672–2680, 2014.

[11] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[12] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1125–1134, 2017.

[13] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4401–4410, 2019.

[14] M. Ghafoorian, C. Nugteren, N. Baka, O. Booij, and M. Hofmann, "El-gan: Embedding loss driven generative adversarial networks for lane detection," in *proceedings of the european conference on computer vision (ECCV) Workshops*, pp. 0–0, 2018.

[15] Y. Galama and T. Mensink, "Iterative gans for rotating visual objects,," aug 2018. Accessed: Aug. 09, 2023. [Online] Available: https://openreview.net/forum?id=HJ7rdGkPz.

[16] A. Pilzer, D. Xu, M. Puscas, E. Ricci, and N. Sebe, "Unsupervised adversarial depth estimation using cycled generative networks," in *2018 international conference on 3D vision (3DV)*, pp. 587–595, IEEE, 2018.

[17] D.-h. Kwak and S.-h. Lee, "A novel method for estimating monocular depth using cycle gan and segmentation," *Sensors*, vol. 20, no. 9, p. 2567, 2020.

[18] X. K. T. Shimada, H. Nishikawa and H. Tomiyama, "Pix2pix-based monocular depth estimation for drones with optical flow on airsim," *Sensors*, vol. 22, no. 6, p. 2097, 2022.

[19] S. M. P. Hambarde and A. Dhall, "Uw-gan: Single-image depth estimation and image enhancement for underwater images," *IEEE Transactions on Instrumentation and Measurement*, vol. 70, pp. 1–12, 2021.

[20] C. Hesse, "Iterative gans for rotating visual objects,," aug 2017. Accessed: Aug. 09, 2023. [Online] Available: https://affinelayer.com/pix2pix/.

[21] H. Bansal and A. Rathore, "Understanding and implementing cyclegan in tensorflow,," aug 2017. Accessed: Aug. 09, 2023. [Online] Available: https://hardikbansal.github.io/CycleGANBlog/.

[22] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, pp. 2223–2232, 2017.

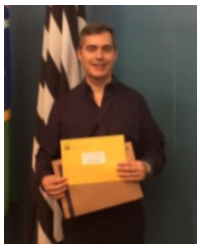
[23] P. K. Nathan Silberman, Derek Hoiem and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *ECCV*, 2012.

[24] I. Alhashim and P. Wonka, "High quality monocular depth estimation via transfer learning," *arXiv preprint arXiv:1812.11941*, 2018.

[25] D. Kim, W. Ka, P. Ahn, D. Joo, S. Chun, and J. Kim, "Global-local path networks for monocular depth estimation with vertical cutdepth," *arXiv preprint arXiv:2201.07436*, 2022.

[26] D. Kim, W. Ka, P. Ahn, D. Joo, S. Chun, and . Kim, Junmo, "Global-local path networks for monocular depth estimation with vertical cutdepth," 2022. Acessado em: Junho de 2023.

Ricardo Salvino Casado obtained his master's degree from the University of São Paulo, São Carlos campus (EESC/USP). He is an RDE professor at the Federal Institute of São Paulo. Currently, he is pursuing a Ph.D. in Computer Science at the Federal University of São Carlos. His areas of interest include computer vision, deep learning, monodepth estimation, and genetic programming. (e-mail: rscasado@ifsp.edu.br).



Emerson Carlos Pedrino is an Associate Professor (Ph.D.) in the Department of Computer Science at the Federal University of São Carlos. He holds a degree in Electrical Engineering from the University of São Paulo and a Bachelor's degree in Computational Physics, both from the University of São Paulo - EESC (2016) and IFSC (2000). He earned a Master's degree in Electrical Engineering from the University of São Paulo - EESC (2003) and a Ph.D. in Electrical Engineering from the University of São Paulo - EESC (2008). Additionally, he completed a Post-doctorate in Electronic Engineering (as a Visiting Professor) at the Department of Electronic Engineering, University of York, England, with research funding provided by FAPESP (2018-2019). He continues to collaborate in the development of hardware applications involving intelligent systems. (e-mail: emerson@dc.ufscar.br).