Universidade Federal do ABC
Centro de Engenharia, Modelagem e Ciências Sociais Aplicadas
Trabalho de Graduação em Engenharia de Informação

Estudo de Redes Neurais Convolucionais aplicadas ao *matching* de uma imagem estéreo

João Pedro Poloni Ponce

Santo André

Maio de 2020

João Pedro Poloni Ponce

Estudo de Redes Neurais Convolucionais aplicadas ao *matching* de uma imagem estéreo

Trabalho de Graduação apresentado ao concluir a Graduação em Engenharia de Informação, como parte dos requisitos necessários para a obtenção do Título Bacharel em Engenharia de Informação.

Universidade Federal do ABC

Orientador: Ricardo Suyama

Santo André

Maio de 2020

Sistema de Bibliotecas da Universidade Federal do ABC
Elaborada pelo Sistema de Geração de Ficha Catalográfica da UFABC
com os dados fornecidos pelo(a) autor(a).

Poloni Ponce, João Pedro
Estudo de Redes Neurais Convolucionais aplicadas ao matching de uma imagem estéreo / João Pedro Poloni Ponce. — 2020.

59 fls.

Orientador: Ricardo Suyama

Trabalho de Conclusão de Curso — Universidade Federal do ABC, Bacharelado em Engenharia de Informação, Santo André, 2020.

1. Aprendizado de Máquina. 2. Processamento de Imagens. 3. Imagens Estéreo. 4. Redes Neurais Convolucionais. I. Suyama, Ricardo. II. Bacharelado em Engenharia de Informação, 2020. III. Título.

Aos meus Familiares, Professores e Amigos, pois tudo que sou é graças a eles e sem eles não seria possível chegar até aqui.

Agradecimentos

Primeiramente gostaria de agradecer a UFABC por fornecer uma formação acadêmica de excelência, voltada a descoberta e produção científica, promover uma visão crítica e multidisciplinar da Natureza e da Humanidade, que certamente agregou muito aos meus princípios pessoais, ser um ambiente de livre pensamento e divulgação de ideias, e pela amizades que levarei para vida graças a ela. O meu eterno agradecimento a instituição, seus funcionários e a todas as pessoas que fazem dela esse lugar único.

Um agradecimento especial ao meu orientador, Professor Doutor Ricardo Suyama, que durante o período do projeto me guiou e fez toda a diferença. Sempre solicito e compreensível para atender as necessidades do projeto.

Agradeço aos meus Familiares e Amigos por todo o suporte recebido durante a execução do projeto e pelo carinho recebido por eles, que proporcionaram cometários valorosos e momentos de lazer importantes para a execução do trabalho. Sem vocês eu não seria metade do que sou.

Resumo

Imagens estéreo são imagens formadas a partir de duas ou mais fontes que captam a mesma cena de forma que é possível inferir a profundidade da cena em analise. O uso de redes neurais convolucionais para computar essas imagens tem se mostrado como uma alternativa viável devido a sua rapidez em encontrar a correspondência entre as imagens. Tal fato levanta questões relacionadas a influência dos parâmetros estruturais, como tamanho do *kernel*, *stride* e política de amostragem no desempenho da rede neural. Para tanto este trabalhou buscou reproduzir um artigo que trata sobre o tema e explorar a influência dos parâmetros citados acima em função dos resultados de taxa de erro e perdas do modelo neural. Os resultados obtidos revelam melhora na alteração de alguns valores e piora em outros, chegando ao ponto de se mostrarem prejudicial ao modelo final. Foi notável também a influencia dos parâmetros sobre o tempo de treinamento dos modelos, mesmo fazendo uso de *GPU*, a diferença temporal no período de treino entre os limites máximos e mínimos chegou a uma razão de seis vezes.

Palavra Chave: Imagem estéreo, CNN, correspondência estéreo.

Abstract

Stereo images are images formed from two or more sources that capture the same scene so that it is possible to infer the depth of the scene under analysis. The use of convolutional neural networks to compute these images has been shown to be a viable alternative due to its speed in finding the correspondence between the images. This raises questions related to the influence of structural parameters, such as size of *kernel*, *stride* and pooling policy on the performance of the neural network. To this end, this work sought to reproduce an article that deals with the topic and to explore the influence of the parameters mentioned above in function of the results of error rate and losses of the neural model. The results obtained reveal an improvement in the alteration of some values and a worsening in others, reaching the point of being harmful to the final model. The influence of the parameters on the training time of the models was also notable, even using GPU, the temporal difference in the training period between the maximum and minimum limits reached a ratio of six times.

Keywords: Stereo images, CNN, stereo matching.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ANN	*Atificial Neural Network*
CNN	*Convolutional Neural Network*
DSI	*Disparity Spactial Image*
GPU	*Graphical Processor Unit*
LIDAR	*Light Detection And Ranging*
ReLu	*Rectified Linear Unit*

Sumário

1 Introdução

Redes Neurais Artificiais podem ser consideradas uma das principais técnicas de aprendizagem de máquina, e têm sido empregadas em diferentes problemas relacionados à classificação e regressão a partir de um grande volume de dados.

O grande interesse de pesquisa observado nos últimos anos se justifica pelos resultados expressivos obtidos com tais técnicas, que só foram possíveis com a disponibilidade de novas plataformas de *hardware*, que se tornaram mais potentes do ponto de vista computacional, e de avanços de *software*, que se tornaram mais eficientes e simplificaram a implementação das redes.

Há diferentes arquiteturas de redes já propostas na literatura, dentre as quais é possível destacar a estrutura clássica do *Perceptron* multicamadas [4], e as propostas mais recentes de redes profundas e redes convolucionais [2].

A escolha por um tipo em particular de estrutura é bastante dependente do problema a ser solucionado. Por exemplo, no problema de classificação de imagens, uma rede neural do tipo *Perceptron* multicamadas pode demandar muito tempo e dados de treinamento para obter um modelo relativamente bom e, em geral, será grande do ponto de vista estrutural, ou seja, com um alto número de neurônios e camadas. Por outra lado, uma rede neural convolucional, que faz uso da operação matemática de convolução para exaltar características do dado a se classificar, tende a apresentar uma estrutura mais enxuta, com menos neurônios, que em geral, resulta em um modelo mais robusto e computacionalmente menos custoso.

As redes neurais convolucionais têm sido amplamente empregadas em tarefas relacionadas a Visão Computacional pois são capazes de processar informação visual permitindo que a máquina onde o software rode reconheça padrões [5]. Por exemplo, a tarefa de classificação de imagens tem grande número de aplicações dentro dos chamados carros autônomos, sendo capaz de fornecer dados valorosos no apoio a tomada de decisão. A Figura 1 ilustra um tipo de topologia de rede neural convolucional que pode ser utilizada nesta tarefa.

Ser capaz identificar o objeto que se encontra no caminho do veículo é fundamental para o avanço desta tecnologia, permitido que decisões adequadas sejam tomadas dependendo da natureza do objeto à frente do veículo. No entanto, em geral, a partir de uma única imagem, é bastante difícil estimar a qual distância determinado objeto está do veículo, o que demanda uma análise da profundidade da cena em análise.

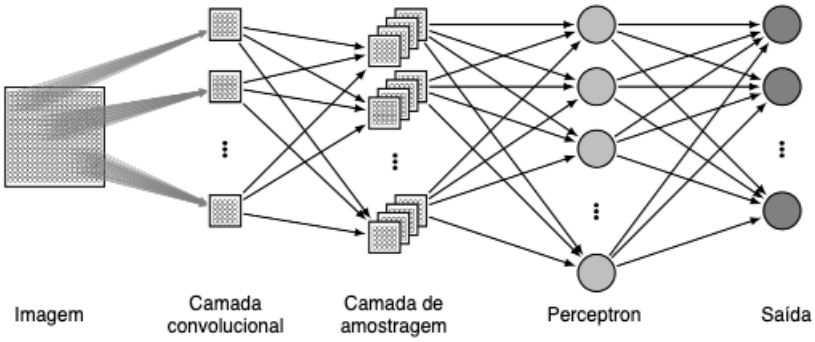

Figura 1 – Exemplo de topologia de rede neural convolucional. Onde a imagem segmentada serve como entrada da camada convolucional, responsável por extrair e exaltar as características da imagem, sua saída funciona como entrada da camada de amostragem, responsável por selecionar uma característica da imagem de acordo com uma política, e em seguida um Perceptron comum com sua saída.

1.1 Estimação de Profundidade a partir de Imagens Estéreo - Correspondência Estéreo

Existem diferentes dispositivos que podem ser utilizados para prover a informação sobre a distância dos obstáculos – como sensores ultrassônicos e dispositivos LIDAR (*Light Detection And Ranging*). Entretanto, é possível obter tal informação explorando técnicas para processamento de *Imagem Estéreo*, como é mostrado em [2].

Essencialmente, considerando que tenhamos acesso a duas imagens de uma mesma cena, obtidas a partir de dois pontos de vista ligeiramente diferentes, as técnicas buscam estimar a profundidade de cada elemento em cena gerando um *Mapas de Profundidade* [2]. Esses mapas são imagens onde cores são usadas para prover a percepção da profundidade por meio da intensidade da cor – regiões mais escuras no mapa indicam que um determinado objeto está mais distante e cores mais claras que indicam que o objeto está mais próximo, conforme mostra a figura 2.

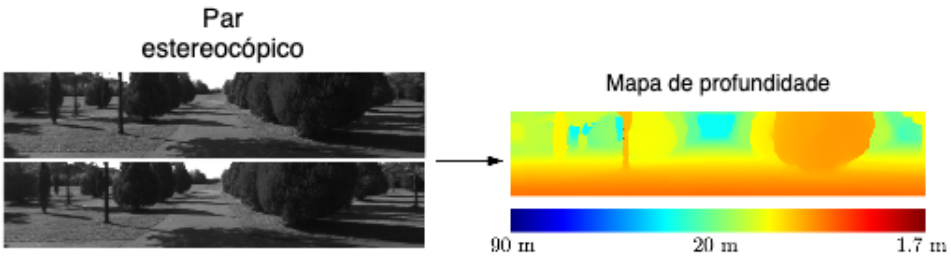

Figura 2 – Par estereoscópico e seu respectivo mapa de profundidade.

Os algoritmos para processamento de imagens estéreo que se encontram no estado da arte, apesar de fornecerem um excelente resultado são custosos do ponto de vista computacional, ou seja, consomem muitos recursos da máquina e costumam levar um

tempo considerável para finalizar a tarefa, mesmo fazendo uso de GPU [3]. Uma das parte mais custosas desses algoritmos se refere à tarefa de encontrar *pixels* correspondentes no par de imagens estereoscópicas. A tarefa consiste, basicamente, em selecionar um *pixel* de uma imagem que, por exemplo, corresponde ao vértice de um cubo presente na imagem, e percorrer a outra imagem a fim de encontrar qual *pixel* corresponde ao mesmo vértice do cubo presente na primeira imagem. Isto é o chamado *correspondente estéreo*, ou seja, trata-se de um processo lento que percorre todos os *pixels* da imagem até encontrar seu similar na outra imagem do par.

Os métodos que lidam com a formação de imagens estéreo em geral são determinísticos, ou seja, eles buscam ser exatos na busca e extração dos *pixels* para formação da imagem estéreo. Nesse sentido, o aumento da capacidade computacional do hardware envolvido leva a uma redução no tempo de computação necessário. Contudo, é possível que ganhos significativos sejam obtidos ao considerar uma nova abordagem para o método de correspondência estéreo.

1.2 Outras Abordagens para Correspondência Estéreo

Uma abordagem probabilística do problema passa por entender quais variáveis são importantes ao tratar sobre o tema, dentro do escopo de imagens estéreo. Uma alternativa é a obtenção de filtros que sejam capazes de realçar os *pixels*, fazendo com que a correlação dos dados estereoscópicos seja maior, e consequentemente sua identificação posterior seja mais rápida, ou seja, ao tratar o problema de modo probabilístico o que se busca é a capacidade de prever o mapa de disparidade, diminuir as perdas da imagens através de regularizações melhores da imagem e encontrar as melhores características da imagem [6, 7, 8].

Outra abordagem que tem se mostrado promissora nos últimos anos é a criação de modelos de aprendizado de máquina que sejam capazes encontrar um padrão dentro dos dados utilizados, fornecendo assim uma maneira rápida de avaliar os dados. As principais pesquisas encontradas utilizam modelos para computar de forma mais eficiente o custo de *matching* de uma imagem estereoscópica como pode ser visto em [3, 2, 8]. Nesse contexto, existem uma série de técnicas que podem ser utilizadas para a criação de modelos de aprendizado de máquina, como por exemplo a criação de árvores de decisão, máquina de vetores suporte, redes neurais, etc. Além disso, há bibliotecas de uso livre que auxiliam na implementação de tais técnicas, como o *Scikit-Learn* [9] e *Tensorflow* [10].

Uma das vantagens de se trabalhar com tais modelos reside na velocidade de execução. Há, no entanto, um custo computacional elevado para o treinamento do modelo, além de exigir um *dataset* que contenha dados de qualidade. Mesmo assim, o resultado final é um modelo rápido e funcional para aplicações que demandem esse tipo de performance.

1.3 Objetivo

Este trabalho de graduação tem como objetivo estudar as técnicas para correspondência de imagens estéreo (*stereo matching*) por meio de uso de redes neurais convolucionais, tendo como base o trabalho [3]. O presente trabalho buscará reproduzir o artigo base e verificar se a alteração de parâmetros do mesmo produz alguma variação de acurácia e perda sobre o modelo final. Tal entendimento será buscado por meio da análise das métricas de taxa de erro e perda que os modelos experimentados resultarem.

2 Fundamentação Teórica

2.1 Disparidade em Imagens Estéreo

Uma imagem estéreo é o resultado da combinação de duas ou mais imagens da mesma cena obtidas de diferentes posições. A forma mais comumente utilizada é aquela que faz uso de duas câmeras idênticas com a geometria de lente conhecida posicionadas ao longo do mesmo eixo, a uma distância d entre elas, conforme ilustrado na Figura 3.

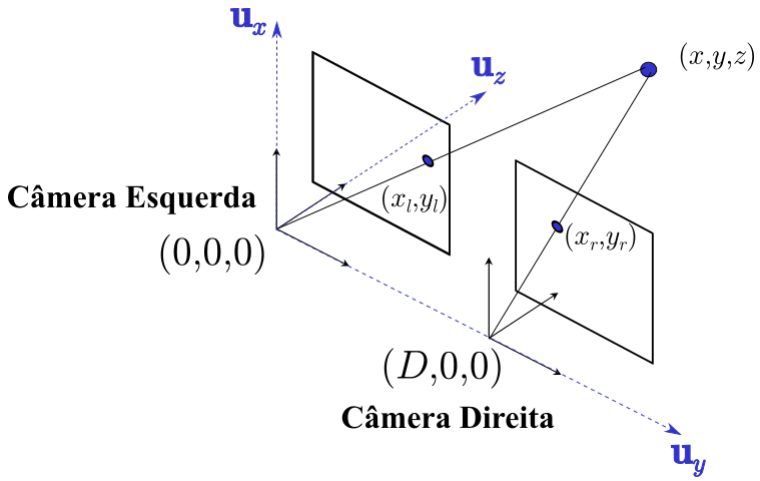

Figura 3 – Diagrama conceitual da captação estéreo. Um mesmo ponto espacial em (x, y, z) é retratado em *pixels* diferentes nas imagens esquerda (L) e direita (R).

Um conceito importante no contexto de imagens estéreo é a *disparidade espacial* entre as imagens. O termo disparidade foi introduzido na literatura relacionada à visão humana para descrever a diferença na localização de características correspondentes vistas pelo olho esquerdo e olho direito[1]. Matematicamente, suponha um par de imagem **L** e **R** tiradas a partir de duas câmeras posicionadas sobre o mesmo eixo a uma distância D entre elas. Tomando *pixels* correspondentes em cada uma das imagens, ou seja, os *pixels* que contém a mesma informação em ambas as imagens, $(x_l;y_l)$ na imagem **A** e $(x_r;y_r)$ na imagem **B**, a disparidade é dada por:

$$d_x = x_r - x_l \tag{2.1}$$

$$d_y = y_l - y_r \tag{2.2}$$

O cálculo da disparidade espacial é importante ao se trabalhar com imagens estéreo pois, implicitamente, provê a informação de profundidade nas imagens. Tomando a situação ilustrada na Figura 3, quanto mais distante o ponto (x, y, z) estiver das duas câmeras,

mais próximos serão os *pixels* correspondentes na imagem esquerda e direita; quanto mais próximo, os *pixels* correspondentes estarão mais distantes entre si, i.e., a disparidade é inversamente proporcional à profundidade [11, 12].

Além disso, ela é a principal componente da *Imagem de Disparidade Espacial*(DSI, *Disparity Space Image*)[13, 14]. A DSI é uma imagem que contém a informação da disparidade entre todos os pares de pixel correspondentes no par de imagens, ela é formada após encontrar a correpondência entre os *pixels*. A Figura 4 traz um exemplo da DSI calculada para um par específico de imagens. A partir dessa informação é possível criar representações em 3 dimensões para a cena capturada e assim auxiliar, por exemplo, em aplicações como reconhecimento de objetos 3D e navegação autônoma.

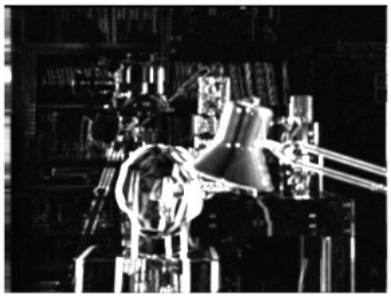

Figura 4 – Exemplo de DSI, no caso fixou-se a disparidade num valor e montou-se uma imagem com as distâncias relativas[1].

Na literatura é possível encontrar uma grande variedade de métodos para correspondência estéreo, propondo diferentes formas para obter a informação sobre profundidade. Entretanto, a fim de comparar as diferentes técnicas, em [1], foi estabelecida uma taxonomia para os diferentes métodos, identificando quatro etapas fundamentais que estão presentes em praticamente todos os métodos propostos:

- Cálculo do custo para *matching* de imagens
- Cálculo do Custo de Agregação
- Computação da disparidade e Otimização
- Refinamento das disparidades

Ao longo das últimas décadas muitos algoritmos de formação de imagens estéreo foram desenvolvidos[3]. E todos eles possuem em maior ou menor escala os itens apresentados acima. Na sequência detalhamos cada uma das etapas mencionadas.

2.2 Cálculo do Custo de *matching*

A etapa de *matching* na análise de imagens estéreo é responsável por identificar os *pixels* que são similares e representam um mesmo ponto da cena. Tal fato a torna essencial para o objetivo final, pois somente ela é capaz de fornecer meios de comparar os *pixels* de ambas as imagens para inferir o grau de similaridade existente entre elas. O termo "custo de *matching*" se refere ao valor da função custo associada à operação de *matching* entre os *pixels*.

Uma escolha bastante comum para função custo é denominada de *diferenças de intensidades quadradas*(SSD, *Sum of Squared Differences*) [15, 1, 16], que consiste basicamente em extrair a diferença de intensidade entre uma imagem de referência do par estereoscópico com o seu respectivo par, e obter o quadrado desse resultado, onde o menor valor significa maior semelhança entre os *pixels* comparados.

Outra opção muito utilizada é denominada de *diferenças de intensidades absolutas*(SAD, *Squared Absolute differences*)[17, 1]. Similarmente ao SSD, ela também extrai a diferença de intensidade entre os *pixels* de uma imagem do par estereoscópico utilizada como referência com a outra imagem do par, porém ao invés de se obter o quadrado, extrai-se o módulo do resultado e novamente o menor valor significa maior semelhança. Ainda, é possível utilizar a correlação cruzada normalizada como custo de *matching* [16, 1], que se comporta de modo similar à SSD.

É importante destacar que a busca pela correspondências entre os *pixels* é um processo não linear pois a distância entre os *pixels* não é constante ou obedece uma função. Se assim fosse seria possível inferir a correspondência de todos os *pixels* apenas encontrando a correspondência de um *pixel* nas imagens do par.

2.3 Custo de Agregação

A comparação do par de imagens a fim de obter a DSI, em geral, não é tão simples de ser realizada *pixel* a *pixel*. Há diferentes fatores que podem interferir nesse processo. Por exemplo, determinado objeto pode ser visível apenas em uma das imagens, e assim não haverá *pixels* correspondentes no par de imagens - que é normalmente denominado um problema de *oclusão*. O ruído também é outro componente importante nas imagens e pode introduzir erros no processo de *matching*.

Dessa forma, em geral, os métodos introduzem restrições para lidar com os problemas mencionados anteriormente. Por exemplo, em vários métodos denominados *locais*, o *matching* não é feito *pixel* a *pixel*, mas tentando maximizar o casamento entre blocos – pequenas regiões ao redor do pixel de interesse.

Métodos locais de correspondência estéreo são tipicamente mais utilizados devido

a sua fácil implementação e por serem mais rápidos do que os globais [18]. O motivo de tal performance pode ser atribuído ao fato de que este tipo de método limita a região a ser analisada e por ter um escopo reduzido é mais rápido do que métodos globais.

Em linhas gerais métodos locais de correspondência estéreo fixam um *pixel* e analisam a janelas onde ele se encontra, buscando seu correspondente através da etapa de *matching*, que é o *pixel* com maior similaridade, e compõem a região do seu entorno combinando os *pixels* de ambas as imagens e homogenizando a região através de uma função de custo, desta forma o processo como todo acaba sendo otimizado.

É importante destacar, entretanto, que o custo de *matching*, nesse caso, envolve a soma (agregação) do custo de *matching* entre vários pares de *pixels*. Em diversas técnicas, a agregação do custo de *matching* é feita somando ou obtendo a média da DSI em uma região de suporte [1]. A diferença entre elas está relacionada à forma como a região de suporte é selecionada e a função utilizada para calcular o custo [19].

Existem diversas abordagens para determinar a região de suporte. É possível utilizar várias janelas ancoradas em diferentes pontos [20], ou ainda empregar um janelamento adaptativo, onde o tamanho da janela de amostragem se adapta conforme a situação [1].

Ao final desta etapa é esperado a agregação dos *pixels* das regiões analisadas de modo que o resultado obtido seja utilizado como base para computar a disparidade entre os *pixels* e formar a DSI.

2.4 Computação da disparidade e Otimização

Após o *matching*, definindo quais *pixels* da região analisada possuem correspondente na imagem de referência, é necessário calcular a disparidade da imagem. Uma vez que o cálculo é feito a partir do resultado do *matching*, naturalmente as técnicas para cálculo de DSI também se dividem entre técnicas locais e globais [1].

Nas técnicas locais, o cálculo das disparidades é relativamente simples. Uma vez que já há a associação entre *pixels* (e regiões) correspondentes nas imagens e o respectivo custo agregado, a disparidade para cada pixel é obtida simplesmente escolhendo o menor valor de custo agregado que foi calculado para esse pixel [1].

Diferentemente dos algoritmos locais, o custo de computar a disparidade e otimizá-la é muito superior aos custos da duas etapas anteriores para os métodos globais. Não é raro que esses algoritmos cheguem até a pular a etapa de agregação, uma vez que ele vai percorrer todos os *pixel* da imagem a analisar um a um buscando minimizar a disparidade de um modo geral.

A grande maioria dos algoritmos globais faz uso de uma abordagem de *minimização de energia* [21], onde o objetivo é encontrar uma função de disparidade que minimiza a

energia global. Matematicamente, a função pode ser modelada por [1]:

$$E(d) = E_{data}(d) + \lambda E_{smooth}, \tag{2.3}$$

onde $E_{data}(d)$ quantifica o quão bem: a função disparidade d acertou para o par de imagens de entrada[1], e é definida por

$$E_{data} = \sum_{(x,y)} M(x, y, d(x, y)), \tag{2.4}$$

onde M é o custo de *matching* para a DSI.

O temo de suavização $E_{smooth}(d)$ carrega informação das hipóteses que o algoritmo explora. Para que a otimização seja tratável e não se torne um problema maior que a própria otimização, ela é restrita à comparação somente com a disparidade da vizinhança do *pixel* analisado, ou seja, o termo de suavização busca suavizar a transição de disparidade dos *pixels* ao entorno do *pixel* analisado. Matematicamente, ela pode ser modelada por:

$$E_{smooth}(d) = \sum_{(x,y)} \rho(d(x, y)) - d(x + 1, y) + \rho(d(x, y)) - d(x, y + 1), \tag{2.5}$$

onde o termo ρ é uma função de disparidade que aumenta monotonicamente.

Uma outra forma de se calcular DSI e otimizá-la é utilizando programação dinâmica. A otimização 2D da equação 2.3 é um problema NP difícil para algumas classes de funções de suavização [22], o que significa que não há um meio determinístico de encontrar um solução em tempo polinomial. Contudo, o uso de programação dinâmica pode contornar esse problema, uma vez que ela é capaz de encontrar bons mínimos independente da varredura utilizada em tempo polinomial.

Algumas abordagens que utilizam programação dinâmica baseados na intensidade focam em otimizar o problema da varredura [20, 1]. Essa abordagem computa o caminho mínimo de uma matriz, que podemos entender como uma imagem, que contenham todos os *pixels* emparelhados por duas varreduras correspondentes. O problema de se utilizar programação dinâmica aplicado ao cálculo de disparidade e otimização em métodos globais para correspondência estéreo está relacionado à seleção correta do custo para *pixels* que sofreram oclusão, algo que pode levar a erros que prejudiquem seriamente a imagem final.

Outra classe de algoritmos capazes de computar a disparidade e otimizá-las para métodos globais são os algoritmos cooperativos, inspirados em modelos computacionais para a visão estéreo humana [1]. Os métodos cooperativos operam por meio de operações locais não-lineares, de maneira que o comportamento global é semelhante ao observado nos métodos globais descritos anteriormente.

2.5 Refinamento das disparidades

Grande parte dos algoritmos de correspondência estéreo produzem uma imagem disparidade discreta [1], i.e., valores inteiros de disparidade apenas. Porém nem todas as aplicações conseguem utilizar um espaço quantizado, sendo necessária uma etapa de *refinamento*, que busca adequar o espaço que a imagem final é gerada para atender aos requisitos de aplicações que necessitam de uma informação de profundidade mais apurada.

Dessa forma, técnicas para se obter disparidades que são frações de *pixels* (sub-pixel), é necessário aumentar a resolução da imagem estéreo, o que pode ser feito, por exemplo, iterativamente por meio do algoritmo de gradiente descendente, ou mesmo por meio do ajuste de curvas para interpolação do custo de *matching* discreto de níveis de disparidade[23, 1]. Tais métodos resultam em aumento da resolução da imagem final e suavização da imagem como um todo.

2.6 Erro de Oclusão

Suponha duas câmeras posicionadas sobre o mesmo eixo, com uma distância d entre elas. Ambas estão registrando uma cena qualquer porém com detalhes relevantes ao fundo. Agora, imagine que exista um objeto na cena que oculte de uma das câmeras parte do fundo; a perda de informação por parte de um dos dispositivos é definido como *erro de oclusão*. Em outras palavras, a oclusão ocorre quando um grupo de *pixels* é registrado em apenas uma das imagens estereoscópica e na outra não [24], conforme ilustrado na Figura 5.

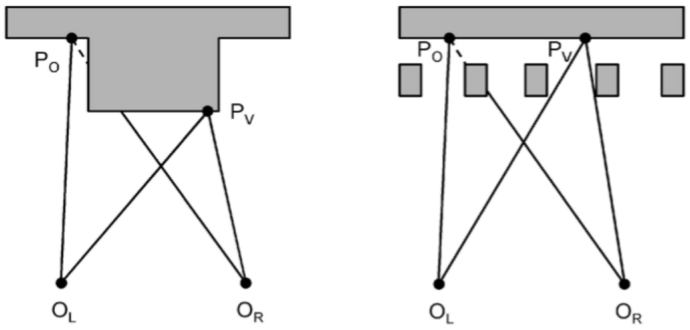

Figura 5 – As duas figuras, da esquerda e da direita, representam um problema de oclusão, onde o ponto P_o não está visível para a câmera da direita.

Em [20] é proposto um algoritmo que faz uso de programação dinâmica para corrigir parte dos erros oriundos de oclusão na geração da DSI. Eles implementam um algoritmo de programação dinâmica que identifica áreas de oclusão, ordena as restrições, ou seja, os *pixels* nos quais ocorre oclusão e finalmente busca para determinar qual seria a disparidade entre os *pixels* e assim obter a DSI [20].

Em [24] o problema da oclusão é tratado de outra forma. É proposto um modelo de *matching* de imagens estéreo simétrico para lidar com os erros provenientes de oclusão que faz uso das limitações de visibilidade da imagem[24], através dessas restrições é possível recuperar superfícies horizontais inclinadas e pequenos objetos ao fundo[24].

2.7 Erros Radiométricos

Diferentemente dos erros de oclusão, os erros radiométricos constituem uma classe de erros cuja origem são as características internas dos dispositivos fotográficos utilizados, esses erros acontecem devido as configurações internas das câmeras serem diferentes[25].

Erros radiométricos podem ser causado por uma série de fatores intrínsecos das câmeras, entre eles diferenças nas configurações das câmeras, ruído na formação da imagem ou vinhetas, que são áreas escurecidas da imagem. De acordo com[25] a calibração do equipamento pode corrigir parte dos erros, contudo nem todos são possíveis de se resolver, o que inviabiliza a produção de uma imagem estéreo.

2.8 Redes Neurais Artificias Convolucionais

Modelos de aprendizado de máquina como *redes neurais convolucionais*(CNN, *Convolutional Neural Network*) tem sido testada desde de 2015 e mostrou-se uma ótima abordagem ao se trabalhar com prolemas relacionados com extração de características de imagens[26], como o *matching* de imagens estéreo.

Quando aplicada em imagens, basicamente elas funcionam extraindo características de baixo nível para os diferentes objetos presentes na cena. Estas características são aprendidas e posteriormente usadas, convergindo para o filtro que resulta no modelo mais adequado[26].

A configuração de uma CNN aplicada a imagens pode envolver uma grande quantidade de parâmetros, cada um deles relacionados a uma característica passível de aprendizado, tal fato torna o uso da rede computacionalmente pesado. O uso de GPU é recorrente quando se trabalha nesta situação.

Como os outros modelos mais básicos de redes neurais, a CNN também passa pela fase de treinamento com uma porção selecionada do *dataset*, e uma fase de teste, uma vez que o modelo está suficientemente treinado é possível válida-lo, uma vez alcançado o nível satisfatório é possível utilizar o modelo gerado na resolução do problema.

Há muitas formas de implementar uma CNN em ambiente computacional. Pode-se fazer do jeito mais pragmático seja implementando a equação diferencial que descreve o comportamento do neurônio e da rede em si em qualquer linguagem capaz de resolver tal

algoritmo, ou utilizando bibliotecas que já possuem a implementação pronta, bastando apenas instanciar a rede, configurar seus parâmetros e executá-la.

2.9 Redes Neurais Artificias aplicadas ao Matching

O uso de redes neurais artificiais para verificar o *matching* de um par de imagens estereoscópicas tem crescido muito nos últimos anos, principalmente após a publicação de [2], onde é descrito como uma CNN pode ser utilizada para computar o custo de *matching*.

Desde sua publicação, o artigo [2] tem sido base para os trabalhos futuros, uma vez que muitos dos trabalhos que lhe sucederam tem usado o mesmo modelo de CNN, porém testando com novas modificações.

A rede neural proposta por eles age sobre um pequeno grupo de imagens de disparidades conhecidas, podendo ter sido adquiridas por um LIDAR por exemplo. O resultado do treinamento da rede é utilizado para computar o custo de *matching*.

A arquitetura proposta por [2] pode ser visualizada na Figura 6, nela podemos visualizar um total de oito camadas que constituem a técnica como um todo. De acordo com seus autores somente a primeira camada é convolucional.

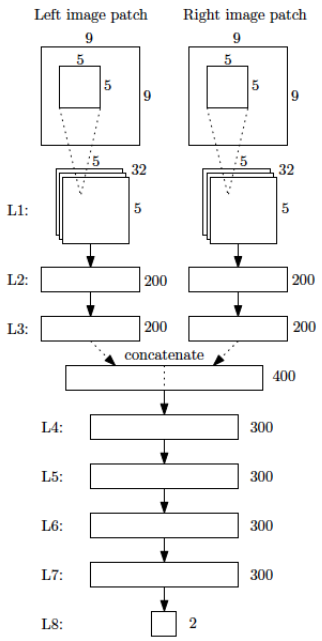

Figura 6 – Arquitetura da CNN utilizada por [2].

Como mostra a Figura 6, a rede possui dois ramos, uma para imagem direita e outro para a imagem esquerda, que são agregadas posteriormente. Cada camada indicada é composta por uma quantidade de neurônios, que são responsáveis por extrair características úteis ao processo. A entrada dela, ou seja, o nível L1 deve receber uma parcela das imagens

amostradas com tamanho de 9x9 em escala de cinza, reduzindo assim o volume de dados analisados.

O nível convolucional L1 é composto por 32 *kernels* de tamanho 5x5x1, sua saída é conectada ao nível L2 com 200 neurônios e cuja saída é o nível L3 também com 200 neurônios. Os três primeiros níveis são responsáveis por extrair as características utilizadas para agregar as imagens.

Com as caraterísticas particulares de cada imagem extraída é possível concatena-las em um vetor de tamanho 400 e dar continuidade ao processo. Dos níveis L4 ao L7 há 300 neurônios em cada um deles, neles ocorrem o refinamento da extração de características. O nível L8, o último, produz uma saída com dois possíveis resultados, um que indica *matching* correto e outro que indica *matching* incompatível [2], portanto trata-se de classificador binário de *matching*.

O modelo apresentado em [2] possui custo de *matching* $C_{CNN}(\mathbf{p}, d)$ representado pela equação 2.6.

$$C_{CNN}(\mathbf{p}, d) = f_{neg}(< P^E_{9x9}(\mathbf{p}), P^D_{9x9}(\mathbf{p}) >) \tag{2.6}$$

Onde $f_{neg}(< P^E, P^D >)$ é a saída da rede que resulta em *matching* incompatível.

Outra proposta interessante de utilização de rede neural para *matching* de imagens estéreo foi abordada por [3], no artigo eles formam uma arquitetura mais enxuta quado comparada com [2], como pode ser visto na Figura 7, porém com bons resultados.

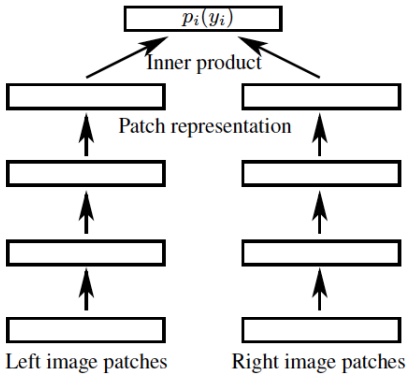

Figura 7 – Arquitetura siamesa utilizada por [3]

Como visto na Figura 7, a rede neural proposta possuí dois ramos, um para cada imagem do par estereoscópico. Mais uma vez cada camada possui uma quantidade de neurônios que extraem características úteis para realizar o *matching*.

A arquitetura siamesa recebe como entrada as imagens estereoscópicas, uma para cada ramo que seguem por quatro níveis que executam uma convolução espacial com um

filtro. O último nível concatena o resultado dos ramos em algo único através do produto escalar.

O uso do produto escalar para computar o custo de *matching* acelera o processo de modo significativo, uma vez que implementações anteriores que não o usavam necessitavam de aproximadamente um minuto de computação em GPU enquanto o método utilizam requer menos de um segundo[3].

O treinamento da rede neural foi feito selecionando uma pequena parte aleatória da cena com *ground truth* conhecido para o ramo esquerdo e a imagem completa para o ramo direito. Durante o treinamento foi utilizado a equação 2.7 para minimizar a perda de entropia cruzada[3].

$$min(w) \sum_{i,y_i} p_{gt} \log p_i(y_i, w) \tag{2.7}$$

Onde:

$$p_{gt} = \begin{cases} \lambda_1, y_i = y_i^{GT} \\ \lambda_2, |y_i - y_i^{GT}| = 1 \\ \lambda_3, |y_i - y_i^{GT}| = 2 \\ 0, \text{ c.c} \end{cases} \tag{2.8}$$

Ao final, o que a rede treinada produz é o mapa de disparidade dado um par estereoscópico. O modelo aplica a arquitetura da Figura 7 de forma que um dos ramos pegue apenas uma parte da imagem, selecionada aleatoriamente, e extrai suas características, enquanto que no outro ramo esse processo é aplicado para a imagem inteira, incluindo a parcela analisada no ramo paralelo.

A divisão em ramos do modelo está associada ao fato de que ao separar a análise das imagens é esperado que ambos os ramos cheguem em resultados próximos para a região que contenha a imagem escolhida aleatoriamente do par, isso justifica o uso do produto escalar ao final do processo pois resultados similares vão indicar forte correlação entre as imagens, resultando na correspondência entre elas.

A Figura 8 busca ilustrar como a rede funciona, desde a seleção da região de *matching* para um dos ramos, posicionada na base da imagem, até a correlação entre os resultados, produzindo o mapa de disparidade da imagem.

O artigo de [3] é fortemente baseado no artigo [27], há grandes semelhanças entre eles, principalmente no modo como a arquitetura foi proposta, ambos fazem uso de uma arquitetura siamesa com uso de ReLU para retificar os valores obtidos. O uso do ReLU foi proposto em [28] e aumenta as não linearidades dentro da rede, tornando a função de discriminação mais efetiva.

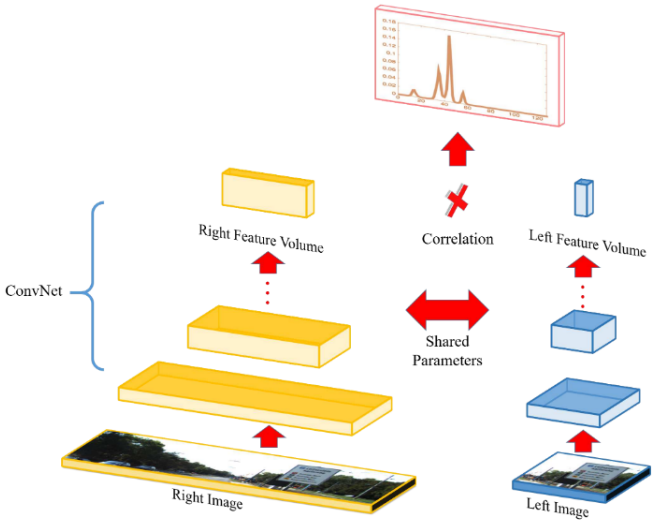

Figura 8 – Diagrama ilustrando o funcionamento da rede criada

O artigo de [29] utiliza o modelo proposto por [2] para estimar a profundidade de uma imagem estéreo. Eles utilizam a arquitetura como base, inserindo modificações, como a implementação do método proposto em [30], que compara a intensidade para os três canais de cores.

Outra abordagem interessante relacionado ao assunto foi proposta em [18], que utiliza uma cascata de CNN composta por dois estágios, onde o primeiro estágio é responsável por produzir uma DSI inicial enquanto o segundo assume a tarefa de refinar e ou retificar o resultado, este processo eles denominaram como 'Aprendizado em cascata residual' em tradução livre(*Cascade Residual Learning*)[18].

O artigo [26] busca criar uma rede neural convolucional que seja computacionalmente leve, de alta performance e que seja capaz de operar em tempo real num dispositivo com baixo poder de processamento.

A arquitetura utilizada pelo artigo se baseia na que foi proposta por [31], e através dela busca-se estimar a profundidade da cena analisada por meio de aprendizado supervisionado. De acordo com [26] a vantagem desta abordagem está no fato de ter uma rede capaz de predizer a confiança da medida para cada *pixel*.

2.10 *Dataset* Kitti 2015

O *dataset* Kitti é um *dataset* utilizado como *benchmark* para aplicações que envolvem aprendizado de máquina sobre imagens. Ele contém ao todos 200 imagens, todas elas compostas pelo par estereoscópico. Trata-se de uma *dataset* com imagens reais cujo conteúdo é a movimentação de ruas e avenidas da perspectiva de um carro em

movimento[18, 32, 33], a Figura 9 constitui um exemplo das imagens presentes no *dataset*.

Figura 9 – Exemplo de imagem presente no *dataset* KITTI 2015

As imagens contidas no *dataset* Kitti funcionam como *ground truth*, ou seja, a imagem estéreo resultante da junção do par estereoscópico é conhecida, de forma que é possível verificar se o modelo é capaz de produzir uma imagem próxima da correta, em outras palavras, o *ground truth* é a imagem estéreo real e que baliza o processo de treinamento do modelo. O processo de treinamento nada mais é do que a busca por convergência entre os resultados do modelo com a realidade representada pelo *dataset*.

É importante citar que o Kitti possuí duas versões, uma de 2012 e outra de 2015. A diferença entre as versões restringe-se apenas a quantidade de imagens presentes em cada uma das versões, a de 2012 com 194 pares de imagens, todas no formato png e a de 2015 com 200 pares de imagens.

Outro fato interessante sobre o *dataset* kitti é que sua organização mantenedora disponibiliza um kit de desenvolvimento com arquivos escritos em C++ e Matlab voltado a calibração da rede neural.

Ele constitui o principal *benchmark* dos artigos citados acima, e em seu site, ele provém um ranking com os algoritmos que possuíram melhor desempenho em realizar o *matching* das suas imagens estéreo.

3 Comparações entre Redes Neurais Artificiais

3.1 MLP

As Redes neurais artificial, do inglês *Artificial Neural Networks* ou também *Multi Layer Perceptron*(MLP), constituem uma classe de algoritmos capazes de classificar um determinado dado ou clusterizar um amontoado de dados com base num *dataset*, seja lá ele qual for. O neurônio artificial foi modelado para se comportar de modo semelhante ao modelo biológico.

Por tentar simular um neurônio, ela possui funcionamento similar, com vários neurônios interligados formando uma rede, com sua própria taxa de disparo e de aprendizado, além de outras medidas computacionalmente interessantes como o número de iterações para se chegar a um modelo e a quantidade de camadas de neurônios.

As primeiras tentativas de simular um neurônio como máquina data de 1943 com McCulloch e Pitts[34], contudo a tecnologia da época não era adequada para seu desenvolvimento, restringindo-se ao campo teórico. Importantes avanços foram feitos ao longo dos anos seguintes como o artigo de Klenee [35], onde ele explorar como representar os eventos numa rede neural. Outro artigo importante é o do Rosenblatt [36] que introduziu o uso de modelos estatísticos ao *Perceptron*.

O relatório produzido por Minsky e Papert [37] que revisa os principais avanços dos últimos anos e já relaciona o uso do *Perceptron* a Inteligência Artificial. Um avanço importante para as redes neurais foi a criação do método de *Backpropagation* que melhorou a performance da rede neural para realizar predições como demonstrado por Werbos [38], contudo sua data de origem é um tanto discutível. Hoje em dia com o advento da GPU com maior poder de processamento foi possível utilizá-las para outras aplicações. Seu alto custo computacional advém do fato de que os modelos fornecidos por ela são não-lineares, o que demanda tempo para fornecer um modelo de predição com boa acuracidade.

O tipo mais comum de rede neural são as chamadas *Perceptron*, é também o mais simples pois trata-se de uma rede de neurônios cuja entrada são as características com as quais se deseja criar um modelo de predição, a saída dos neurônios são somadas e aplicadas a uma função. A Figura 10 representa um Perceptron com uma camada oculta de neurônios.

Em outras palavras o que o *Perceptron* faz receber o valor das características selecionadas, multiplicar por uma função, que funciona como um peso e enviar para a

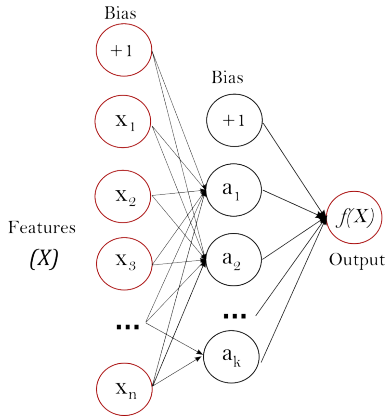

Figura 10 – Perceptron com uma camada oculta.

camada seguinte, esta camada pode ou não ter mais neurônios, contudo ela recebe como entrada todas as saídas da camada anterior, somando todas elas, esse processo é repetido até chegar o momento em que não há mais camadas, ao final os valores são novamente somados e inseridos numa função de disparo, classificando o dado e gerando um modelo de predição.

3.2 CNN

Redes neurais convolucionais, do inglês *Convolutional Neural Networks*, são capazes de aplicar filtros nos dados ao início do processo. Tal característica permite trabalhar diretamente com os dados do *dataset* ou com baixo grau de preparação, pois os filtros convolucionais extraem a característica para utilizar com entrada da camada neural. O estudo de redes neurais convolucionais surge ao tentar aplicar redes neurais artificial para o estudo de classificação e identificação de imagens e vídeos.

As redes neurais convolucionais buscam replicar esse processo de modo que os neurônios localizados na entrada de rede analisam apenas um conjunto de *pixels*, aplicando os filtros necessários para limpar imperfeições indesejadas. A atribuição de um conjunto de *pixels* para cada neurônio reduz erros na vizinhança do *pixels* central.

Uma rede neural convolucional típica pode ser dividia em três partes: a camada convolucional, uma camada de amostragem e um *Perceptron* tradicional. Cada uma dessas partes aprecem de forma sequencial na rede de modo que o produto de uma é a entrada de outra, ou seja, a saída de camada convolucional é entrada da camada de amostragem.

A camada convolucional é composta por uma quantidade fixa de *kernels* determinadas pelo arquiteto da rede. Um *kernel* é uma matriz que filtra parte dos dados, ou seja, sua aplicação sobre um conjunto de dados resulta na extração de uma características. O *kernel* deve ser aplicado para todo o conjunto de modo a cobrir todos os dados, fazendo isso obtém se um mapa de característica, que nada mais é do que um mapa com as características

dos dados realçadas.

Com o conjunto de dados inteiramente coberto pelo *kernel* é necessário realizar a operação de convolução entre os mapas de características e o conjunto de dados, o resultado desse processo é uma matriz, comumente chamada de mapa de ativação e é o produto da camada convolucional.

A camada de amostragem funciona reduzindo o tamanho do espaço do mapa de ativação, em outras palavras ele amostra o conjunto e produz um valor que o representa. Ele pode fazer isso através de qualquer função, porém duas são as mais utilizadas: por valor máximo e por média. Ao amostrar o por valor máximo o resultado será o maior valor da amostra, já ao amostrar por valor médio o resultado será a média dos valores encontrados. O produto da camada de amostragem é um conjunto de dados filtrados pela camada convolucional e menor graças a camada de amostragem.

Uma vez que o conjunto de dados foi filtrado, suas características principais foram realçadas e ele foi reduzido somente a elas, é possível utilizar um *Perceptron* para encontrar padrões e gerar um modelo de capaz de classificar ou prever um determinado dado.

O uso do *Perceptron* está ligado ao fato de que ele possui todos os neurônios da mesma camada conectados, sendo ativados por pesos diferentes e, após passar por todos os níveis estabelecidos, serão somados, resultando em um modelo de aprendizado de máquina.

Com as características básicas de uma CNN estabelecidas é possível analisar com maior rigor quais elementos e parâmetros constituem cada camada de um rede e qual a sua importância e influência dentro da mesma.

Os dados de entrada da rede são valores numéricos que representam a forma como enxergamos eles no mundo real, no caso de uma imagem, esse dados são representados por meio de um tensor, tendo como base o padrão de cores RGB o tensor será composto por três matrizes, uma para cada canal de cor e cada *pixels* é formado pela união das intensidades de cada canal, esta variando de zero até 255 como grau de intensidade para cada canal, ou seja, a união dos valores das três matrizes representam o espectro de cores que podemos representar.

A imagem recebida na entrada da rede será dividida de forma equivalente para cada neurônio presente nesta camada. Logo o que ocorreu foi uma operação de amostragem e definiu uma parcela da imagem, que pode ser representada por uma matriz, para cada neurônio.

Esta primeira camada trata-se de uma camada convolucional pois a operação a ser realizada na entrada é uma convolução que é matematicamente descrita pelo Equação 3.1. O que a operação descrita faz é montar uma matriz conhecida como mapa de características, percorrendo a matriz inicial, multiplicando ela por um filtro e somando os valores obtidos

naquela amostra.

$$C(i,j) = \sum_{m=0}^{(M_a-1)} \sum_{n=0}^{(N_a-1)} A(m,n)B(i-m,j-n) \tag{3.1}$$

Onde A é uma matriz de dimensões (M_a, N_a) e B é uma matriz com dimensões (M_b, N_b), e as condições de contorno são: $0 \leq i < M_a + M_b - 1$ e $0 \leq j < N_a + N_b - 1$.

É necessário definir alguns parâmetros para executar convolução. O tamanho do *kernel* a ser utilizado como filtro é de suma importância bem como os valores que ele assume, pois dependendo dessas duas variáveis é possível criar um filtro com funcionalidades específicas como normalizar uma imagem ou extrair as bordas do objeto em cena.

Logo, a construção do filtro deve ser voltada a extração de características que distinguem uma imagem da outra, em outras palavras o filtro deve valorizar a principal característica da amostra.

A taxa com que o *kernel* percorre os dados é chamada de *stride*, ele representa o passo com que o *kernel* percorre os dados, podendo avançar um ou mais *pixels* por vez. Ele também pode implicar em diferenças no resultado final da operação uma vez que a repetição de alguns *pixels* ou a perda dos mesmos pode ocorrer.

Pode acontecer que o *kernel* não consiga preencher a parcela amostrada da imagem deixando uma parte sem percorrê-la ou não havendo colunas ou linhas suficientes para encaixar todo o *kernel* na imagem. Para esse problema existem duas soluções: a primeira seria ignorar a parcela restante e prosseguir com a convolução, e a segunda opção, mais comum, seria preencher com zeros a parcela restante até que seja possível realizar a convolução na parcela.

O preenchimento com zeros não afeta o resultado final pois na multiplicação entre os fatores e posterior soma, os elementos com zero ao serem utilizados na operação de convolução trariam um resultado nulo, e ao serem somados no final da operação não afetariam resultado final da operação de convolução.

Outro parâmetro importante na etapa convolucional é a escolha do modelo de retificação de valores pois ao convoluir os dados é possível que exista valores negativos, contudo esse tipo de dado não possui sentido físico, uma vez que não admite esse tipo de valor para imagens e para isso é preciso retificá-los.

Existem algumas técnicas que podem ser utilizas para retificar a matriz, dentre elas a mais comumente utilizada por cientistas de dados é o ReLu, do inglês *ReLu Rectified Linear Unit* ou *Retificador linear Unitário*, trata-se de uma função unitário que converte

valores negativos em zero, conforme a Equação 3.2.

$$ReLu = \begin{cases} a_{ij}, se\ a_{ij} \geq 0 \\ 0, \text{ c.c} \end{cases} \tag{3.2}$$

Outras funções podem servir como retificadores, como a tangente hiperbólica ou uma sigmóide positiva, porém como dito acima a ReLu é a mais comumente utilizada.

O produto final da camada convolucional é o mapa de características, que é matematicamente descrita como uma matriz, devido ao processo convolucional seu tamanho é menor que a parcela da imagem original, porém realça as características marcantes da imagem, a matriz de características possuí o tamanho expressa pela Equação 3.3, onde h é o numero de linhas e w o número de colunas e d o número de canais de cores da parcela da imagem, f_h é o numero de linhas e f_w o número de colunas e d o número de canais de cores do *kernel* utilizado para filtro.

$$\begin{gathered} Parcela\,da\,imagem\,original:\ h \times w \times d \\ Filtro:\ f_h \times f_w \times d \\ Matriz\,de\,Caracter\acute{i}sticas:\ (h - f_h + 1) \times (w - f_w + 1) \end{gathered} \tag{3.3}$$

É importante citar que o número de canais da parcela da imagem e do *kernel* utilizado como filtro devem ser iguais.

A camada convolucional pode ser resumida como mostra a Tabela 1

Tabela 1 – Resumo da camada convolucional.

Entrada	Ação	Parâmetro	Saída
Imagem amostrada ou matriz	Aplicar *kernel* sobre a entrada Extrair características	Tamanho do *kernel* Valores do *kernel* Valor do *stride*(passo) Função de retificação	matriz ou imagem das características

Outra camada importante dentro de uma rede neural convolucional é camada de amostragem, que tem a função de amostrar a matriz de características dando relevância a principal característica encontrada.

O processo de amostragem trata-se de uma operação de percorrer o resultado obtido na camada convolucional com uma janela bem definida e montar uma nova matriz somente com as características mais relevantes, ou seja, é um processo um tanto quanto parecido com o da camada convolucional, contudo a operação executada é diferente.

Alguns parâmetros são importantes para esta camada como por exemplo o tamanho da janela a ser utilizada. Uma janela pequena amostra poucos valores e produz uma redução mínima, enquanto uma janela muito grande pode gerar perdas de detalhes importantes na análise.

Além do tamanho da janela, a operação executada por ela é de suma importância. É possível realizar inúmeras operações nesta secção, contudo elas precisam ter um certo sentido ao serem executadas, e as mais utilizadas são as seguintes:

1. O valor máximo da janela
2. A média dos valores da janela
3. A soma dos valores presentes na janela

Tais operações são necessárias para que as características extraídas não sejam perdidas durante a operação de amostragem e servem para realçar o produto da extração. A escolha de qual técnica utilizar deve ser adequada a situação analisada.

É importante citar que no processo de amostragem o *stride* também é importante nesta etapa pois a taxa com que a janela percorre a matriz de características produz diferenças consideráveis no resultado. Para um *stride* igual a um pode haver valores repetidos caso a operação seja a extração do valor máximo por exemplo. A Figura 11 ilustra bem o processo realizado pela camada de amostragem.

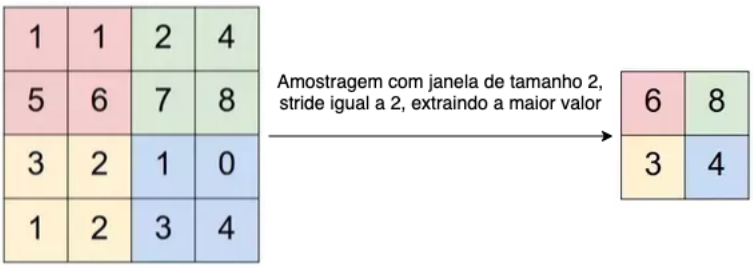

Figura 11 – Ilustração da camada de amostragem

A camada de amostragem tem como produto a uma matriz de características com dimensão espacial menor e com as características extraídas da camada convolucional realçadas, a Tabela 2 resume o processo como um todo.

Tabela 2 – Resumo da camada de amostragem.

Entrada	Ação	Parâmetro	Saída
Matriz ou imagem das características	Reduzir a dimensionalidade Realçar as características extraídas	Tamanho da janela Valor do *stride*(passo)	Matriz ou imagem das características com dimensão espacial reduzida e características realçadas

A última camada dentro de uma rede neural convolucional é o *Perceptron*, que nada mais é do que a utilização de uma rede neural MLP, ou seja, a grande diferença entre uma rede neural MLP e uma convolucional é que esta última a entrada de dados passa por um filtro convolucional e por um processo de amostragem para realçar as as características do dado a ser classificado.

Apesar de ser uma rede neural comum existem parâmetros importantes que devem ser analisados ao se trabalhar com redes neurais como um todo. A quantidade de neurônios presentes na camada oculta pode provocar mudanças nos resultados.

É importante dizer que um maior número de neurônios na camada oculta do *Perceptron* não necessariamente implica em melhora dos resultados, contudo representam um aumento no custo de processamento utilizado pelo hardware em questão.

A função de ativação dos neurônios constitui parâmetro importante dentro da camada do *Perceptron*, pois dado uma entrada num neurônio qualquer é ela que vai provocar os disparo do mesmo ou não. A mais utilizada é o *ReLu*, o mesmo da Equação 3.2.

Outro parâmetro importante dentro desta camada é a taxa de aprendizado a ser utilizado pelos neurônios e qual a dinâmica delas. Taxas pequenas podem resultar em muitas iterações dentro da camada para ela encontre um padrão.

A taxa de aprendizado do neurônio segue alguns formatos, ela pode ser constante, e isso significa ela vai ser a mesma para todos os neurônios presentes na camada, ela também pode ser adaptativa, ou seja, conforma ela se adapta conforme o sistema vai evoluindo, podendo aumentar ou diminuir. A escolha da taxa de aprendizado está diretamente ligada ao poder computacional do *hardware* utilizado para treinar a rede.

A camada do *Perceptron* pode ser resumida de acordo com a Tabela 3.

Tabela 3 – Resumo da camada *Perceptron*.

Entrada	Ação	Parâmetro	Saída
Matriz ou imagem das características com dimensão espacial reduzida e características realçadas	Classificar ou estimar os dados de acordo com o *target* estabelecido	Quantidade de neurônios Função de ativação Taxa de aprendizado	Dado classificado de acordo com o padrão encontrado pela rede

As três camadas descritas acima constituem o básico de uma rede neural convolucional, ou seja, esses três elementos são necessários para construir uma CNN, isso não significa que eles não podem ser combinados das mais diversas formas ou que seja adicionado novas camadas ao processo como um todo.

Dentro de uma rede neural convolucional é comum que as camadas convolucionais e de amostragem sejam repetidas, isso acontece justamente pois elas são utilizadas para grandes volumes de dados e o produto da combinação delas é um grande filtro onde somente as características mais relevantes chegam até o *Perceptron*, onde os dados são efetivamente classificados. A Figura 12 ilustra muito bem uma rede onde há repetição das camadas convolucionais e de amostragem.

As camadas citadas acima, bem como os parâmetros, são as principais dentro de qualquer rede neural convolucional, independente do modelo, estes sempre estarão presentes, independente do peso estes são elementos fundamentais para a construção,

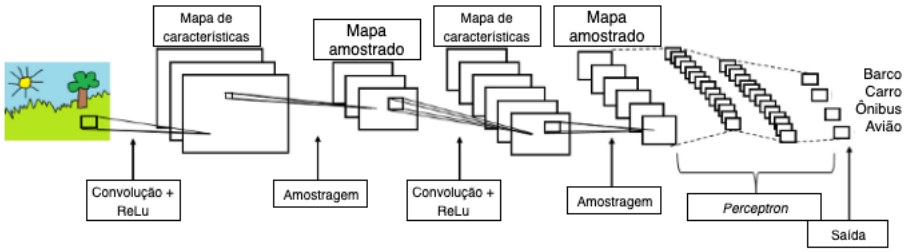

Figura 12 – Exemplo de CNN com repetição das camadas convolucionais e de amostragem

configuração e aplicação de uma CNN em situações em que ela se mostra útil e necessária.

4 Metodologia

O presente trabalho busca reproduzir o artigo [3], averiguando a reprodutibilidades dos resultados e modificar os parâmetros de configuração da rede neural montada pelos pesquisadores e comparar os resultados obtidos, de forma que seja possível visualizar a relevância de cada parâmetro dentro da rede criada.

Por se tratar de um projeto com dupla finalidade, a reprodução de um artigo e a comparação de dados oriundos da alteração de parâmetros de configuração da CNN, logo, isso significa que a metodologia do projeto deve seguir a apresentada no artigo base para que a primeira parte do projeto seja alcançada.

Uma vez de posse dos mesmos dados que o artigo, será possível avançar com a alteração da configuração da CNN, atuando nos seguintes parâmetros:

1. Tamanho do *kernel* convolucional
2. Tamanho do *kernel* de amostragem
3. Técnicas de amostragem(Valor máximo, média)

É preciso comentar um fato relevante a respeito da estrutura da rede neural convolucional montada em [3], dentro do modelo criado por eles não há camada de amostragem, a análise do artigo não deixa claro os motivos do desuso de tal camada. Para que seja possível avaliar os efeitos da camada de amostragem ela será adicionada para que seu efeito possa ser mensurado.

A geração de métricas para comparação dos resultados será feita de forma individual, ou seja, escolhendo um parâmetro de configuração de cada vez, ele será iterado do seu limite mínimo ao máximo, enquanto as outras configurações permanecem estáticas. A métrica final gerada será os valores de acurácia e perda para o conjunto de treinamento e teste para cada alteração de parâmetro.

A análise dos dados será focada em entender e explicar os comportamentos observados, sendo assim possível inferir qual a influência e eficiência do parâmetro nos valores finais do modelo criado.

Serão realizadas duas análises, uma quantitativa e outra qualitativa. A analise quantitativa será construída em cima do crescimento, decrescimento e constância dos dados. Através da analise qualitativa será possível aferir a influência de determinado parâmetro através de melhora ou piora relativa ao sistema estudado e de forma geral o reflexo dos dados no mesmo.

A obtenção dos *dataset* é parte essencial deste trabalho. Logo é necessário realizar o *download* das imagens dos seus respectivos repositórios online. Outra parte vital ao projeto é o desenvolvimento do *software* a ser utilizado para criar o modelo de aprendizado.

O *software* desenvolvido em [3] foi disponibilizado e tornado público pelos seus autores. Ele foi escrito utilizando a linguagem de programação *Lua* e faz uso do *framework Torch* para computar de forma eficiente a solução final do problema. Apesar do software original ser público ele não será usado pois será desenvolvida uma nova versão em *Python* tendo o *Tensorflow* como biblioteca para manipular modelos de aprendizado de máquina com eficiência, que terá como base o código descrito em [39].

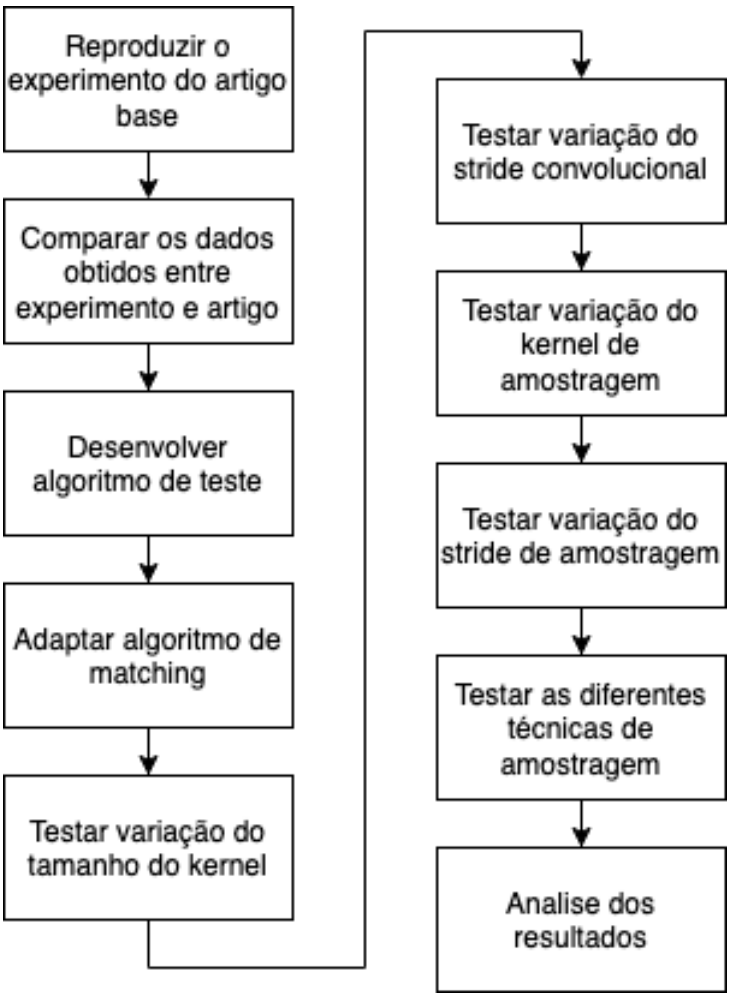

Figura 13 – Fluxograma do esquema de trabalho

A metodologia consiste basicamente em reproduzir o artigo original para garantir a reprodutibilidade dos dados originais, contudo é preciso observar que as condições de reprodução do dados não serão iguais, pois, tendo em vista que no artigo original usa-se uma GPU NVIDIA TITAN-X a qual não será utilizada neste projeto. Como a principal medida do artigo [3] é tempo de computação, é preciso ter em mente que o resultado que projeto pode obter precisam ser analisados de forma proporcional e correspondente as recursos utilizados neste projeto.

Apesar da métrica temporal original poder ser diferente da obtida é esperado que os valores sejam parecidos, uma vez que o modelo gerado deve ser próximo ao dos pesquisadores. E o que será buscado serão os valores de acurácia e perda do modelo gerado, com esses valores poderá se dizer o quão prático e utilizável é o modelo.

Uma vez que o experimento original foi reproduzido, partir-se-á para a proposta do projeto, que é testar e observar o comportamento da rede neural convolucional, através das métricas utilizadas no artigo original[3], ou seja, taxa de erro e perdas, ao variar os parâmetros de configuração da CNN.

A taxa de erro se encaixa bem como métrica para avaliar o desempenho do modelo treinado, seu propósito é verificar o quão bem o modelo consegue acertar dado uma entrada. De acordo com [33] a correspondência entre os *pixels* é considerada correta se a distância relativa entre eles é menor do três *pixels* ou que fluxo óptico seja menor do que 5%.

Outra métrica interessante para mensurar o desempenho do modelo é a perda. Ela varia de zero ao infinito e mede o grau de otimização do modelo criado, com zero representando a máxima otimização do modelo, desta forma ela é capaz de avaliar se a função utilizada para convergir o modelo está otimizada ou não, em linhas gerais a perda busca representar o quão confuso o modelo está em relação ao *dataset* utilizado.

4.1 Configurações do ambiente de experimental

É importante explicitar as características do *hardware* utilizado, para que na necessidade de reprodução dos experimentos contidos no presente trabalho ele não seja um elemento desconhecido. O *hardware* tem influência direta sobre os resultados obtidos, logo eles são válidos apenas para esse conjunto especifico de recursos.

Foi utilizada uma máquina virtual baseada em nuvem devido a facilidade e praticidade em criar um *hardware* adequado sem a necessidade tê-lo fisicamente. Junto a máquina virtual foi utilizada uma GPU NVIDIA TESLA P100 para acelerar a computação dos resultados, em [3] foi utilizada uma NVIDIA TITAN-X.

O *Sistema Operacional*(SO) utilizando para operar o *hardware* foi o Ubuntu 18.04, enquanto que o SO original [3] é desconhecido. Outro ponto importante a ser citado é em relação as características do *software* utilizado, como citado nos parágrafos anteriores, foi utilizado a linguagem de programação *Python* para codificar as operações necessárias ao invés de *Lua*, por praticidade e abundância de bibliotecas e recursos facilitando o desenvolvimento do experimento.

Outro ponto de destaque se refere ao *framework* utilizado. Ao invés de *Torch*, mais adequada a linguagem *Lua*, foi empregado o *Tensorflow* [10], que é amplamente aplicado em *softwares* relacionados ao tópico de rede neurais. A Tabela 4 lista e compara

os elementos contidos no *hardware* e *software* original e deste experimento.

Tabela 4 – Comparativo entre o ambiente utilizado em [3] versus o do presente trabalho.

	Original	Reprodução
Hardware	-	Máquina Virtual
RAM	-	60 GB
SO	-	Ubuntu 18.04
GPU	GPU NVIDIA TITAN-X	GPU NVIDIA TESLA P100
Linguagem de programação	*Lua*	*Python 3.6*
Framework	*Torch*	*Tensorflow*

4.2 Pré-processamento

Ao fazer uso de modelos de aprendizado de máquina sempre existe uma etapa de pré processamento. O artigo original [3] estabelece um forma de realizá-lo, e ela foi preservada de modo que para todos os experimentos que vieram a seguir tiveram esta etapa como base.

O pré-processamento estabelecido em [3] é necessário para adequar a imagem a rede neural. O processo consiste em acessar o arquivos que contêm as imagens, já na forma de DSI, particionar os arquivos em treino e validação, e identificar em cada imagem do *dataset* quais *pixels* serão analisados, extraindo deles suas respectivas posições, tanto na imagem esquerda quanto na direita. O fluxograma da Figura 14 ilustra muito bem o processo de preprocessamento.

Figura 14 – Fluxo do processo de pré-processamento

É preciso deixar claro que a finalidade do pré-processamento é selecionar uma região da imagem estéreo, ou DSI, e identificar nela a posição correspondente no par estereoscópico, ou seja, uma vez definida a região será encontrada a posição dos *pixels* na imagem direita e esquerda, funcionado como *ground truth* para a rede neural.

Ao final do pré-processamento serão gerados três arquivos, dois contendo as informações de posição dos *pixels* na imagem direita e esquerda, e a identificação de qual imagem está sendo analisa pois elas foram permutadas que serão utilizados para treino e validação; e um arquivo que identifica as imagens permutadas.

A etapa de pré-processamento é de extrema importância pois ela serve como base para que a rede consiga apurar se os pixel escolhidos para formar a imagem estéreo possuem alto grau de correlação e portanto são iguais.

4.3 Alteração do Tamanho do *Kernel* Convolucional

O método utilizado para realizar o experimento de verificar a influência do tamanho do *kernel* convolucional no tamanho da rede, busca preservar a estrutura original, ou seja, as nove camadas originais foram mantidas para verificar somente a influência do tamanho *kernel*. Com a exceção da modificação do tamanho do *kernel*, todos os outros parâmetros foram mantidos, ou seja, parâmetros como: taxa de aprendizado, otimizador e quantidade de iterações foram mantidas iguais a [3].

O experimento foi montado de modo a obter-se os valores de acurácia e perda da rede. O procedimento adotado consiste em criar um modelo para cada um dos tamanhos de *kernel* ímpares, começando em três e indo até 11, ou seja, foram criados cinco modelos cujos tamanhos dos respetivos *kernels* eram três, cinco, sete, nove e onze.

Durante o processo de treinamento é possível visualizar os valores de perda do modelo naquele instante, sendo que a perda final é a última a aparecer. Com o modelo treinando é possível executar o algoritmo de modo que ele extraia a taxa de erro(SER, *Stereo Error Rate*) e disponibiliza ela na tela. Os valores obtidos foram armazenados numa Tabela de controle para análise posterior.

O valor da acurácia é obtida de modo indireto, pois com modelo pronto é realizado um teste com as imagens do *dataset* afim de se obter a taxa de erro estéreo (SER), e o valor da acurácia (acc) pode ser obtida pela equação 4.1.

$$acc = 1 - SER/100 \tag{4.1}$$

Uma vez de posse dos números, será feita a análise qualitativa e quantitativa dos dados. A Figura 15 constitui o fluxograma de procedimentos adotados para este experimento e ilustra de modo visual todo o processo a ser realizado. O resultado esperado

deste experimento é que os valores de taxa de erro estéreo fiquem melhores com o uso de *kernels* maiores.

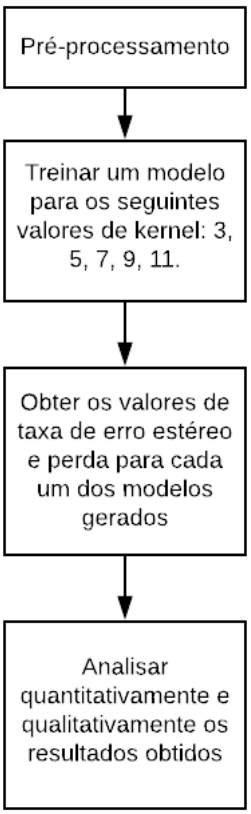

Figura 15 – Fluxo do processo de alteração do tamanho do *kernel*.

4.4 Alteração do Tamanho do *Kernel* de Amostragem

O experimento relacionado a verificação da influência do tamanho do *kernel* de amostragem possuí algumas peculiaridades interessantes. O fato da rede original não possuir uma camada de amostragem constituiu um desafio, pois qualquer mudança estrutural dentro da rede é capaz de fazê-la perder suas propriedades originais. Devido a este fato o procedimento adotado buscou ser o menos intrusivo possível.

Para poder averiguar a influência do tamanho do *kernel* de amostragem, foi inserido logo após a última camada convolucional uma camada de amostragem, para que tal experimento fosse possível. A localização dela foi escolhida tendo como base o propósito da camada convolucional, que é realçar as características da imagem por meio da operação de convolução, logo a opção de posicionar a camada de amostragem logo após a última camada convolucional se mostrou razoável, pois após a imagem passar por nove camadas convolucionais as característica buscadas já estariam bem realçadas e prontas para serem amostradas.

O procedimento adotado busca gerar um modelo para cada tamanho de *kernel* de amostragem e obter os valores de acurácia e perda de cada treinamento realizado, o procedimento para obter estas métricas é similar ao adotado no variação do tamanho do *kernel* convolucional, ou seja, para o valor de perda do modelo será considerado o último valor obtido durante o treinamento e para a acurácia será obtido de modo indireto através da SER e da equação 4.1.

Desta forma buscou-se fixar o tamanho do *kernel* convolucional em cinco e variar o tamanho do *kernel* de amostragem de modo a gerar um modelo para os seguintes valores: três, cinco, sete e nove, sempre o *stride* de amostragem fixo em um. A amostragem realizada irá extrair o valor máximo do quadrante analisado e desta forma bastará comparar com os valores do modelo original e verificar a influência da camada de amostragem sobre a rede montada.

A hipótese a ser testada neste experimento é verificar qual a influência da camada de amostragem sobre os valore de taxa de erro e perda para o problema em questão. É esperado que os resultados de taxa de erro e perda sejam altos devido a natureza da camada de amostragem de selecionar valores e perder informação durante o processo.

Uma vez de posse dos números, será feita a análise qualitativa e quantitativa dos dados. A Figura 16 constitui o fluxograma de procedimentos adotados para este experimento e ilustra de modo visual todo o processo a ser realizado.

4.5 Alteração da Técnica de Amostragem

Para verificar a influência da técnica de amostragem sobre a rede o procedimento elaborado é similar ao adotado para verificar a influência da amostragem sobre o modelo, a única diferença é em ralação ao tipo de técnica utilizada para extrair os valores, no caso da influência da amostragem foi extraído o valor máximo do quadrante, enquanto que para o experimento será extraído o valor médio.

O procedimento adotado busca gerar um modelo para cada tamanho de *kernel* de amostragem e obter os valores de acurácia e perda de cada treinamento realizado, o procedimento para obter estas métricas é similar ao adotado no variação do tamanho do *kernel* convolucional, ou seja, para o valor de perda do modelo será considerado o último valor obtido durante o treinamento e para a acurácia será obtido de modo indireto através da SER e da equação 4.1.

Desta forma buscou-se fixar o tamanho do *kernel* convolucional em cinco e variar o tamanho do *kernel* de amostragem de modo a gerar um modelo para os seguintes valores: três, cinco, sete e nove, sempre com o *stride* de amostragem fixo em um. A amostragem realizada irá extrair o valor médio do quadrante analisado e desta forma será possível

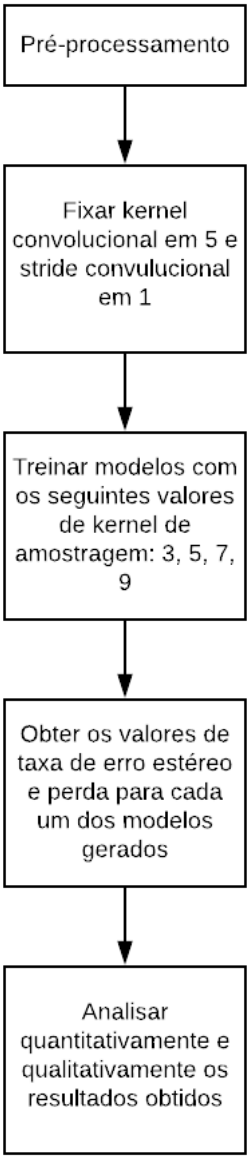

Figura 16 – Fluxo do processo de alteração do tamanho do *kernel* de amostragem.

comparar com os valores do modelo original e aqueles cuja técnica de extração foi o valor máximo e verificar a influência da técnica de extração sobre a rede.

A hipóteses a ser testada neste experimento é verificar a influência da técnica utilizada para amostrar os valores tem sobre as métricas selecionadas. É esperado que a extração do valor médio produza valores melhores por harmonizar a região e conter informação de todo a área, mesmo de forma indireta.

Uma vez de posse dos números, será feita a análise qualitativa e quantitativa dos dados. A Figura 17 constitui o fluxograma de procedimentos adotados para este experimento e ilustra de modo visual todo o processo a ser realizado.

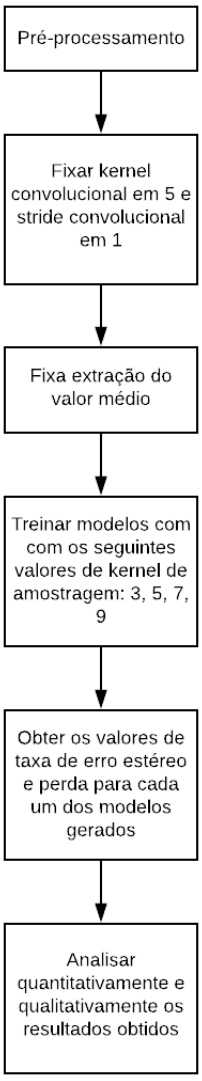

Figura 17 – Fluxo do processo de alteração da técnica de amostragem.

5 Resultados

5.1 Reprodução dos resultados

Como descrito na seção de Metodologia, a parte experimental foi divida em duas, a primeira focada em reproduzir os resultados do artigo original dentro das limitações existentes; enquanto a segunda foca em variar parâmetros da rede neural convolucional com intuito de verificar a influencia deles sobre os valores de taxa de erro e perda do modelo.

É preciso lembrar que as condições de reprodução do experimento não foram exatamente iguais, por isso não são esperados resultados idênticos, mas sim valores similares. Logo abaixo na Tabela 5 se encontram os resultados obtidos comparando com os originais.

Tabela 5 – Comparação entre os tempos execução do modelo original e do reproduzido.

	Tempo de execução(s)
Original	0,34
Reprodução	0,3110

Como pode ser observado na Tabela 5 o tempo de execução da reprodução do experimento ficou próximo do original, variando apenas $8,59\%$, diferença está dentro do esperado, uma vez que é impossível reproduzir o ambiente original. A Figura 18 representa o par estereoscópico utilizada como entrada para o modelo, enquanto a Figura 19 representa a imagem gerada, ou seja, a DSI resultante do processo de formação da imagem estéreo.

(a) Imagem esquerda. (b) Imagem direita

Figura 18 – Imagem utilizada como entrada para o modelo

Com o quesito tempo satisfeito e reproduzido de acordo com o esperado, passou-se para as métricas de acuracidade do modelo treinado. Com treinamento realizado sendo feito em GPU, o tempo total para averiguar tal tarefa foi de aproximadamente 4 horas, o período dedicado ao treinamento do modelo não é relevante neste caso, pois pode-se levar mais ou menos tempo de acordo com o hardware selecionado. Uma métrica importante para o modelo é o valor da acurácia dele, tais resultados se encontram na Tabela 6.

Figura 19 – DSI resultante da aplicação do modelo no par estereoscópico

Tabela 6 – Taxa de Erro dos modelos.

	Taxa de erro (%)
Original	7,23%
Reprodução	6%

De acordo com a Tabela 6 os resultados da reprodução estão compatíveis com o original, apesar da diferença de 1,23% entre as medidas a diferença parece razoável para afirmar que a reprodução dos resultados foi garantida e assegurada de modo que foi possível avançar com as mudanças propostas.

É interessante comentar que uma taxa de erro de 6% implica numa acurácia de 94% para o modelo apresentado, contudo é relativo dizer se esse valor é bom ou ruim, pois essa afirmação necessariamente precisa considerar o tipo de aplicação envolvida que consome o dado, para um veiculo autônomo que utiliza visão estéreo para obter informações sobre o quão distante determinado objeto está do veículo para apoio a tomada de decisão 94% pode ser ruim, pois um dos objetos da cena pode ser um pedestre, porém para uma aplicação que busca apenas identificar a profundidade de uma cena qualquer sem que essa informação influencie algo mais que 94% é excelente.

5.2 Alteração do Tamanho do *Kernel* Convolucional

Seguindo a metodologia apresentada na secção de Metodologia, o resultado para a variação da taxa de erro em função do tamanho do *kernel* se encontra na Tabela 7, localizada logo abaixo.

Tabela 7 – Taxa de erro e perda em função da variação do tamanho do *kernel*.

Tamanho do *kernel*	Taxa de Erro	Perda
3	6,442	2,095
5	6,482	1,870
7	7,662	1,805
9	2,726	1,675
11	2,308	1,800

Primeiramente é preciso observar que o valor da taxa de erro diminui com o aumento do tamanho do *kernel* convolucional, sendo a única exceção tendo ocorrido com o tamanho sendo sete. Tal fato se mostra intrigante uma vez que o aumento do *kernel* produz um aumento na área coberta pela convolução, e por conseguinte uma generalização maior da região analisa, o que deveria causar pouco realce das características.

Tal fato é corroborado pelos valores de perda da rede após 40 mil interações, tais valores pode ser verificados na Tabela 7, que de acordo com a mesma, ficou com valores baixos em progressão, gerando uma taxa de erro estéreo baixa progressivamente.

É preciso lembrar que tais valores foram obtidos sem realizar nenhuma mudança estrutural da rede original, ou seja, neste ponto ainda não há adição de camada de amostragem dentro da rede. Foi apenas variado o tamanho do *kernel* em busca de se obter seus efeitos.

É interessante notar que a melhor taxa de erro estéreo, e por consequência o modelo de melhor acurácia, foi obtido com o tamanho de *kernel* fixado em 11, tal resultado pode ser explicado pela seguinte preposição, o aumento do *kernel* convolucional implica que o valor final do *pixel* será composto por múltiplas fontes de informação, no caso, os *pixels* ao redor. Tal preposição significa que o *pixel* resultante da camada convolucional será formado pela soma dos *pixels* ao seu redor multiplicado pelos valores sobrepostos do *kernel*, e que ao final ele carregará informação de todos eles, somando isso ao fato de que a rede possui nove camadas convolucionais, dificilmente o valor final do *pixel* será igual a outro dentro da imagem, logo a correlação entre o *pixel* da imagem direita e esquerda será alta. É preciso deixar claro que a preposição funciona para os caso em que as imagens analisadas não possuem grandes regiões homogêneas, ou seja, a riqueza de detalhes dentro do *kernel* é grande o suficiente para gerar valores únicos.

Um ponto interessante dentro do experimento proposto e que se confirmou verdadeiro foi a relação direta de proporcionalidade entre o tempo de treinamento e o tamanho do kernel, quanto maior o tamanho do *kernel* utilizado mais longo era o período de treinamento do modelo. Para o menor *kernel* o treinamento durou cerca de cinco horas, enquanto para o maior foi próximo de 24 horas, mesmo utilizando GPU para computar os dados. A explicação para isso é simples, um *kernel* de tamanho três possuí nove células que necessitam ser calculadas enquanto um de tamanho 11 possuí 121, logo fica evidente a diferença entre realizar nove operações e 121 para cada *pixel* da região analisada e em nove camadas.

Outro dado interessante que a Tabela 7 revela ao analisar o tamanho do *kernel* fixo em sete, que se revela fora da tendência identificada de decaimento da taxa de erro conforme o crescimento do tamanho do *kernel*. Tal fato pode ser explicado devida a natureza da estrutura que não convergiu com a taxa esperada, é possível, e altamente provável, que os valores se alterem caso a metodologia seja executada várias vezes, contudo

espera-se que os valores fiquem dentro de um intervalo esperado, a causa da taxa de erro destoante da tendência pode ser atribuída a processos que estavam sendo executados paralelamente e que alteraram o estado da máquina influenciando o resultado final, tanto para a taxa de erro quanto para o perda. Para se certificar que a real taxa de erro e perda, o procedimento contido na metodologia precisa ser repetido mais vezes de modo a se obter valores estatisticamente corretos.

Outro fato interessante de se comentar é em relação a perda dos modelos gerados, onde ela vem decaindo conforme o tamanho do *kernel* aumenta, tendo a minimização das perdas dentre o valores amostrados, ocorrido no *kernel* de tamanho nove. A perda do modelo na realidade busca maximizar a acurácia pela minimização do erro, contudo nem sempre o modelo de menor perda resulta no de melhor acurácia, a Tabela 7 deixa isso claro pois a menor taxa de erro é o modelo de *kernel* 11 enquanto que a menor perda é o *kernel* de tamanho 9. A forma como a taxa de erro se relaciona com a perda do modelo se dá pelo fato de que a taxa de erro é obtida pela minimização da perda do mesmo, logo elas estão intrinsecamente ligadas.

5.3 Alteração do Tamanho do *Kernel* de Amostragem

Conforme apresentado na seção de metodologia o procedimento experimental seguiu exatamente o que foi seguido de modo que os resultados obtidos para a taxa de erro estéreo e perda se encontram na Tabela 8.

Tabela 8 – Taxa de erro em função da variação do tamanho do *kernel* de amostragem.

Tamanho do *kernel*	Taxa de Erro	Perda
3	5,510	1,964
5	4,176	1,755
7	5,186	2,253
9	6,276	2,261

Observando a Tabela 8 é interessante notar que o melhor resultado é obtido quando o tamanho do *kernel* de amostragem é igual ao do convolucional, também é interessante o fato de que houve uma melhora nos índices três e cinco e uma piora em sete e nove quando comparados com rede de *kernel* fixado em cinco.

A relativa melhora nos valor da taxa de erro para o índice três pode ser explicada pelo fato de que a camada de amostragem com *stride* fixo em um e aplicada para pequenas regiões, ou seja, valores de *kernel* pequenos, somando isso ao fato de que a camada estava configurada para extrair o maior valor da região. Todas essas condições reunidas conseguem um realce melhor pois a riqueza de detalhes numa região menor é obviamente menor, pois ela é mais homogênea, o que resulta na extração do mesmo *pixel* porém realçado. Tal hipótese se mostra razoável para explicar tal efeito.

Com relação ao índice cinco, o ótimo desempenho pode ser explicado pelo mesmo argumento usado para explanar o *kernel* de tamanho três, contudo para o tamanho em questão deve-se considerar o balanceamento entre a região analisada e detalhes presentes na imagem que formaram o melhor filtro para encontrar *pixels* correspondentes dentro de uma imagem estéreo. No caso do *kernel* de tamanho cinco os dados obtidos e disponibilizados na Tabela 8 indicam que a relação foi maximizada e obteve o seu melhor resultado, apontando para que o *kernel* de amostragem cinco possui a melhor relação entre região analisada e detalhes presentes.

Para os índices sete e nove, os valores obtidos e disponibilizados na Tabela 8 novamente podem ser explicados pelo mesmo argumento utilizado pelos outros tamanhos. Neste caso fica claro observando a Tabela 8 para os tamanhos de *kernel* em questão a relação entre região analisada e detalhes presente é ruim e indica piora na taxa de erro, o fato do tamanho do *kernel* de amostragem ser relativamente grande, faz com se tenha uma grande quantidade de *pixels* na região e no caso de encontrar uma região rica em detalhes, ou heterogenia, o valor do *pixel* extraído nem sempre é o mesmo analisado pois neste experimento a extração se deu pelo valor máximo, e tal valor pode não ser um mero realce do pixel, mas sim o *pixel* de outra parte do *kernel* de amostragem.

Outro fato notório contido na Tabela 8 é predileção por tamanhos ímpares de amostragem. Tal ocorrência pode ser atribuída ao ao preprocessamento, pois nesta fase é realizado o procedimento de preparar corretamente os dados e para tamanhos de *kernel* de amostragem com valores pares fossem possíveis o preprocessamento deveria gerar uma imagem com medidas invalidas para o restante da rede, ou seja, para se utilizar um *kernel* de amostragem par o preprocessamento deve gerar uma imagem incompatível com o resto da rede, e como rede foi montada buscando preservar os aspectos originais só foram avaliados valores impares para o *kernel* de amostragem.

É interessante comentar também os valores de perdas obtidos pelos modelos construídos. O modelo de menor valor de perda neste experimento foi de *kernel* de amostragem com tamanho cinco, por ventura este valor também foi o de melhor taxa de erro, logo este tamanho representa a melhor minimização das perdas e maximização da acurácia, situação diametralmente oposta ao *kernel* de tamanho nove, que obteve os maiores valores de perda e taxa de erro dentre os valores analisados, ou seja, o tamanho nove obteve os piores valores na tentativa de maximiza-los.

É relativo dizer se os valores obtidos e disponibilizados na Tabela 8, são bons ou ruins, pois tudo depende do que se está comprando. A melhor analise que podemos fazer é compara-los uns com os outros pois assim garantimos que há coesão entre as métricas. De fato o melhor valor é aquele com *kernel* de amostragem com tamanho e cinco enquanto o pior é o de tamanho nove.

5.4 Alteração da Técnica de Amostragem

Conforme apresentado na seção de metodologia o procedimento experimental seguiu exatamente o que foi seguido de modo que os resultados obtidos para a taxa de erro estéreo e perda se encontram na Tabela 9.

Tabela 9 – Taxa de erro e perda para extração do valor médio amostrado.

Tamanho *Kernel*	Taxa de erro		Perda	
	Máximo	Médio	Máximo	Médio
3	5,510	6,734	1,964	1,881
5	4,176	9,406	1,755	1,719
7	5,186	3,862	2,253	1,994
9	6,276	7,432	2,261	2,088

Observando a Tabela 9 podemos comparar os valores para as duas técnicas de amostragem diferentes, uma buscando extrair o valor máximo da área analisada pelo *kernel* de amostragem, cujos valores são os mesmos do experimento sobre análise do efeito da camada de amostragem sobre a estrutura; e outro que extrai o valor médio da região analisada. Tal comparação pode ser feita pois *kernel* convolucional foi fixado em cinco e com *stride* convolucional e de amostragem fixados em um, sendo variado somente os valores do *kernel* de amostragem para ambas as técnicas de amostragem.

Analisando a acurácia dos valores encontrados na Tabela 9 fica constatado o melhor desempenho da extração do valor máximo, tendo esta sendo superior na maioria dos caso, com a única exceção quando o *kernel* de amostragem foi sete. Tal fato mostra que a extração do valor máximo é a melhor opção para o caso analisado.

O melhor desempenho da extração do valor máximo pode ser atribuído ao fato de que sua extração preserva o valor convolucional das camadas anteriores operadas sobre a imagem de entrada. Como a operação convolucional é responsável por enaltecer as características da imagem, a extração do valor médio quebra esse realce de modo a harmonizar a com a região harmonizada, logo a correlação produzida não encontra valores melhores que a extração do valor máximo

Contudo é preciso ter em mente que ao extração de qualquer valor de uma região, o algoritmo estará selecionando um dado a partir daqueles amostrados que busca representar aquele universo. Um cuidado que deve ser tomado ao utilizar o valor máximo, são zonas heterogenias pois o valor máximo pode ser o mesmo para toda região escolhida para análise.

O mesmo fenômeno se repete com a perda do modelo, porém dessa vez somente o *kernel* de tamanho três obteve melhores para para a extração do valor máximo do que do médio. Desta forma fica evidente que ao utilizar o valor médio o modelo é capaz de minimizar as perdas.

Apesar do experimento ter funcionado e obtido valores passíveis de análise é necessário refletir sobre a necessidade de do uso da camada de amostragem e suas técnicas. Dentro do escopo do problema proposto, ou seja, uso de rede neural convolucional capaz de correlacionar os *pixels* do par estereoscópico dentro de uma região, logo o uso de uma camada capaz de extrair *pixels* significa selecionar dados conforme um critério e necessariamente implica em perda de informação original, fazendo com que a correlação seja executada com dados que não são originais daquela posição, em outras palavras, a camada de amostragem altera a posição dos *pixels* logo a DSI produzida não condiz com a realidade.

6 Conclusão

Após seguir o procedimento adotado na secção de Metodologia e analisar os resultados na secção correspondente, chega o momento de concluir o estudo. Os experimento foram todos bem sucedidos e os resultados obtidos foram suficientes para fornecerem uma análise da influência dos parâmetros de configuração de um de neural convolucional.

A reprodutibilidade do artigo original [3] foi garantida no momento em que a taxa de erro obtida é próxima do original e o tempo necessário para realizar uma predição ficou um pouco abaixo do original. Como o experimento não foi realizado sob as mesmas condições do original, ou seja, o *hardware* utilizado era diferente, logo era esperado uma variação nas métricas, mas como a variação foi pequena é seguro afirmar que a reprodução do experimento original foi um sucesso.

Ao todo foram realizados cinco experimentos, dois referentes a influência do *stride* se encontram no Apêndice A por não terem uma base de comparação adequada e por não ser possível realizar todos os experimentos propostos devido a limitações do *hardware*.Com relação aos outros três, cujo foco era investigar a influência do tamanho do *kernel* e da política de amostragem sobre o modelo criado foram bem sucedidos, pois produziram valores capazes de serem analisados de modo que fosse possível medir o efeito da alteração de parâmetros sobre as métricas de taxa de erro e perda para cada experimento realizado.

O primeiro experimento buscou analisar o efeito que o tamanho do *kernel* convolucional tem sobre as métricas selecionadas. Neste caso a base de comparação foram os resultados obtido pelo artigo original, onde fica estabelecido um *kernel* de tamanho cinco como referência de medida. O resultado obtido revela a tendência em de atingir melhores valores de acurácia e perda conforme o tamanho do *kernel* aumenta, ou seja, o valor da acurácia é diretamente proporcional ao tamanho do *kernel* escolhido, porém isso é somente uma tendência, para se confirmar como fato será necessário a repetição dos testes por número maior de vezes para que se possa confirmar como fato.

A melhora da acurácia com o aumento do tamanho do *kernel* possui um custo relacionado ao período de treinamento, sendo esta relação diretamente proporcional, ou seja, o aumento do *kernel* convolucional provoca um aumento no tempo de treinamento, isso implica diretamente no valor da acurácia, pois apesar do tamanho *kernel* melhorar a acurácia leva-se um tempo maior para executar tal tarefa. O maior valor testado obteve a melhor métrica de acurácia porém levou aproximadamente 20 horas para treinar.

O segundo experimento investigou o efeito da camada de amostragem em uma rede neural convolucional sobre o problema proposto. Este foi o ponto mais crítico de todos os experimentos realizados pois o original proposto em [3] não possuía uma camada de

amostragem e a adição da camada constituiria uma mudança estrutural dentro do modelo. Guiado sempre pela premissa de baixa intervenção estrutural foi adicionado ao final da estrutura original a camada de amostragem.

Os resultados obtidos indicam uma melhora na métricas quando comparados com a medida de referência, que é *kernel* convolucional fixado em cinco e sem amostragem, tal fato contrasta com a hipótese levantada pois não era esperada uma melhora de índices, e sim piora. Conforme explanada na secção de resultados a melhora nos resultados podem ser atribuídas a relação entre realce provocado pela ação convolucional e a quantidade de detalhe da região analisada, este argumento parece razoável para explicar os resultados.

O terceiro e último experimento buscou investigar o efeito que a técnica de extração de valores na camada de amostragem tem sobre o modelo construído. Ficou constado que a extração do valor máximo possui desempenho melhor que a extração do valor médio. A hipótese inicial esperava que o valor médio obtivesse um resultado melhor pois ele busca harmonizar a região, contudo o melhor resultado do valor máximo pode ser atribuído ao fato de que ele deixa a DSI final mais homogenia e acaba produzindo uma correlação maior.

Dos três experimentos realizados, dois investigaram parâmetros de configuração da camada de amostragem e refletem sobre seu uso. A função da camada como o próprio nome já diz é amostrar valores de uma dada região, ou seja, extrair um valor de uma determinada área analisada de acordo com critério específico, no caso do experimentos realizados valor máximo ou médio, logo por definição o resultado que o uso da camada produz é uma imagem com valores amostrados da original e consequentemente com falta de informação original.

O escopo do problema não permite perdas de informação durante o processo pois ao final do modelo será feito o produto escalar do par estereoscópico de modo a obter a maior correlação entre os *pixels*, logo todo o uso da camada de amostragem não faz sentido dentro do problema analisado. As DSI produzidas pelo experimentos que fazem uso da camada de amostragem são claras quanto a isso, pois elas não revelam uma DSI clara e útil.

Por outro lado a camada convolucional se revelou extremamente útil ao problema e foi capaz de gerar DSI coerentes com a realidade, em alguns caso obtendo resultados superiores que o artigo original, apesar de haver um custo temporal alto.

É justo dizer que este trabalho atingiu com sucesso tudo que ele se propôs a fazer, pois todos os experimentos produziram resultados valiosos para análise e através deles hipóteses elencadas na secção de Metodologia foram confirmadas e falseadas, e suas razões exploradas de modo a obter total compreensão da influência da alteração de determinados parâmetros sobre o resultado final do modelo e aplicação dentro do problema proposto.

Com estas palavras finalizo o trabalho em que espero ter contribuído um pouco para o tema e que possa servir de referência para futuros projetos da área. Que sirva de guia sobre a função de cada camada de uma rede neural convolucional e seus principais parâmetros de configuração. Espero que ajude a quem interessar possa.

Referências

1 SCHARSTEIN, R. S. e. R. Z. D. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Proceedings IEEE Workshop on Stereo and Multi-Baseline Vision (SMBV 2001)*, 2001. Citado 7 vezes nas páginas 11, 5, 6, 7, 8, 9 e 10.

2 ZBONTAR, J.; LECUN, Y. Computing the stereo matching cost with a convolutional neural network. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2015. p. 1592–1599. Citado 7 vezes nas páginas 11, 1, 2, 3, 12, 13 e 15.

3 LUO, W.; SCHWING, A. G.; URTASUN, R. Efficient deep learning for stereo matching. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2016. p. 5695–5703. Citado 14 vezes nas páginas 11, 13, 3, 4, 6, 14, 25, 26, 27, 28, 29, 43, 53 e 57.

4 CHEN, Y.-N.; CHUAN, C.-H.; FAN, K.-C. Stereo-based 3d space handwriting recognition. In: IEEE. *2018 32nd International Conference on Advanced Information Networking and Applications Workshops (WAINA)*. [S.l.], 2018. p. 615–617. Citado na página 1.

5 SZELISKI, R. *Computer vision: algorithms and applications*. [S.l.]: Springer Science & Business Media, 2010. Citado na página 1.

6 ZHONG, Y.; LI, H.; DAI, Y. *Open-World Stereo Video Matching with Deep RNN*. 2018. Citado na página 3.

7 MAYER, N. et al. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2016. p. 4040–4048. Citado na página 3.

8 SEKI, A.; POLLEFEYS, M. Sgm-nets: Semi-global matching with neural networks. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2017. p. 231–240. Citado na página 3.

9 PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011. Citado na página 3.

10 ABADI, M. et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. 2015. Software available from tensorflow.org. Disponível em: <https://www.tensorflow.org/>. Citado 2 vezes nas páginas 3 e 27.

11 BOLLES, R. C.; BAKER, H. H.; MARIMONT, D. H. Epipolar image analysis: An approach to determine structure from motion. *International Journal of Computer Vision*, 1987. Citado na página 6.

12 OKUTOMI, M.; KANADE, T. A multiple-baseline stereo. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, IEEE, n. 4, p. 353–363, 1993. Citado na página 6.

13 BOBICK AARON F E INTILLE, S. S. Large occlusion stereo. *International Journal of Computer Vision*, Springer, v. 33, n. 3, p. 181–200, 1999. Citado na página 6.

14 YANG, Y.; YUILLE ALAN E LU, J. Local, global, and multilevel stereo matching. In: IEEE. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.], 1993. p. 274–279. Citado na página 6.

15 ANANDAN, P. A computational framework and an algorithm for the measurement of visual motion. *International Journal of Computer Vision*, Springer, v. 2, n. 3, p. 283–310, 1989. Citado na página 7.

16 HANNAH, M. J. *Computer matching of areas in stereo images*. [S.l.], 1974. Citado na página 7.

17 KANADE, T. et al. Development of a video-rate stereo machine. In: IEEE. *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots*. [S.l.], 1995. v. 3, p. 95–100. Citado na página 7.

18 PANG, J. et al. Cascade residual learning: A two-stage convolutional neural network for stereo matching. In: *Proceedings of the IEEE International Conference on Computer Vision*. [S.l.: s.n.], 2017. p. 887–895. Citado 3 vezes nas páginas 8, 15 e 16.

19 GONG, M. et al. A performance study on different cost aggregation approaches used in real-time stereo matching. *International Journal of Computer Vision*, Springer, v. 75, n. 2, p. 283–296, 2007. Citado na página 8.

20 BOBICK, A.; INTILLE, S. S. Large occlusion stereo. *International Journal of Computer Vision*, v. 33, p. 181–200, 09 1999. Citado 3 vezes nas páginas 8, 9 e 10.

21 TERZOPOULOS, D. Regularization of inverse visual problems involving discontinuities. *IEEE Transactions on pattern analysis and Machine Intelligence*, IEEE, n. 4, p. 413–424, 1986. Citado na página 8.

22 VEKSLER, O.; ZABIH, R. Efficient graph-based energy minimization methods in computer vision. Cornell University New York, USA, 1999. Citado na página 9.

23 KANADE, T.; OKUTOMI, M. A stereo matching algorithm with an adaptive window: Theory and experiment. In: IEEE. *Proceedings. 1991 IEEE International Conference on Robotics and Automation*. [S.l.], 1991. p. 1088–1095. Citado na página 10.

24 SUN, J. et al. Symmetric stereo matching for occlusion handling. In: CITESEER. *CVPR (2)*. [S.l.], 2005. p. 399–406. Citado 2 vezes nas páginas 10 e 11.

25 HIRSCHMULLER, H.; SCHARSTEIN, D. Evaluation of stereo matching costs on images with radiometric differences. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, v. 31, n. 9, p. 1582–1599, 2009. Citado na página 11.

26 SCHENNINGS, J. *Deep convolutional neural networks for real-time single frame monocular depth estimation*. 2017. Citado 2 vezes nas páginas 11 e 15.

27 ZAGORUYKO, S.; KOMODAKIS, N. Learning to compare image patches via convolutional neural networks. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2015. p. 4353–4361. Citado na página 14.

28 SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. Citado na página 14.

29 JORDAN, T. S.; SHRIDHAR, S.; THATTE, J. Usings cnns to estimate depth from stereo imagery. Citado na página 15.

30 ZHANG, K.; LU, J.; LAFRUIT, G. Cross-based local stereo matching using orthogonal integral images. *IEEE transactions on circuits and systems for video technology*, IEEE, v. 19, n. 7, p. 1073–1079, 2009. Citado na página 15.

31 HE, K. et al. Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2016. p. 770–778. Citado na página 15.

32 GEIGER, A. et al. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013. Citado na página 16.

33 MENZE, M.; GEIGER, A. Object scene flow for autonomous vehicles. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2015. Citado 2 vezes nas páginas 16 e 27.

34 MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, Springer, v. 5, n. 4, p. 115–133, 1943. Citado na página 17.

35 KLEENE, S. C. *Representation of events in nerve nets and finite automata*. [S.l.], 1951. Citado na página 17.

36 ROSENBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, American Psychological Association, v. 65, n. 6, p. 386, 1958. Citado na página 17.

37 MINSKY, M.; PAPERT, S. A. Artificial intelligence progress report. 1972. Citado na página 17.

38 WERBOS, P. Beyond regression:" new tools for prediction and analysis in the behavioral sciences. *Ph. D. dissertation, Harvard University*, 1974. Citado na página 17.

39 WANG, Y. *stereo_matching*. Github, 2019. Disponível em: <https://github.com/wangy12/stereo_matching>. Citado na página 26.

Apêndices

APÊNDICE A – Experimentos relativos ao *Stride* das camadas convolucionais e de amostragem

Os experimentos relacionados a verificar a influência do *stride* nas métricas do modelo não possuem uma base comparação de adequada no presente trabalho e limitação do *hardware* e do *software* de pré processamento não permitiram a aquisição de valores diferentes de *stride*. Com isso em mente mostrou-se congruente separar os resultados e disponibilizar os relativos ao *stride* sob o formato de apêndice, desta forma, nas seções abaixo estão presentes a metodologia utilizada, os resultados obtidos e algumas considerações sobre eles para quem interessar possa em reproduzir e ter uma base de comparação em experimentos relacionados.

A.1 Metodologia

A.1.1 Alteração do *Stride* do *Kernel*

A analise da variação do *stride* convolucional tem como objetivo verificar a influência do tamanho do passo do avanço do *kernel* tem sobre o resultado final. Tal experimento precisou ser elaborado com cuidado para não prejudicar a estrutura original da relatada em [3].

A estrutura original conta com nove camadas convolucionais com *kernel* de tamanho cinco e *stride* com passo igual a um. O experimento busca variar o passo do *kernel* convolucional de dois até cinco, pois é o valor máximo que faz sentido para o *kernel* fixado, o *stride* de valor um foi ignorado pois este é valor padrão, logo, ele possui o mesmo valor do experimento anterior onde foi aplicado o mesmo tamanho de *kernel* que este. A variação do *stride* provoca uma redução maior da dimensionalidade, pois está sendo selecionado e extraído uma parte da imagem, logo, se todas as nove camadas variassem o *stride* a imagem usada como entrada da rede deveria ser maior do que ela própria para que a rede fornecesse um resultado.

Para resolver esse problema e ao mesmo tempo buscando sempre preservar a estrutura original foi adicionado na última camada convolucional a propriedade de *stride*, desta forma será possível avaliar o efeito do atributo no resultado final, a base de comparação a ser utilizada será os valores obtidos de acurácia e perda para o *kernel* convolucional de tamanho cinco.

Devido a limitações técnicas não foi possível buscar a máxima preservação da estrutura da rede original, tais limitações estão ligadas a quantidade de memória a ser alocada para gerar o modelo final. Logo foi constado que o hardware utilizado só era capaz de gerar um modelo com *kernel* convolucional de tamanho três com *stride* convolucional também três. Qualquer valor maior tornou-se inviável pela limitação do hardware e também pelo fato de que o preprocessamento extrapola os limites da imagem.

Desta forma o procedimento elaborado busca fixar o *kernel* convolucional em três e apenas adicionar na última camada convolucional a propriedade de *stride*, e realizar um treinamento para o valor fixo de *stride* igual a três, de modo a obter os valores de acurácia e perda para o treinamento realizado, para ao final comparar com os valores do modelo criado sem o *stride* variante, ou seja, *stride* fixado em um. A Figura 20.

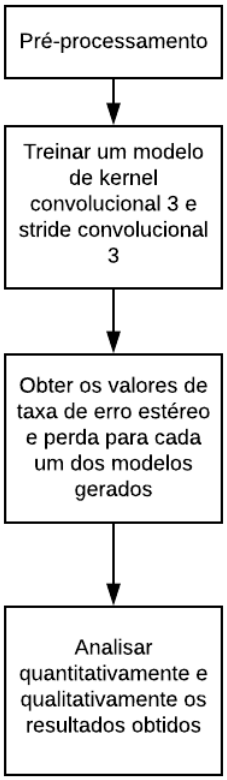

Figura 20 – Fluxograma do processo de alteração do *stride* convolucional.

A hipótese a ser testada neste experimento é influência do *stride* convolucional sobre os valore de acurácia e perda da rede montada para o experimento. O resultado esperado para este experimento são valores altos ou nulos para taxa de erro estéreo e perda devido a natureza do *stride* de perder informação ao longo do processo da rede.

A.1.2 Alteração do *Stride* de Amostragem

Para a verificação da influência do *stride* de amostragem sob uma rede neural convolucional os problemas enfrentados são os mesmo da secção de verificação do tamanho do *kernel* de amostragem, ou seja, o fato da rede original não possuir uma camada de amostragem constituiu um desafio técnico pois mudanças estruturais podem acarretar em perda de similaridade com a rede original. Com isto em mente o procedimento adotado buscou ser o menos intrusivo possível.

Assim como na alteração do *stride* convolucional, também há limitações técnicas que impediram a preservação da estrutura do modelo original pelos mesmos motivos citados na seção anterior, ou seja, limitação de memória do hardware escolhido e que pré processamento extrapola os limites d imagem. Desta forma optou-se por utilizar um *kernel* convolucional de tamanho três bem como o *kernel* de amostragem de tamanho também igual a três.

Tomando como base a rede original, com nove camada convolucionais em sequência, foi inserido após a última camada convolucional uma camada de amostragem, afim de obter a influência do *stride* de amostragem, os parâmetros da rede foram fixados, mantendo o *kernel* convolucional com tamanho três e *stride* um, na camada de amostragem foi fixado o tamanho *kernel* em três e o *stride* fixo em três também pois era o máximo que o hardware suporta sem estourar memória e a própria imagem.

O procedimento adotado buscou ao máximo preservar as características da rede original, pois isso os parâmetros convolucionais foram fixados. O tamanho do *kernel* de amostragem foi fixado em três, pois ele era o menor tamanho permitido e é necessário que sua influência seja mínima para avaliar somente o efeito de *stride* de amostragem.

A hipótese a ser testada neste experimento busca verificar a influência do *stride* de amostragem sobre as métricas de taxa de erro estéreo e perda. É esperado que os valores de taxa de erro e perda sejam altos ou nulos devido a natureza do *stride* e da camada de amostragem, que geram perdas de informação.

Uma vez de posse dos números, será feita a análise qualitativa e quantitativa dos dados. A Figura 21 constitui o fluxograma de procedimentos adotados para este experimento e ilustra de modo visual todo o processo a ser realizado.

A.2 Resultados

A.2.1 Alteração do *Stride* Convolucional

Seguindo a metodologia apresentada na secção acima, o resultado da taxa de erro e perda do modelo em função do *stride* convolucional se encontra na Tabela 10, localizada

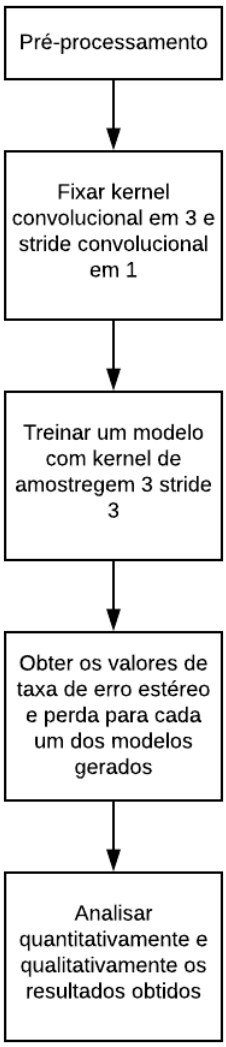

Figura 21 – Fluxo do processo de alteração do tamanho do *stride* de amostragem.

logo abaixo.

Tabela 10 – Taxa de erro e perda em função da variação da variação do *stride* convolucional.

Stride	Taxa de Erro	Perda
3	0	1,651
5	-	-
7	-	-
9	-	-

Analisando a Tabela 10 é notório o fato de que a Tabela não está totalmente preenchida. Assim como descrito na secção de Metodologia, isso ocorreu devido a limitações técnicas de hardware pois no momento de treinamento do modelo é alocado mais memória do que o disponível, que por sua vez isso acontece porque a etapa de preprocessamento extrapola os limites da imagens para *stride* maior que três. Para permitir valores de *stride*

superiores a três seria necessário modificar radicalmente a estrutura original[3].

No que tange aos valores obtidos, a hipótese inicial se comprovou verdadeira ao analisar o valor da taxa de erro, pois a função do *stride* é fazer com que o *kernel* se mova mais rapidamente sobre a matriz, isso causa a redução de dimensionalidade dos dados e por conseguinte a perda de parte da informação necessária para atingir o objetivo final da rede.

A intenção do artigo original era montar uma rede convolucional capaz de identificar os *pixel* de uma par estereoscópico sendo capaz de montar uma DSI. O uso do *stride* na camada convolucional para esse propósito é irrelevante e prejudicial pois não é admissível a perda de informação durante o processo de treinamento pois é preciso que todos os *pixels* estejam presentes para que a correlação faça sentido e se monte uma imagem estéreo.

Portanto o valor de zero para a taxa de erro é coerente, pois a perda de informação gerada pelo *stride* impede que a correlação entre os *pixels* funcione corretamente, pois parte dos *pixels* não estão presentes para formar a DSI.

A.2.2 Alteração do *Stride* de Amostragem

Seguindo a metodologia apresentada na secção acima, o resultado da taxa de erro e perda do modelo em função do *stride* de amostragem se encontra na Tabela 11, localizada logo abaixo.

Tabela 11 – Taxa de erro em função da variação do tamanho do *stride* de amostragem.

Tamanho do *kernel*	Taxa de Erro	Perda
3	0	1,726
5	-	-
7	-	-
9	-	-

Analisando a Tabela 11 é nítido que ela não está completa, e a razão de tal ocorrência é similar aos motivos apresentados para justificar a mesma ocorrência na Tabela 10. Como descrito na secção de Metodologia, limitações técnicas foram constituem o principal motivo por só haver uma linha da preenchida na Tabela, pois durante o pré processamento a alocação de memória para valores de *stride* superiores a três não eram possíveis pois a placa de vídeo não tinha disponível o quantidade necessária.

A alocação excessiva de memória foi causada pois, ainda durante o pré processamento as dimensões da imagem foram extrapoladas, ou seja, para que o modelo conseguisse gerar uma saída na dimensão esperada a imagem deveria ser muito maior do que ela realmente era, devido a este fato somente o *stride* igual a três foi capaz de gerar um

modelo compatível pois durante o pré processamento a imagem ficou dentro dos limites da imagem.

Um ponto que merece destaque é o fato da Tabela 11 começar no *stride* três. A ausência do valor dois tem origem novamente na fase de pré processamento. Durante o experimento ficou constatado que para índices pares o pré processamento gera dados incompatíveis com a estrutura do modelo utilizado, em outras palavras o modelo não consegue gerar uma saída compatível devida a incompatibilidade dimensional dos dados ingeridos no modelo, ou seja, a estrutura utilizada não funciona para valores pares pois pré processamento realizado para tais valores não gera valores adequados.

Apesar das justificativas serem similares para explicar os valores nas Tabelas 10 e 11 existem particularidades que merecem atenção. O *stride* de amostragem tem como função percorrer a imagem de modo que o *kernel* extraia um determinado valor que representa a área percorrida pelo *kernel*, a partir dessa premissa o uso de *stride* na camada de amostragem faz total sentido quando ele é igual ou superior ao tamanho de *kernel* de amostragem pois isso garante que não serão percorridos áreas já amostradas.

Como relação ao valores obtidos a hipótese inicial foi confirmada para o valor da taxa de erro, pois o uso de *stride* implica em perda de informação, no caso específico de seu uso na camada de amostragem, significa extrair um valor que represente uma determinada área, e como a premissa do cálculo da taxa de erro é ao final do final do modelo realizar um produto escalar entre a imagem direita e esquerda, fica evidente que a perda de informação impacta negativamente e com força no resultado final da taxa de erro. Portanto o valor de zero é coerente com o esperado e confirma a hipótese e inicial.

O valor da perda busca minimizar o valor da taxa de erro durante o processo de treinamento. A primeira vista pode parecer incoerente o valor da perda ser diferente da taxa de erro uma vez que o resultado indica que não há acerto algum, contudo é preciso considerar que a perda é uma minimização matemática que busca diminuir os erros durante o processo não implicando na taxa de erro final.

A.3 Considerações sobre o uso de *stride*

O primeiro experimento deste apêndice buscou analisar o efeito do *stride* na camada convolucional. O experimento confirmou a hipótese levantada na secção de Metodologia de que o uso de *stride* acarretaria em um modelo ruim com acurácia próxima de zero, pois o efeito que ele possui implica em perda de informação pois deixa de analisar algumas regiões devido a movimentação de *kernel* sobre a área analisada, logo o resultado obtido faz sentido dentro do contexto de analise.

O segundo experimento contido neste apêndice buscou analisar o efeito de *stride*

dentro da camada convolucional. Os resultados obtidos confirmaram a hipótese inicial, pois era esperados uma acurácia baixa ou nula pois o uso do *stride* implica em perda de informação, e devido a essa perda o produto escalar realizado não era capaz de encontrar os *pixels* de maior correlação. Através do experimento ficou constatado esse fato pois a acurácia obtida foi nula e os motivos são coesos com a realidade.

Este foi um caso peculiar onde o uso do *stride* faz todo sentido, pois tendo em vista que a função da camada de amostragem é selecionar um valor que represente uma área, e o uso correto do *stride* impede de amostrar o mesmo valor várias vezes, contudo o escopo do problema não permitia tal utilização e seu uso acarretou em um modelo de acurácia nula, totalmente inutilizável.

O uso do *stride* se mostrou ruim diante do problema em questão pois para ambos os caso testado não conseguiu produzir algo útil porem o experimento deixou claro que seu uso não faz sentido dentro do escopo do problema. A camada de amostragem produziu modelos satisfatórios porém DSI incoerentes com a realidade devido a própria natureza da camada pois há perda de informação, algo que o problema não permite.