UNIVERSIDADE FEDERAL DE PELOTAS
Centro de Desenvolvimento Tecnológico
Programa de Pós-Graduação em Computação

Tese

Extração de Mapas de Profundidades de *Dense Light Fields* usando *Deep Learning*

Anderson Priebe Ferrugem

Pelotas, 2022

Anderson Priebe Ferrugem

Extração de Mapas de Profundidades de *Dense Light Fields* usando *Deep Learning*

Tese apresentada ao Programa de Pós-Graduação em Computação do Centro de Desenvolvimento Tecnológico da Universidade Federal de Pelotas, como requisito parcial à obtenção do título de Doutor em Ciência da Computação.

Orientador: Prof. Dr. Bruno Zatt
Coorientador: Prof. Dr. Luciano Volcan Agostini

Pelotas, 2022

Universidade Federal de Pelotas / Sistema de Bibliotecas
Catalogação na Publicação

F399e Ferrugem, Anderson Priebe

Extração de mapas de profundidades de dense light fields usando deep learning / Anderson Priebe Ferrugem ; Bruno Zatt, orientador ; Luciano Volcan Agostini, coorientador. — Pelotas, 2022.

163 f. : il.

Tese (Doutorado) — Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, 2022.

1. Campo de luz denso. 2. Função plenóptica. 3. Aprendizado profundo. 4. Redes neurais artificiais. 5. Visão computacional. I. Zatt, Bruno, orient. II. Agostini, Luciano Volcan, coorient. III. Título.

CDD : 005

Elaborada por Simone Godinho Maisonave CRB: 10/1733

Anderson Priebe Ferrugem

Extração de Mapas de Profundidades de *Dense Light Fields* usando *Deep Learning*

Tese aprovada, como requisito parcial, para obtenção do grau de Doutor em Ciência da Computação, Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas.

Data da Defesa: 25 de novembro de 2022

Banca Examinadora:

Prof. Dr. Bruno Zatt (orientador)
Doutor em Microeletrônica pela Universidade Federal do Rio Grande do Sul.

Prof. Dr. Luciano Volcan Agostini (coorientador)
Doutor em Ciências da Computação pela Universidade Federal do Rio Grande do Sul.

Prof. Dr. Edson Mintsu Hung
Doutor em Engenharia de Sistemas Eletrônicos e de Automação pela Universidade de Brasília.

Prof. Dr. Guilherme Ribeiro Corrêa
Doutor em Engenharia Electrotécnica e de Computadores pela Universidade de Coimbra, Portugal.

Prof. Dr. Ismael Seidel
Doutor em Ciências da Computação pela Universidade Federal de Santa Catarina.

Dedico esse trabalho

A minha esposa

Andréia

Aos meus pais

Hélio (*in memoriam*) e **Loraci**

Aos meus irmãos

Robinson (*in memoriam*) e **Michele**

Ao meu filho

Theo

AGRADECIMENTOS

Nenhuma maratona da vida é trilhada de forma solitária, a vida não é uma corrida individual, é um campeonato de equipes, sem linha de chegada, com passagem de bastão e se necessário carregamento no colo.

Sem minha amada esposa **Andréia** jamais teria chegado aqui. Sem teu apoio estaria rastejando no chão. Com ele, pude sonhar e voar. No início de minha caminhada, meu querido pai **Hélio**, hoje não mais ao meu lado, apontou o caminho certo e justo; minha adorável mãe **Loraci** me tirou do berço e me carregou nos braços até eu aprender a andar sozinho. Na minha caminhada tive o prazer de contar com minha carinhosa irmã **Michele** e meu saudoso irmão **Robinson**. Como todo maratonista, aprendi o peso da responsabilidade com meu querido filho **Theo**, quem disser que correr com esse peso torna a maratona mais dura não sabe nada de paternidade. Nesse caminho pude contar com mestres que se tornaram amigos, como meus professores **Gil Medeiros** e **Dante Barone**. Tive o prazer de ter também amigos que se tornaram professores como **Bruno Zatt** e **Luciano Agostini** e também colegas que se tornaram amigos como o professor **Marilton Sanchotene**. Enfim, não sei quando minha maratona termina, mas posso dizer que foi um privilégio percorrer ela com essa companhia.

A todos eles meu profundo e estimado agradecimento.

"Beware of his false knowledge: it is more dangerous than ignorance."
— GEORGE BERNARD SHAW

RESUMO

FERRUGEM, Anderson Priebe. **Extração de Mapas de Profundidades de *Dense Light Fields* usando *Deep Learning***. Orientador: Bruno Zatt. 2022. 163 f. Tese (Doutorado em Ciência da Computação) – Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, Pelotas, 2022.

Diversos sistemas de imageamento utilizam a **metrologia por imagem** para medir, identificar, inspecionar e diagnosticar. A demanda crescente por sistemas de metrologia visual em diversas áreas que necessitam de sensores compactos e robustos tem impulsionado o desenvolvimento de dispositivos para captura. Estes sensores utilizam diferentes grandezas físicas para o imageamento e cálculo das distâncias, cada um com suas limitações e vantagens. Entre as tecnologias emergentes de imageamento, que usam apenas a informação de luz visível, temos destaque para o uso de imagens *light field* capturadas através de câmeras *light field* densas ou esparsas, que possuem vantagens intrínsecas em relação aos dispositivos tradicionais. Por exemplo, essas câmeras são robustas em situações específicas de oclusão e também em cenas com ambientes ruidosos (chuva, neve, etc.). Isso faz com que câmeras *light field*, também chamadas de câmeras plenópticas, possuam potencial de uso como um versátil sensor com múltiplas aplicações. Essa capacidade é pouco aproveitada devido as características ópticas complexas relacionadas ao sistema de captura e ao custo computacional do processamento relacionado. Para se extrair o mapa de profundidade usando métodos geométricos tradicionais é necessário estimar n-variáveis, atualizar seus valores e realizar novos cálculos a cada mudança de parâmetro. Ao se usar redes neurais artificiais, essas relações já ficam implícitas na própria rede neural, o que permite uma resposta imediata a modificação dinâmica dos parâmetros. Essa tese apresenta duas técnicas para extração de mapas de profundidade de imagens *light field* densas baseadas em redes neurais com aprendizado profundo. A primeira proposta simplifica a rede EPINET, reduzindo o fluxo de quatro entradas para apenas duas entradas. Já a segunda proposta explora a rede de entrada multifluxo em uma rede neural convolucional *u-shaped*. Cada proposta é explorada e por fim são apresentadas suas vantagens e desvantagens. Ambas propostas calculam mapas de profundidade em tempos menores que a EPINET original.

Palavras-chave: campo de luz denso. função plenóptica. aprendizado profundo. redes neurais artificiais. visão computacional.

ABSTRACT

FERRUGEM, Anderson Priebe. **Depth Map Extraction of Dense Light Fields using Deep Learning**. Advisor: Bruno Zatt. 2022. 163 f. Thesis (Doctorate in Computer Science) – Technology Development Center, Federal University of Pelotas, Pelotas, 2022.

Several imaging systems use **image metrology** to measure, identify, inspect and diagnose. The growing demand for visual metrology systems in several areas that require compact and robust sensors has driven the development of capture devices. These sensors use different physical quantities for imaging and calculating distances, each with its limitations and advantages. Among the emerging imaging technologies, which use only visible light information, we highlight the use of *light field* images captured through dense or sparse *light field* cameras, which have intrinsic advantages over traditional devices. For example, these cameras are robust in specific occlusion situations and also in scenes with noisy environments (rain, snow, etc.). This makes *light field* cameras, also called plenoptic cameras, potentially useful as a versatile sensor with multiple applications. This capacity is little used due to the complex optical characteristics related to the capture system and the computational cost of the related processing. To extract the depth map using traditional geometric methods, it is necessary to estimate n-variables, update their values and perform new calculations at each parameter change. When using artificial neural networks, these relations are already implicit in the neural network itself, which allows an immediate response to the dynamic modification of the parameters. This thesis presents two techniques for extracting depth maps from dense *light field* images based on neural networks with deep learning. The first proposal simplifies the EPINET network, reducing the flow from four inputs to just two inputs. The second proposal explores the multistream input network in an *u-shaped* convolutional neural network. Each proposal is explored and its advantages and disadvantages are presented. Both proposals calculate depth maps in less time than the original EPINET.

Keywords: Dense Light Field. plenoptic function. Deep Learning. artificial neural network. Computer Vision.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

2D	Bidimensional
3D	Tridimensional
AE	*Autoencoders*
BN	*Batch normalization layer*
CNN	*Convolutional neural networks*
CoC	*Circle of confusion*
ConvNets	*Convolutional neural networks*
Conv	*Convolutional layer*
DLF	*Dense Light Field*
DLV	*Depth likelihood volume*
DNN	*Deep neural networks*
DoF	*Depth of Field*
DS	*Depth step*
DSLR	*Digital Single-Lens Reflex*
EDOF	*Extended Depth of Field*
EPI	*Epipolar-Plane Image*
ESLF	*External Standardized Light Field*
FCN	*Fully convolutional network*
FF	*Full Frame*
FLF	*Focused light field*
FoV	*Field of View*
IEC	*International Electrotechnical Commission*
ISO	*International Standards Organization*
LCD	*Liquid crystal display*
LiDAR	*Light Detection And Ranging*
LIT	*Light-field inference of transparency*

LF	*Light field*
LFR	*Light Field raw file*
LFT	*Light Field Toolbox*
LSTM	*Long short-term memory*
MCL	*Monte Carlo location*
MLA	*Microlens array*
MLP	*Multilayer perceptrons*
MPI	*Multi-Plane Image*
MSI	*Multi-Sphere Image*
PFM	*Portable FloatMap format*
PMC	Perceptrons de multicamadas
PPI	*Pixels Per Inch*
PPM	*Portable Pixmap Format*
Radar	*Radio Detection and Ranging*
ReLU	*Rectified Linear Unit*
RGB-D	*RGB Depth Image*
RN	*Rede Neural*
RNN	*Recurrent neural network*
ResNet	*Residual Networks*
SAI	*Subaperture Image*
SLR	*Single-Lens Reflex*
SLF	*Sparse Light Field*
ToF	*Time of Flight*
ULF	*Unfocused light field*

SUMÁRIO

1 INTRODUÇÃO

Sistemas de metrologia, telemetria, fotogrametria, reconstrução 3D, odometria visual, carros autônomos, robótica, microscopia, e outros sistemas que usam imageamento para extrair medidas de uma cena, em muitas situações necessitam de sensores compactos e robustos. Estes sensores usam vários tipos de grandezas físicas para realizar o imageamento e o cálculo das distâncias, entre os mais usados pode-se citar: o radar (ondas eletromagnéticas), sonar (ondas sonoras), LiDAR/ToF (luz/laser pulsado), câmeras estéreo (imagem 3D) e câmeras RGB-D (uma câmera 2D e uma câmera infravermelho combinadas). Sendo que cada tipo de sensor possui suas limitações e vantagens.

Por exemplo, sistemas como o LiDAR possuem um alto desempenho no cálculo de distâncias, mas a medida que os objetos se distanciam de seu ponto de dispersão radial se perde a resolução e, dependendo da localização e distanciamento, objetos entre dois feixes ou em uma área de sombreamento não são detectados. Tal comportamento pode ser observado na Figura 1, onde a sombra escura é na verdade a região onde o laser não é projetado, visto que o mesmo foi bloqueado por um objeto mais próximo.

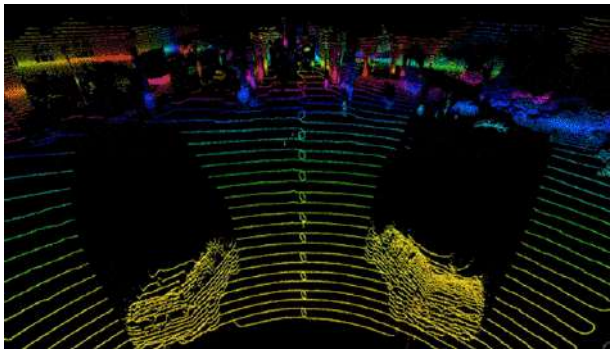


Figura 1 – Objetos podem ficar oclusos na região escura. Fonte: Luminar Technologies/IEEE Spectrum - Hecht (2017)

O mesmo pode ocorrer em câmeras RGB-D que projetam um padrão salpicado de pontos em frequência infravermelha (Figura 2), portanto não visível para humanos, em uma cena. Como as posições relativas e diferenças entre os pontos é fixa, o cálculo

da distância dos objetos e do mapa de profundidade de uma cena é feito a partir das distorções causadas nesse padrão. Como se trata de uma projeção, também existem regiões de sombra. Além disso, um sistema RGB-D sofre interferência de outras fontes de infravermelho (como o sol) e pode gerar falsos positivos através da detecção de padrões semelhantes gerados de forma aleatória por essas fontes externas.

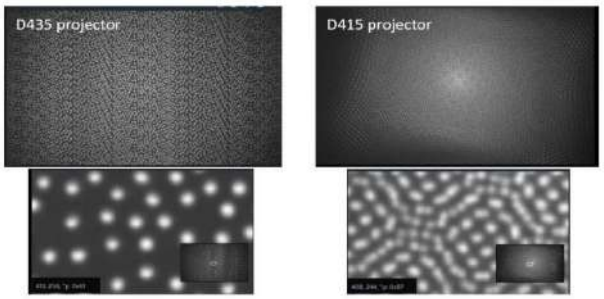


Figura 2 – Padrões de pontos infravermelhos projetados pelas câmeras RGB-D Intel Realsense D435 e D415. Fonte: Grunnet-jepsen et al. (2020)

.

Os sistemas citados, dependendo das condições de uso e necessidades, são robustos, mas também com limitações severas. O LiDAR, por exemplo, se for usado de forma massiva em carros autônomos, pode vir a causar lesões nas retinas de pedestres e danificar câmeras digitais, justamente por ser um feixe laser pulsado, além de ter baixo desempenho em situações climáticas adversas e possuir um custo mais elevado em relação a outros sensores. Já câmeras digitais, que usam a faixa de frequência eletromagnética visível para humanos, possuem um potencial de aplicação que vai além do cálculo de distâncias. Elas podem ser usadas para reconhecimento de objetos, padrões, avaliação de contexto, reconhecimento de estruturas, etc. Câmeras monoculares podem calcular a distância entre objetos através do conhecimento da geometria projetiva da câmera usada ou através de marcos pré-estabelecidos na imagem. Já sistemas de visão estéreo podem calcular a distância através da triangulação entre dois planos focais, tendo essa estrutura a capacidade de tratar alguns tipos de oclusões e ruídos. Entre tecnologias emergentes de imageamento, que usam apenas a informação de luz visível, temos destaque para o uso de imagens *light field* capturadas através de câmeras *light field* densas ou esparsas, que possuem vantagens intrínsecas em relação aos dispositivos tradicionais citados.

Câmeras **LF** ou *light field*[1] possuem potencial de uso como um versátil sensor com múltiplas aplicações (aquisição de imagens, cálculo de distâncias, geração de mapas de profundidade), desde que superados alguns desafios técnicos comuns em tecnologias emergentes (calibração, retificação, tratamento de ruído, extração de informação relevante para construção de mapas de profundidade, etc.). Importante observar que se em termos comerciais podemos considerar a tecnologia de *light field* recente, com

[1] em português e espanhol campo de luz

a empresa Raytrix[2] anunciando a produção e comercialização de câmeras com essa tecnologia em 2010 (câmera plenóptica 2.0) e a empresa Lytro [3] em 2011 (câmera plenóptica 1.0). Mesmo sendo uma tecnologia recente, seus conceitos remontam ao século XIX. Já em 1846, Michael Faraday (FARADAY, M., 1846) sugeria que a luz e outras radiações ocorrem em “linhas de força”, semelhantes às “linhas de força magnética”, em outras palavras, um campo de luz. Em 1908 Lippmann (LIPPMANN, G., 1908), apresenta a *“Photographie intégrale”* e a primeira menção a *light field* é no trabalho de Gershun em 1939 (GERSHUN, 1939).

Observa-se que em meados dos anos 90 e início do século XXI temos o que pode se chamar de **"primeira onda LF"**. Essa fase começa com o interesse teórico na área e a construção de protótipos de câmeras LF, fato correlacionado com o aumento da capacidade de processamento, barateamento de sistemas de captura digital e novas demandas na área de visão computacional. A partir de 2010 temos uma **"segunda onda LF"** com o lançamento comercial das câmeras das empresas Lytro e Raytrix e o crescimento do interesse na área e disponibilidade de *datasets*. Já nos últimos seis anos, percebe-se uma **"terceira onda LF"** com desenvolvimento de novos dispositivos, tanto comerciais como protótipos, baseados em *light field* para captura e imageamento, junto com um aumento significativo de pesquisas e projetos. Isso se deve ao amadurecimento da área e à demanda por informações 3D e possibilidade técnica de criação dos dispositivos. Por se tratar de uma área em franco desenvolvimento, muitos problemas permanecem em aberto, por exemplo, como representar, transmitir e armazenar uma imagem *light field*, quais técnicas devem ser usadas para a calibração das imagens das câmeras, como tratar regiões fronteiriças entre as microlentes, como construir monitores e telas para esse tipo de formato, etc. Portanto é uma área com vários desafios em aberto e com crescente desenvolvimento teórico, prático e comercial.

1.1 Motivação

Como já dito, as câmeras plenópticas possuem um potencial de uso como um sensor versátil com múltiplas aplicações no campo da metrologia, robótica, carros autônomos e etc. Esse potencial tem sido pouco explorado em virtude das características ópticas complexas dessas câmeras e a quantidade de processamento necessário para tratar a geometria projetiva usada na captura de imagens, principalmente quando não se tem acesso a informações estruturais geométricas dos dispositivos. As imagens LF capturadas por câmeras plenópticas densas são convertidas para sequências de vistas com pequenas variações, o que permite a construção de mapas de profundidade,

[2] raytrix.de

[3] A empresa Lytro encerrou suas atividades em março de 2018

com a devida retificação, mesmo em ambientes externos sem controle da iluminação. A principal limitação desse tipo de dispositivo são as linhas de base muito estreitas e com ruído, o que torna as estimativas de profundidades difíceis (SHIN et al., 2018).

Apesar dessas adversidades, câmeras plenópticas possuem vantagens intrínsecas em relação a sistemas LiDAR (*Light Detection And Ranging*), Radar e câmeras RGB-D. Por exemplo, câmeras LF, também chamadas plenópticas, capturam informação espacial e angular da cena, sendo mais robustas em condições extremas climáticas que geram muito ruído, como chuva forte e neve, possuindo bom desempenho em situações específicas de oclusão e com capacidade de remoção de reflexos e melhoramento de imagens com baixa iluminação (BAJPAYEE; TECHET; SINGH, 2018; ZANG et al., 2019).

O processamento de uma LF, estática ou dinâmica, possui várias etapas: decodificação, calibração, retificação, cálculo da faixa refocável, construção do mapa de profundidade, etc. Essas etapas em geral usam transformações lineares para a construção do mapa de profundidade. Mas uma das principais barreiras é gerar o mapa de profundidade em **tempo real** a partir de imagens *light field*, uma vez que temos algo semelhante a várias câmeras de captura gerando dados com alta dimensionalidade [4], o que leva esse procedimento apresentar um custo considerável em matéria de tempo de processamento e uso de recursos computacionais (WU et al., 2017). Uma forma de reduzir esses cálculos é usando um aproximador universal de funções que aprenda essas relações e retorne os valores diretamente. Um dos mais conhecidos algoritmos aproximador universal de função usado na área de aprendizado de máquina é o **perceptron multicamada**, que pode ser usado para receber como entrada as imagens LF, **aprender** as relações ópticas e os parâmetros usados na captura, diretamente do volume de entrada, e fornecer na saída o mapa de profundidade estimado. Essa abordagem reduz a complexidade ao trabalhar apenas com a imagem LF, substituindo os cálculos geométricos usados para inferir a disparidade entre vistas, que devem ser recalculados a cada nova cena, por uma rede neural que aprende a relação **entrada**/**saída**

1.2 Objetivos

O objetivo principal desse trabalho foi a construção de um sistema com aprendizado de máquina, baseado em redes convolucionais, que gera o mapa de profundidade a partir do aprendizado das relações entre as disparidades entre as vistas e a profundidade associada a esses deslocamentos, usando apenas como entrada imagens **LF** produzidas de forma sintética ou real por uma câmera plenóptica 1.0 do tipo denso, virtual ou real, e com velocidade de processamento que permitam o seu uso

[4] do inglês *high dimensional data*

em sistemas *near real time* ou mesmo em sistemas *real time*.

No desenvolvimento da pesquisa foram identificados os seguintes subproblemas:

- Definir um *dataset* adequado para a abordagem proposta;
- Estudo das abordagens existentes para a solução do problema apresentado;
- Construção de um *pipeline* de aprendizado de máquina para testar variações das redes neurais utilizadas;
- Desenvolvimento da solução para o problema exposto.

No decorrer do levantamento dos dados para esse projeto foi possível identificar que esse é um tema de pesquisa promissor e que existe um crescente interesse comercial na área.

1.3 Organização do texto

O texto está organizado de forma a apresentar inicialmente o fundamento teórico necessário para o desenvolvimento e compreensão do trabalho e depois apresenta a solução proposta. Os Capítulos 2 e 3 apresentam as bases em óptica e das técnicas de aprendizado de máquina usadas. O **Capítulo 2** apresenta as informações sobre imageamento, óptica, *light field* e cálculo de profundidade. Já o **Capítulo 3** apresenta uma breve introdução a redes neurais, com enfoque no aprendizado profundo. No **Capítulo 4** são apresentados trabalhos relacionados e como o mercado tem se desenvolvido na área de *light field*. No **Capítulo 5** são apresentados os materiais utilizados e a abordagem utilizada para o desenvolvimento da solução proposta, que é apresentada no **Capítulo 6**. Por fim, os resultados dos testes são apresentados no **Capítulo 7** e o fechamento da tese com as conclusões são feitas no **Capítulo 8**.

2 CONCEITOS BÁSICOS DE IMAGEAMENTO

Em sistemas monoculares, informações ópticas, como distância focal, em geral não são necessários para o processamento das imagens. Já em sistemas de imageamento n-dimensionais, onde em um único disparo do obturador são capturadas n-pontos de vista, é necessário a compreensão de conceitos elementares da formação da imagem, para que se possa trabalhar com alinhamento, triangulação, retificação e outros procedimentos relacionados. Desta forma, para a definição do escopo do trabalho é necessário o conhecimento de conceitos ópticos básicos de formação de imagem em um dispositivo *light field*. Esta seção apresenta essas noções elementares envolvidas na captura de imagens usando **câmeras digitais**. Os princípios apresentados são fundamentais para o entendimento da escolha e construção do *dataset* e no cálculo do mapa de profundidade.

2.1 Aquisição de imagens ou imageamento

Imagear é obter ou capturar uma imagem por meio de equipamento imageador. Um equipamento imageador é um dispositivo que gera imagens de acordo com um sistema de captura de entrada. Ele pode ser um instrumento óptico como uma câmera fotográfica, aparelhos de diagnóstico que formam imagens a partir de fontes diferentes (radiografia, ressonância magnética, tomografia computadorizada etc.) ou mesmo um sistema optoeletrônico como um escâner, além de vários outros tipos de equipamentos. Existem vários tipos de dispositivos para aquisição de imagens fotográficas conforme a Figura 3. Em virtude das particularidades do trabalho desenvolvido, os equipamentos com as características ópticas que são objetos de estudo são as câmeras **SLR**/**DSLR** e as câmeras digitais sem espelho (***mirrorless***).

Uma câmera "padrão" possui um visor óptico separado da lente da câmera. Dessa forma, tanto a lente como o visor focalizam a cena separadamente, levando a discrepâncias entre o que foi enquadrado pelo visor e o que foi capturado pelo conjunto lente/sensor. Já em uma câmera SLR (abreviação de "*single-lens reflex*" em inglês) existe um espelho que desvia a imagem capturada pela lente para o visor óptico. Isso

Figura 3 – No sentido horário, acima a partir da esquerda, *smartphone* com câmera, câmera compacta, câmera sem espelho (*mirrorless*), e câmera SLR digital. Fonte: Mchugh (2019)

permite que o fotógrafo saiba exatamente o que está sendo enquadrado pela câmera. Em uma DSLR (abreviação de " *digital single-lens reflex*" em inglês) esse processo é repetido através de uma tela LCD que mostra a imagem advinda do sensor de captura. Desta forma, assim como nas SLR analógicas, as DSLR permitem ao fotógrafo visualizar diretamente o enquadramento realizado. Um avanço em relação as DSLR são as câmeras com lentes intercambiáveis sem espelho (*mirrorless*) introduzidas em meados de 2010 (ANG, 2018). Basicamente essas câmeras removem o visor óptico da DSLR usando apenas a tela de LCD.

2.1.1 Lentes, conjunto de lentes e distância focal

O que é chamado de forma genérica como lente, na verdade é um **conjunto de lentes**, referenciado nominalmente pela sua distância focal. Sendo que a **distância focal** é a medida em milímetros do ponto de convergência (nodal) dos feixes de luz capturados até a superfície do sensor da câmera conforme a Figura 4 (TAYLOR, 2015).

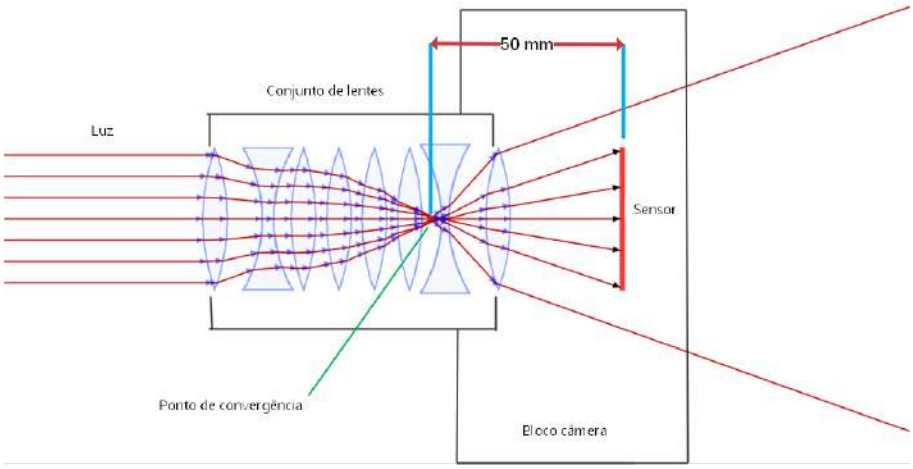


Figura 4 – Exemplo de um conjunto de lentes com distância focal de 50mm. Fonte: De autoria própria.

O comprimento focal da lente determina a área/campo de visão capturada cha-

mado de **FoV** (do inglês: Field of View), ou seja, ele determina o **ângulo de visão**, e o quão ampliados os objetos aparecem no quadro de captura. Por ângulo de visão se entende a medida em graus da cena capturada (TAYLOR, 2015). A Figura 5 apresenta um quadro ilustrativo da relação entre a distância focal, ângulo de abertura e ampliação.

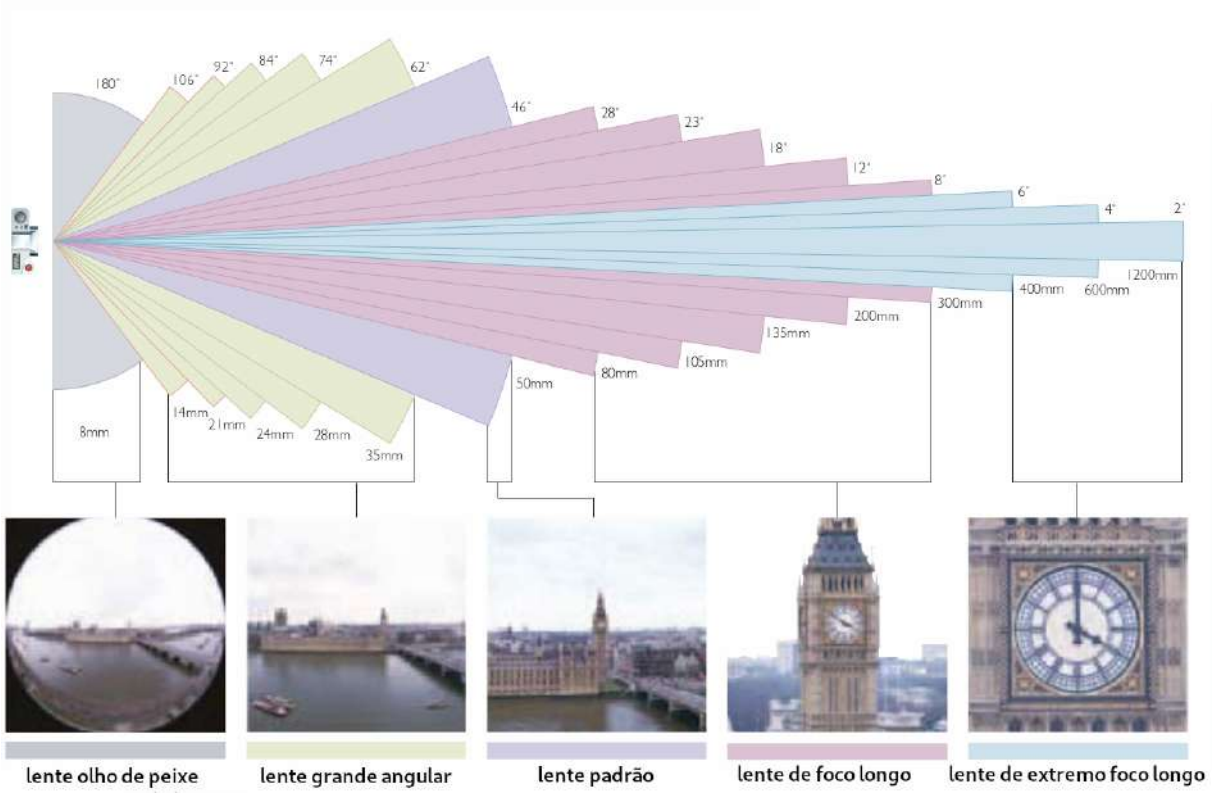


Figura 5 – Distância focal e ângulo de visão. Fonte: Adaptado de Taylor (2015)

2.1.2 Exposição

A quantidade de luz que irá alcançar o sensor de captura pode ser ajustada diretamente por dois controles físicos (TAYLOR, 2015): A abertura do diafragma e a velocidade do obturador. A sensibilidade da captura de imagem em relação a quantidade de luz incidente também é determinada pela ISO aplicada ao sensor. A seguir são detalhadas essas características.

2.1.2.1 Abertura do diafragma

A abertura ou diâmetro do diafragma da lente (Figura 6) pode ser alterada através do anel de abertura ou sistema equivalente. A abertura afeta a quantidade de luz que chegará ao sensor ou filme, o que também altera a profundidade de campo (HEDGECOE, 2005).

O diâmetro da abertura do diafragma é reduzido por um fator de 1.4 ($\approx \sqrt{2}$). Observa-se que o padrão *f-number* é criado a partir dos valores arredondados en-

Figura 6 – Abertura do diafragma. Fonte: Tabora (2020)

contrados a partir deste fator de redução, o que corresponde a uma redução de área pela metade conforme as relações a seguir:

$$1.4 \cdot 1.4 = 1.96 \approx 2 \Rightarrow f/2$$

$$2.0 \cdot 1.4 = 2.8 \Rightarrow f/2.8$$

$$2.8 \cdot 1.4 = 3.92 \approx 4 \Rightarrow f/4$$

$$\ldots$$

Esses valores são usados para calcular o *f-number*/ *f-stop*. O *f-number* representa uma abertura específica do diafragma definida como a razão entre o comprimento focal (f) e o diâmetro da abertura do diafragma (D): $N = f/D$ (MAÎTRE, 2017)(BLACK, 2015). Através desse cálculo e as variações padrão da abertura do diafragma pode-se gerar a série padrão internacional de *f-number*:

$$f/1 \rightarrow f/1.4 \rightarrow f/2 \rightarrow f/2.8 \rightarrow f/4 \rightarrow f/5.6 \rightarrow f/8 \rightarrow f/11 \rightarrow f/16 \rightarrow f/22 \rightarrow \ldots$$

O cálculo do diâmetro absoluto do diafragma depende do comprimento focal da lente (BLACK, 2015). Por exemplo, fotografando com uma lente com comprimento de 50 mm com uma abertura de *f/1.4*, o diâmetro da abertura real da lente é 35,7mm (50/1.4)(BLACK, 2015). Por este princípio, pode-se obter o mesmo diâmetro de abertura real mudando essas variáveis.

2.1.2.2 Velocidade do Obturador

O obturador controla o tempo (*t*) de exposição total (*H*) dada ao sensor. O total de iluminação da imagem (E_i) é dado pelo *f-stop* da abertura (Equação 1)(RAY, 1999).

$$H = E_i t \tag{1}$$

O valor de *t* é chamado de velocidade do obturador. A estrutura mecânica do obturador pode variar conforme a Figura 7.

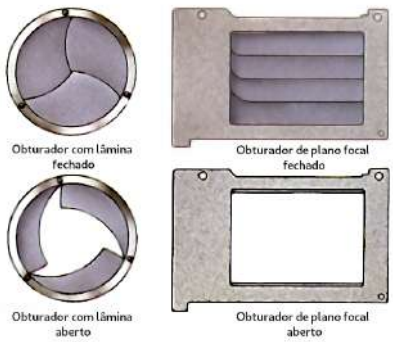


Figura 7 – Tipos de obturador. Fonte: Adaptado de Hedgecoe (2005)

A velocidade do obturador em geral é utilizada para objetos em movimento ou em situações em que se deseja alterar a quantidade de exposição a luz de forma complementar.

2.1.2.3 Número ISO ou ISO Speed

O número ISO ou ISO *Speed* representa o grau de sensibilidade do sensor a quantidade de luz incidente. Quanto mais alto o valor de do número ISO utilizado mais sensível o sensor é a luz. Esse valor influencia na qualidade da imagem e no ruído obtido no sensor de captura. O termo *speed* é associado aos antigos filmes fotográficos, onde o tempo de exposição para a fixação química da imagem poderia ser mais rápido ou mais lento. O número ISO é separado em três categorias: (i) rápidos; (ii) médios; (iii) lentos. Se o sensor está ajustado com um valor de ISO alto, isso significa que ele necessita de pouca luz para capturar a imagem, em outras palavras, ele é mais sensível e reage **rapidamente** a exposição da luz, por isso essa ISO também é chamada de *high ISO speed*, quanto maior o valor ISO, mais sensível a luz. O uso de um valor errado pode gerar muito ruído, ou baixa qualidade da imagem. Portanto, um ISO alto/rápido é usado em situações de pouca luz, e um ISO baixo é ideal quando os níveis de luz são adequados e se busca mais qualidade com detalhes finos (HEDGECOE, 2005). A Figura 8 mostra efeito de se usar um valor de ISO alto em uma cena com iluminação adequada.

Em termos técnicos, um filme monocromático o valor de ISO *speed* é definido como $S = 0.8/H_m$. Onde H_m representa a exposição em um ponto *m*. H é medido em lux por segundo. Uma boa exposição para uma cena externa, neste caso, corresponde a aproximadamente $H_0 = 9.4/Hm$ (JOHNSON JR., 2017) .

2.1.3 Profundidade de Campo

Profundidade de Campo ou DoF (do inglês *Depth of Field*) é a região em torno do plano em que se encontra o ponto focal da imagem que permanece em foco (ANG, 2018). Essa região se encontra tanto a frente como atrás do plano do ponto focal

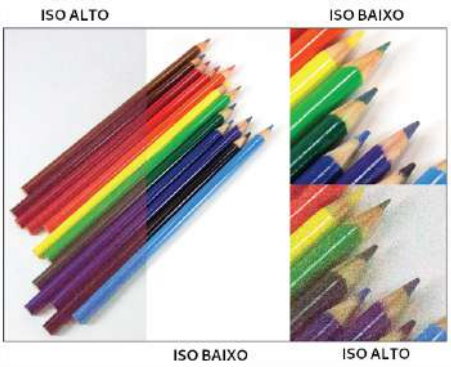


Figura 8 – ISO diferentes aplicadas a mesma cena, com a mesma velocidade de obturador. Ao se usar um ISO de valor alto, a imagem que possui iluminação adequada, passa a apresentar ruído. Fonte: Adaptado de Correll (2021)

conforme se vê na Figura 9.

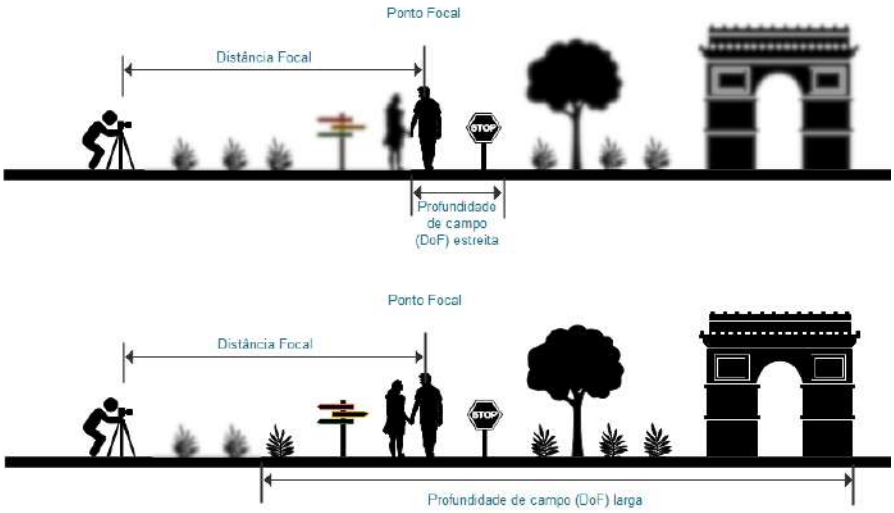


Figura 9 – Dois tamanhos distintos de profundidade de campo aplicadas na mesma cena. Fonte: De autoria própria.

A principal forma de controlar a DoF é através da abertura das lentes. Se diminuirmos a abertura de lente (por exemplo usando f/11 no lugar de f/4) a profundidade de campo aumenta. Outro fator é a distância focal da lente. Quanto menor a distância focal da lente, maior a profundidade de campo. Por exemplo, para uma mesma distância, se usarmos uma abertura de lente fixa no valor de f/11, a profundidade de campo é maior em uma lente de 28 mm se comparada a uma lente de 300mm. Na Figura 10 podemos observar essa relação entre quantidade de luz recebida pelo sensor de acordo com abertura do obturador e sua influência na profundidade de campo.

O cálculo do DoF já é bem estabelecido na literatura. Conforme se observa na Equação 2 e na Figura 11, ele possui dois limites, um valor superior e um inferior, que engloba toda região em foco na imagem.

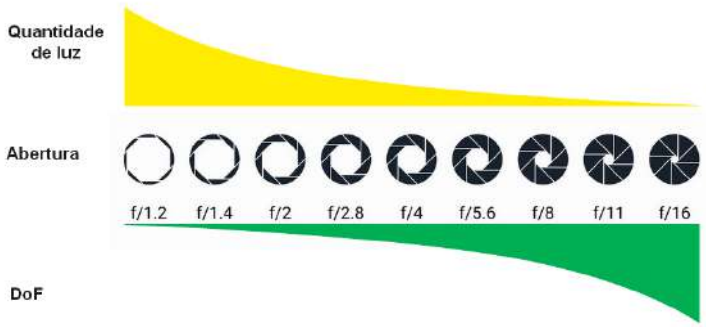


Figura 10 – Quando a abertura de diafragma diminui o sensor recebe uma quantidade de luz menor e a profundidade de campo aumenta. Fonte: De autoria própria.

$$DoF = DoF_{\textit{Limite distante}} - DoF_{\textit{Limite próximo}} \tag{2}$$

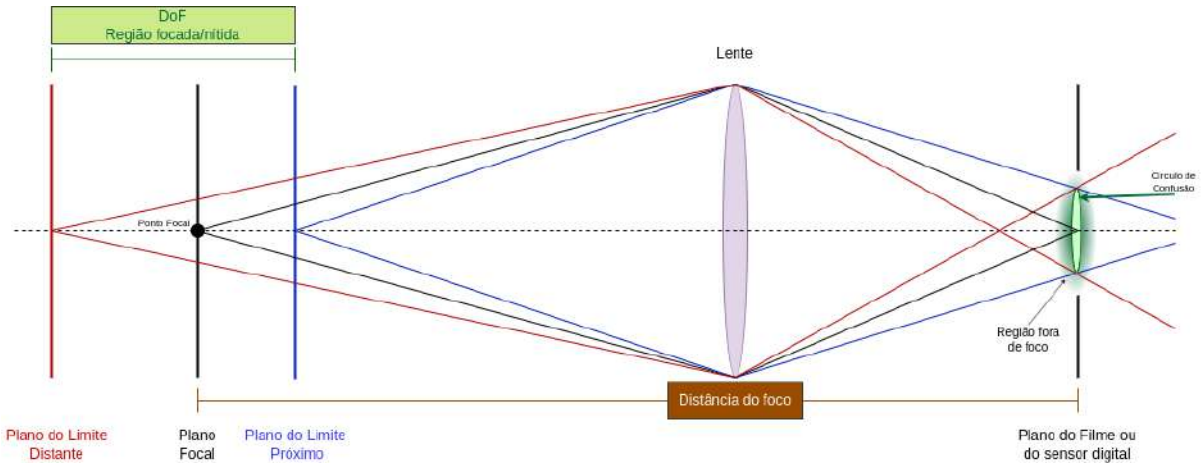


Figura 11 – Diagrama representando o círculo de confusão e sua relação como o DoF. Fonte: De autoria própria.

Para se determinar os limites inferior e superior do DoF, é necessário usar o conceito de **círculo de confusão** ou **CoC** (do inglês *circle of confusion*). O CoC define a região que se percebe como focada/nítida no sensor (Figura 11). Uma região fora da profundidade de campo possuí o CoC embaçado/desfocado de forma perceptível (MCHUGH, 2019).

O limite superior, ou ponto mais distante dentro da profundidade de campo é dado pela Equação 3, e o limite inferior pela Equação 4, onde H é a **distância hiperfocal**, f é o **comprimento focal da lente** usada e u a **distância do foco**.

$$DoF_{\textit{Limite distante}} = \frac{H.u}{H - (u - f)} \tag{3}$$

$$DoF_{\textit{Limite próximo}} = \frac{H.u}{H + (u - f)} \tag{4}$$

A **distância hiperfocal** é a distância focal que dá o máximo de profundidade de campo. Ela é calculada pela Equação 5 , onde C representa o **limite do CoC** e N a **abertura** (*f-number*).

$$H = f + \frac{f^2}{N.C} \tag{5}$$

A informação DoF é **importante para determinar a região focada na imagem e permitir o cálculo da distância dos pontos que se encontram no mesmo plano focal**.

2.1.4 Resolução, tamanho do sensor e fator de corte

Uma imagem digital pode ser visualizada como uma matriz 2D onde em cada posição é armazenado um valor de intensidade luminosa associada ao pixel que se encontra na respectiva coordenada espacial da imagem projetada no sensor de captura. No caso de imagens coloridas pode-se tratar cada posição como um vetor com três valores, um para cada canal de cor RGB, ou como três matrizes de tamanhos idênticos onde cada matriz representa uma imagem com valores de intensidade associados a um canal de cor. O tamanho dessa matriz junto com a quantidade de dados armazenados para cada pixel individual determinam respectivamente a resolução espacial e a quantização da imagem (*bit resolution*) (SOLOMON; BRECKON, 2010).

A **resolução espacial** é a quantidade total de *pixels* independentes usados para cobrir o espaço visual capturado da cena projetada no sensor. Em geral, esse valor é representado pelo total de colunas *versus* linhas, por exemplo: 640x480.

A **profundidade de bit** determina o valor máximo de intensidade de um *pixel*, ou seja, a quantização da informação da imagem (SOLOMON; BRECKON, 2010).

O tamanho do sensor de captura influencia na geometria das características que serão expressas na imagem. O tamanho típico de filmes fotográficos era de 35mm, esse tamanho foi transportado para os sensores digitais (JOHNSON JR., 2017). Sensores com esse tamanho são chamados *full frame* ou FF. A Tabela 1 mostra alguns tamanhos de sensores usados em câmeras digitais. Para uma comparação visual a Figura 12 mostra os tipos mais comuns de sensores.

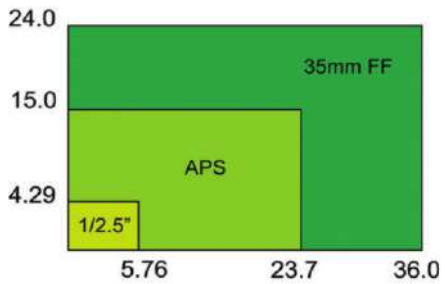


Figura 12 – Tamanho relativo entre os tipos de sensores. Fonte: Johnson jr. (2017).

Tabela 1 – Sensores típicos de câmeras digitais (2015). Adaptado de (JOHNSON JR., 2017)

Tipo	Proporção de tela	Largura (mm)	Altura(mm)	Diagonal(mm)
1/3”	4:3	4,8	3,6	6,0
1/2.5”	4:3	5,76	4,29	7,18
1/1.7”	4:3	7,6	5,7	9,6
1.0”	4:3	13,2	8,8	15,9
4/3”	4:3	17,3	13,0	21,6
APS-C	3:2	22,2	14,8	27,04
FF (35 mm)	3:2	36,0	24,0	43,27
Formato Médio	4:3	43,8	32,8	54,72

Um pixel não possui um valor fixo em termos de área ocupada e seu tamanho depende das dimensões do sensor, diagonal do sensor, largura do pixel e distância entre os centros de cada pixel (*pixel pitch*). Por exemplo, em um sensor de $1/3$ polegadas com 8 megapixels (3264 x 2448), o *pixel pitch* é de $1,48\ \mu m$ com 17.272 PPI[1]. Em sensores APS-C e FF 35mm (que são mais comuns em câmeras digitais) o *pixel pitch* é mais largo (JOHNSON JR., 2017). Em uma câmera com sensor FF 35mm é de $8,42$ μm para $12,2$ megapixel e $4,13\ \mu m$ para $50,6$ megapixel (JOHNSON JR., 2017).

O **fator de corte** (do inglês *crop factor*) reflete a relação da lente com o tamanho do sensor de captura. Para uma mesma lente fotográfica ou objetiva a projeção sobre o sensor será a mesma, mas a captura total da cena depende do tamanho do sensor. Sensores menores capturam uma região menor. Por exemplo, na Figura 13 temos a comparação entre a área coberta por um sensor FF e por um sensor APS-C (conforme a Tabela 1) para a mesma objetiva.

Entre os sensores padrão o **FF**-*full frame* é o que captura a maior área possível. Isso faz com que ele seja usado como referencial. Desta forma, o fator de corte sempre indica o quanto de cena é capturada em comparação a um sensor *full frame*. Por exemplo, ao se utilizar uma lente de 50mm em um sensor FF, se tem a máxima área de captura (**fator de corte 1**). Ao manter a objetiva e todas outras variáveis constantes, mas utilizar um sensor APS-C com fator de corte 1.6, a região capturada pela Equação 6 é equivalente a de uma lente de **80mm** ($50mm$ x $1.6 = 80mm$).

$$\text{Área} = \text{Lente x Fator de Corte} \tag{6}$$

Portanto para cobrir a mesma área de um sensor FF com uma lente de 80mm em um sensor APS-C a objetiva usada deve ser de 50mm. Deve-se ressaltar que o único fator que é alterado é a região da imagem capturada. A lente de 50mm não se comporta como uma de 80mm em um sensor com fator de corte 1.6. Suas características como profundidade de campo e aberrações ópticas continuam as mesmas. Apenas

[1] *Pixels Per Inch*

a região projetada é equivalente. A Figura 5 permite visualizar essa equivalência em termo de abertura angular.

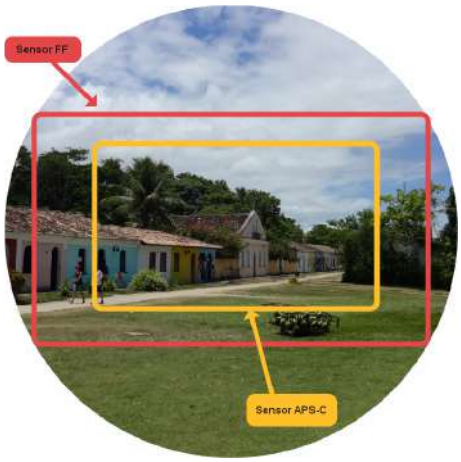


Figura 13 – Simulação aproximada do fator de corte para um sensor FF e um APS-C com a mesma lente objetiva. Fonte: De autoria própria.

2.1.5 Considerações

Essa seção apresentou definições basilares necessárias para a criação de datasets utilizando câmeras comerciais. Essas informações também foram úteis no processo de seleção de datasets já existentes.

2.2 Imagens com múltiplas perspectivas

As câmeras citadas até aqui fazem a captura de imagens a partir de um único ponto de vista. Isso resulta na perda de informações de profundidades da cena. Para se obter dados de profundidade, usa-se duas ou mais câmeras com características e configurações idênticas, mas com variações de posição em relação a seus eixos horizontal, vertical ou em ambos. No caso de uma cena estática, podemos mover uma câmera nesses eixos e capturar a cena a partir de novas perspectivas com sobreposição de regiões entre as imagens.

2.2.1 Visão estéreo

O processo de extração da informação 3D feito a partir de duas perspectivas distintas é chamado de visão estéreo (IKEUCHI, 2014). A visão estéreo serve como base para entendimento dos métodos usados em imagens com mais de duas perspectivas, como é o caso das imagens *light field*.

Para fins de definição dos parâmetros usados, em imagens estéreo, cada vista possui uma matriz de projeção P e P' associadas a cada uma das câmeras (câmera P e câmera P'). Sendo que o apóstrofo (') é usado para apontar parâmetros referentes ao que é designado como segunda perspectiva, e sua ausência denota parâmetros da

primeira vista. Exemplificando, um ponto espacial tridimensional X, com as coordenadas espaciais (x, y, z), é projetado como a transformação geométrica $x = PX$ na primeira perspectiva e como $x' = P'X$ na segunda (HARTLEY; ZISSERMAN, 2004). Os pontos x e x' são correspondentes, pois representam a projeção bidimensional do mesmo ponto X tridimensional. Essa abordagem aponta para três problemas básicos (HARTLEY; ZISSERMAN, 2004):

1. **Geometria da correspondência**- De que forma a localização de um ponto x na primeira perspectiva restringe a localização de x' na segunda vista?

2. **Geometria da câmera (movimento)** - Dada uma imagem e seu conjunto de pontos correspondentes $x_i \leftrightarrow x'_i$, $i = 1, ..., n$, quais são as câmeras (matrizes) P e P' para as duas visualizações?

3. **Geometria (estrutura) da cena** - Dado os pontos de imagem correspondentes $x \leftrightarrow x'$ e matrizes P e P', qual é a posição de X no espaço 3D?

A última questão é o objetivo final dessa proposta quando aplicada a n-vistas. Mas para se chegar neste ponto é necessário responder as duas questões anteriores. Convém ressaltar que o problema da retificação das imagens não está sendo abordado nesse trabalho, pois as imagens geradas já estão retificadas pelos softwares utilizados.

2.2.2 Geometria Epipolar

A primeira questão apresentada é também conhecida como **problema da correspondência estéreo**. Como existem dois pontos de vista do objeto, um ponto localizado na superfície de um objeto 3D ocupa coordenadas distintas em cada perspectiva. Por exemplo, na Figura 14 o ponto X é projetado nos planos de captura da esquerda nas coordenadas x e o mesmo ponto é projetado nas coordenadas x' no plano da direita.

Desta forma, é necessário localizar para cada ponto $x \in I$ a projeção x' em I' para poder determinar a diferença entre as coordenadas. Como saída desse processo temos um **mapa de disparidade** onde o valor de intensidade representa a diferença entre x e x' (ZHANG, 2013). Portanto, encontrar a **correspondência estéreo** do ponto X é identificar o local de sua projeção em cada plano com suas respectivas coordenadas.

Segundo (DAVIES, 2017) existem duas principais abordagens para mapear a informação de profundidade em imagens: *light striping* em imagens monoculares e linhas epipolares em imagens estéreo. A primeira técnica é baseada em iluminação estruturada (*light striping*) para marcar pontos conforme a Figura 15 e se usa a distorção do

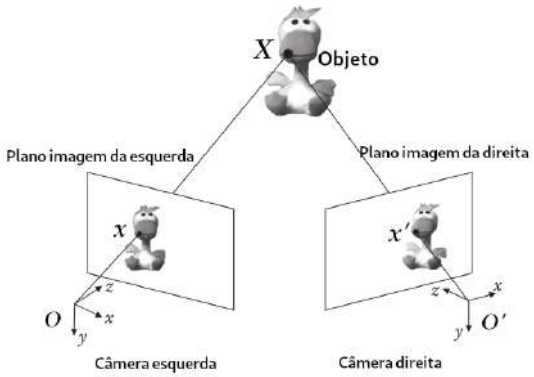


Figura 14 – Projeção de uma imagem 3D. Fonte: Adaptado de Zhang (2013)

padrão para calcular o mapa de profundidade. Essa abordagem não é usada nesse projeto.

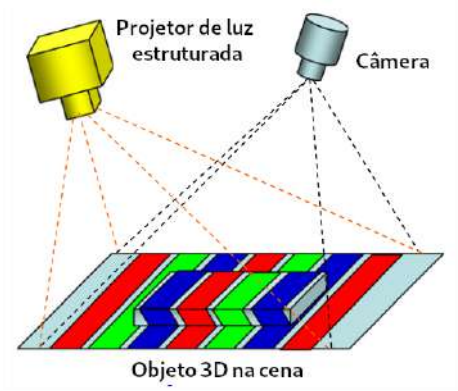


Figura 15 – Luz estruturada monocular usando faixas coloridas. Fonte: Geng (2011).

A segunda abordagem, linhas epipolares, é a mais comum em se tratando de imagens estéreo sem controle estruturado de iluminação. Por **geometria epipolar** se entende as relações geométricas associadas a geometria projetiva intrínseca entre duas vistas de um objeto espacial (OROZCO et al., 2017) . Essa geometria é independente da estrutura da cena e depende apenas dos parâmetros internos das câmeras (**matriz intrínseca**) e das poses relativas (HARTLEY; ZISSERMAN, 2004).

A ideia básica é reduzir a região de busca para encontrar o pares x e x'. Conforme a Figura 16, existe um ponto x correspondente a projeção de X. Em vez de varrer toda imagem da direita atrás das coordenadas x' se estabelece uma linha e' para a busca do ponto correspondente a X nesse plano. Essa linha é chamada de **linha epipolar**. Ela pode variar de acordo com as características geométricas das imagens, por exemplo, distorções das lentes.

Para se definir a linha epipolar é necessário se definir a **geometria epipolar**. Na Figura 16 cada câmera é indicada por seus centros de convergência C e C' e seus respectivos planos de imagem. A **linha de base** (*baseline*) une os centros das câme-

ras C e C', sendo que os pontos de intersecção entre ela e os planos de imagens são chamados de **epipolos**, representados na Figura 16a como e e e'. Na mesma Figura se observa que o **plano epipolar** π é determinado pelos centros das câmeras e pelo ponto espacial X. Para cada intersecção de um plano epipolar com um plano visual temos uma **linha epipolar**. Como o plano epipolar intersecta ambos planos visuais temos pares de linhas epipolares (l,l') correspondentes.

Se for conhecida apenas as coordenadas x, pode-se restringir a busca do ponto x' a linha epipolar no plano da imagem I'. O epipolo e' funciona como limite para a busca. Na Figura 16b temos a visão da linha epipolar em ambos planos, após a retificação das vistas I e I'. As perspectivas são **retificadas** de forma a ficarem **coplanares**, desta forma as linhas epipolares para cada par de pontos (x,x') passam a ser **colineares**.

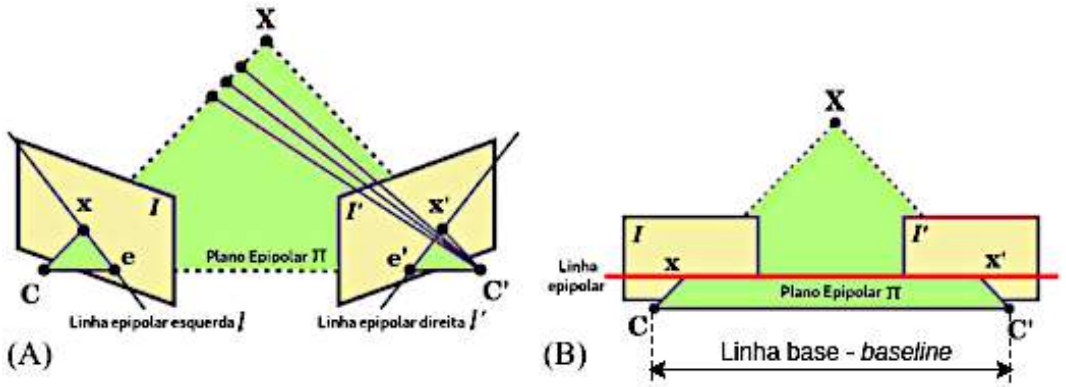


Figura 16 – Geometria das linhas epipolares. A **linha de base** (*baseline*) une os centros das câmeras C e C', sendo que os pontos de intersecção entre ela e os planos de imagens são chamados de epipolos (**e**, **e'**). Fonte: Adaptado de Orozco et al. (2017).

Uma vez estabelecidos os pares de linhas epipolares, se busca em l' o ponto x' relativo a x usando uma função de custo que retorne os pares de pontos com máxima correspondência.

2.2.3 Matrizes de projeção das câmeras

A projeção de uma imagem em um plano visual (Figura 14) depende das características ópticas e físicas da câmera. As relações entre as coordenadas 3D de um ponto X e sua projeção 2D em uma câmera é descrita por uma matriz de transformação chamada **matriz intrínseca** (P). Ao multiplicarmos a matriz intrínseca P pelo vetor que contém as coordenadas espaciais (X_i, Y_i, Z_i) do ponto X obtemos os valores das coordenadas (x_i, y_i) na projeção 2D (Equação 7). Na Equação 8 temos os parâmetros da matriz P: A distância focal ($f_x = f_y$); o ponto central da imagem (c_x, c_y); e o valor w que é igual a Z_i. Se a distância focal for modificada, as coordenadas 2D mudam. Para fins de simplificação a matriz extrínseca será chamada apenas de câmera a partir de agora.

$$x = PX \tag{7}$$

$$\begin{bmatrix} x_i \\ y_i \\ w_i \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_x & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix} \tag{8}$$

Uma projeção estéreo é calculada usando os conjuntos de pares (P,P') e as coordenadas espaciais X conforme a Equação 9.

$$\begin{aligned} x &= PX \\ x' &= P'X \end{aligned} \tag{9}$$

Quando os sistemas de coordenadas da câmera e do mundo real não estão usando o mesmo referencial de posição, ou seja, o plano focal de projeção 2D da imagem não está paralelo ao plano da imagem 3D, é necessário realizar o ajuste através de uma transformação geométrica. Isso é feito através da **matriz extrínseca** que converte as coordenadas do mundo real para o mesmo referencial do sistema de coordenadas da câmera. Essa abordagem não é necessária nesse trabalho.

2.2.4 Estrutura da cena e cálculo de distâncias

A última questão envolve a busca da posição de X no espaço 3D dado os pontos de imagem correspondentes $x \leftrightarrow x'$ e as matrizes P e P', ou seja, busca a estrutura da cena capturada em termos de posição e distância em relação ao ponto de observação. O conceito para o cálculo de distâncias envolve triangulação dos pontos conhecidos, lembrando que sempre existe um erro associado, visto que existe ruído na captura.

Antes de se calcular as relações trigonométricas na projeção estéreo é necessário entender o comportamento da captura de um ponto em uma única câmera. Para fins de simplificação na Figura 17 é usada uma câmera *pinhole*. Observa-se que o ponto tridimensional X é projetado na posição x_i do sensor de captura, equivalente ao ponto x_f no plano frontal. A variável f é a **distância focal** da lente, sendo o deslocamento de x_i e f proporcional a X_c e Z_c, estabelecendo as relações dadas na Equação 10.

$$\begin{aligned} \frac{x_i}{f} = \frac{X_c}{Z_c} \Rightarrow x_i = \left(\tfrac{f}{Z_c} X_c\right) \\ \frac{y_i}{f} = \frac{Y_c}{Z_c} \Rightarrow y_i = \left(\tfrac{f}{Z_c} Y_c\right) \end{aligned} \tag{10}$$

Para que se possa calcular a estrutura da cena por um sistema estéreo a partir

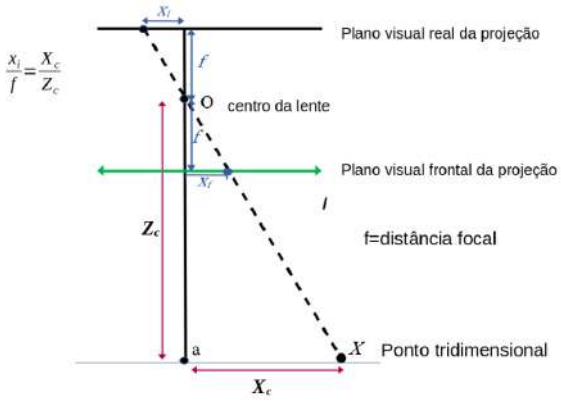


Figura 17 – Relações trigonométricas na projeção de um ponto em uma câmera *pinhole*. Fonte: Adaptado de Shapiro; Stockman (2001)

.

dessas relações básicas é necessário realizar a **retificação**, onde os planos visuais são ajustados de forma a ficarem coplanares. Desta forma, as câmeras ficam em eixos ópticos colineares, tendo apenas deslocamento horizontal entre elas (Figura 18). Uma vez que se encontre o mesmo ponto em ambas imagens (x e x') pode se estimar sua distância.

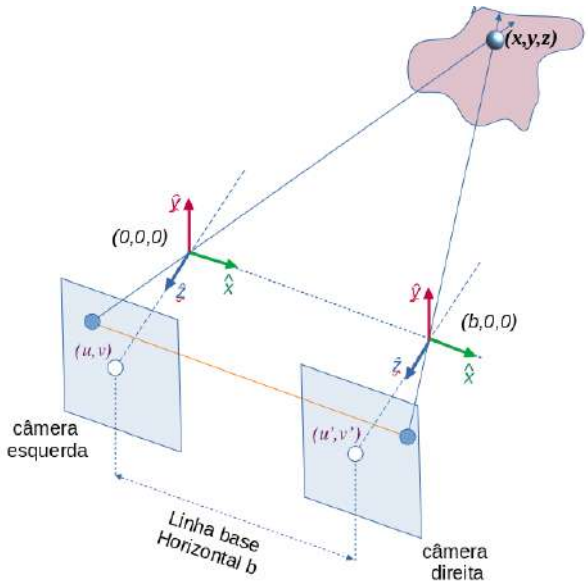


Figura 18 – Planos visuais retificados, prontos para a triângulação. Fonte: De autoria própria.

Pelas relações trigonométricas da Figura 19 temos a disparidade d entre os pontos x e x' dada pela Equação 11. Uma vez conhecida a disparidade é possível calcular a profundidade pela linha base horizontal b e a distância focal f (Equação 12), ou pela descoberta dos ângulos α e β determinados pelos valores x e x' (Equação 13).

$$d = x - x' \tag{11}$$

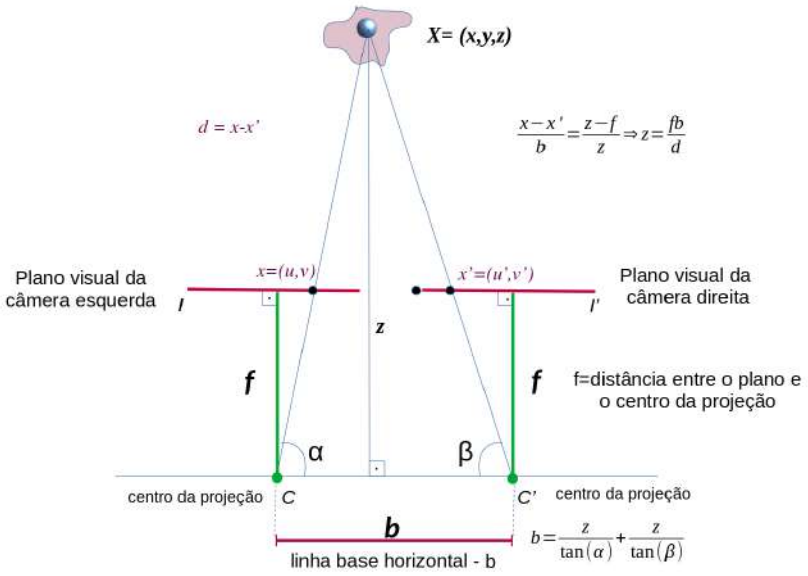


Figura 19 – Relações trigonométricas em um sistema de captura estéreo retificado. Fonte: De autoria própria.

.

$$\frac{x - x'}{b} = \frac{z - f}{z} \Rightarrow z = \frac{fb}{d} \tag{12}$$

$$b = \frac{z}{\tan(\alpha)} + \frac{z}{\tan(\beta)} \tag{13}$$

Observa-se pela Equação 11 que a profundidade é inversamente proporcional a disparidade.

2.2.5 Mapa de disparidade e mapa de profundidade

O conjunto de disparidades entre todos os pontos das imagens I e I' (Figura 19) formam o **mapa de disparidade**. Conforme já citado, dado o mapa de disparidade é possível calcular a profundidade de cada ponto através da linha base horizontal **b** e a distância focal f pela Equação 12, criando uma imagem em tons de cinza chamada **mapa de profundidade**, onde o valor de intensidade de cada pixel é proporcional a distância (Figura 20). O mapa de disparidade pode ser convertido para uma nuvem de pontos 3D representando a cena.

2.3 Função plenóptica e representação da imagem

A função plenóptica (Equação 14) descreve a intensidade de cada raio de luz no mundo como uma função do ângulo visual, comprimento de onda, tempo e posição do observador (IKEUCHI, 2014).

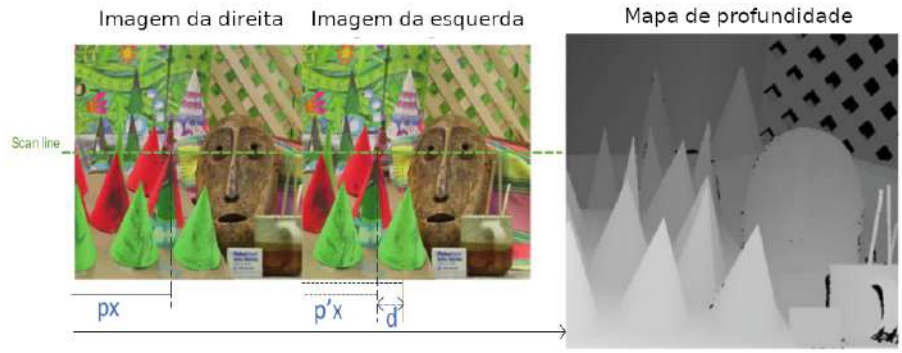


Figura 20 – O mapa de disparidade é obtido a partir da imagem esquerda (I) e da imagem direita (I'), onde a correspondência estéreo é realizada para encontrar os pontos correspondentes (px e $p'x$), localizados na mesma linha horizontal. O valor da disparidade d corresponde a portanto a diferença horizontal entre eles. Fonte: Shahnewaz; Pandey (2020)

.

$$P = P(\theta, \phi, \lambda, t, V_x, V_y, V_z) \tag{14}$$

Em uma câmera fotográfica convencional, se considerarmos o sensor de captura como ponto de convergência dos raios luminosos (Figura 21), nem todos raios serão observados. Mas ao contrário do que ocorre nesses sistemas usuais, a função plenóptica parametriza todos raios que chegam a todos pontos no espaço. Desta forma, a função plenóptica descreve o comportamento de todos raios luminosos, mesmo os que não podem ser capturados por sistemas usuais.

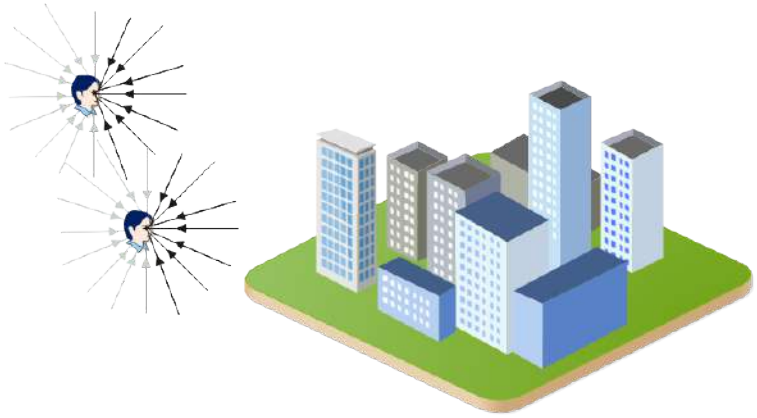


Figura 21 – A função plenóptica representa toda a informação visual que chega a um ponto do espaço em um determinado momento independente do observador. Na Figura temos dois observadores que não percebem os raios luminosos em cinza. Fonte: De autoria própria.

Podemos representar um raio luminoso na forma de um vetor em um sistema de coordenadas esféricas como na Figura 22. O ponto *O* representa o ponto de convergência (observador) e centro do sistema de coordenadas. Neste sistema se descreve a direção ou ângulo de visão através de duas coordenadas: θ que representa a colatitude (ângulo polar ou ângulo zenital) e ϕ o azimute. Desta forma todos os raios que

chegam em um ponto de vista são especificados como $P(\theta, \phi)$. Essa equação não leva em conta o comprimento de onda do raio luminoso (cor).

O sistema de coordenadas esféricas é uma forma simples de representar a esfera completa de raios luminosos que chegam a um ponto no espaço (Figura 23a), mas pode se optar por usar um sistema de coordenadas cartesianas, onde (x, y) representam as coordenadas espaciais de um plano imaginário da cena a uma distância pré-definida do ponto de observação como na Figura 23b (ADELSON; BERGEN, 1991).

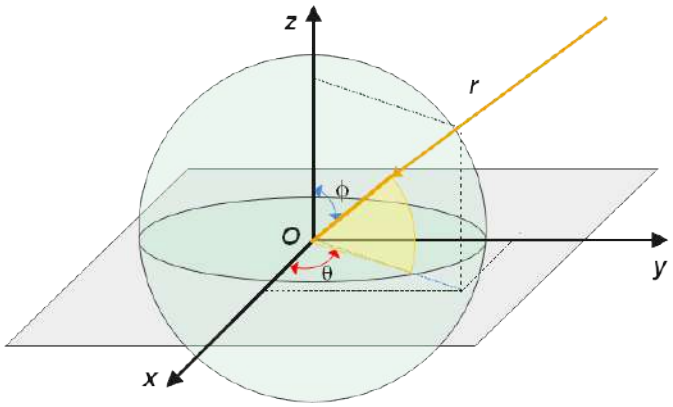


Figura 22 – Sistema esférico de coordenadas - r representa o raio de luz incidente com as coordenadas $P(\theta, \phi)$. Fonte: De autoria própria.

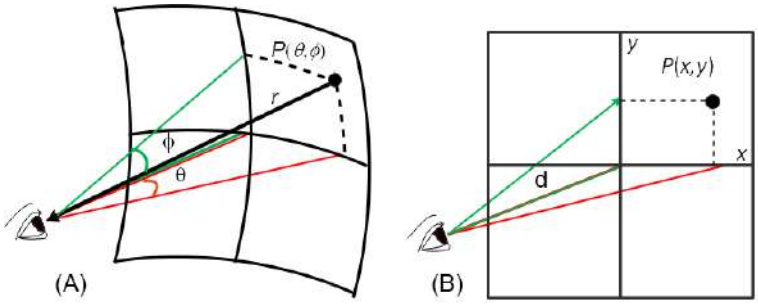


Figura 23 – A função plenóptica pode parametrizar um raio de luz ***r*** através de coordenadas esféricas (A) ou de coordenadas cartesianas(B), onde **d** representa a distância do observador ao plano imaginário da imagem. Fonte: De autoria própria.

A função plenóptica com sistema de coordenadas cartesianas fica representada na forma da Equação 15 .

$$P = P(x, y, \lambda, t, V_x, V_y, V_z) \tag{15}$$

Podemos representar uma cena ou ponto de vista conforme a Figura 24a, em tons de cinza pelo conjunto de raios de luz que a compõe pela função $P(\theta, \phi)$. Ao adicionar o parâmetro λ que representa o comprimento de onda, cor, de cada raio de luz, passamos a ter a função $P(\theta, \phi, \lambda)$ que representa uma cena colorida a partir de um único ponto de vista conforme a Figura 24b.

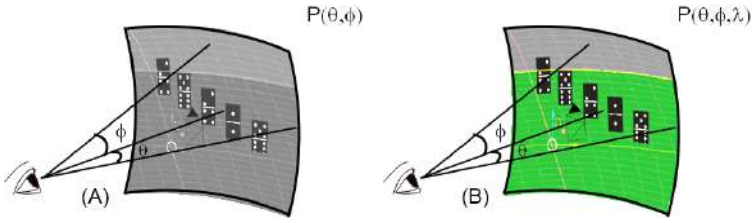


Figura 24 – (A) Cena representada pela totalidade de raios $P(\theta, \phi)$ que são capturadas em um único ponto de vista. (B) Informação de comprimento de onda λ acrescentada. Fonte: De autoria própria.

O parâmetro t representa o momento temporal da cena que está sendo observada, geralmente usado em vídeos para representar o momento temporal de captura de um *frame* dentro de uma sequência um vídeo. Já os parâmetros V_x, V_y, V_z representam todos pontos de vista possíveis (Figura 25). Chegamos dessa forma ao Equação 15 .

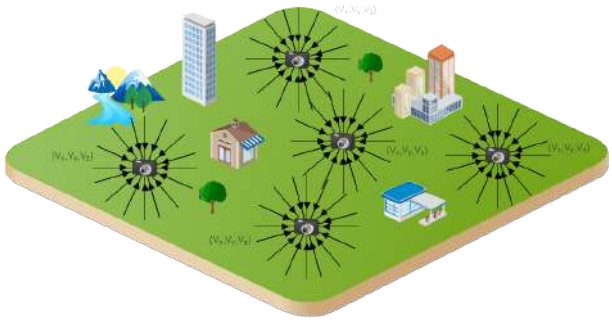


Figura 25 – Os parâmetros V_x, V_y, V_z que representam todos pontos de vista possíveis. Na Figura são apresentados alguns pontos de vista na forma de câmeras. Fonte: De autoria própria.

2.3.1 Função plenóptica 5D

A alta dimensionalidade proposta pela função plenóptica da Equação 14 é difícil de gravar e manipular na prática (WU et al., 2017). Mas pode-se simplificar essa função assumindo que a imagem é monocromática e invariante no decorrer do tempo como na Equação 16.

$$L(V_x, V_y, V_z, \theta, \phi) \tag{16}$$

Nessa proposta, o comprimento de onda de cada raio de luz é gravado de forma independente em canais de cores, e a sequência de tempo t é armazenada como uma sequência de *frames* se for o caso de um vídeo *light field*. Por fim se substitui (V_x, V_y, V_z) por (x, y, z) que indica a posição do ponto de origem (observador) no espaço 3D. Com essas modificações se chega à descrição da função plenóptica 5D na forma representada na Equação 17.

$$L(x, y, z, \theta, \phi) \tag{17}$$

2.3.2 Light Field 4D

Levoy e Hanraham (LEVOY; HANRAHAN, 1996) e Gorler *et al.* (GORTLER et al., 1996) fizeram uma nova simplificação na Equação 17. Considerando que o campo de luz (*light field*) está sendo medido em um **espaço livre**, pode-se assumir que a radiância permanece constante ao longo da linha de propagação. Desta forma se remove mais uma dimensão e se tem a representação chamada 4D *light field*.

Para se representar a 4D *light field*, a solução mais comum é parametrizar os raios de luz através da intersecção interna entre dois planos colocados em posições de forma arbitrária (WU et al., 2017). O sistema de coordenadas do primeiro plano é representado pelos eixos (u, v) e do segundo plano por (s, t) conforme a Figura 26, chegando assim à representação de um raio de luz L em um sistema 4D *light field* pelos pares de coordenadas $L(u, v, s, t)$, onde um raio de luz intercepta o primeiro plano nas coordenadas (u, v) e o segundo plano nas coordenadas (x, y).

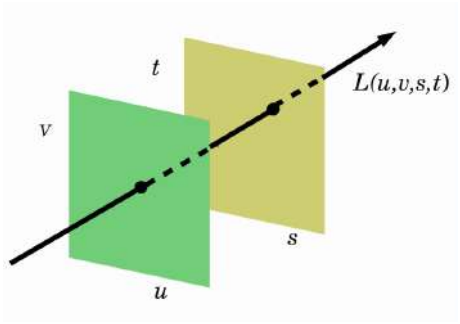


Figura 26 – Sistemas de coordenadas em um sistema 4D light field. Fonte: De autoria própria.

2.4 Câmeras LF

A estratégia de aquisição de imagens LF pode variar entre **esparsa** ou **densa** organizadas de forma **estruturada** ou **não estruturada**, usando câmeras comuns ou desenvolvidas especificamente para a captura de *light fields*. O termo **LF esparsa** ou **SLF** (do inglês *sparse light field*) é aplicado a grupamentos de câmeras distintas e individuais que realizam a aquisição de imagens LF, tanto de forma **estruturada** ou **não estruturada**.

Em uma **LF esparsa estruturada** se conhece a posição de cada câmera usada na aquisição. A forma de disposição da câmeras pode ser uma matriz, um vetor, ou mesmo uma única câmera que se desloca em um sistema mecânico capturando poses em posições pré-determinadas (BROXTON et al., 2020). Na Figura 27a temos uma LF esparsa composta por uma matriz de câmeras 8x12 (WILBURN et al., 2005), capaz de gerar vídeos e imagens. Na Figura 27b (FLYNN et al., 2019), a proposta é usar 16 câmeras GoPro® Hero4 dispostas em um plano variando os ângulos de visão (câmeras) usados na construção do LF dataset. Na Figura 27c temos um array de 16

GoPro® Hero4 que rotacionam sobre um eixo capturando as LF (OVERBECK et al., 2018). A Figura 27d mostra um conjunto de 47 câmeras Xiaomi® dispostas na parte côncava de disco parabólico (BROXTON et al., 2019).

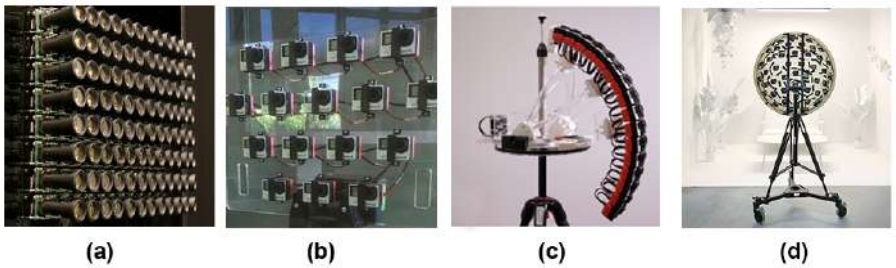


Figura 27 – Quatro sistemas de aquisição de light fields esparsas estruturadas.a (WILBURN et al., 2005),b (FLYNN et al., 2019),c (OVERBECK et al., 2018),d (BROXTON et al., 2019)

Em uma **LF esparsa não-estruturada** o conjunto de imagens é adquirido de forma livre onde a imagem LF é construída através da triangulação das imagens (DAVIS; LEVOY; DURAND, 2012). A Figura 28 mostra o processo. É feita a captura em vários ângulos, que irão formar um conjunto de pontos de vista de uma cena em comum. A **LF** é construída pela integração dessas perspectivas via algoritmos.

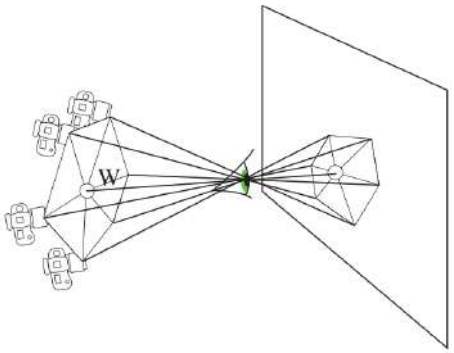


Figura 28 – Construção de uma **LF** esparsa não-estruturada. Fonte: Davis; Levoy; Durand (2012)

A abordagem **LF densa** ou DLF (do inglês Dense Light Field) utiliza um conjunto de microlentes responsável pela variação do ponto de vista, conforme será apresentado a seguir.

2.5 Câmeras Dense Light Fields - DLF

Dense Light Fields (DLF) são câmeras plenópticas compactas; elas usam uma matriz de microlentes entre a lente principal e o sensor de imagem. As câmeras DLF são geralmente divididas em duas categorias (ZHU et al., 2018):

- **ULF** *Unfocused Light Field* (desfocada) ou plenóptica 1.0;
- **FLF** *Focused Light Field* (focada) ou plenóptica 2.0;

Cada tipo será descrito a seguir de forma mais detalhada.

2.5.1 Plenóptica 1.0 (Unfocused Light Field)

A UFL foi proposta pela primeira vez por (ADELSON; WANG, 1992) em 1992. Essa configuração de lentes para captura de LF foi aprimorada em 2006 (NG, 2006). Posteriormente, esse aprimoramento se tornou uma câmera comercial produzida e vendida pela empresa Lytro®[2].

A Figura 29 mostra a configuração básica de uma câmera UFL (ZHU et al., 2018). Pode-se observar o sensor de captura no plano focal localizado a uma distância B após a matriz de microlentes. Essa matriz é chamada de **MLA** (do inglês *microlens array*).

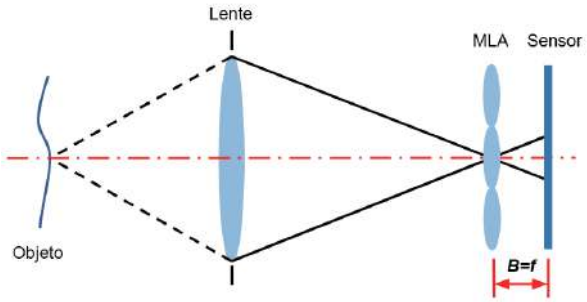


Figura 29 – Câmera plenóptica 1.0. Fonte: Adaptado de Zhu et al. (2018)

A MLA é responsável por mapear raios de luz de um mesmo local com diferentes direções para *pixels* vizinhos no sensor de captura (IKEUCHI, 2014). Dessa forma, a imagem registrada é uma matriz de *macropixels*; onde cada *macropixels* contém o conjunto de *pixels* vizinhos que armazenam informações de raios com determinado conjunto de direções (IKEUCHI, 2014). Esse conjunto de *macropixels* é chamado de microimagem. Na Figura 30 pode se observar uma fotografia *light field* bruta como é capturada por uma **DLF** com destaque para as microimagens na região ampliada.

A partir das microimagens é possível reconstruir as subaberturas (vistas). Na Figura 31a temos uma imagem LF bruta já retificada, composta por um conjunto de microimagens. Cada microimagem possui um grupamento de 3x3 *pixels* que representam perspectivas diferentes do mesmo **ponto** do **FoV**, com três vistas em destaque representadas por cores distintas na Figura 31a (azul, amarela e verde). Na Figura 31b estão reconstruídas as subaberturas correspondentes a cada perspectiva, processo que é feito associando cada *pixel* de uma microimagem a sua subabertura na coordenada correspondente. Pode-se observar que a Figura 31 usa uma MLA 3x3 já que gera microimagens desse mesmo tamanho. Já as imagens resultantes de cada plano (u, v) são matrizes 6x6.

[2]Essa empresa encerrou suas operações em 2018

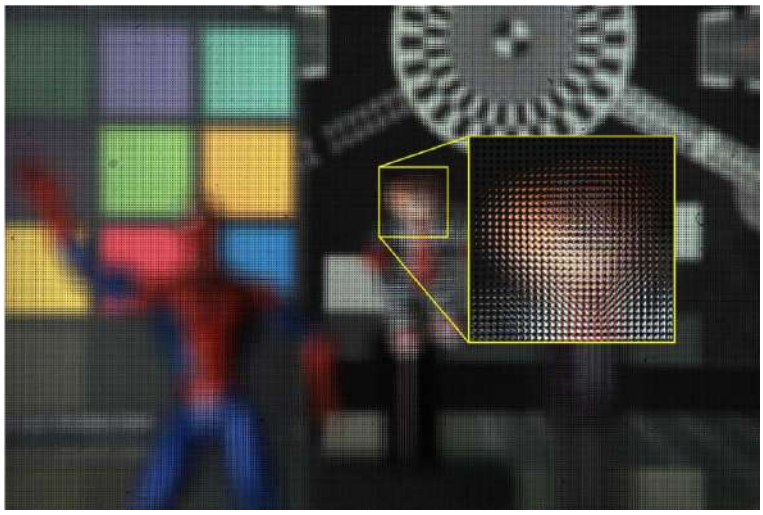


Figura 30 – Imagem bruta (*raw image*) de uma fotografia *light field*. A região ampliada mostra detalhes das micro imagens. Fonte: Hahne et al. (2016).

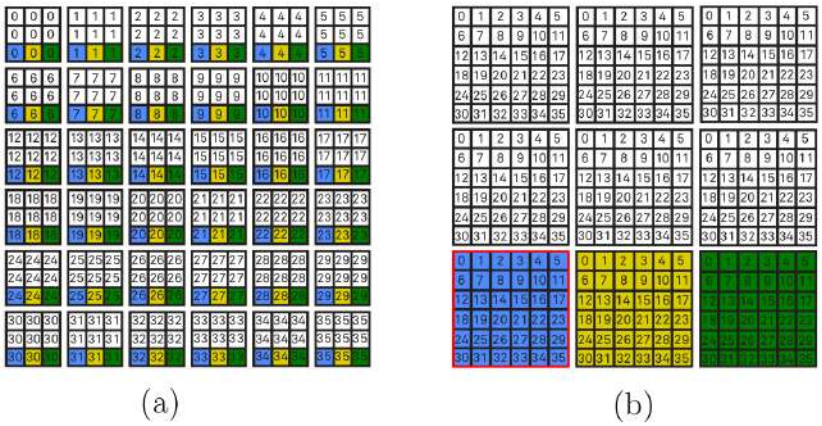


Figura 31 – (a) imagem *light field* bruta composta por microimagens. (b) Subaberturas extraídas. Fonte: Hahne (2016).

2.5.2 Plenóptica 2.0 (Focused Light Field)

As câmeras plenópticas 2.0 foram propostas por Lumsdaine e Georgiev (LUMSDAINE; GEORGIEV, 2009). Em termos gerais, uma câmera **FLF** difere de uma **ULF** na distância do MLA ao sensor de captura e no ponto onde a imagem intermediaria é projetada. Na Figura 32 são demostrados os dois tipos de configurações do conjunto de lentes: Kleperiana e Galileana (ZHU et al., 2018).

Na configuração Kepleriana o plano da imagem (imagem intermediária) é projetado em frente à MLA (LIU; JIN; DAI, 2017). Desta forma, as microlentes estão focadas em uma imagem intermediária real (LUMSDAINE; GEORGIEV, 2009). Já na configuração Galileana, o plano da imagem é projetado atrás do sensor, fazendo com que as microlentes estejam focadas em uma imagem intermediária virtual (LUMSDAINE; GEORGIEV, 2009). As câmeras produzidas pela empresa Raytrix®[3] usam esse tipo de

[3]https://raytrix.de

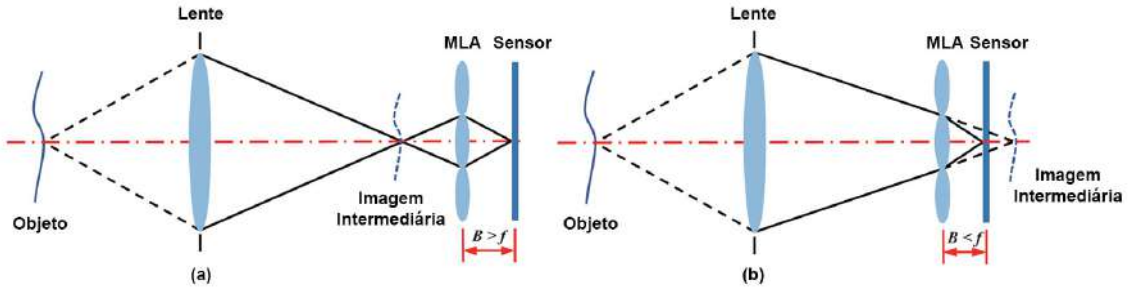


Figura 32 – Câmera plenóptica 2.0. (a) Configuração Kepleriana. (b) Configuração Galileana . Fonte: Adaptado de Zhu et al. (2018)

configuração.

2.6 A câmera Lytro Illum®

A câmera DLF usada na obtenção das imagens do *dataset* construído para esse trabalho e usada para os testes é a Lytro Illum®(Figura 33). Essa é uma câmera **plenóptica 1.0** densa do tipo SLR digital sem espelho que foi produzida pela empresa Lytro®. A Lytro ILLUM® possui uma abertura constante de diafragma com *f-stop* igual a $f/2$. Conforme a Tabela 2, a distância focal varia de 30 à 250 mm, como a câmera possui um fator de corte (*crop factor*) de $3,19$, a faixa focal efetiva é de $9,5 - 77,8$ mm. Convém ressaltar que ao aumentar a distância focal ocorre a compressão da faixa de profundidade de campo- **DoF** como em qualquer outra câmera.

Segundo (SCHAMBACH; PUENTE LEÓN, 2020), o sensor de captura possui resolução total de 7728×5368 *pixels*, tendo $1,4\mu m$ de área ocupada por pixel com profundidade de 10 bits por pixel e fator gama de $0,4$. As microlentes são dispostas em uma grade hexagonal, sendo estimado o desvio padrão ocasionado por ruído em $0,1\%$ do diâmetro da microlente, ou seja, $\sigma = 0,0143$ *pixels*. Como as microlentes possuem diâmetro aproximado de $20\mu m$ e *f-stop* fixo igual a $f/2$, o comprimento focal ideal é de $40\mu m$ (SCHAMBACH; PUENTE LEÓN, 2020).

Figura 33 – Câmera Lytro Illum usada nos experimentos e na construção do *dataset*. Fonte: De autoria própria.

Cada microlente cobre em torno de 225 pixels no sensor de captura (15x15) (SILVA, 2016), mas são geradas apenas 196 SAI's em vez de 225 (RANGAPPA et al., 2019).

Tabela 2 – Especificações Técnicas da Lytro Illum: Lentes. Fonte: De autoria própria.

Lentes	
Distância Focal	9.5 – 77.8 mm (equivalente a 30-250 mm)
Fator de Corte	3.19
Zoom	8.3x Óptico
Abertura de lente	Constante f/2.0
Macro Focus	0 mm a partir da frente da lente
Macro Ratio	1:3
Microlentes	≈ 200.000 lentes hexagonais

Tabela 3 – Especificações Técnicas da Lytro Illum: Sensor de Imagem e características de captura. Fonte: De autoria própria.

Sensor de Imagem e características de captura	
Tecnologia sensor	CMOS
Image ratio w:h	3:2
Sensor size	1/2"(6.4 x 4.8 mm)
Resolução da light field	40 Megaray
Resolução efetiva máxima	$\approx$ 4 Megapixels (2450x1634)
Resolução total do sensor	$\approx$ 40Megapixel (5300x7600 pixels)
Formato do sensor	1/1.2”
Área ativa	10.82 x7.52 mm
Faixa ISO	80-3200
Shutter speed mínimo	32 segundos
Shutter speed máximo	1/4000 segundos
Disparo contínuo	3.0 fps

Isso se deve ao fato de nem todos pixels possuírem informação viável por ocuparem posições limítrofes da microlente, sendo descartados na geração das SAI's. Deste conjunto de pixels desprezados, aproximadamente metade possuem informação parcial e o restante pouca ou nenhuma informação. Esse perda ocorre devido ao fenômeno de *vignetting* da lente, onde o brilho ou saturação da imagem capturada é reduzido a medida que se aproxima da borda da microlente (ZHANG, 2021). Desta forma, a microimagem na prática é uma matriz de 14x14 *pixels*[4].

A Tabela 3 apresenta algumas características da câmera. Nela se observa que apesar de não produzir vídeo, a Illum pode realizar capturas sequenciais de **3 frames por segundo**. Outras informações úteis são a resolução total, efetiva e de *light field*, que permitem dimensionar os *datasets* e a precisão intrínseca. A imagem de saída bruta sem processamento é de 5368x7728 (ŘEŘÁBEK; EBRAHIMI, 2016; SCHAMBACH; PUENTE LEÓN, 2020). A resolução da imagem light field capturada é de 40 *Megaray* de resolução angular. Esse valor registra o número de raios de luz capturados pelo sensor. Considerando a resolução total do sensor existem aproximadamente

[4]Esse valor é o usado pelo Lytro Desktop

200.000 microlentes com uso efetivo (Equação 18).

$$Microlentes = \frac{5300.7600}{196} \approx 205510$$
$$Microlentes = \frac{5300.7600}{225} \approx 179022 \tag{18}$$

O plano do sensor da câmera fica localizado na posição indicada pelo símbolo ϕ conforme a Figura 34. Para que se possa inferir as distâncias dos objetos capturados é necessário conhecer a distância da lente até o sensor (Figura 35).

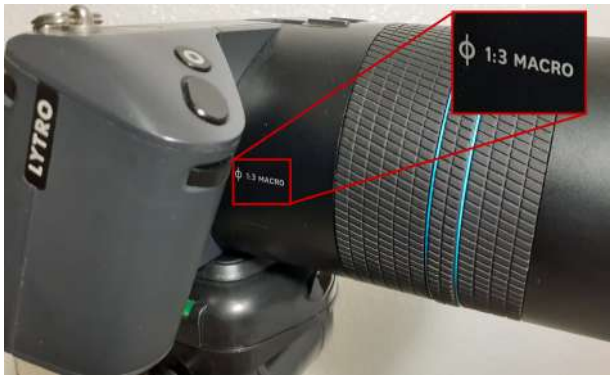


Figura 34 – O símbolo ϕ indica a posição do sensor CCD da câmera. O plano do sensor está a aproximadamente 115 mm da borda da lente. Fonte: De autoria própria.

A Figura 35 mostra de forma esquemática a incidência de dois raios luminosos em duas microlentes distintas. Esse comportamento dos raios luminosos na aquisição das imagens é usado na triangulação da imagem para o cálculo do mapa de profundidade.

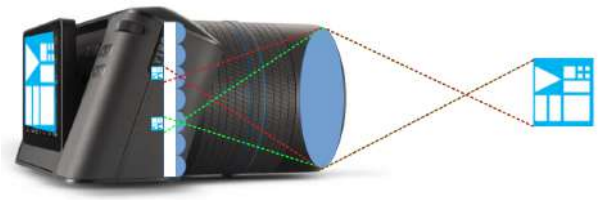


Figura 35 – O plano do sensor fica aproximadamente no local indicado na figura acima. Fonte: De autoria própria.

2.6.1 Lytro Desktop

O software Lytro Desktop é uma aplicação para manipulação e gerenciamento das imagens adquiridas via câmera Lytro Illum (formato LFR). Esse software permite gerar todas 196 subaberturas de um arquivo LFR em duas resoluções com 24 bits por pixel: 1620x1080 e 1080x720 no formato PNG. Além de criar e permitir manipular o mapa de profundidade. Observa-se que essas resoluções ultrapassam a capacidade máxima

do sensor de captura de acordo com a Tabela 3, desta forma o software usa uma técnica *não informada* de super-resolução para fazer o *upscaling* das imagens.

2.6.2 Lytro Power Tools

O Lytro Desktop é basicamente um editor de imagens, o que limita bastante seu uso na pesquisa das imagens light fields capturadas. Visando ampliar a utilização da câmera, a empresa Lytro criou o **Lytro Power Tools** tendo como público alvo programadores, desenvolvedores e pesquisadores, permitindo a realização de experiências com os dados do light field.

O **Lytro Power Tools** é um conjunto de ferramentas em **Python 2.7** para controle das imagens, uso na web e construção de aplicativos (LYTRO, 2015b). Nele existe um módulo que permite processar, importar, exportar e realizar operações de metadados em arquivos LFR chamado **Light Field Processing Tool**. As operações de interesse para essa proposta são:

- Processar arquivos brutos (*raw*) de imagens *light fields* (.LFR);
- Produzir imagens de profundidade de campo (*Extended Depth of Field* - EDOF) com tudo em foco (*all in focus*) em cinco formatos possíveis (jpeg, png, tiff, bmp, exr), de resolução de 2022 x 1404 x 24 bbp, com variações no eixos (u, v) de $[-0.3464, -0.2000]$ até $[0.3464, 0.2000]$;
- Gerar o mapa de profundidade em tons de cinza dos objetos de cena com resolução de 541x326x8 bbp e formatos png, bmp ou dat;
- Gerar imagens focadas em diferentes planos de distância;
- Gerar EDOF e imagens refocadas em diferentes valores de perspectiva (mudança do ponto de vista).

Cada imagem *raw* (.LFR) processado no formato .ESLF (*external standardized light field*) possui 7574×5264 *pixels*. Esse módulo também permite gerar sequências com mudanças de perspectiva nas coordenada U (eixo horizontal) e V (eixo vertical). O intervalo recomendado é de -0,5 a 0,5, mas são aceitos valores na faixa de -1,0 a 1,0. Essa ferramenta também gera uma mapa de profundidade relativo, usando uma técnica de alongamento de histograma no mapa de profundidade gerado. Isso faz com que não se possa usar o mapa de profundidade gerado **diretamente** como *ground truth*.

2.6.3 Light Field Toolbox for Matlab

Light Field Toolbox for Matlab (**LFT**) (DANSEREAU, 2020) é um conjunto de ferramentas para trabalhar com imageamento de *light fields* em MATLAB. Essa toolbox é

usada na geração dos datasets disponíveis no **JPEG Pleno framework**[5] de acordo com o as condições de teste comum estabelecidas em (ISO; IEC; JTC, 2019). A LFT representa as LF como uma pilha de imagens RGB de baixa resolução. As imagens possuem resolução de 434x625, com 3 canais de cor RGB mais um canal de ponderação e 15x15 SAI (ŘEŘÁBEK; EBRAHIMI, 2016).

As características estabelecidas nesses datasets [6] são (ISO; IEC; JTC, 2019):

- Aquisição feita em câmera Lytro Illum;
- Formato - imagens PPM (componentes de cores RGB, não entrelaçados);
- Conteúdo - natural, ao ar livre;
- 15 × 15 SAI, mas apenas as visualizações centrais 13 × 13 são usadas para evitar o uso das visualizações escuras associadas ao *vignetting*;
- Resolução espacial - 625 × 434;
- Profundidade de bits - 10 bits;
- Mapa de profundidade para imagem de sub-abertura central.

Existe um problema em usar a LF Toolbox para estimar profundidade. As câmeras Lytro sofrem **distorções duplas**, na lente principal e nas microlentes. A **Light Field Toolbox** lida apenas com a distorção da lente principal, desta forma, para se fazer uma estimativa precisa da profundidade deve-se usar uma *toolbox* geométrica, como a proposta em (BOK; JEON; KWEON, 2017) para uma estimativa precisa da profundidade.

2.6.4 Biblioteca Plenpy.

Este é uma biblioteca em Python para calibrar, processar e analisar imagens LF (SCHAMBACH; PUENTE LEÓN, 2020). Por ser escrito em Python permite a conexão direta com o OpenCV e outras toolboxes de processamento de imagens e de aprendizado de máquina, o que torna um ferramenta versátil na proposta apresentada (SCHAMBACH, 2021).

2.7 Considerações finais

Nesse capítulo foram introduzidos conceitos básicos de óptica e captura de imagens em câmeras digitais. Esses princípios são importantes para o entendimento da

[5] https://jpeg.org/jpegpleno/
[6] https://jpeg.org/jpegpleno/plenodb.html

formação e captura de retratos em dispositivos com sensores digitais e como modificações em parâmetros, tais como profundidade de campo e abertura de diafragma, influenciam o resultado final. Na sequência foram apresentados conceitos de imagens estéreo, com o objetivo de caracterizar a construção de mapas de profundidade e detalhar a triangulação usada para o cálculo de distâncias em uma cena. Convém destacar que a triangulação com apenas 2 vistas (estéreo) é a versão mais simples quando se trata de cálculo de distâncias envolvendo n-vistas. Os conceitos de imagem plenóptica foram detalhados a partir do conhecimento previamente apresentado. Por fim, foi introduzida a câmera plenóptica usada no projeto e junto suas características básicas. Os principais softwares usados também foram comentados nesse capítulo.

As informações desse capítulo são a base para o entendimento de como se dá a formação e a captura de uma imagem plenóptica. Esse conhecimento é necessário para justificar as escolhas tomadas no desenvolvimento do projeto. No próximo capítulo serão abordado conceitos básicos de redes neurais e aprendizado de máquina.

3 REDES NEURAIS ARTIFICIAIS E PROBLEMAS DE REGRESSÃO

Em um problema de regressão o objetivo é predizer o valor de uma ou mais variáveis contínuas t dado um vetor de entrada D-dimensional com x valores (BISHOP, 2006). Esse é um problema característico de aprendizado supervisionado, onde se tem um conjunto de dados de treinamento compreendendo N observações x_n, sendo $n = 1, ..., N$, com seus valores alvos correspondentes t_n, o objetivo é predizer o valor de t para um valor x de entrada inédito (BISHOP, 2006). Existem várias técnicas para análise de regressão: modelos lineares paramétricos, modelos não lineares paramétricos, métodos baseados em *kernels*, etc. Uma das principais abordagens usadas em problemas de regressão são redes neurais artificiais.

O termo **Rede Neural Artificial** ou apenas **Rede Neural** (**RN**) é utilizado para classificar um subconjunto de algoritmos de aprendizado de máquina dentro da IA que possuem uma abordagem **livremente** inspirada nas redes neurais biológicas. Por livremente se entende que elas não possuem por objetivo simular *ipsis litteris* o comportamento biológico de uma rede neural, e desta forma, existe liberdade na seleção e na forma de uso das características a serem implementadas. Esses algoritmos apresentam grande variação em suas arquiteturas e aplicações, mas possuem elementos em comum que recebem a mesma nomenclatura de seus análogos biológicos.

Em termos mais gerais uma rede neural é composta por um conjunto de unidades ou nodos que realizam processamento dos sinais entradas de forma paralela chamados **neurônios artificiais** ou simplesmente **neurônios**. Esses neurônios possuem entradas ponderadas de sinais, e se conectam entre si formando a rede neural propriamente dita. A forma como esses neurônios são estruturados, o fluxo de informação, como se dá o aprendizado, as funções de transferência usadas, a quantidade de camadas, as conexões entre neurônios e outras características compõe o que se chama **arquitetura** da rede neural. A seguir serão detalhados alguns conceitos básicos.

3.1 Rede Neural Artificial

Uma rede neural artificial é um modelo não-linear paramétrico, sendo que a arquitetura mais bem sucedida no contexto de reconhecimento de padrões é o *feedforward* ou perceptron multicamadas (BISHOP, 2006). Atualmente as redes neurais de aprendizado profundo (DNN-*Deep neural network*) são o estado da arte em termos de desempenho quando são usados grandes *datasets* no treinamento. As redes neurais tradicionais podem ser classificadas de forma geral em três arquiteturas (HAYKIN, 2009):

- Redes *feedforward* de camada única;
- Redes *feedforward* com múltiplas camadas;
- Redes recorrentes.

As **redes *feedforward* de camada única** possuem uma camada de entrada ligada diretamente aos neurônios da camada de saída conforme a Figura 36a. Nessa rede a propagação é apenas adiante e apesar de possuir "duas"camadas, a camada de entrada não é contabilizada como camada da rede neural por não realizar nenhuma computação. Esse tipo de rede é um grafo acíclico direcionado com nós de entrada e saída pré-estabelecidos (RUSSELL; NORVIG, 2020). O fluxo da informação se dá dos nós de entrada para saída, onde cada nó computa suas entradas de acordo com uma função (função de ativação) e passa os resultados para sua saída.

As **redes *feedforward* com múltiplas camadas** (do inglês *multilayer feedforward networks*) se diferenciam das de uma única camada, justamente por possuírem uma ou mais camadas ocultas (Figura 36b). Essas camadas recebem este nome por encontrarem-se entre a camada de entrada e a camada de saída conforme a Figura 36b. Por último, as **redes recorrentes** (do inglês *recurrent networks*[1]) se diferenciam das redes *feedforward* por apresentarem ao menos um *loop* de realimentação, conforme a Figura 36c.

3.2 Funções de Ativação

A escolha da função de ativação $\Phi(.)$ é um ponto crítico no projeto de uma rede neural (AGGARWAL, 2018), sendo seu principal objetivo introduzir não-linearidade na saída de um neurônio e limitar a amplitude do sinal de saída de um neurônio (HAYKIN, 2009). Entre as características de uma função de ativação $\Phi(.)$ pode se observar a existência ou ausência das seguintes características (KOVÁCS, 2006):

- **Monotonicidade** - comportamento monotônico sobre uma faixa dinâmica;

[1] Não confundir com *recursive neural network* que são uma rede de aprendizado profundo.

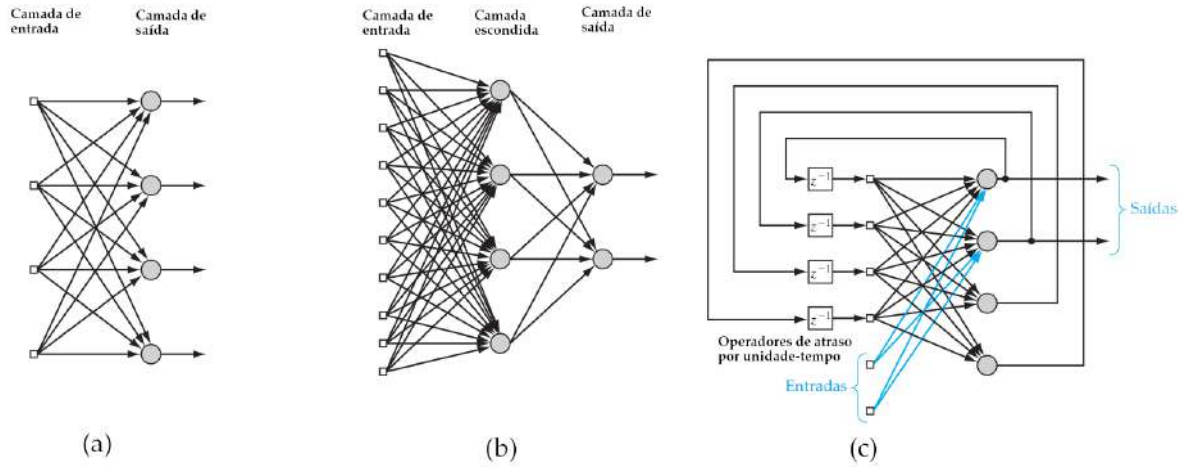


Figura 36 – Arquiteturas tradicionais de redes neurais. Fonte: Haykin (2009)

- **Saturação** - saída saturada fora da faixa dinâmica x.

Além dessas características, a função de ativação deve ser derivável para que a rede neural aprenda. Essa propriedade é necessária para o cálculo da derivada do erro em relação aos pesos em cada camada, da saída em direção a entrada, de forma a minimizar a função de perda definida na camada de saída. Pelo mesmo motivo, é importante que a função de ativação não desloque o gradiente para zero [2], pois isso pode eliminar totalmente o valor de gradiente durante a retropropagação a medida que se avança na direção das primeiras camadas. Outra característica importante é $\Phi(.)$ ser centrada no valor zero de forma a evitar que os gradientes mudem para uma direção específica.

As funções de ativação utilizadas no desenvolvimento inicial das RN e consideradas clássicas são: de sinal, sigmoide e tangente hiperbólica (Figura 37 b-d). Já as funções como a ReLU (*Rectified Linear Unit*) e Hard Tanh (*Hard hyperbolic tangent*) 37 e-f) dominam aplicações que usam aprendizado profundo. Na Figura 38 estão representadas as derivadas das funções de ativação citadas.

A função de ativação depende do tipo de aplicação e saída da rede ou neurônio. Por exemplo, em um perceptron na camada de saída onde se faz uma rotulagem binária, o uso de uma função do tipo sigmóide (Figura 37c) é a mais adequada. Se a saída da rede for um valor real, a função identidade ou função linear (Figura 37a) pode ser aplicada. Observa-se que a função identidade não fornece nenhuma não linearidade, e portanto serve como um mapeamento entre os valores de entrada e a saída $\Phi(.)$. Pela Equação 19 se observa que a mesma é monotônica, mas não saturada (KOVÁCS, 2006). A função linear também pode ser aplicada em saídas discretas quando se necessita criar uma função de perda suavizada na saída (AGGARWAL, 2018).

[2] Problema de desaparecimento de gradiente, do inglês:*vanishing gradient problem*

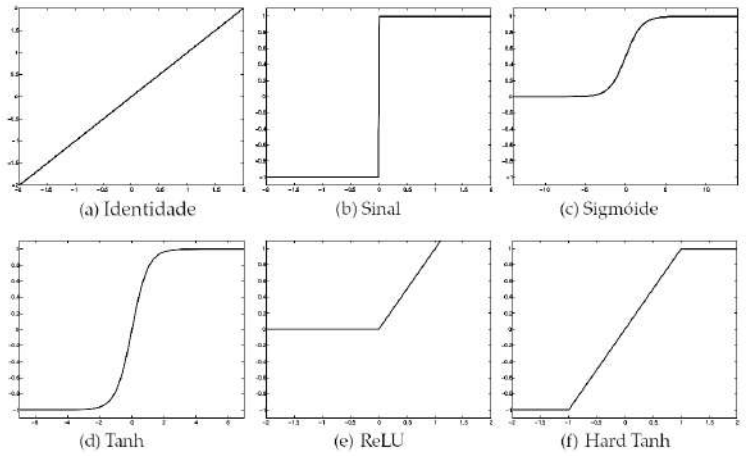


Figura 37 – Funções de ativação. Fonte: Aggarwal (2018).

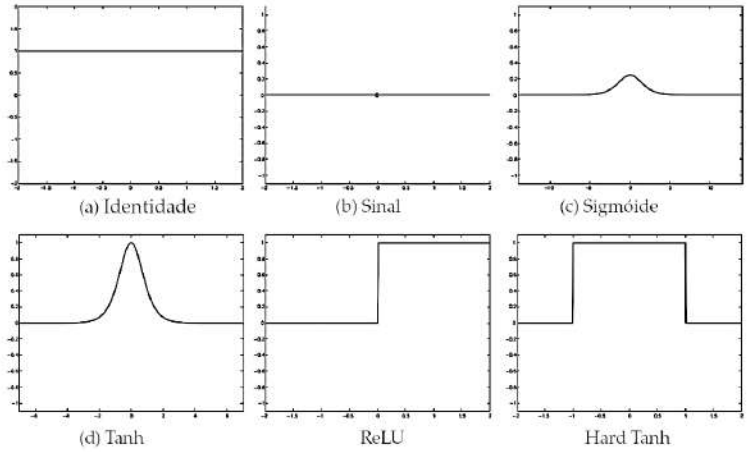


Figura 38 – Derivadas das funções de ativação. Fonte: Aggarwal (2018).

$$\Phi(x) = x \tag{19}$$

A função sinal usada no modelo de McCulloch & Pitts (KOVÁCS, 2006), também chamada de degrau (Equação 20), não é monotônica, apenas a saturação é mantida. O principal problema dessa função é o fato de não ser derivável no ponto onde $x = 0$ e sua derivada ser igual a **zero** para valores onde $x \neq 0$, o que impossibilita o aprendizado em redes multicamadas (Figura 38b).

$$\Phi(x) = \begin{cases} 0, & \text{se } x < 0 \\ 1, & \text{se } x \geq 0 \end{cases} \tag{20}$$

A função sigmoide da Equação 21 é uma das formas mais comuns de função de ativação usadas em redes neurais tradicionais (HAYKIN, 2009). Ela é contínua e portanto derivável. Quando essa função é usada em redes neurais com muitas ca-

madas (redes de aprendizado profundo) apresenta o **problema de desaparecimento de gradiente** (*vanishing gradient problem*), ou seja, em redes com muitas camadas o valor de gradiente é zerado na retropropagação, impedindo o aprendizado. A função tangente hiperbólica da Equação 22, possui a vantagem, em relação a função sigmoide, de apresentar valores negativos na saída. Infelizmente ela também apresenta o problema de desaparecimento de gradiente, o que dificulta seu uso em redes de aprendizado profundo.

$$\Phi(x) = \frac{1}{1 + e^{-x}} \tag{21}$$

$$\Phi(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{22}$$

A função **ReLU** (do inglês *rectified linear unit*), é amplamente usada em camadas escondidas de DNN, pois é de fácil computação (baixa complexidade), não satura e não causa o problema de desaparecimento de gradiente. Pela a Equação 23 observa-se que a função devolve zero para valores negativos e a identidade para valores positivos ($\Phi(x) = max(0, x)$). Apesar de não ser derivável para $x = 0$, pode-se escolher de forma arbitrária um valor de 0 ou 1 nesse ponto. Ao substituir valores positivos pela identidade e tornar qualquer valor negativo em zero, a ReLU evita o efeito em cascata de um valor baixo puxar outras entradas também para baixo e acabar por zerar os valores a medida que avança na retropropagação, causando o problema de desparecimento do gradiente. Um efeito colateral do uso dessa função é o problema da **ReLU morimbunda** (do inglês *Dying ReLU problem*) que ocorre quando temos uma taxa de aprendizado alta com muitos valores ao mesmo tempo negativos, isso faz com que o ReLU torne todas suas entradas inativas reduzindo a capacidade de aprendizado da rede.

$$\Phi(x) = \begin{cases} 0, & \text{se } x < 0 \\ x, & \text{se } x \geq 0 \end{cases} \tag{23}$$

A função ***hard hyperbolic tangent*** (Equação 24) tem sido utilizada nas modernas RN pelos mesmos motivos da **ReLU**.

$$\Phi(x) = \begin{cases} -1, & \text{se } x < -1 \\ x, & \text{se } -1 \leq x \leq 1 \\ 1, & \text{se } x > 1 \end{cases} \tag{24}$$

Outra função de ativação atualmente usada é a **Softmax**, também chamada de softargmax ou função normalizada exponencial (do inglês *normalized exponential function*). Esta função é geralmente usada na saída de um classificador para representar

a distribuição de probabilidade entre as classes (GOODFELLOW; BENGIO; COURVILLE, 2016). Por exemplo, na Equação 25 tem-se a função **Softmax** $\Phi(x)_i$ sendo aplicada sobre x, que é o vetor de saída da última camada, onde K é o número de classes e, portanto, x obrigatoriamente deve ter a mesma dimensão de K. Nessa mesma equação j representa a classe cuja probabilidade categórica se deseja calcular.

$$\Phi(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_i}} \tag{25}$$

3.3 Redes *feedforward* com camada única: O Perceptron

O Perceptron é uma rede neural artificial de camada única com alimentação adiante (*feedforward*) proposta por Frank Rosenblatt em 1958. O neurônio do Perceptron é composto por um combinador linear v associado a limitador abrupto $\Phi(v)$ (função de ativação sinal) que introduz não linearidade em sua saída e funciona como um classificador linear (binário) (HAYKIN, 1998). Um nodo de entrada adicional chamado viés ou *bias* b pode ser adicionado (Figura 39). O *bias* é um termo constante que não depende de qualquer valor de entrada e permite ajustar a saída aumentando ou diminuindo a entrada da função de ativação (HAYKIN, 2009) e dessa forma deslocar a saída y da função de ativação para esquerda ou direita em relação ao ponto de origem dos eixos $(0, 0)$ adiantando ou atrasando o disparo de $\Phi(.)$.

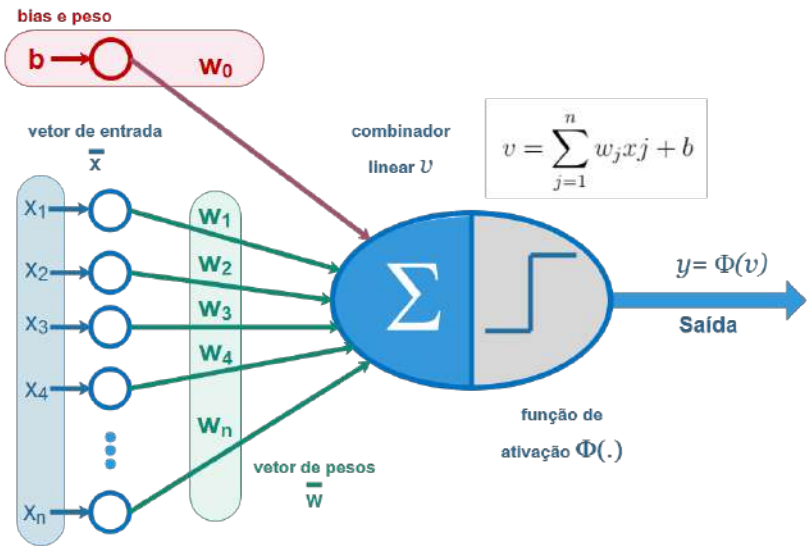


Figura 39 – Arquitetura básica de um Perceptron. Fonte: De autoria própria.

Pela Figura 39 observa-se que o combinador linear v recebe o vetor de entradas $\overline{X}$ junto com seu vetor de pesos $\overline{W}$, onde $\overline{X} = [x_1, ..., x_n]$ e $\overline{W} = [w_1, ..., w_n]$, podendo

ou não possuir o *bias* b, e aplica uma soma ponderada sobre a entrada (Equações 26,27,28).

$$b = 1.w_0 \tag{26}$$

$$v = \sum_{j=1}^{n} w_j xj = \overline{X}.\overline{W}^{\mathsf{T}} \tag{27}$$

$$v = \sum_{j=1}^{n} w_j xj + b = \overline{X}.\overline{W}^{\mathsf{T}} + b \tag{28}$$

A saída y do perceptron (Equação 29) gera uma etiqueta de classe binária através da aplicação da função de ativação sinal $\Phi(.)$ (Equação 20) sobre o valor agregado v.

$$y = \Phi(v) \tag{29}$$

A maioria dos modelos básicos de aprendizado de máquina como regressão de mínimos quadrados com alvos numéricos, máquina de vetores de suporte, ou classificador de regressão logística, podem ser facilmente representados nessa arquitetura de rede neural simples através da escolha de diferentes funções de ativação (AGGARWAL, 2018).

A principal limitação dessa rede é o fato do Perceptron ser incapaz de trabalhar com classes de problemas não separáveis linearmente, pois como o mesmo é uma combinação de discriminadores lineares, e toda combinação de discriminadores lineares pode ser substituída por uma única função discriminadora linear, é impossível resolver problemas não separáveis linearmente como o representado na Figura 40.

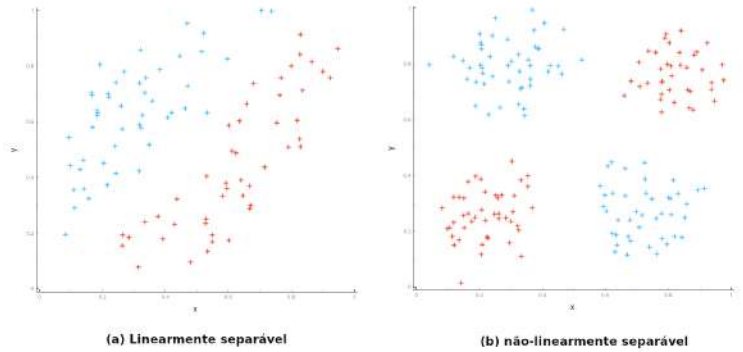


Figura 40 – Exemplo de dados com duas classes distintas. Em (a) os dados estão distribuídos de forma a serem linearmente separáveis, em (b) as classes são linearmente inseparáveis. Fonte: De autoria própria.

3.4 Redes *feedforward* com múltiplas camadas: *Multilayer Perceptron*

A incapacidade do Perceptron de camada única de separar/classificar classes não-linearmente separáveis (Figura 40) fez com que a área de redes neurais ficasse estagnada com poucos investimentos na década de 70, apesar da pesquisa continuar em alguns nichos acadêmicos nessa época. A resposta para o problema do perceptron, entre outras contribuições de vários cientistas, foi o desenvolvimento do algoritmo *backpropagation* relatado por Rumelhart, Hilton e Willians em 1986 [3] (HAYKIN, 1998) e permitiu o treinamento de redes perceptrons de multicamadas e a resolução de problemas não separáveis linearmente.

Uma rede perceptrons de multicamadas (PMC), ou em inglês multilayer perceptrons (MLP), é a base do que chamamos de modelo de aprendizado profundo (do inglês *deep learning*) (GOODFELLOW; BENGIO; COURVILLE, 2016).

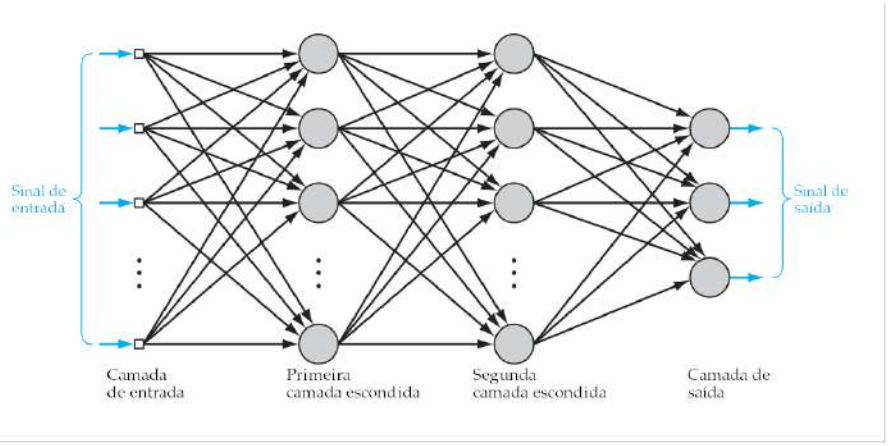


Figura 41 – Perceptron Multicamada com duas camadas escondidas. Fonte: Haykin (2009).

O MLP, conforme se observa na Figura 41, é uma rede de tipo *feedforward* composta por mais de um perceptron e com n camadas escondidas, cujo objetivo é aproximar alguma função f^*. Em um classificador, isso significa termos uma saída $y = f(x)$ que mapeia uma entrada x em uma classe y. Esse tipo de rede define um mapeamento na forma $y = f(x; \theta)$ onde θ representa os valores dos parâmetros que precisam ser aprendidos de forma que a saída y apresente a melhor aproximação a função $f(x)$ (GOODFELLOW; BENGIO; COURVILLE, 2016). Na MLP o treinamento é feito de forma supervisionada através do algoritmo de retropropagação de erro (em inglês *error backpropagation*) (HAYKIN, 1998). Para que a rede possa aprender de forma supervisionada, são apresentados para a rede os pares de vetores de entrada $\overline{X}$ e o vetor de resposta desejada $\overline{Y}$ (alvo) e então é aplicado o algoritmo de aprendizagem por correção de erro. A ideia básica do treinamento supervisionado é modificar os pesos das conexões da rede de forma que a saída gerada para o vetor de entrada pela

[3] O algoritmo de *backprogation* foi proposto de forma independente em mais dois outros lugares.

rede (valor real) seja o mais próximo possível do vetor de saída apresentado (valor desejado). Essa estrutura permite que a MLP aprenda e possa generalizar de forma a apresentar saídas corretas para entradas não treinadas.

O *backpropagation* possui basicamente dois passos através das camadas da rede: um passo para frente, a **propagação**, e um passo para trás, a **retropropagação**. Na **propagação**, é apresentado na camada de entrada o padrão (vetor de entrada $\overline{X}$) que desejamos que a rede aprenda, sendo seu efeito de ativação propagado por toda a rede, camada por camada, até gerar a ativação dos neurônios na camada de saída. Durante o passo de propagação, os pesos sinápticos da rede são todos fixos (HAYKIN, 1998). Na **retropropagação** os pesos sinápticos são todos ajustados de acordo com uma regra de correção de erro. Na camada de saída é calculado o sinal de erro que é a resposta real gerada pela rede subtraída da resposta desejada. Esse sinal de erro é propagado em direção as camadas de entrada. Ao realizar a retropropagação os pesos sinápticos são ajustados de modo que a resposta real da rede se mova na direção da resposta desejada, reduzindo dessa forma o erro (descida de gradiente).

3.5 Redes neurais com aprendizado profundo

Conforme já citado, algumas funções de ativação possuem o problema de desaparecimento de gradiente quando aplicadas a redes que usam métodos de aprendizado baseado em gradientes. Algumas das principais soluções propostas para mitigar esse problema foram (SCHMIDHUBER, 2015):

- Pré-treinamento não supervisionado de **RNN** (*recurrent neural network*) hierárquicas;
- Desenvolvimento de redes **LSTM** (*long short-term memory*), arquitetura não afetada pelo problema;
- Criação e uso de hardware mais potente que o utilizado durante os anos 1990. Mesmo em redes neurais tradicionais, isso permitiu aumentar a propagação de erros em mais algumas camadas dentro de um tempo razoável de processamento. Essa abordagem mitigou, mas não resolveu o problema;
- Uso da otimização *Hessian-free*;
- Uso de outras funções de ativação como ReLU.

Essas soluções permitiram vários avanços na área nos últimos 20 anos. Atualmente temos uma grande variedade de arquiteturas e aplicações que exploram esses desenvolvimentos. Nas próximas seções será apresentada a DNN relacionada com essa proposta de tese: as *convolutional neural networks* (CNN).

3.6 Filtragem espacial e filtros convolucionais

A convolução é uma das técnicas de filtragem utilizada quando se trabalha com imagens no domínio espacial. O objetivo de aplicar um filtro em uma imagem é destacar ou atenuar certas características de interesse. A ideia básica da convolução é transformar uma imagem original em uma imagem destino, percorrendo todos *pixels* da imagem original e aplicando sobre cada *pixel* alvo uma matriz de convolução que irá gerar um novo valor de acordo com o valor original do *pixel* e dos *pixels* em sua vizinhança. A matriz de convolução é chamada de *kernel* ou máscara. As dimensões de um *kernel* e seus valores determinam o efeito de transformação do processo de convolução.

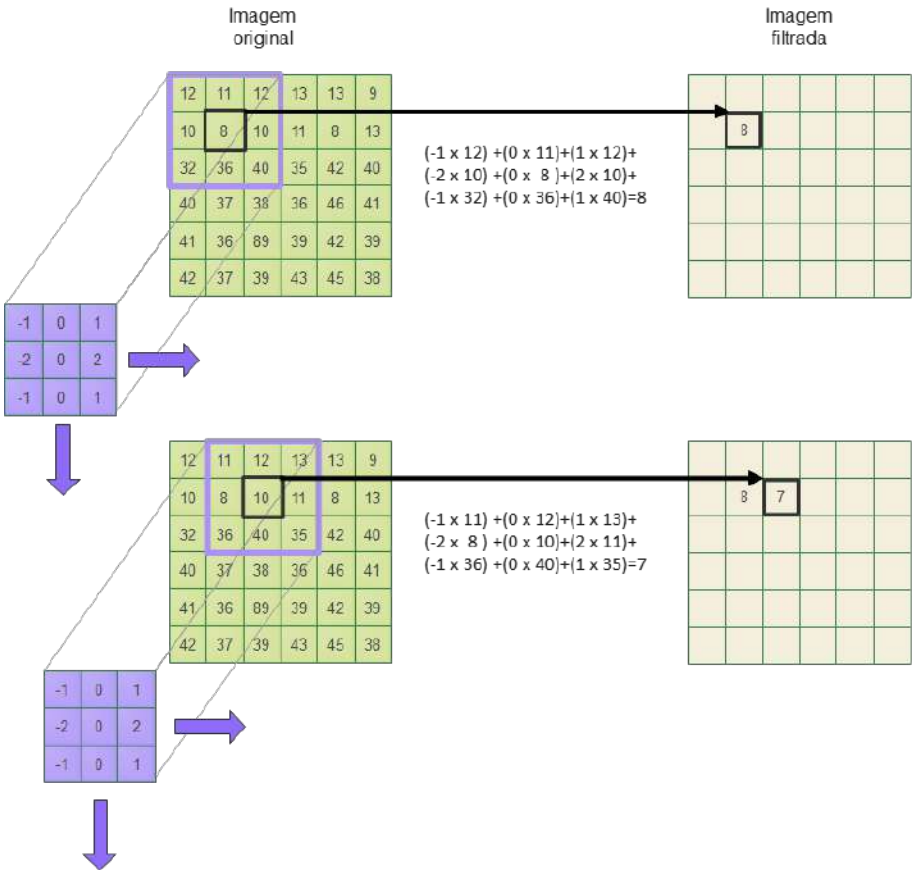


Figura 42 – Convolução de um *kernel* de tamanho 3x3. Essa máscara é o filtro de Sobel G_x. Fonte: De autoria própria.

Na Figura 42 pode se observar o efeito de se usar um filtro de Sobel G_x sobre os dois primeiros *pixels* alcançáveis pelo *kernel* de tamanho 3x3. Os *pixels* não alcançáveis são chamados de bordas e devem receber tratamento especial dependendo da aplicação.

As técnicas de filtragem são usadas na extração de características, buscando reduzir a complexidade das imagens de entrada de uma rede neural tradicional. Em redes neurais convolucionais é exatamente isso que a camada convolucional faz. Ela

aprende *kernels* que destacam características que melhor descrevem a imagem, com a vantagem dos filtros aprendidos não terem necessariamente estrutura e valores de *kernels* usuais de processamento de imagens.

3.7 Rede Neural Convolucional

As Redes Neurais Convolucionais (ConvNets ou CNNs) são redes neurais artificiais profundas que podem ser usadas para classificar imagens, agrupá-las por similaridade e realizar reconhecimento de objetos como indivíduos e sinais de rua em uma imagem.

Uma CNN (do inglês *convolutional neural networks*) pode ser descrita, de forma sintetizada, como separada em dois módulos (Figura 43): um módulo para extração de características e outro para classificação.

O aprendizado dessa rede é do tipo supervisionado, onde são apresentadas as entradas com as respectivas saídas desejadas. A CNN recebe o padrão de entrada do objeto a ser reconhecido. Esse padrão é submetido a uma camada de convolução que aprende quais filtros/*kernels* (matrizes de convolução) devem ser usadas para representar os dados de entrada, em outras palavras, aprende que padrões espaciais são característicos do objeto a ser classificado. Após a etapa de convolução segue-se uma etapa de subamostragem, também chamada de *pooling*, em que se reduz a dimensionalidade dos filtros da camada anterior. Esse processo continua até o fim do módulo de extração/aprendizado de características. Por fim, os padrões espaciais extraídos são apresentados a uma rede neural totalmente conectada, por exemplo, uma rede MLP com algoritmo de aprendizado *backpropagation*.

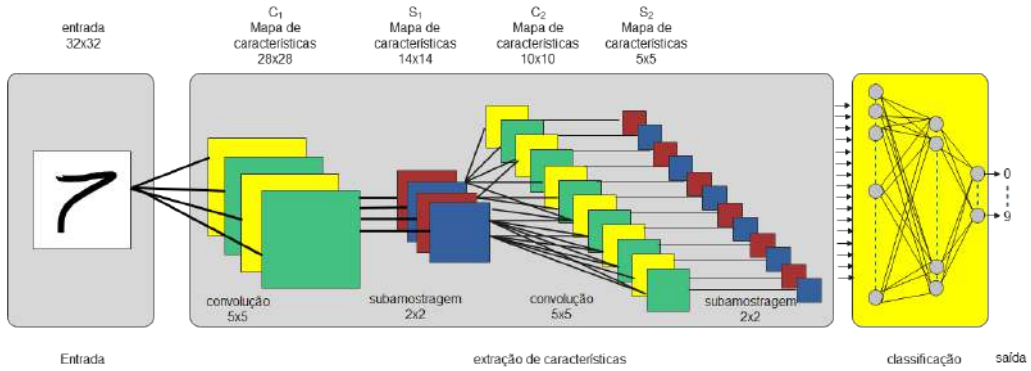


Figura 43 – CNN sendo usada para reconhecimento de dígitos escritos a mão. A entrada é uma imagem e a saída o digito identificado na faixa de 0 a 9. Fonte: De autoria própria.

A **camada convolucional**, conforme já citado, é responsável pelo aprendizado dos filtros independentes que irão representar as características do objeto de entrada. Para que a camada de convolução funcione corretamente é necessário determinar como se dá o deslocamento do *kernel* do filtro a ser aprendido, em outras palavras,

qual será o tamanho da passada do *kernel*, ou em inglês ***stride***. Também é necessário determinar qual será o tratamento das bordas da imagem. A técnica de preenchimento, chamada em inglês de ***padding***, é a normalmente usada. Essa técnica, acrescenta bordas a imagem de entrada visando manter coerente a saída da convolução. Essa duas técnicas, *stride* e *padding*, determinam o tamanho da camada de saída.

Cada filtro aplicado gera uma imagem chamada **mapa de características**. Na Figura 43 foram aplicados, na primeira camada de convolução C_1, quatro *kernels* 5x5 que geraram quatro (4) mapas de características 28x28. Observa-se então que a **profundidade da camada de convolução** indica quantos filtros são aprendidos na respectiva camada (ex. 28x28x4).

Após a camada de convolução segue-se geralmente uma camada de subamostragem (*pooling*). O objetivo é diminuir a dimensionalidade buscando reduzir o número de parâmetros e custo computacional. Esse processo abrevia o tempo de treinamento e controla o *overfitting*. As técnicas mais usadas são o *max pooling* que pega o valor mais alto de uma janela do *kernel* aplicado. Por exemplo, na Figura 43, na saída resultante do *pooling* foram gerados quatro mapas de características com dimensões reduzidas para 14x14.

Outra técnica importante aplicada é o ***dropout***, ou abandono de neurônios. O objetivo do *dropout* é simplificar a rede neural removendo conexões que não contribuem para o aprendizado. Por fim a **normalização em lote** ou ***batchnorm*** faz a regularização através da estandardização e normalização de valores para evitar o *overfitting* e fixar os valores da rede dentro de uma faixa específica. As funções de ativação mais usadas são as **ReLU** e a **softmax**.

Existem inúmeras variações da CNN como: R-CNN (Region Based Convolutional Neural Networks), Fast R-CNN, Faster R-CNN, U-Net, etc.

3.8 Autoencoders e Redes Neurais artificiais U-shaped

Esta seção trata de um grupo de arquiteturas de redes neurais convolucionais nas quais a principal característica é apresentar na saída de cada grupamento de convoluções uma redução nas dimensões em relação a resolução de entrada (*enconder*) e depois reconstruir o sinal original seguindo o processo inverso (*decoder*).

3.8.1 Autoencoder

Autoencoders (**AE**) são redes neurais treinadas para gerar em sua saída uma cópia da entrada (GOODFELLOW; BENGIO; COURVILLE, 2016). A ideia básica é usar redes de convolução para aprender filtros que capturem a representação da entrada a cada camada, reduzindo as dimensões de entrada, e dessa forma chegar a um con-

junto de atributos relevantes chamado de **espaço de características latentes**, que será usado para reconstruir a imagem original. Apesar do propósito primário de um autoencoder ser a reconstrução de sua entrada, uma vez que os atributos da imagem de entrada são aprendidos, pode-se manipular essa informação na reconstrução do sinal, de forma a recuperar apenas características específicas (MICHELUCCI, 2022). Essa propriedade seletiva permite o uso de autoencoders em aplicações de segmentação, filtragem e restauração, por exemplo.

Conforme se observa na Figura 44, a estrutura básica de um autoencoder é formada por três componentes básicos: (i) entrada/codificador **e**; (ii) espaço de representação latente **h**, que codifica as características latentes; e, (iii) saída/decodificador **d** (YALÇIN, 2021).

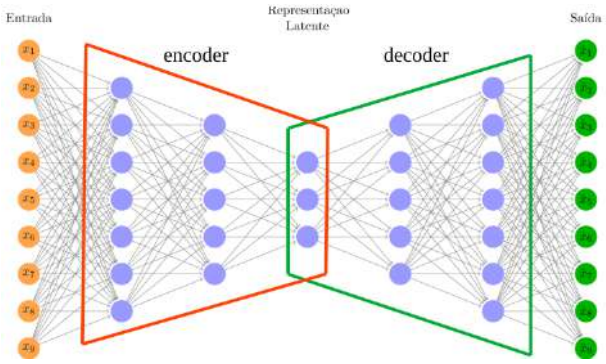


Figura 44 – Componentes básicos de um *autoencoder*: encoder (codificador), espaço de representação latente, e decoder (decodificador). Fonte: De autoria própria.

As camadas de entrada mais o espaço latente formam a etapa de codificação **e**, cuja saída h (espaço de representação latente) é a função de codificação $h = e(x)$, aplicada ao dado de entrada x. Já o espaço latente h mais as camadas de saída formam a etapa de decodificação **d**, que reconstrói o sinal de entrada x, gerando a imagem reconstruída de saída r, através de uma função $r = d(h)$. Autoencoders funcionam como gargalos em que a dimensionalidade dos dados é reduzida a cada etapa e o número de filtros para extração de características sofre um aumento, até a geração do espaço de representação latente. Na etapa de decodificação o processo se inverte culminando na imagem reconstruída (Figura 45).

3.8.2 U-Net

A rede U-Net proposta em (RONNEBERGER; FISCHER; BROX, 2015) para segmentação semântica também usa a estrutura de gargalo para redução de dimensionalidade e extração de características. Essa rede usa o conceito de *skip connections*, que transportam a saída de uma etapa de contração para a etapa de expansão, o que permite recuperar a informação espacial original na reconstrução do sinal de entrada.

A rede U-Net, conforme a Figura 46, possui uma etapa de contração e uma etapa de expansão. No artigo original (RONNEBERGER; FISCHER; BROX, 2015), na etapa

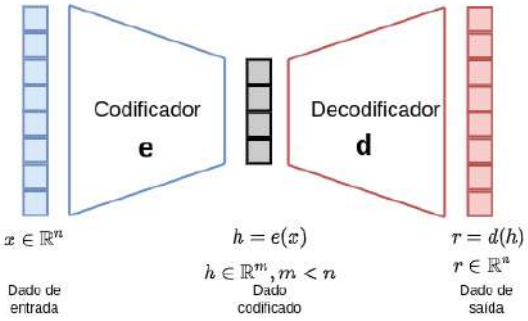


Figura 45 – Estrutura simplificada de um *autoencoder* e suas relações entre a dimensionalidade de entrada/saída e o número de camadas convolucionais. Fonte: De autoria própria.

de contração são aplicadas duas convoluções 3x3 sem preenchimento (*unpadded*) seguidas da função de ativação ReLU e uma operação de *max pooling* 2x2 com *stride* 2, que reduz para a metade a dimensão da imagem no *downsampling*. Para cada passo de *downsampling* o número de filtros de características é dobrado. Na etapa de expansão é realizado a convolução transposta *transposed convolution* do mapa de características de saída de cada nível. No nível imediatamente superior são concatenados a saída da convolução transposta com o mapa de características (*skip connections*) oriundo da etapa de contração pertencente ao mesmo nível. Como o mapa de características da etapa de contração não possui dimensões maiores que o mapa de características da etapa de expansão, em virtude de não usar o preenchimento (*padding*), é necessário recortar (*cropping*) o mesmo para ajustá-lo antes de realizar a concatenação. A esse processo se segue duas convoluções $3\text{x}3$ seguidas por uma camada ReLU. Ao final é usada uma camada com convolução $1\text{x}1$ para mapear o vetor de características com 64 componentes para o número desejado de classes (no artigo original a rede era usada para segmentação semântica).

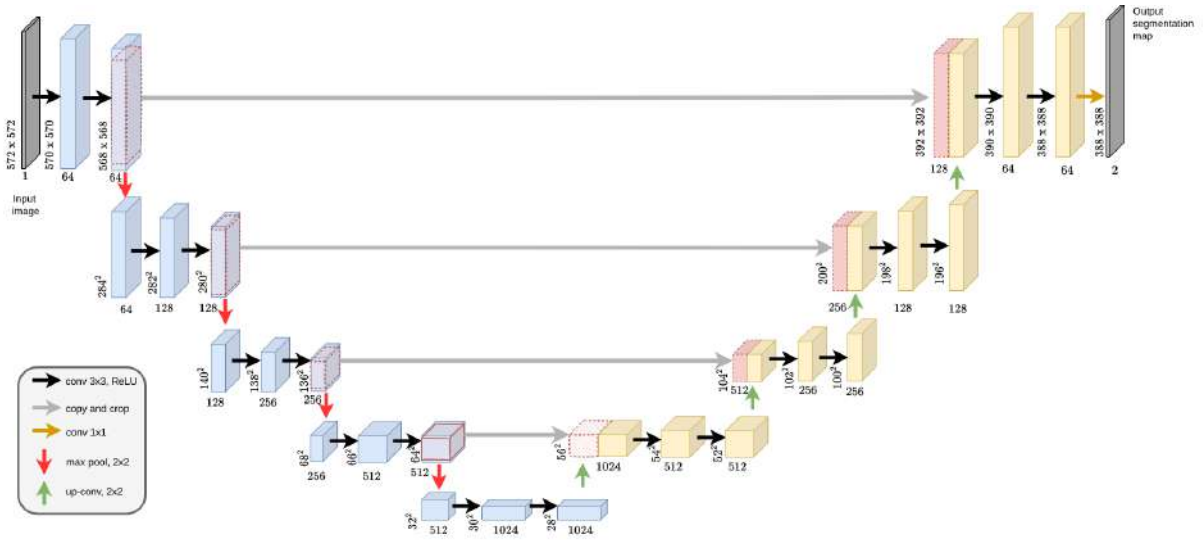


Figura 46 – Estrutura básica de um *U-Net*. Fonte: Adaptado de Ronneberger; Fischer; Brox (2015)

3.8.3 SegNet

A SegNet (BADRINARAYANAN; KENDALL; CIPOLLA, 2017) é uma *deep fully convolutional neural network architecture* para segmentação semântica por pixel. A SegNet se diferencia da U-Net na forma como faz o *upsamples*. Nesse processo, no lugar de usar convolução transposta, o decodificador utiliza índices de agrupamento (*pooling indices*) computados durante o processo de *maxpooling* na etapa de codificação e os utiliza para realizar um *upsampling* não linear, tornando desnecessário que o *decoder* aprenda esses filtros.

Pela Figura 47 observa-se que o decodificador faz *upsample* de sua entrada usando os índices de *poolling* transferidos da etapa de codificação para produzir os chamados mapas de recursos esparsos. Em seguida, ele realiza a convolução com um banco de filtros treinável para densificar o mapa de recursos. Os mapas de recursos de saída do decodificador final passam por uma função de ativação softmax.

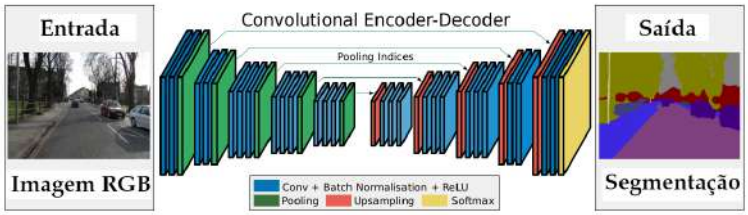


Figura 47 – Estrutura básica da *SegNet*. Fonte: Badrinarayanan; Kendall; Cipolla (2017)

3.8.4 LinkNet

A LinkNet (CHAURASIA; CULURCIELLO, 2017) é também uma arquitetura *u-shaped*, em formato de U, como a U-Net. Sua principal diferença está nas *skip connections*. A Figura 48 mostra um comparativo entre ambas. Enquanto a U-Net concatena a saída de cada bloco do *encoder* a entrada equivalente do bloco de decodificação, na LinkNet a saída de cada bloco de codificação, ao invés de ser concatenada, é somada a entrada do bloco de decodificação.

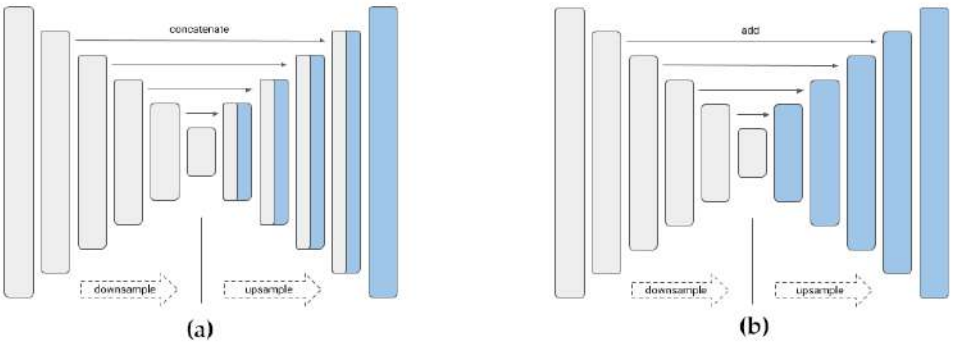


Figura 48 – (a) Arquitetura da U-Net , (b)Arquitetura da LinkNet. Fonte: Iakubovskii (2019).

A Figura 49 apresenta a arquitetura completa da LinkNet (CHAURASIA; CULURCIELLO, 2017). Essa estrutura permite o aprendizado sem aumento significativo no número de parâmetros.

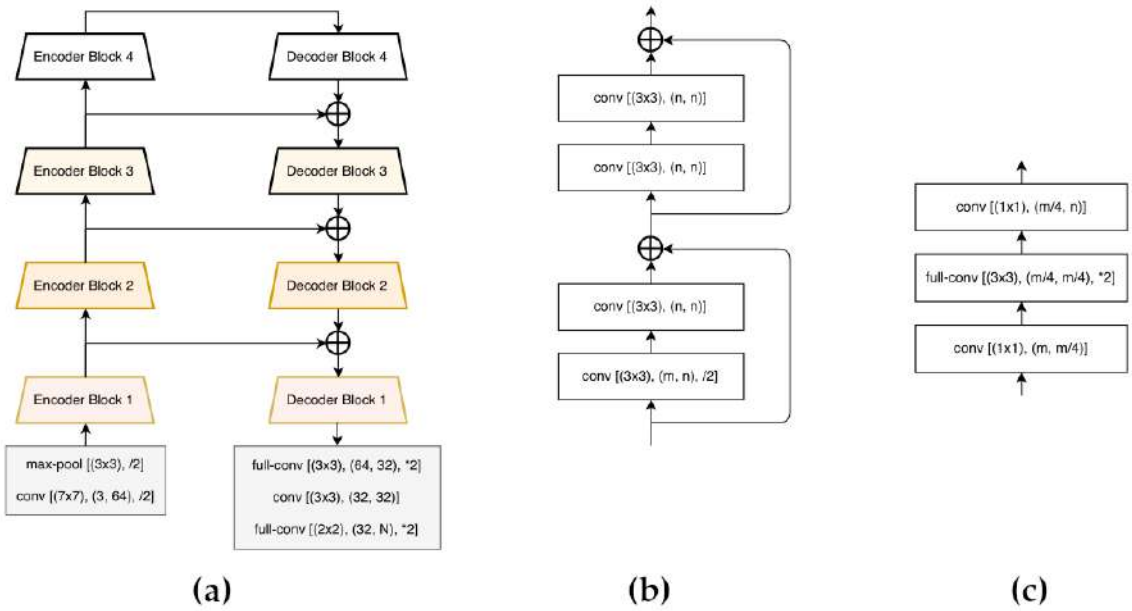


Figura 49 – Estrutura básica da *LinkNet*. (a) Arquitetura da LinkNet, (b) Módulos convolucionais no *encoder-block*, (c) Módulos convolucionais no *decoder-block*. Fonte: Chaurasia; Culurciello (2017).

3.9 Considerações finais

Nesse Capítulo foram apresentadas as definições básicas de redes neurais artificiais. Nas primeiras seções foram introduzidos o funcionamento de uma rede neural tradicional, as principais arquiteturas e problemas relacionados. Na sequência o processo de filtragem espacial usando *kernels* foi discutido brevemente. A partir do conhecimento previamente apresentado foi construída a noção de funcionamento de uma rede neural convolucional e de redes *u-shaped*. Esses conceitos são necessários para o entendimento do trabalho que será detalhado nos próximos capítulos.

4 TRABALHOS RELACIONADOS E APLICAÇÕES COMERCIAIS

De forma a justificar a existência desse trabalho, é importante precisar a teoria e tecnologia *light field* como uma área emergente. Para tal propósito, torna-se importante demonstrar o interesse tanto da indústria quanto da academia no tema. Neste capítulo, primeiramente são relacionados artigos no domínio de concentração da tese, partindo dos trabalhos mais abrangentes até chegar no assunto específico da tese. Por fim, apresenta-se uma sinopse do mercado comercial no setor de dispositivos *light field*.

4.1 Trabalhos relacionados

O problema de correspondência estéreo continua em aberto na visão computacional apesar de soluções eficientes terem sido propostas. Alguns dos maiores desafios enfrentados são: (i) **oclusão total do objeto em uma das vistas**, essa situação impossibilita encontrar a disparidade nessa região oclusa, uma vez que essa informação está ausente; (ii) **tratamento de regiões homogêneas ou com textura repetitiva**, nesse contexto existe problemas em encontrar os pontos homólogos nas vistas, uma vez que as características dos pontos em questão, ou são iguais, ou são cíclicas, criando impasses no cálculo de disparidades usado para a construção dos mapas de profundidade; e, (iii) **distorções ópticas e ruídos intrínsecos ao equipamento de captura**, que também podem gerar impasses e dificuldades na localização de ponto homólogos entre as vistas, por exemplo, as câmeras Lytro sofrem distorções tanto na lente principal quanto nas microlentes, o que pode provocar problemas na etapa de retificação das vistas.

Para o cálculo de distâncias, em sistemas estéreos ou com n-vistas, são usadas duas ou mais imagens com variações na perspectiva, de forma que se possa deduzir as informações de profundidade a partir das disparidades do mesmo ponto espacial nas n-vistas. Uma aplicação com crescente demanda é o uso de visão estéreo em carros autônomos (KEMSARAM; DAS; DUBBELMAN, 2020) e em sistemas robóticos.

Câmeras do tipo LF também possuem aplicações em áreas com demanda de informações espaciais como: robótica (Z.ZHOU; CHEN; O.C.JENKINS, 2019)(Z.ZHOU; SUI; C.JENKINS, 2018), imagens médicas (DE FARIA et al., 2019), carros autônomos (BAJPAYEE; TECHET; SINGH, 2018), etc. Por exemplo, no artigo (Z.ZHOU; SUI; C.JENKINS, 2018) é proposto um método que permite um braço robótico pegar um objeto localizado atrás de um obstáculo translúcido usando imagens plenópticas de uma câmera Lytro. Em (DE FARIA et al., 2019) é proposto um *dataset* contendo 250 imagens LF de lesões cutâneas classificadas em oito categorias clínicas. Esse tipo de *dataset* permite desenvolvimento de novas técnicas e algoritmos em estudos dermatológicos. Outro exemplo é o uso de sistemas de imageamento LF em carros autônomos, como em (BAJPAYEE; TECHET; SINGH, 2018), onde é proposto um método capaz de suportar qualquer matriz de múltiplas câmeras formando uma **SLF**. Esse tipo de sistema de visão pode substituir ou ser usado em conjunto com sensores LIDAR, radar, sonar e etc.

As principais abordagens atuais com deep learning para extração de mapas de profundidade utilizam rede neural convolucional (**CNN**) em diferentes combinações e arquiteturas. As propostas de extração de mapas de profundidade usando aprendizado de máquina podem ser divididas por tipo entrada: uma entrada simples, com apenas um fluxo; ou n-entradas, com vários fluxos tratados inicialmente de forma independente. De uma perspectiva abrangente, ambas as abordagens têm uma unidade dedicada para extração de recursos e uma unidade para construção de mapas de profundidade. A principal diferença é a forma como a entrada é processada.

A estimativa de profundidade com base em imagens LF sucede a tradicional correspondência estéreo binocular e profundidade de imagens monoculares (HAN et al., 2021). A criação de mapas de profundidade a partir de imagens LF avançou de forma significativa, mas ainda existe problemas em **balancear o tempo de processamento com a precisão da estimativa da profundidade**. Como a base do processo é a geometria das LF, essa área se mostra um campo de exploração promissor para a aplicação de algoritmos de aprendizado profundo (HAN et al., 2021).

4.1.1 Cálculo de profundidade e distâncias

Os métodos para estimar a profundidade e as distâncias em uma imagem podem ser divididos em três classes de acordo com sua abordagem (WU et al., 2017): **(i) baseados em correspondência de valores das subaberturas da imagem LF de entrada**; **(ii) baseados em geometria epipolar**; **(iii) baseados em aprendizado de máquina**. O tipo de entrada também influencia na complexidade dos métodos usados. Como esse trabalho utiliza apenas imagens, os tipos de entradas podem ser divididos em duas classes básicas: **par de imagens** (imagem estéreo); e, **imagens com n-vistas**. Na última categoria se encaixam os formatos *light field* já citados (esparsa,

densa, focada, desfocada, etc). A seguir serão detalhados alguns trabalhos relacionados por categoria, com especial ênfase nos métodos baseados em aprendizado de máquina.

4.1.1.1 Baseados em correspondência de valores entre subaberturas

Em uma LF densa, cada par de subabertura possui uma **linha base muito estreita**[1] (WU et al., 2017), fazendo com que a faixa de disparidade entre as imagens de subaberturas também seja muito pequena. Outro problema é que o deslocamento de um ponto espacial entre as projeções nas SAI's pode ser causado por uma interpolação com borramento, e não necessariamente por uma real disparidade. Isso leva a baixo desempenho em câmeras DLF de abordagens baseadas em correspondência, sendo improvável o uso de pares de subaberturas para buscar correspondências estéreo em virtude de a linha base ser estreita (WU et al., 2017). A alternativa é não usar correspondência estéreo, mas de aplicar restrições que favoreçam o uso de toda a informação gerada pelas LF e use todas as subaberturas para estimar um mapa de profundidade inicial (WU et al., 2017).

Em (WANG; EFROS; RAMAMOORTHI, 2015) essa técnica é usada para criar um mapa de profundidade inicial. Os autores propõem um modelo de oclusão para *light field* com base na formação da imagem física. Como existe variação do ponto de vista, alguns blocos da imagem se mantêm consistentes, enquanto regiões com oclusões não se mantêm. Uma vez identificadas essas regiões, pode-se tratar cada caso separadamente. Desta forma, a técnica serve para informar locais de oclusão modeladas explicitamente usando orientação da borda para dividir com uma linha reta o bloco angular em duas regiões iguais. Após identificar as regiões inconsistentes o mapa de profundidade é refinado através de outras técnicas (Figura 50). O método melhorou os resultados de profundidade para situações de oclusão única em linha, mas uma linha reta não é suficiente para lidar com situações de oclusões múltiplas (AI; XIANG; YU, 2019).

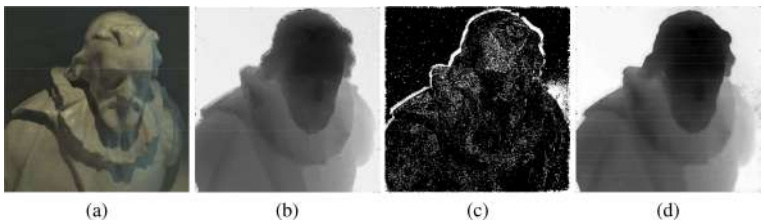


Figura 50 – (a) imagem colorida. (b) Mapa de profundidade inicial. (c) Mapa de detecção para pontos ocluídos em outras visualizações (regiões claras). (d) mapa de profundidade refinado. Fonte: Ai; Xiang; Yu (2019).

Em (YU et al., 2013) é explorada a própria estrutura geométrica das linhas 3D no espaço dos raios luminosos para melhorar a triangulação com *light field* e fazer

[1] Na câmera Lytro é menor que 1 pixel

a correspondência estéreo, conforme a Figura 51. O problema da triangulação visa preencher o espaço de raios de luz de forma contínua e não sobreposta ancoradas em alguns raios de luz usados como referências. No artigo é demostrado que o espaço de *light field* é altamente bilinear, então, visto que a triangulação feita diretamente no espaço bilinear leva a grandes erros, o artigo propõe mapear os subespaços bilineares para linhas delimitadas e aplicar a triangulação restrita de *Delaunay* para encontrar os *pixels* com correspondências nas subaberturas.

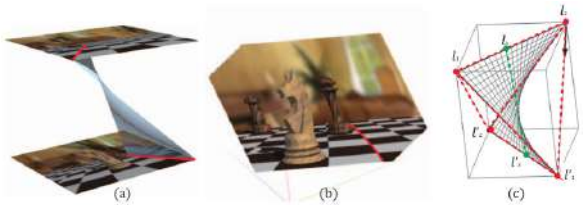


Figura 51 – Estruturas de raios bilineares. (a) Um linha de segmento 3D l mapeia para um subespaço bilinear em uma LF; (b) l mapeia para uma curva em um corte diagonal; (c) O volume é criado através da triangulação usando força bruta. Fonte: Yu et al. (2013).

Em (ZHOU; SUI; JENKINS, 2018) é utilizado o método de monte carlo para achar a localização de um objeto que se encontra atrás de um obstáculo translúcido. Conforme a Figura 52, a câmera Lytro instalada no efetor final de um braço robótico captura sequências de imagens LF. Para cada *light field*, as imagens das subaberturas são extraídas (vista central em vermelho na Figura 52). O volume de probabilidade de profundidade (DLV [2]) é calculado como uma matriz 3D com probabilidades de profundidade ao longo de certos *pixels* (i, j) localizados em uma profundidade d. O DLV é um comparador de semelhança de cor e gradiente entre a vista central e outras imagens de subaberturas. Assumindo uma geometria conhecida e região de interesse, a posição do objeto com 6-DOF (seis graus de liberdade) é estimada através da localização por filtro de partículas ou MCL (do inglês *Monte Carlo Location*) sobre o volume de probabilidade DLV.

4.1.1.2 Baseados em geometria epipolar

A geometria epipolar é usada em imagens estéreo na reconstrução de cenas a partir de duas perspectivas diferentes (WU et al., 2017). Com o advento das câmeras *ligh field* densas a EPI (do inglês *Epipolar-Plane Image*) ou imagem do plano epipolar pôde ser criada diretamente das imagens LF, visto que as inclinações das linhas são indicativas das profundidades dos diferentes objetos (WU et al., 2017). Devido a essa característica das LF, grande parte das técnicas que calculam o mapa de profundidade tendo como estrutura base a EPI acabam usando essas inclinações e propondo variações nesse tipo de abordagem (WU et al., 2017). Em (KIM et al., 2013) é feita a reconstrução de cenas de ambientes complexos e detalhados a partir de EPIs de alta

[2] depth likelihood volume

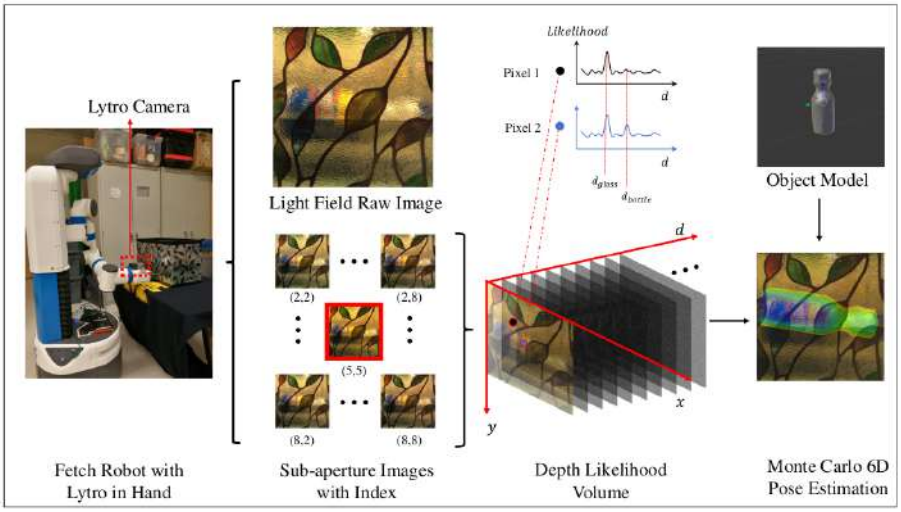


Figura 52 – Pipeline da técnica PMCL. Fonte: Zhou; Sui; Jenkins (2018)

resolução angular, através das informações de profundidade 3D para todos os pontos visíveis da cena. Nesse artigo, é aplicada uma medida de confiança no espaço EPI para calcular a confiabilidade da profundidade estimada (WU et al., 2017).

4.1.1.3 Baseados em aprendizado de máquina

Nos últimos anos houve um aumento no uso de técnicas de aprendizado de máquina na área de LF, justamente pelas vantagens que existem em relação as técnicas anteriores citadas que demandam um alto consumo de computação devido a quantidade de parâmetros que envolvem uma simples imagem LF. As principais abordagens usam redes CNN em variadas combinações e arquiteturas. A seguir são destacados alguns artigos.

Em (ZHOU; CHEN; JENKINS, 2020), os autores apresentam um algoritmo com dois estágios para estimativa de pose de objetos transparentes usando uma câmera LF e renderização fotorrealística. Na Figura 53 pode se observar os dois estágios. O primeiro estágio recebe como entrada imagens LF e fornece como saída a segmentação do material translúcido e o centro estimado do objeto. Os resultados da segmentação são apresentados para uma rede de detecção que rotula o objeto. No segundo estágio, para cada centro estimado, é prevista sua probabilidade através da estimativa de profundidade local calculada a partir do volume de probabilidade de profundidade. Nesse estágio uma otimização de partículas é iniciada com base nas estimativas de rede e profundidade, que convergem para as poses finais 6D (ZHOU; CHEN; JENKINS, 2020).

Muitas propostas atuais envolvem redes siamesas. Uma rede siamesa (do inglês siamese networks) consiste de duas ou mais sub-redes neurais idênticas com os mesmos parâmetros conforme a Figura 54. Nesse tipo de rede são fornecidas duas ou mais imagens e através do acréscimo de uma etapa de integração, os valores de

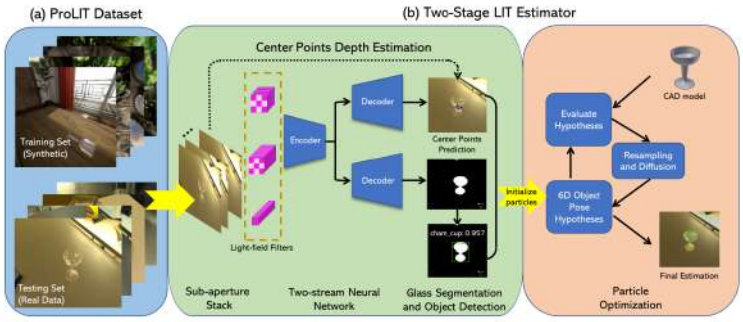


Figura 53 – Sistema LIT - *Light-field Inference of Transparency* (ZHOU; CHEN; JENKINS, 2020). Fonte: Zhou; Chen; Jenkins (2020)

saída das redes são comparados visando verificar a similaridade das imagens de entrada. Desta forma, a saída da rede não é a probabilidade de predição de cada classe como em CNNs tradicionais, mais sim a distância entre as imagens. Essa característica permite apenas calcular o grau de similaridade entre as imagens avaliadas como no caso da rede SigNet para verificação de assinatura (DEY et al., 2017).

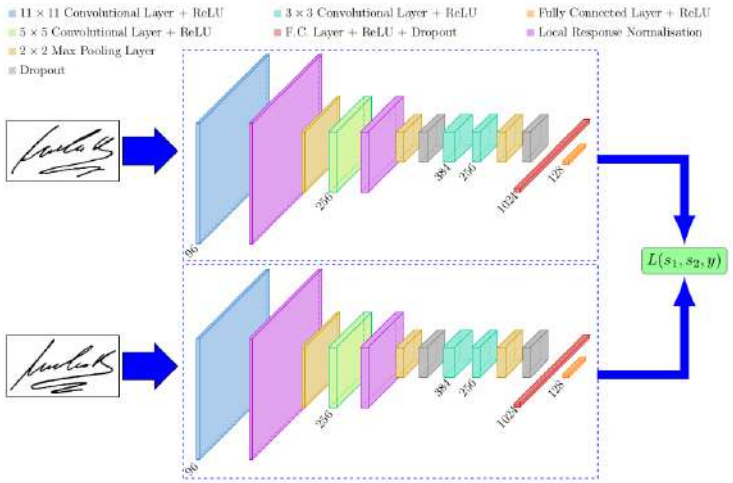


Figura 54 – Rede siamesa SigNet usada para comparação entre duas assinaturas de entrada. Fonte: Dey et al. (2017).

Em (ŽBONTAR; LECUN, 2016) e (LUO; SCHWING; URTASUN, 2016) é proposto o uso de uma rede siamesa para computar o mapa de disparidade entre duas imagens capturadas de forma estéreo. Para evitar apresentar na saída apenas o grau de similaridade entre duas imagens, foi modificada a abordagem original. Conforme a Figura 55, a imagem da esquerda é dividida em K blocos de tamanho nxn. A ideia básica é descobrir o valor da disparidade para cada pixel dos K blocos da imagem da esquerda. Dado que as imagens estão retificadas, o deslocamento é feito horizontalmente entre a imagem direita e a esquerda. Uma vez estabelecidos os K blocos da imagem esquerda, repete-se o processo dividindo a imagem da direita também em K blocos de tamanho nxn ao longo da mesma linha usada para a criação dos blocos da imagem da esquerda, isso faz com que a disparidade máxima de cada pixel seja igual

a K (disparidade $\leq$ K). A imagem direita é apresentada para sua CNN, gerando uma representação que pode ser associada a cada bloco da imagem. O mesmo ocorre para cada bloco da imagem esquerda. Por fim, cada representação produzida a partir da imagem esquerda é passada ao longo do volume gerado da imagem direita (convoluído) criando uma matriz nxn para cada K bloco. Desta forma é produzida uma saída Kxnxn, onde para cada *pixel* da imagem da esquerda é calculado uma pontuação para as disparidades associadas, que é usada para se encontrar o valor de disparidade mais provável. Em (ŽBONTAR; LECUN, 2016) é proposta uma rede semelhante, onde o treinamento da CNN é feita através de pares de pequenos blocos com disparidade conhecida. Na Figura 56 se observa o mapa de disparidade gerado por essa rede.

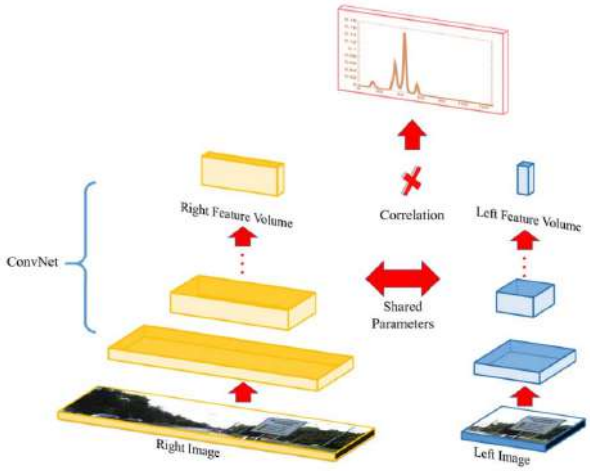


Figura 55 – Rede siamesa para cálculo de disparidade entre duas imagens apresentada em (LUO; SCHWING; URTASUN, 2016). Fonte: Luo; Schwing; Urtasun (2016)

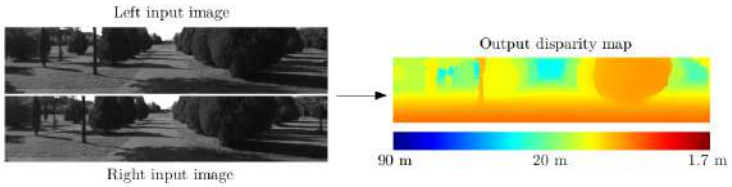


Figura 56 – Mapa de disparidade gerada pela rede CNN proposta em (ŽBONTAR; LECUN, 2016). Fonte: Žbontar; Lecun (2016)

(SHI; JIANG; GUILLEMOT, 2019) propõe um *framework* para estimar a profundidade da cena baseado em aprendizado supervisionado, onde a entrada é um subconjunto flexível de vistas que compõe uma *light field*. Nessa proposta é construído o mapa de disparidade para todas subaberturas da LF. O uso de subconjuntos de aberturas da LF tem como objetivo aumentar a acurácia e limitar a complexidade computacional. As estimativas de disparidade iniciais são computadas entre pares de subaberturas alinhadas usando a arquitetura FlowNet 2.0 (ILG et al., 2017). O

FlowNet 2.0 é basicamente um *optical flow* com aprendizado de máquina profundo, que estima a disparidade em LF densos e esparsos. Essas disparidades são usadas para remodelar um conjunto flexível de subaberturas referenciais gerando um ponto de vista destino. A fusão das estimativas iniciais de disparidade, na forma do vencedor leva tudo, permite ter uma maior acurácia em regiões com oclusão e ao longo dos contornos.

A Figura 57 mostra um exemplo com uma LF 5x5. A imagem alvo L_t (em azul), é a vista para a qual se busca estimar a disparidade. Os retângulos amarelo e vermelho são respectivamente visualizações estéreo horizontais e verticais (L_s). A imagem L_t e as imagens L_s são usadas para calcular os mapas de disparidade preliminares d_i usando o modelo FlowNet 2.0. Vistas âncora ou referenciais L_a (retângulos azuis escuros) podem ser compostas por qualquer subconjunto de subaberturas, à exceção de L_t que são usadas para estimar o erro de distorção (*warping error*) para a fusão das estimativas iniciais. Uma rede de aprendizado residual multiescalar corrige os artefatos presentes na fusão e suaviza o mapa de disparidade final em uma última etapa de refinamento.

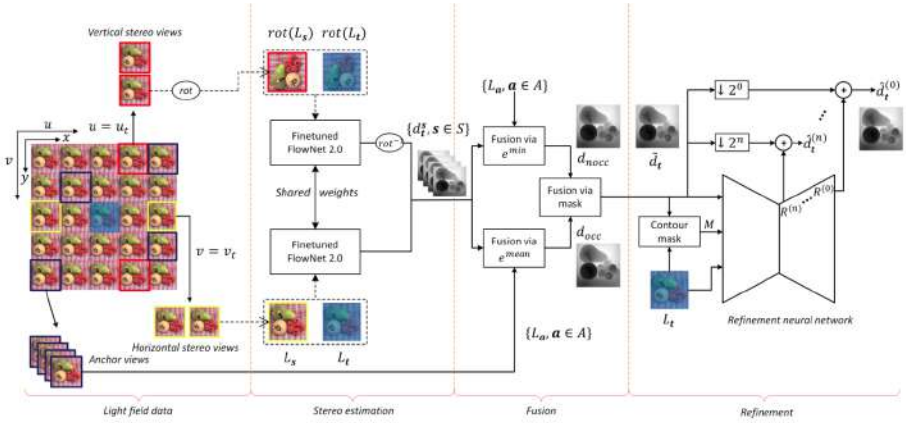


Figura 57 – Visão geral do *framework* proposto sobre a arquitetura FlowNet 2.0. Fonte: Shi; Jiang; Guillemot (2019)

O artigo (GUO; WEN; HAN, 2020) apresenta uma proposta diferente, com foco no reconhecimento e segmentação de objetos baseado em seus materiais/texturas. Na abordagem, conforme se observa na Figura 58, é feito o desacoplamento de informações angulares e espaciais estabelecendo correspondências no domínio angular, sendo depois empregada a regularização para se ter invariância rotacional. A rede recebe como entradas subaberturas espaçadas selecionadas de acordo com os deslocamentos estimados de cada subimagem através da transformada de Fourier.

Em (HEBER; YU; POCK, 2017) é usada uma rede do tipo U-Net para extrair a informação geométrica de uma LF esparsa, conforme a Figura 59. A rede proposta recebe como entrada uma imagem LF capturada na forma esparsa e usa camadas convolucionais 3D que permitem propagar a informação espacial bidimensional junto

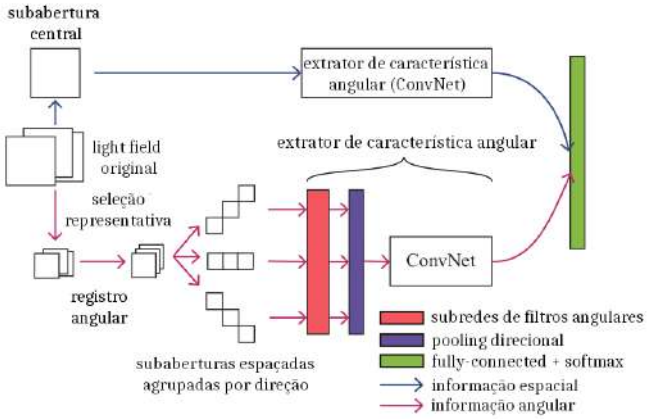


Figura 58 – Framework proposto em (GUO; WEN; HAN, 2020). Fonte: Guo; Wen; Han (2020)

com uma dimensão direcional. Cada nível possui quatro camada convolucionais seguidas por uma camada ReLU. As camadas convolucionais realizam convoluções 3D com núcleos de tamanho $3 \times 3 \times 3$ (HEBER; YU; POCK, 2017).

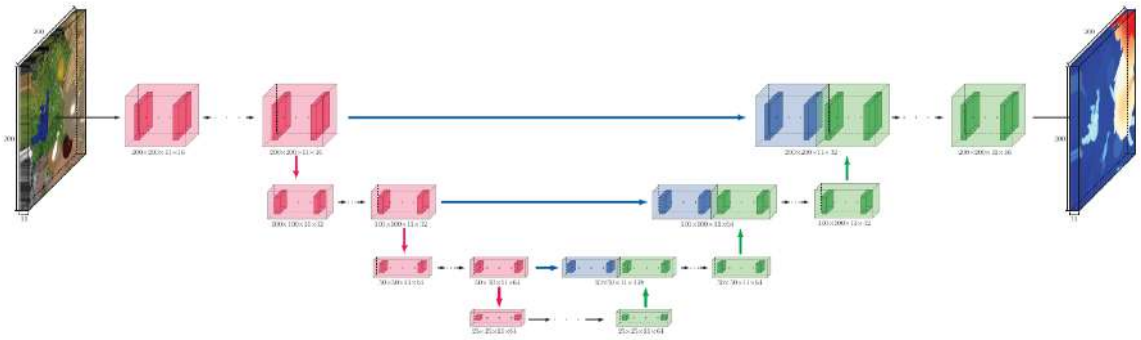


Figura 59 – Rede U-Net proposta em (HEBER; YU; POCK, 2017). Fonte: Heber; Yu; Pock (2017)

Em (SHIN et al., 2018) é apresentada a rede EPINET, que possui uma arquitetura chamada pelos autores de multifluxo que lembra as redes siamesas, pois usa redes neurais em paralelo que compartilham a mesma estrutura, mas que possuem valores diferentes de pesos. Cada conjunto de subaberturas na mesma direção angular representa um "fluxo": horizontal, vertical, diagonal esquerda e diagonal direita. Portanto, cada imagem LF é dividida nesses conjuntos de subaberturas e apresentadas como uma pilha de imagens para sua respectiva rede neural conforme a Figura 60.

Essa separação das subaberturas faz com que as redes neurais produzam filtros representativos restritos ao seu tipo de fluxo (SHIN et al., 2018). Cada rede é composta por três camadas de FCN (do inglês *fully convolutional network*): Conv-ReLU-Conv-BN-ReLU responsaveis por medir a disparidade por pixel em uma região/camilho local. Os autores usam um kernel 2x2 com passo 1 para medir pequenas disparidades (± 4 *pixels*). Isso é necessário devido a linha base ser estreita em imagens LF produzi-

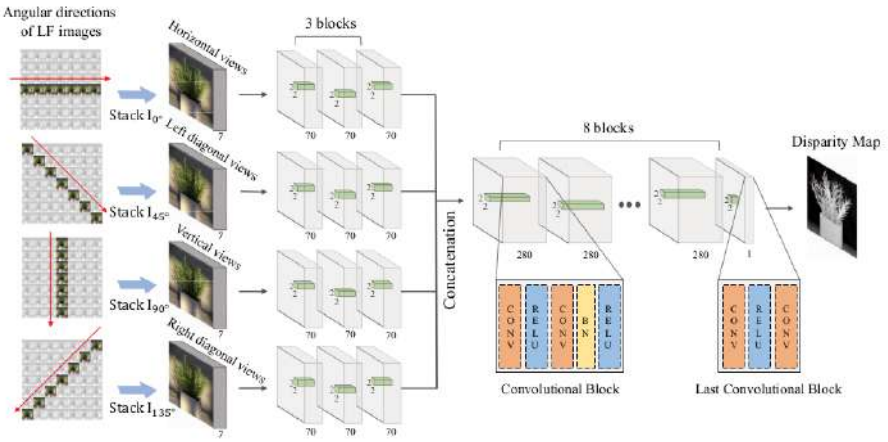


Figura 60 – EPINET. Fonte: Shin et al. (2018)

das por câmeras densas conforme já citado. As saídas das quatro redes de multifluxo são concatenadas e apresentadas como entrada para um rede com oito blocos convolucionais. Os sete primeiros blocos são idênticos aos usados nos blocos multifluxos, apenas o último bloco responsável por inferir os valores de disparidade apresenta uma configuração distinta (Conv-ReLU-Conv).

O *framework* com aprendizado não-supervisionado proposto em (LIN et al., 2022), utiliza uma estratégia mista com aprendizado de máquina e técnicas usuais de processamento de imagem. Essa abordagem utiliza uma técnica de **aprendizado aproximado**, que gera funções de perda diferenciáveis, combinada com restrições geralmente aplicadas em imagens LF para redução de complexidade. Primeiramente é estimado o mapa de profundidade através de uma rede com aprendizado não-supervisionado, e depois é projetada a perda de consistência espaço-angular adaptativa combinada com as versões diferenciáveis das restrições usuais. A Figura 61 mostra a estrutura completa.

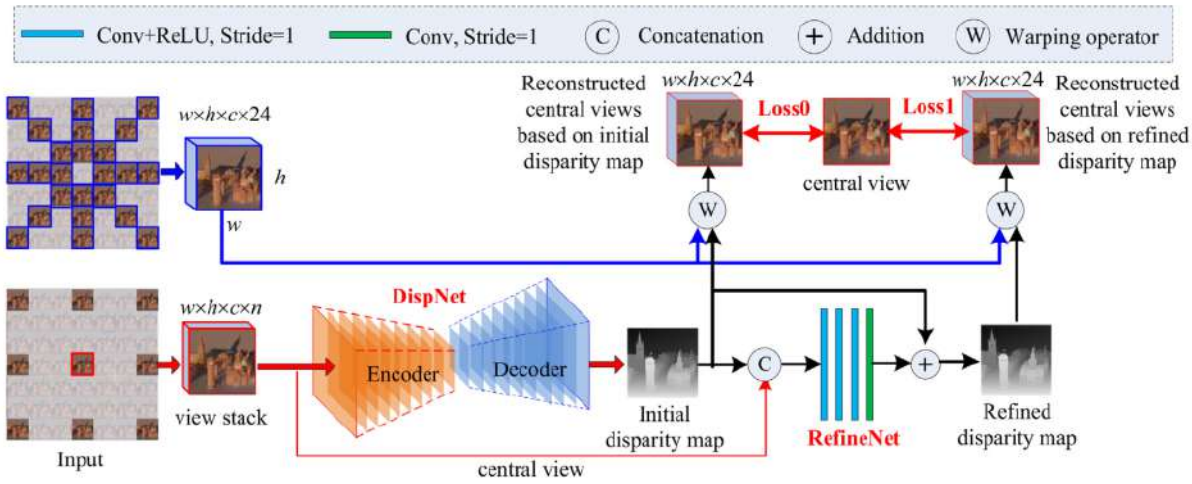


Figura 61 – Arquitetura proposta em (LIN et al., 2022). Fonte: Lin et al. (2022)

Em (KHAN; KIM; TOMPKIN, 2021), os autores estimam o mapa de profundidade

utilizando um conjunto esparso de bordas de profundidade e gradientes. A abordagem deduz que as arestas verdadeiramente profundas são mais sensíveis às restrições locais do que bordas compostas por textura e, portanto, podem ser diferenciadas de forma confiável por meio de um processo de difusão bidirecional. Primeiro o sistema usa o plano epipolar para estimar a disparidade de subpixel em um conjunto esparso de pixels. Para encontrar pontos esparsos de forma eficiente, é feito um refinamento baseado na entropia da estimativa da linha de um conjunto limitado de bancos de filtros orientados. Em seguida, para estimar a direção de difusão a partir dos pontos esparsos, é aplicado o método de difusão bidirecional. Isso resolve o problema de ambiguidade que pode surgir ao tentar encontrar a superfície a qual a borda observada pertence, separando de forma confiável bordas com profundidade de bordas pertencentes a texturas (KHAN; KIM; TOMPKIN, 2021).

Em (LI et al., 2021), é apresentada uma estrutura de aprendizado autosupervisionado para a construção do mapa de profundidade. Esse sistema usa como entrada uma pilha de EPI. A rede estima a mudança de disparidade de EPI por meio da refocalização. Para reduzir a sensibilidade do EPI ao ruído, o artigo propõe um novo modo de entrada chamado EPI-Stack, que empilha EPIs. A Figura 62 apresenta a estrutura geral proposta.

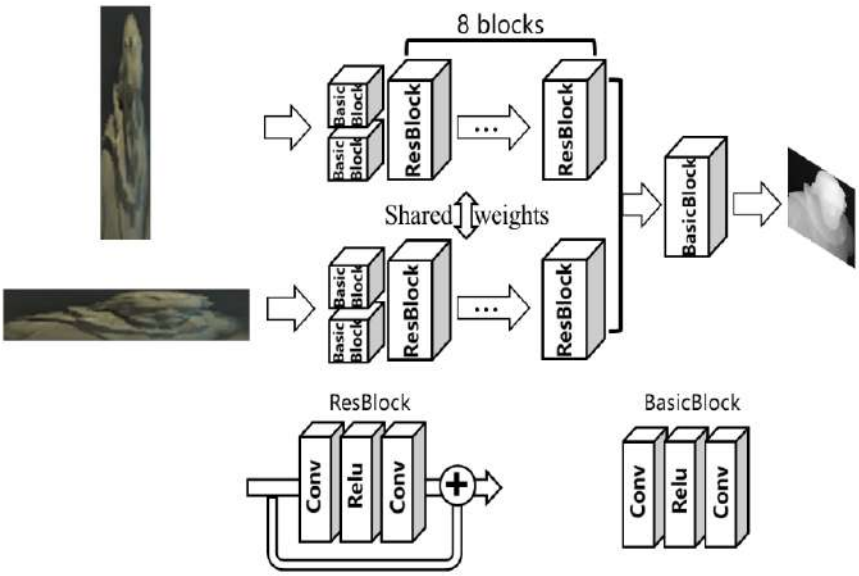


Figura 62 – Estrutura proposta em (LI et al., 2021). Fonte: Li et al. (2021)

4.1.2 Resumo comparativo e Desafios de Pesquisa

Esse trabalho encontrou vários desafios por ser uma área onde ainda estão se criando os padrões a respeito da forma de captura e representação. Por exemplo, tipos diferentes de sistemas podem capturar uma LF se concentrando em aspectos distintos como: resolução espacial, densidade angular e **FoV**, já os dados adquiri-

dos na captura podem ser representados em formatos como: *Lenslet*, 4D *Multiview*, 4D volumétrico, *Geometry-Assisted*, modelo MPI (*Multi-Plane Image*) e modelo MSI (*Multi-Sphere Image*). Essa falta de padronização leva a alguns lapsos de informação como é o caso das especificações técnicas da câmera Lytro Illum. Em diferentes artigos se encontra variações no tamanho máximo do sensor de captura e do tamanho máximo de uma subabertura sem *upscaling*. A compreensão do comportamento óptico de uma DLF também representa um desafio, dado o número de microlentes e distorções apresentadas por seu diminuto tamanho.

Nas técnicas citadas para o cálculo de profundidade e distâncias entre objetos, pela Tabela 4, se nota uma maior concentração de abordagens baseadas em aprendizado de máquina a partir do final da década de 2010. Essa correlação não surpreende, pois ambas as técnicas tiveram seu desenvolvimento justamente durante esse período. Mas o **motivo real que leva o aprendizado profundo ser uma boa estratégia de abordagem para o problema de cálculo de distâncias é a complexidade de se trabalhar com imagens LF somada a incerteza associada as características internas das câmeras DLF**. Todas técnicas citadas, com aprendizado profundo, permitem a criação de mapas de profundidade, variando o tempo de treinamento e precisão da rede neural. O principal problema abordado por essa tese é o **balanceamento entre acurácia e tempo de execução**, pois a maioria dos trabalhos não se preocupa com o tempo de execução.

4.2 Aplicações comerciais, mercado e empresas afins

O mercado de *light field* tem crescido com o surgimento tanto de dispositivos de captura como de dispositivos de apresentação de imagens e vídeos *light field*. Isso se deve a demanda cada vez maior para representar e apresentar informação tridimensional, através de dispositivos robustos e de fácil uso. Dentro desse escopo, o uso de imagens plenópticas tem se apresentado como uma abordagem promissora com soluções comerciais já em uso, conforme será apresentado no decorrer dessa seção.

Pode se apontar o surgimento do mercado de *light field* com a criação dos primeiros dispositivos comerciais entre 2010 e 2011. Em 2010, a empresa **Raytrix GmbH** anunciou a produção e comercialização da primeira câmera **plenóptica 2.0**. Em 2011 foi a vez da empresa **Lytro, Inc.** lançar uma nova câmera, com o diferencial de ser do tipo **plenóptica 1.0**. Enquanto a Lytro investiu no mercado fotográfico e de vídeos, a Raytrix investiu em mercados específicos como o de microscopia, inspeção industrial e pesquisa científica. O ponto de maior convergência entre ambas empresas foi o setor de sistemas imersivos.

A empresa Lytro encerrou suas atividades em março de 2018, após sete anos tentando propor um novo paradigma no centenário mercado fotográfico. Mas apesar

Tabela 4 – Resumo comparativo entre os algoritmos citados. . Fonte: De autoria própria.

Categoria	Abordagem	Característica	Entrada	Saída
Baseado em correspondência	(WANG; EFROS; RAMAMOORTHI, 2015)	Modelo de oclusão para LF	LF 1.0	mapa de disparidade
	(YU et al., 2013)	Estrutura geométrica	Estéreo LF esparsa LF 1.0	mapa de disparidade
	(ZHOU; SUI; JENKINS, 2018)	Estrutura geométrica Monte Carlo Localization	LF 1.0	posição estimada
Baseado em em EPI	(KIM et al., 2013)	Reconstrução da cena	LF 1.0	mapa de disparidade
Baseado em aprendizado	(ŽBONTAR; LECUN, 2016)	CNN (Siamesa)	Estéreo	mapa de disparidade
	(LUO; SCHWING; URTASUN, 2016)	CNN (Siamesa)	Estéreo	mapa de disparidade
	(ZHOU; CHEN; JENKINS, 2020)	CNN com Filtro de partículas	LF 1.0	posição estimada
	(HEBER; YU; POCK, 2017)	CNN (U-Net)	LF esparsa	mapa de disparidade
	(SHIN et al., 2018)	CNN (EPINET) Multifluxo	LF 1.0	mapa de disparidade
	(SHI; JIANG; GUILLEMOT, 2019)	CNN (Autoencoder) FlowNet 2.0	LF 1.0 LF esparsa	mapa de disparidade
	(GUO; WEN; HAN, 2020)	CNN (Multifluxo) Transformada de Fourier	LF 1.0	segmentação por tipo de material
	(LIN et al., 2022)	Aprendizado não-supervisionado Autoencoder	LF 1.0	mapa de disparidade
	(LI et al., 2021)	CNN EPI-Stack	LF 1.0	mapa de disparidade

desse revés, novas empresas estão surgindo e tradicionais empresas também investem em pesquisa na área. Pode-se dizer que um dos principais focos são sistemas com realidade aumentada e realidade mista, hologramas e sistemas imersivos em geral.

4.2.1 Aplicações

Uma aplicação típica do uso de processamento de imagens e visão computacional em plantas industriais é a **inspeção óptica automática**. Tipicamente são dispositivos para inspeção da qualidade de uma superfície, medições e verificação de integridade de estruturas. Entre as principais vantagens do uso de câmeras DLF em sistemas de inspeção está a visualização espacial-**3D** do objeto e o uso de apenas um dispositivo de captura, ao contrário de sistemas de captura estéreo ou baseados em laser. Essas qualidades também são convenientes em dispositivos de imagem usados no cálculo da velocidade de partículas (*particle image velocimetry*). Esses sistemas medem a velocidade de fluídos ou do ar.

Inspeção industrial, microscopia, exames de diagnóstico por imagens são algumas das áreas beneficiadas com o uso de tecnologia **LF**. Para exemplificar pode-se citar a microscopia 3D, que tem tanto aplicações industriais como laboratoriais, onde o uso de sistemas LF permite realizar aferições do tamanho de células, identificar

conexões eletrônicas e detectar falhas em estruturas mecânicas de forma mais precisa e ampla.

O desenvolvimento de **telas** baseadas em *light field* permitirá o uso de 3D em televisores, celulares e tablets sem a necessidade de aparatos adicionais como óculos e outros acessórios. A simples mudança de posicionamento permite ao usuário a sensação de um novo ponto de vista por estar observando um conjunto diferente de raios de luz, desta forma, os *displays* LF podem finalmente ampliar o uso do 3D no cotidiano. Dentro da mesma linha de pesquisa, **óculos de imersão** baseados nessa tecnologia devem ampliar o uso de 3D em plantas industriais e jogos imersivos, ou qualquer sistema que use **realidade virtual** ou **realidade aumentada**.

O uso de câmeras plenópticas em **robótica** e **veículos autônomos** também possui amplo apelo. Uma única câmera DLF pode capturar dados 2D e 3D, calcular a profundidade e medir a distância dos elementos na cena a partir desses dados. Outras áreas de aplicação que podem ser citadas são: modelagem, escaneamento e renderização 3D de objetos, reconstrução de imagens e animação.

4.2.2 Empresas do setor

Apesar da empresa **Lytro**[3] encerrar suas atividades em 2018, várias câmeras ainda estão disponíveis para compra. Uma das grandes vantagens das câmeras Lytro são o seu preço, visto elas terem sido desenvolvidas para ocupar um nicho no mercado fotográfico. A empresa desenvolveu duas câmeras portáteis: A câmera **Lytro®** de primeira geração e a câmera **Lytro Illum®**. A empresa também tentou se inserir no mercado de realidade virtual com a câmera **Lytro Immerge®**, câmera de captura para gravação de volumes com 360 graus de cobertura e 6DoF (seis graus de liberdade). Outro dispositivo da empresa foi a **Lytro Cinema Camera®**, câmera de captura de vídeo e de uso cinematográfico.

A empresa **Raytrix**, pioneira no mercado de câmeras *light field*, continua ativa, desenvolvendo novos produtos e criando parcerias com laboratórios acadêmicos de pesquisa. A empresa possui várias câmeras do tipo plenóptica 2.0, tanto de captura de imagem quanto vídeo, além de softwares e suportes específicos para uso em sistemas de inspeção, microscopia, reconhecimento de faces, robôs cirúrgicos, *scanners* dentais, medições de volume e outras aplicações semelhantes. Em virtude de seu uso específico, as câmeras da Raytrix possuem preços elevados, de ordem superior a €5.000, o que impede o seu uso de forma mais ampla.

Outras empresas possuem investimentos na área de LF. A empresa **Google Inc.**, possui várias iniciativas no setor, como o lançamento no serviço **Steam®** de um aplicativo baseado em imagens LF[4], onde prometia uma experiência em realidade virtual

[3] web.archive.org/web/20110627085842/http://www.lytro.com/

[4] store.steampowered.com/app/771310/Welcome_to_Light_Fields

com reflexos, profundidade e translucidez do mundo real. Para a captura dessas imagens a empresa desenvolveu um sistema de aquisição, processamento e renderização *Light Field* (OVERBECK et al., 2018). A empresa também desenvolveu uma abordagem com aprendizado profundo para sintetizar cenas LF usando várias imagens com multiplanos (FLYNN et al., 2019) e propôs um *array* de câmeras de baixo custo para captura de vídeos panorâmicos de LF (BROXTON et al., 2019). Em maio de 2021, a Google apresentou o projeto *Starline*, um sistema de vídeo conferência onde as pessoas ficam posicionadas em frente a um monitor LF de 65 polegadas e possuem a sensação de volume 3D na imagem projetada (Figura 63) .

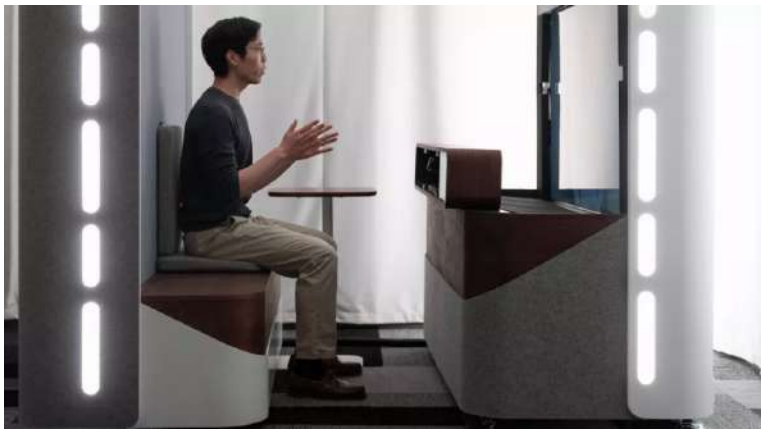


Figura 63 – Protótipo apresentado do projeto *Skyle*. Fonte: Google

Empresas tradicionais com **Apple Inc.** e **LG Corporation** também apresentam interesse no setor. Em 2018, a **LG Corporation** patenteou um telefone celular com 16 câmeras (KIM et al., 2018). Em 2020, foi publicada pelo US Patent & Trademark Office o pedido de patente da **Apple Inc.**[5] de um sistema de câmera panorâmica *light field* para *iDevices* e HMD que criará cenas imersivas com 6 graus de liberdade, através da captura convencional de fotos e da criação, via software, de imagens *light field*.

No desenvolvimento de hardware de captura com custos mais acessíveis pode-se destacar as empresas: **K|Lens**, **Doitplenoptic** e **Wooptix**. A empresa **K|Lens** desenvolveu um conjunto de lentes de captura de *light field* para uso em câmeras DSLR padrão, gerando várias imagens com diferentes exposições em um único disparo. Já a **Doitplenoptic** patenteou um dispositivo chamado **Doit 3D Micro** que permite transformar qualquer microscópio óptico em um microscópio 3D ou 4D. A empresa **Wooptix** possui uma proposta diferenciada em seu dispositivo de captura de *light field*. Em vez de usar um sistema com várias lentes em um *array*, ou um conjunto de lentes, como a proposta da **K|Lens**, ela utiliza uma única lente com foco variável.

Em termos de dispositivos para apresentação de imagens e vídeos *light field*, pode-se destacar as empresas: Avegant, Leia inc e Light Field Lab. A empresa **Light Field**

[5] https://www.patentlyapple.com/patently-apple/2020/04/apple-invents-a-light-field-panorama-camera-system-for-idevices-hmd-that-will-create-immersive-scenes-with-6-degrees-of-fre.html

Lab apresentou em 2021 a tecnologia Solidlight holograms, uma tela que permite visualizar hologramas 3D de forma realista. A empresa **Leia inc** está produzindo o tablet **Lume Pad** que permitem a experiência 3D sem necessidade de óculos. Já a **Avengant** vai no sentido oposto, na construção de dispositivos compactos que possam ser acoplados em óculos e permitir imersão e realidade aumentada.

Na Tabela 5 temos uma amostra do setor e das empresas envolvidas.

Tabela 5 – Empresas que trabalham com tecnologias *Light Fields*. Fonte: De autoria própria.

Empresa	Produtos e protótipos	*Homepage*
Apple	Patente de um sistema panorâmico de captura de imagens	`www.apple.com`
Avegant	Dispositivos para sistemas imersivos	`www.avegant.com/light-field`
Doitplenoptic	Microscopia	`www.doitplenoptic.com`
Google	Câmeras esparsas, monitores e sistemas de realidade aumentada e virtual	`https://augmentedperception.github.io/deepviewvideo/`
K|Lens	Conjunto de lentes para uso em câmeras DSLR padrão	`www.k-lens.de`
Leia inc	Tablet (Lume Pad) com tela 3D *light field*	`www.leiainc.com`
Light Field Lab	Monitores e ecossistemas	`www.lightfieldlab.com`
Raytrix	Câmeras para aplicações profissionais e pesquisa	`www.raytrix.de`
Wooptix	Câmera de captura com uma única lente variável	`www.wooptix.com`

4.2.3 Investimentos e desenvolvimento de padrões

O desenvolvimento recente de novas tecnologias *light field* também foi impulsionado por projetos como o SAUCE[6]. Esse projeto teve como objetivo produzir, testar e demonstrar um conjunto de ferramentas e técnicas profissionais visando reduzir os custos na produção de conteúdo digital em indústrias criativas. Um do principais tópicos abordados foi referente a possibilidades e desafios de integrar captura e processamento de LF em produções de mídia (TROTTNOW et al., 2019). O projeto SAUCE durou três anos (de janeiro de 2018 até dezembro de 2020) e envolveu laboratórios e pesquisadores das seguintes entidades: Universitat Pompeu Fabra, Foundry, DNEG, Brno University of Technology, Filmakademie Baden-Württemberg, Saarland University, Trinity College Dublin, Disney Research.

Outra ação importante para a indústria é a criação do JPEG Pleno[7] pelo comitê JPEG (Joint Photographic Experts Group) que tem uma longa tradição na criação de

[6] `https://www.sauceproject.eu/Technology/Light-Fields`
[7] `https://jpeg.org/jpegpleno/`

padrões de codificação de imagens estáticas. JPEG é um grupo de trabalho conjunto da International Standardization Organization (ISO) e da International Electrotechnical Commission (IEC). O objetivo do núcleo JPEG Pleno é fornecer uma estrutura padrão para representar novas modalidades de imageamento, como *light field*, nuvem de pontos e imagens holográficas. O *framework* JPEG Pleno busca integrar todas as ferramentas necessárias em um único sistema para representar a mesma realidade visual, considerando diferentes modalidades, requisitos e funcionalidades. Em agosto de 2022, grupo do JPEG Pleno organizou o **1st JPEG Pleno Workshop on Learning-Based Light Field Coding Proceedings**.

Desta forma, a área de *light field* possui um potencial que tem sido capturado por várias empresas da área tecnológica, mas por ser um tecnologia comercial nova, ainda depende da criação de padrões, métodos de transmissão e compactação.

4.3 Considerações finais

Esse capítulo apresentou o crescente uso de tecnologias baseadas em *light field*, com a descrição de diversos dispositivos desenvolvidos ou em fase de testes para uso comercial. Foram citados também os esforços na criação de padrões por parte do comitê JPEG (Joint Photographic Experts Group), responsável pela criação de padrões de codificação de imagens. Na seção de trabalhos relacionados foram apresentadas técnicas que constroem os mapas de profundidade que são necessários para o cálculo de distâncias, ao mesmo tempo que se enfatizou a falta de estudos mais consistentes que reúnam informação visual com informações numéricas conhecidas e gerem na saída uma imagem com dados de distância entre os objetos na cena e o observador (câmera) de forma plástica.

5 MATERIAIS E MÉTODOS DE PESQUISA

Conforme citado nos capítulo prévios, essa tese busca apresentar técnicas que melhorem o **balanceamento entre acurácia e tempo de execução** na criação de mapas de profundidades a partir de imagens DLF. De todas as técnicas citadas, as que usam estratégias com aprendizado profundo são as que possuem maior plasticidade, podendo variar o tempo de processamento e a precisão da rede neural de acordo com a arquitetura apresentada. Desta forma, o aprendizado profundo é uma boa estratégia de abordagem para o problema apresentado devido sua flexibilidade em relação a mudança de parâmetros de entrada, o que se apresenta como vantajoso considerando a complexidade de se trabalhar com imagens LF. Com base nessa observação, essa tese usa conhecimentos de aprendizado de máquina, óptica e *light field* para propor modelos de redes neurais profundas capazes de inferir o mapa de profundidade a partir de imagens **LF** densa. Por possuir um escopo amplo, o método de abordagem foi guiado pelos seguintes objetivos gerais:

- Propor e avaliar soluções para extração de mapas de profundidade baseadas em *deep learning*;
- Filtrar estratégias mais apropriadas ao problema proposto;
- Investigar as redes neurais mais propícias para o uso no problema de pesquisa;

Partindo dos objetivos gerais, se estabeleceu os seguintes objetivos específicos:

- Determinar o *dataset* a ser aplicado no treinamento;
- Selecionar as ferramentas mais adequadas;
- Definir o *pipeline* de aprendizado de máquina para o estudo e desenvolvimento das redes neurais;
- Projetar e desenvolver as arquiteturas de redes neurais voltadas a geração dos mapas de profundidade a partir de imagens *ligth field* densa.

5.1 Projeto do Pipeline de aprendizado

Uma pipeline de aprendizado de máquina é uma estrutura que esquematiza o fluxo de trabalho e as etapas envolvidas na construção de modelos de aprendizado de máquina. Essa estrutura representa todos os passos necessários, desde a extração de dados, pré-processamento, treinamento, avaliação e desenvolvimento propriamente dito. Pipelines não são fluxos unidirecionais. Eles são cíclicos por natureza e permitem interações para melhorar as pontuações dos algoritmos de aprendizado de máquina e tornar o modelo escalonável, conforme se observa na Figura 64.

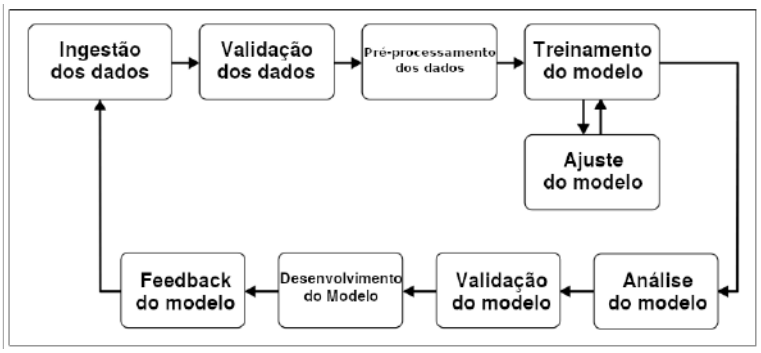


Figura 64 – Pipeline genérico de um sistema de aprendizado de máquina. Fonte: Adaptado de Hapke; Nelson (2020).

O projeto apresentado nessa tese envolve dois ciclos com um total de seis tarefas básicas:

- Definição de dados e *dataset*:
  - Definir o problema e preparar a abordagem a ser seguida;
  - Sumarizar e entender os dados a serem usados no projeto;
- Implementação e testes:
  - Definir e implementar os algoritmos;
  - Processar e avaliar os dados;
  - Avaliar os algoritmos;
  - Melhorar os resultados.

Na primeira etapa do trabalho, **definir o problema e preparar a abordagem a ser seguida**, se estabeleceu o tema e o contexto de aplicação através do levantamento bibliográfico inicial. Com base nas abordagens utilizadas em diversos artigos foi determinado o formato de arquivo a ser usado no *dataset* e na saída do sistema.

O passo para **sumarizar e entender os dados a serem usados no projeto** envolveu três etapas básicas parcialmente independentes: (i) estudo da câmera de captura;

(ii) idealização e construção do *dataset* real; (iii) seleção do *dataset* sintético. Essas etapas são detalhadas na seção 5.2. Por fim, as etapas associadas a **implementação e testes** é abordada no Capítulo 6 que descreve o trabalho propriamente dito.

5.2 Especificação do dataset

A escolha do *dataset* para treinamento é uma etapa básica. Existem *datasets* LF sintéticos e *datasets* LF de imagens reais. Os *datasets* LF de imagens reais são feitos com câmeras comerciais e/ou protótipos. Os principais problemas encontrados ao se trabalhar com os *datasets* LF de imagens reais são: o ruído, a retificação das imagens, problemas nas anotações e descrições das disparidades e principalmente o método usado para extrair o mapa de profundidade que será utilizado como *ground truth* na etapa de treinamento. Por exemplo, o software proprietário da Lytro Illum gera mapas de profundidade e aplica um algoritmo de alongamento de histograma para realçar as diferenças entre as disparidades encontradas. Esse método é interessante quando se trabalha apenas com questões estéticas do tratamento da imagem, mas como ele altera as relações de disparidade no mapa de profundidade, não é útil no processamento de informações de distância da imagem.

A estratégia usada nesse trabalho foi usar *datasets* sintéticos na etapa de treinamento, pois esses dados possuem precisão no *ground truth* e no controle nas características intrínsecas e extrínsecas da câmera. Para potenciais testes do sistema, foi gerado um *dataset* LF de imagens reais próprio com uma câmera LF densa, a Lytro Illum ®, em um ambiente controlado.

5.2.1 Dataset sintético

O dataset sintético utilizado é o proposto em (WANG et al., 2016)[1]. Esse conjunto de imagens foi construído para ser usado com o **4D Light Field Benchmark**, que foi elaborado para avaliar o desempenho de algoritmos na estimativa de profundidade em cenas lambertianas (WANG et al., 2016).

O **4D Light Field Benchmark** apresenta um total de vinte oito cenas, sendo quatro cenas estratificadas, quatro cenas para teste, quatro cenas para treinamento (Figura 65) e dezesseis cenas adicionais (Figura 66). Segundo (WANG et al., 2016), todas as câmeras virtuais estão deslocadas em relação a um plano de foco, mantendo os eixos ópticos paralelos, o que faz a disparidade zero não corresponder a profundidade infinita. Para cada cena são fornecidos:

- Imagens light field com 3 canais de cor e 256 níveis de intensidadede (8 bits) por canal, resolução angular de 9x9 e resolução espacial de 512x512 (9x9x512x512x3), armazenadas como imagens individuais no formato PNG;

[1] `https://lightfield-analysis.uni-konstanz.de/`

- Arquivos de configuração com informações da câmera e da faixa de disparidade da cena;
- Para cada vista central (menos para as quatro cenas de teste):
  - Mapas de profundidade e disparidade com resolução de 512x512 e 5120x5120 no formato PFM;
  - Máscaras de avaliação com resolução de 512x512 e 5120x5120 no formato PNG.

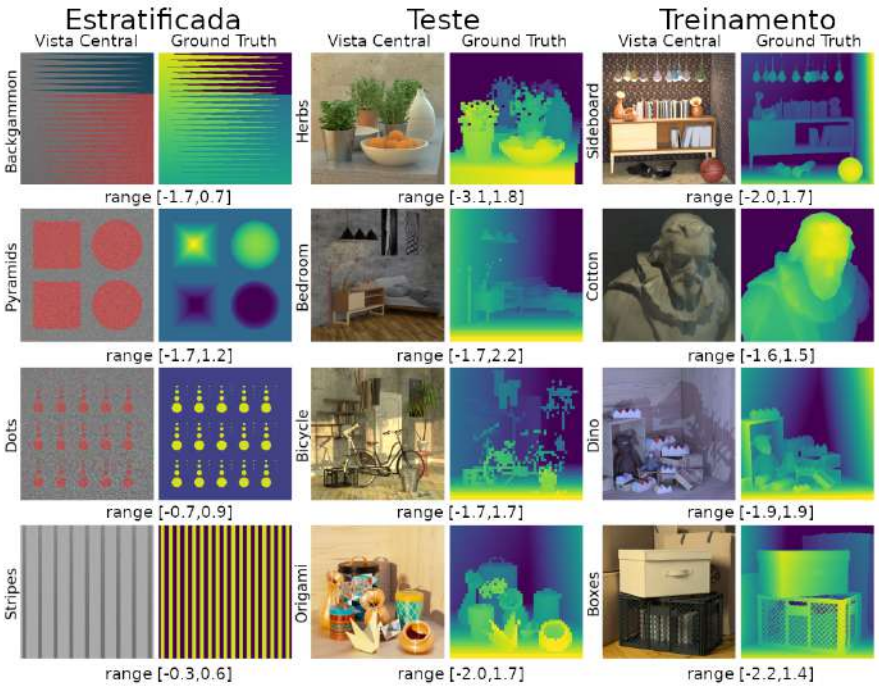


Figura 65 – Dataset sintético do 4D Light Field Benchmark. Fonte: Wang et al. (2016).

Todas cenas foram criadas no software Blender[2]. Nas cenas estratificadas foram usadas o renderizador interno do software, e nas cenas fotorrealistas o renderizador Cycles (WANG et al., 2016).

5.2.2 Dataset de cenas reais

Para esse trabalho foi construído um *dataset* de imagens LF, onde os objetos estão em posições e distâncias conhecidas. O primeiro passo estabeleceu o *dataset* mínimo para a geração de dados. Foi necessário um ambiente controlado, com as configurações da câmera em parâmetros fixos e com marcações visuais das distâncias de cada objeto ao plano de captura. As configurações estabelecidas foram baseadas em princípios ópticos previamente calculados e averiguadas em experimentos de campo.

[2] `https://www.blender.org/`

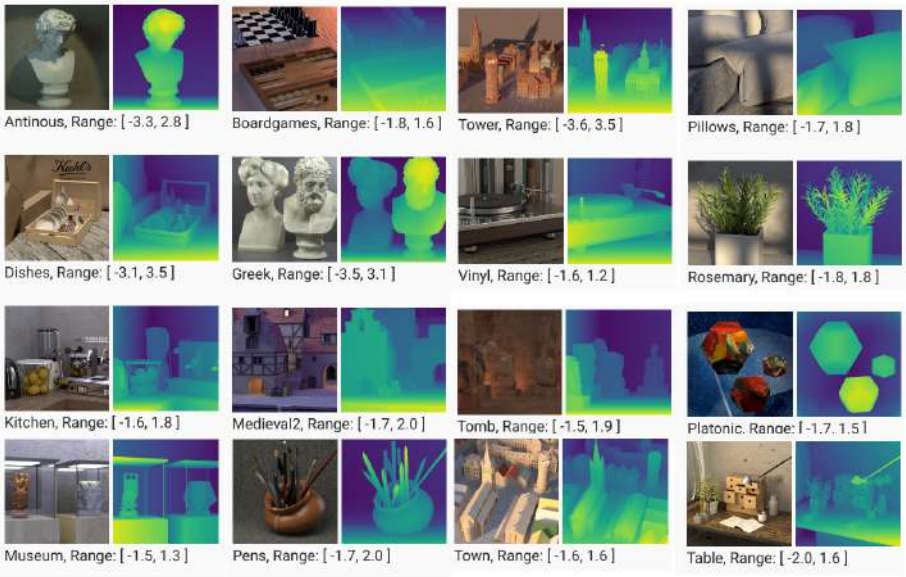


Figura 66 – Dataset sintético adicional do 4D Light Field Benchmark. Fonte: Wang et al. (2016).

Para cada imagem LF é gerado um mapa de profundidade para servir como *ground truth* e os parâmetros da câmera são armazenados em dois arquivos JSON.

5.2.2.1 Câmera Lytro Illum®

A etapa de **estudo da câmera de captura** (câmera Lytro Illum®) foi basilar para o projeto. Uma vez que não existem *datasets* com imagens reais com as características necessárias para o trabalho proposto, foi essencial o completo entendimento dos arquivos gerados pelo equipamento disponibilizado para essa tarefa.

A câmera Lytro Illum® gera, a cada captura, um arquivo LFR (*Light Field Raw*) que contém: a imagem bruta capturada pelo sensor (Figura 30); metadados com a configuração do dispositivo no momento da captura; os números seriais do sensor e da câmera; e, uma miniatura, no formato JPG, da vista central (SILVA, 2016).

Para extrair as informações de configuração da câmera foi usada a ferramenta Lytro Power Tool®, escrita em código aberto na linguagem Python 2.7, que gera um arquivo JSON (Figura 67). Levando em conta a facilidade de manipulação, se optou extrair cada SAI como uma imagem PNG independente. Essa extração pode ser feita através do software comercial da própria câmera (Lytro Desktop®) , das bibliotecas Plenpy (python) [3] e Light Field Toolbox (Matlab)[4], da *toolbox* Lytro Power Tool®, ou do software PlenoptiCam[5].

[3] `https://iiit-public.gitlab.io/plenpy/`
[4] `https://dgd.vision/Tools/LFToolbox/`
[5] `http://www.plenoptic.info/pages/software.html`

```
[
  {
    "picture": {
      "dcfDirectory": "100PHOTO",
      "dcfFile": "IMG_1147",
      "totalFrames": 1,
      "frameIndex": 0
    },
    "generator": "lightning",
    "settings": {
      "flash": {
        "exposureCompensation": 0.0,
        "curtainTriggerSync": "front",
        "zoomMode": "auto",
        "mode": "unknown",
        "afAssistMode": "auto"
      },
      "focus": {
        "roi": [
```


Figura 67 – Fragmento do arquivo JSON gerado pela ferramenta Lytro Power Tool®. Fonte: De autoria própria.

5.2.2.2 Método de construção do dataset

As imagens são adquiridas por uma câmera Lytro Illum com o sensor de captura a uma distância de 600mm do plano onde se encontra os objetos, conforme a figura 68. Observa-se que os objetos são dispostos a cada 100mm, somando um total de 500mm entre o início do plano e o fim. Desta forma, ao adicionar a distância da câmera, os objetos se encontram em uma faixa de 600mm até um total de 1200mm do sensor de captura. Não existe um limite em relação a quantidade de objetos a uma mesma distância do sensor. O critério usado foi manter uma quantidade que permita visualizar todos objetos na cena conforme se vê na Figura 69b, visto que o objetivo do *dataset* é extrair a informação de distância. O plano usado também possui marcações laterais de 100mm conforme se observa na Figura 69a.

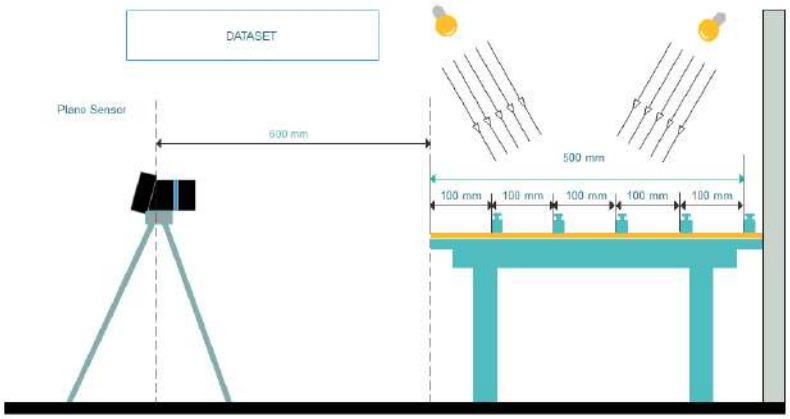
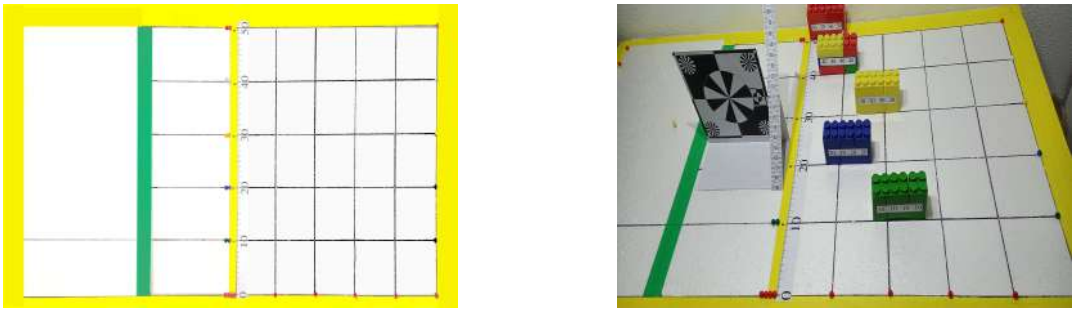


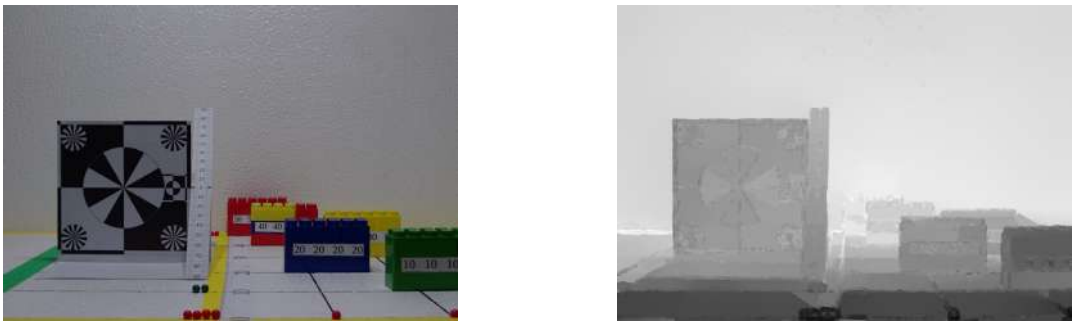
Figura 68 – Esquema usado na captura das imagens. Fonte: De autoria própria.



(a) Plano usado (b) Disposição dos objetos

Figura 69 – Processo de captura. Fonte: De autoria própria.

Na Figura 70 é possível visualizar uma amostra do *dataset* processado pela ferramenta Lytro Power Tools®. A Figura 70a apresenta todos objetos focados. Essa imagem é gerada através da junção das imagens LF capturadas. Já o mapa de profundidade da imagem é mostrado na Figura 70b. Pode-se notar que com as características ópticas escolhidas o mapa gerado possui uma separação visível a cada 100mm. Isso é possível em virtude do comprimento focal ser de 48mm [6], o que faz, para as características da câmera e distância do objeto que está no centro do tabuleiro, haver uma região focável, tanto à frente do objeto quanto atrás, de aproximadamente 10 cm.



(a) Imagem focada (b) Mapa de profundidade

Figura 70 – Exemplo de imagem do *dataset*. Fonte: De autoria própria.

Pode-se fazer uma analogia entre o DoF de uma câmera convencional e a forma como a câmera Lytro captura uma imagem LF. No DoF existe regiões nítidas (focadas) em frente e atrás do ponto focal; na Lytro temos uma **subfaixa refocável em frente e atrás** do ponto focal. A Figura 71 mostra essa organização, onde a faixa refocável abrange todos os pontos que podem assumir foco relativamente nítido depois que a foto é capturada (LYTRO, 2015a). Cada subfaixa refocável oferece um espectro de nitidez relativa, que depende da profundidade em que a imagem é refocada. Na Figura 71, quanto mais brilhante for o tom da faixa azul ou laranja, mais nítidos serão os objetos localizados na distância associada (LYTRO, 2015a). A faixa mais brilhante dentro de cada subfaixa refocável é seu pico - onde os objetos serão mais nítidos na refocalização (LYTRO, 2015a).

[6] 15mm considerando o *crop* de 3.19

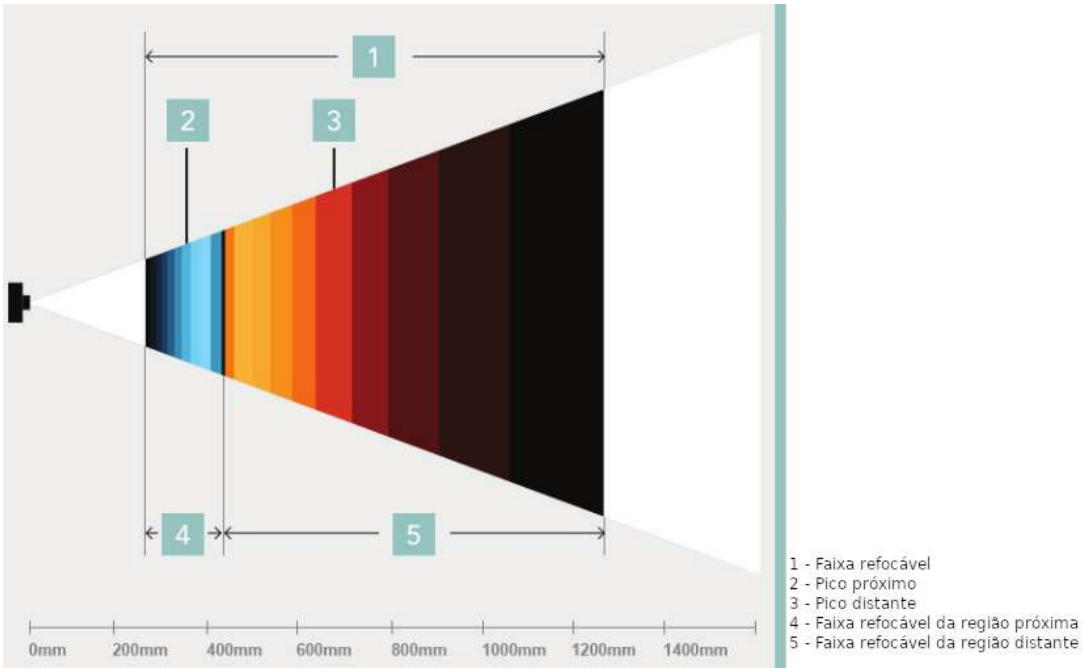
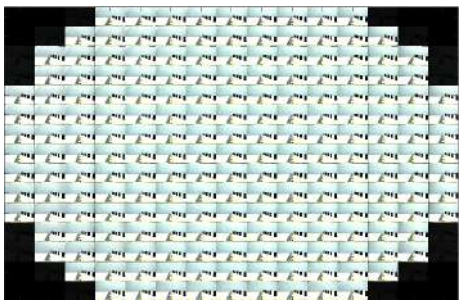


Figura 71 – Faixa refocável com lente ajustada para distância focal de 50 mm (equivalente a 35 mm) e foco óptico em aproximadamente 42cm. As distâncias físicas da câmera são mostradas em cinza. Fonte: Lytro (2015a).

Para garantir imagens com boas características de refocagem, o *focus bracketing* foi ajustado para 5 disparos com 10 *depth steps*. O *focus bracketing* faz com que a câmera tire uma série de fotos cada vez que o obturador é pressionado, incrementando o foco de acordo com os *depth steps*. Isso significa que o intervalo de refocagem é modificado, criando mais chances de capturar uma imagem com maiores possibilidades de refocagem (LYTRO, 2015a). Na Figura 73 é possível ver o efeito desse ajuste. Nesse exemplo foram feitos **três disparos com um *depth step*** (DS). No primeiro disparo a captura é feita com a configuração de foco original (Figura 73a). No segundo disparo é realizado um *depth step* de -1 que comprime a faixa refocável (Figura 73b). Por fim, no terceiro disparo é realizado um *depth step* de $+1$ (em relação ao foco original), que aumenta a faixa refocável (Figura 73c).

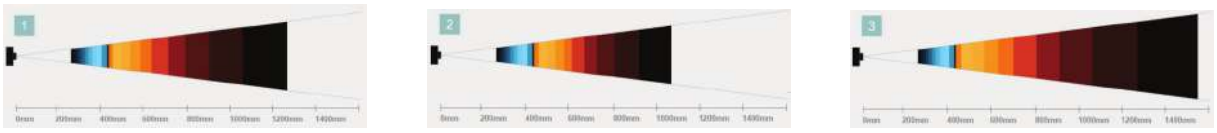
(a) Configuração da câmera



(b) Subaberturas de uma imagem do *dataset*

Figura 72 – Configuração da câmera e SAIs geradas. Fonte: De autoria própria.

Segundo (GEORGIEV et al., 2013), a ferramenta Lytro Desktop consegue aumen-



(a) Faixa refocável original (b) Faixa refocável com -1 DS (c) Faixa refocável com +1 DS

Figura 73 – Faixa refocável com *focus bracketing* definido para 3 fotos e 1 passo de profundidade. Fonte: (LYTRO, 2015a).

tar a resolução de saída usando de forma simultânea as configurações Kleperiana e Galileana (Figura 32). Conforme a Figura 74, a lente principal cria uma imagem em foco em frente das microlentes, e uma imagem virtual focada atrás das microlentes (GEORGIEV et al., 2013), ambas visualizadas pelo sensor. A correção ou mistura apropriada de tais microimagens produz as imagens com resolução mais alta observadas na renderização final.

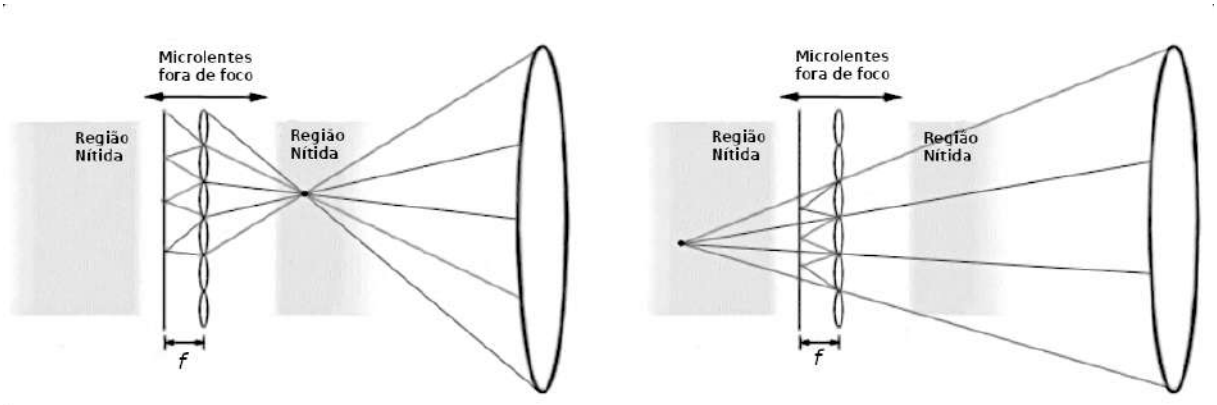


Figura 74 – Captura de imagem Galileana (esquerda) e Kepleriana (direita) em uma câmera plenoptica 1.0. A área sombreada representa a área de boa focagem das microlentes. Fonte: Georgiev et al. (2013).

Como esse método usado para fazer o *upscale* da imagem e a forma de combinação entre as projeções são algoritmos não acessíveis ao usuário das ferramentas Lytro Desktop® e Lytro Power Tool ®, se optou por usar ferramentas que extraiam diretamente as vistas.

Com base nessas considerações foi criado um *dataset* com 81 poses, com dois formatos de saída e duas resoluções para as vistas (SAI), conforme a Tabela 6. Para extrair as SAI's foram usados os softwares **Light Field Toolbox** e **PlenoptiCam**. Já o método usado para gerar o *ground truth depth map* foi o proposto em (JEON et al., 2015), com implementação para MatLab® [7]

A saída com a resolução 1 é gerada pela **Light Field Toolbox** e a resolução 2 pela **PlenoptiCam**. A resolução de 625x434 sem *upscaling* é usada na maioria dos *datasets* consultados. São adicionados ao *dataset* dois arquivos JSON. Um gerado pela ferramente Lytro Power Tools, mantido caso algum pesquisador queira acessar

[7] Disponível em `https://github.com/Vincentqyw/Depth-Estimation-Light-Field/tree/master/LF`

Tabela 6 – Dataset proposto - ver Apêndice B. Fonte: De autoria própria.

Dataset	
Imagens LF	81 poses
SAI	9x9 e 7x7
Resolução 1	625x434
Resolução 2	620x430
Abertura de lente	Constante - f/2.0
All in focus	2022x1494 formato tiff

detalhes padrão da captura, e o outro é uma versão simplificada, com dados ópticos relevantes para o cálculo de distâncias absolutas. Nesse arquivo, estão a informações de ISO, abertura da câmera, distância focal, tempo de exposição, velocidade do obturador. Podem ser adicionados futuramente outros parâmetros, caso se provem necessários.

5.3 Ferramentas

Para implementação do projeto, foi selecionada a linguagem Python (versão 3.x) e o *framework* Tensorflow (versão 2.6.0), que é uma plataforma *end-to-end* e *open source* para aprendizado de máquina. O Tensorflow vem com uma API de alto nível chamada tf.keras que permite criar e treinar modelos de aprendizado profundo. A junção da linguagem Python e o *framework* Tensorflow, permitem a flexibilidade necessária para a prototipagem rápida e integração com outras bibliotecas.

Para essa tomada de decisão foram estudadas as ferramentas descritas abaixo.

- Para uso com Python:
  - Plenpy - `https://gitlab.com/iiit-public/plenpy`.
  - Plenopticam - `http://www.plenoptic.info/pages/software.html`.
  - Lytro Power Tools - `http://lightfield-forum.com/lytro/lytro-archive/`
  - Plenoptic 2.0 Toolbox - `https://github.com/freerafiki/PlenopticToolbox2.0`
- Para uso com MATLAB:
  - Light Field Toolbox for MATLAB - `https://dgd.vision/Tools/LFToolbox/`).
  - FDL toolbox - `https://github.com/LEPENDUM/FDL-Toolbox`.

5.4 Conclusão

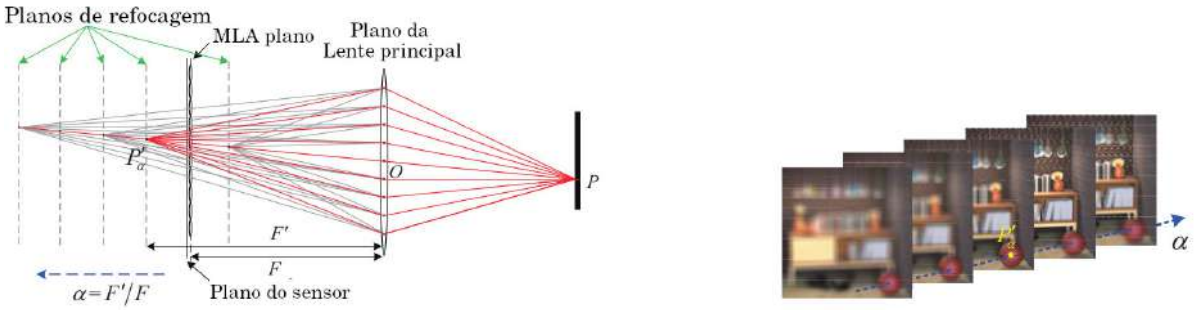
Esse capítulo apresentou a abordagem usada no desenvolvimento do projeto, bem como a seleção de ferramentas. O capítulo sumarizou as estratégias para construção

do dataset real e escolha do dataset sintético. Por fim, apresentou o pipeline utilizado e sua abordagem.

6 TRABALHO DESENVOLVIDO

Existem duas abordagens tradicionais para o cálculo de profundidade: através da **triangulação geométrica** entre vistas diferentes; e através da **variação focal**. A **triangulação geométrica** em *light field* é feita através de uma adaptação do esquema de triangulação convencional estéreo. Para isso é necessário resolver o problema de correspondência entre pontos das vistas, encontrar as disparidades entre as subaberturas e criar o mapa de profundidade. Uma vez feitos esses passos, é necessário calcular as distâncias absolutas através da relação entre as disparidades em *pixels* na imagem e as características intrínsecas da câmera. Na **variação focal** a abordagem empregada utiliza a refocalização. Essa técnica também possui como entrada as imagens de subaberturas LF. Encontra-se a região em foco em cada uma das subaberturas e através do uso das características específicas da câmera, definida pela matriz intrínseca, junto com informações da distância focal e dimensões do sensor, acha-se a distância dos objetos. Na Figura 75, observa-se os vários planos α refocalizados, dados pela equação $\alpha = F'/F$, onde F é a distância entre a lente principal e o *array* de microlentes, e F' é a distância entre a lente principal e o plano α onde a região de interesse se encontra focalizada.

Um ponto crítico nessas abordagens é o **balanceamento entre acurácia e tempo de processamento na criação de mapas de profundidades**. Isso ocorre pela de-



(a) Direção dos raios de luz em relação ao foco (b) Refocagem

Figura 75 – Esquema ilustrativo do refoco em uma LF. Raios vindos do ponto P da cena possuem a mesma radiação e convergem no ponto refocado P'_α. Fonte Zhou et al. (2019)

manda de cálculos envolvidos. Em geral são necessários avaliar n-variáveis, atualizar seus valores e fazer novos cálculos a cada mudança de algum parâmetro. Uma alternativa é usar abordagens com aprendizado profundo, que possuem maior plasticidade, podendo variar o tempo de processamento e a precisão da rede neural de acordo com a arquitetura apresentada. Nos sistemas baseados em aprendizado de máquina e redes neurais convolucionais ocorre uma resposta direta a modificação dinâmica dos parâmetros devido ao processo de aprendizado das relações entre entrada e saída.

O modelo de aprendizado de máquina usado como base nessa proposta é a rede EPINET (SHIN et al., 2018), chamada pelos autores de **arquitetura de multifluxo**. Apesar do artigo original dessa arquitura ser de 2018, ela continua como referência na área. Conforme observa-se na Figura 76, o desempenho dessa arquitetura (**Epinet-fcn9x9**) continua próximo de algoritmos mais novos. Nesse gráfico temos a comparação da rede EPINET com dois algoritmos que estão entre os melhores desempenhos na página de *benchmark* **4D Light Field Dataset**: (i) AttMLFNet (CHEN; ZHANG; LIN, 2021); (ii) CAPNet (LIU et al., 2020). Convém ressaltar que ambos métodos fazem a separação das vistas da imagem LF igual a EPINET.

Figura 76 – Comparação da rede EPINET com outros algoritmos. Fonte: Gerado em **4D Light Field Dataset**, 2022. Disponível em: <https://lightfield-analysis.uni-konstanz.de/>. Acesso em: 24 de novembro de 2022.

A disposição da rede EPINET lembra as redes siamesas, pois usa redes neurais

em paralelo que compartilham a mesma estrutura. Apesar da semelhança, não se pode classificar a EPINET como uma rede siamesa, pois a mesma apresenta valores diferentes de pesos entre as ramificações de entrada.

Esse trabalho fez um estudo da **arquitetura de multifluxo** da EPINET e propõe duas arquiteturas derivadas para extrair as informações de profundidade a partir de imagens plenópticas: a **EPINET-FAST** e a **U-EPINET**. O objetivo é apresentar métodos que possam ser utilizados por pesquisadores que pretendem usar imagens *light field* em aplicações de tempo real ou próximas do tempo real, como é caso da robótica móvel e carros autônomos. Nas próximas seções serão descritos os formatos de entrada usados e as arquiteturas criadas.

6.1 Dados de entrada

Como dados de entrada para treinamento se optou por dados sintéticos, uma vez que os dados reais gerados possuem necessidade de ajustes **manuais** no *ground truth* devido ao ruído e as distorções duplas da câmera Lytro (na lente principal e nas microlentes), além da adequação aos parâmetros da câmera no momento da captura (foco, zoom, DoF, etc.). Como esse ajuste não é o foco principal dessa tese, se optou por usar o *dataset* real apenas nas etapas de testes, e **futuramente** viabilizar o uso do mesmo na etapa de treinamento.

As imagens *light field* sintéticas usadas no treinamento e teste advém do *dataset* **HCI da Heidelberg University**[1] (WANG et al., 2016). Esse dataset fornece imagens *light field* de 8 bits com resoluções de (9×9×512×512×3), ou seja, resolução angular de 9x9 (81 vistas), com resolução espacial de 512×512 e três canais de cor. Estas *light field* estão disponíveis como uma sequência de 81 imagens no formato PNG que devem ser organizadas em uma matriz 9x9 conforme a Figura 78. Além disso, para cada imagem plenóptica são fornecidos os parâmetros da câmera de captura e o mapa de profundidade em formato PFM com resolução de 5120x5120.

Cada vista apresenta variações na localização de um mesmo pixel em relação a vista central, conforme a Figura 77. Essas variações representam a disparidade entre os pixels que pertencem a uma mesma região. A partir de um conjunto de vistas (pilha de vistas) é possível montar um **mapa de disparidade**, que pode ser usado para calcular o **mapa de profundidade** a partir das características intrínsecas da câmera.

Para cada imagem *light field* são criados volumes usando a totalidade das vistas, ou selecionando, de acordo com algum critério prévio, apenas um subconjunto destas. A EPINET usa quatro subconjuntos de 9 vistas, conforme a Figura 78. Cada agrupamento é construído de acordo com quatro critérios:

- Diagonal principal - passa pela vista central (vista 40) com inclinação de $45°$;

[1] `http://hci-lightfield.iwr.uni-heidelberg.de/`

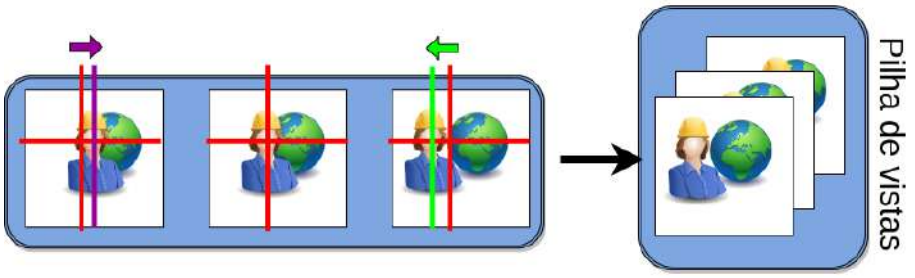


Figura 77 – Exemplo de disparidade entre as vistas. Os eixos vermelhos marcam um pixel na vista central. Na vista mais a esquerda se nota, pela linha vertical roxa, que o mesmo pixel sofreu um deslocamento horizontal para a direita da vista central. Já na vista mais a direita, o mesmo ponto na cena sofreu um deslocamento horizontal para a esquerda em relação a vista central. Essas diferenças em relação a vista central é chamada de disparidade. A junção dessas diferentes perspectivas formam o chamado volume/pilha de vistas. Fonte: De autoria própria.

- Diagonal secundária - passa pela vista central (vista 40) com inclinação de $135°$;
- Eixo de $90°$ - passa pela vista central;
- Eixo de $0°$ - passa pela vista central.

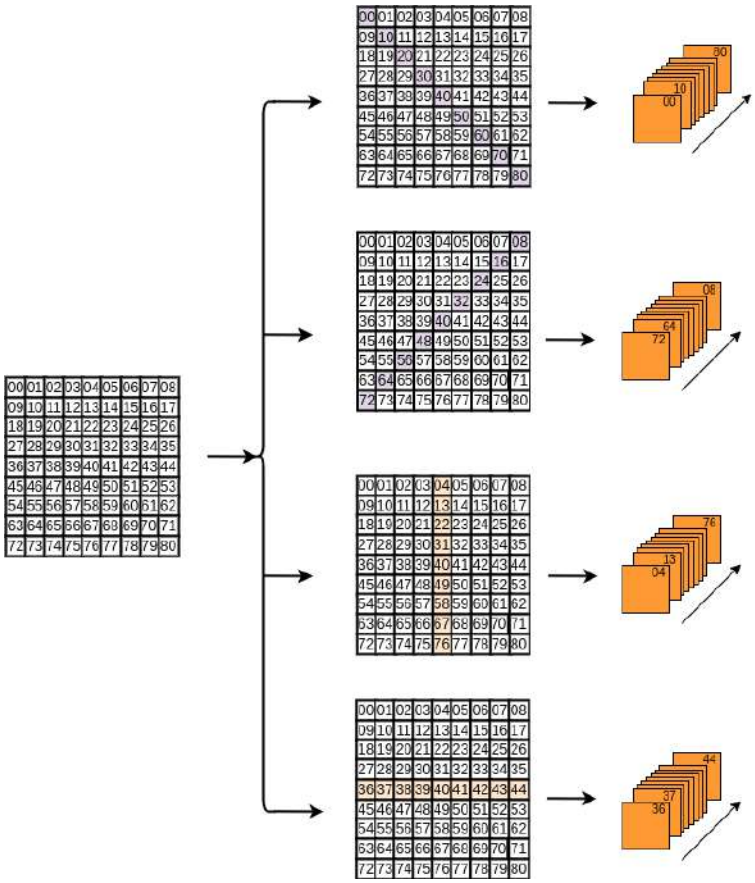


Figura 78 – Geração dos volumes de entrada baseados na disposição das vistas capturadas. Fonte: De autoria própria.

Para gerar esses dados são usadas 16 imagens para treinamento (grupo chamado *additional*) e 8 LF para teste. São criados os quatro volumes de vistas para cada LF de acordo com os critérios citados (Figura 78).

Compete ressaltar que as imagens são convertidas para tons de cinza usando valores ponderados para cada canal, onde: canal R igual a $0,299$; canal G igual a $0,587$; canal B igual a $0,114$.

6.1.1 Data augmentation

Um dos principais problemas de usar imagens *light field* no treinamento de redes neurais é a baixa disponibilidade de imagens com características semelhantes em termos de luminosidade e ao mesmo tempo com alta variabilidade de objetos, materiais e distâncias entre objetos. Outro problema é a qualidade ou precisão dos mapas de disparidade disponíveis nestes *datasets*. Por exemplo, os que são extraídos via softwares como o Lytro Desktop ® não podem ser **diretamente** usados como *ground truth* uma vez que geram mapas de profundidade relativos, onde se aplica alongamento de histograma e outras transformações, fazendo com que se perca as informações necessárias para a construção de um mapa de profundidade absoluto. Para resolver esse problema é necessário fazer um remapeamento entre o *ground truth* gerado e o *ground truth* real [2]. Por sua vez, extrair geometricamente o *ground truth* de imagens reais pode gerar ruído em virtude de características da câmera e do sistema óptico. Desta forma, acaba-se ficando limitado a um pequeno conjunto de imagens LF sintéticas que conseguem atender as demandas necessárias. Mas quando se trabalha com redes neurais convolucionais é necessário usar uma grande quantidade de dados para que a rede neural consiga generalizar. Essa limitação de imagens torna necessário aumentar a quantidade de dados mantendo as relações geométricas entre as imagens de subaberturas (vistas) e ao mesmo tempo contornar o problema de *overfitting* que pode acontecer se apenas apresentarmos o mesmo conjunto de imagens de forma reiterada.

A estratégia para aumentar a quantidade de dados (*data augmentation* em inglês) neste trabalho foi retirada de (SHIN et al., 2018), que consiste nas técnicas de :

- Deslocar a vista central através de translação;
- Rotacionar as imagens LF em $90°$, $180°$ e $270°$;
- Reescalar o tamanho das imagens;
- Espelhar a imagem - *flipping*.

[2] Isso só é possível se as informações de distâncias estiverem disponíveis, como no caso do *dataset* gerado para essa tese.

6.1.2 Deslocar a vista central

Cada volume construído captura variações epipolares em uma das quatro direções. Conforme se observa na Figura 80, cada um desses volumes é associado a uma entrada distinta. Uma estratégia simples para aumentar a quantidade de dados é mudar a subabertura usada como referencial e construir novos volumes baseados nessa nova vista central conforme se observa na Figura 79.

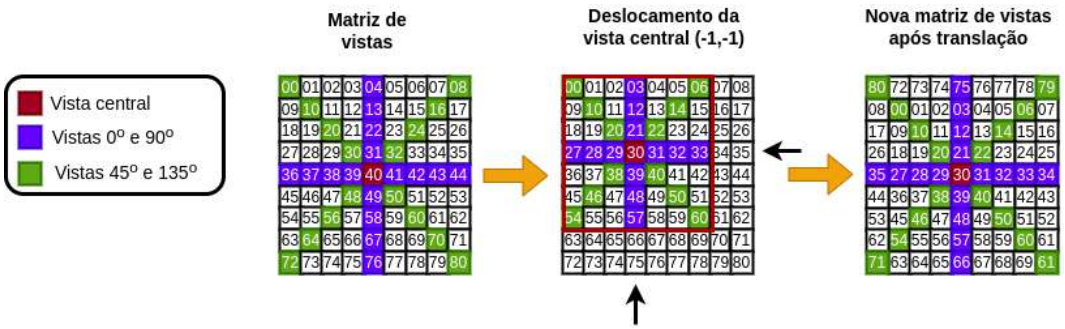


Figura 79 – Deslocamento da vista central em (-1,-1). Fonte: De autoria própria.

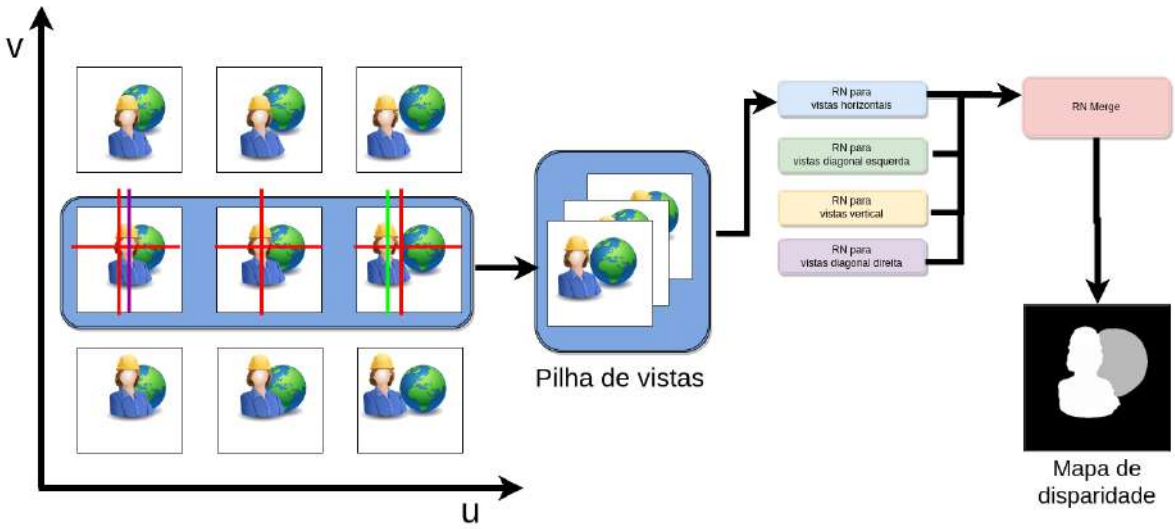


Figura 80 – Cada volume construído é associado a uma entrada. Fonte: De autoria própria.

6.1.3 Rotação das imagens LF

Essa técnica é largamente utilizada para o aumento de dados de entrada. Imagens LF possuem uma particularidade - precisam manter as relações de disparidade. De acordo com (SHIN et al., 2018), pixels na direção horizontal estão fortemente relacionados entre si nos volumes de vistas horizontais. Ao rotacionar a imagem, de forma a manter a informação epipolar, esses pixels passam a representar variações no sentido de variação vertical, desta forma, a rotação do volume muda a entrada associada ao volume criado. Na Figura 80, o volume é associado a entrada horizontal, após a rotação esse conjunto de vistas passa a ser associado a entrada vertical (Figura 81).

A mesma lógica é aplicada as demais entradas e volumes.

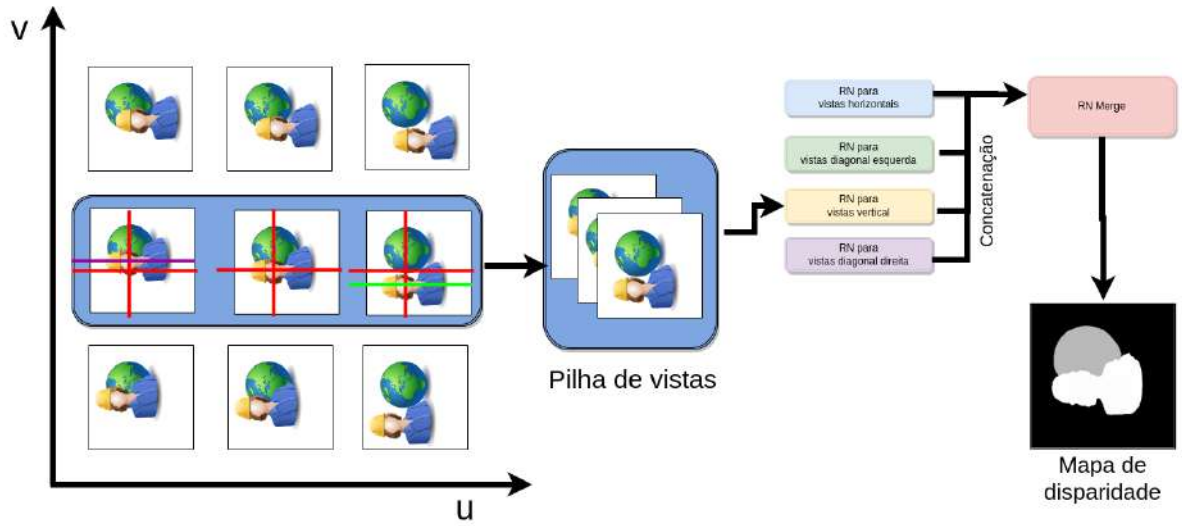


Figura 81 – Mudança da entrada associada ao conjunto de imagens após rotação. Fonte: De autoria própria.

6.1.4 Redimensionar o tamanho das imagens

Ao se redimensionar as imagens os valores de diparidades precisam ser ajustados de forma apropriada. A abordagem novamente é a usada em (SHIN et al., 2018), onde, tanto as dimensões da imagem, quanto os valores de disparidade são ajustados multiplicados por um fator de $1/N$, onde N assume os valores de 1, 2, 3 ou 4.

6.1.5 Espelhamento da imagem - *flipping*

O espelhamento não apresenta os mesmos problemas associados a rotação. O único cuidado a ser observado no espelhamento é a inversão do sinal de disparidade.

6.2 Modelos propostos

Nos últimos anos houve um aumento no uso de técnicas de *deep learning* na área de LF, justamente pelas vantagens que existem em relação as técnicas geométricas tradicionais que demandam um alto consumo de computação devido a quantidade de dados que envolvem uma simples imagem light field. As principais abordagens atuais usam redes neurais convolucionais com variadas combinações e arquiteturas.

Nessa seção são apresentados os modelos propostos. O trabalho é constituído por duas abordagens: (i) modelos derivados diretamente da abordagem EPINET - chamados de EPINET-FAST; (ii) modelos multifluxo com redes em **formato u** (*u-shaped neural networks*) - chamados U-EPINET.

6.2.1 EPINET-FAST

A arquitetura **EPINET-FAST** é uma versão simplificada da rede EPINET (SHIN et al., 2018), buscando maior velocidade de processamento na etapa de extração de mapas de profundidade. Sua abordagem básica é composta de duas redes idênticas com três blocos FCN (*fully convolutional networks*). Cada bloco possui três sequências iguais de camadas *fully convolutional* com a seguinte sequência: Conv-ReLU-Conv-BN-ReLU. Foi usado um *kernel* 2x2 com *stride* 1. O *kernel* e o *stride* apresentam essas dimensões reduzidas para medir as pequenas disparidades apresentadas. Isso é necessário devido a linha base ser estreita em imagens LF produzidas por câmeras densas (±4 pixels), conforme já citado. As saídas das duas redes de multifluxo são concatenadas, criando um novo volume de dados com o dobro do tamanho de cada saída individual, e apresentadas como entrada para uma rede com oito blocos convolucionais. Os sete primeiros blocos são idênticos aos usados nos blocos multifluxos, apenas o último bloco, responsável por inferir os valores de disparidade, apresenta uma configuração distinta (Conv-ReLU-Conv). A Figura 82 apresenta a rede EPINET-FAST.

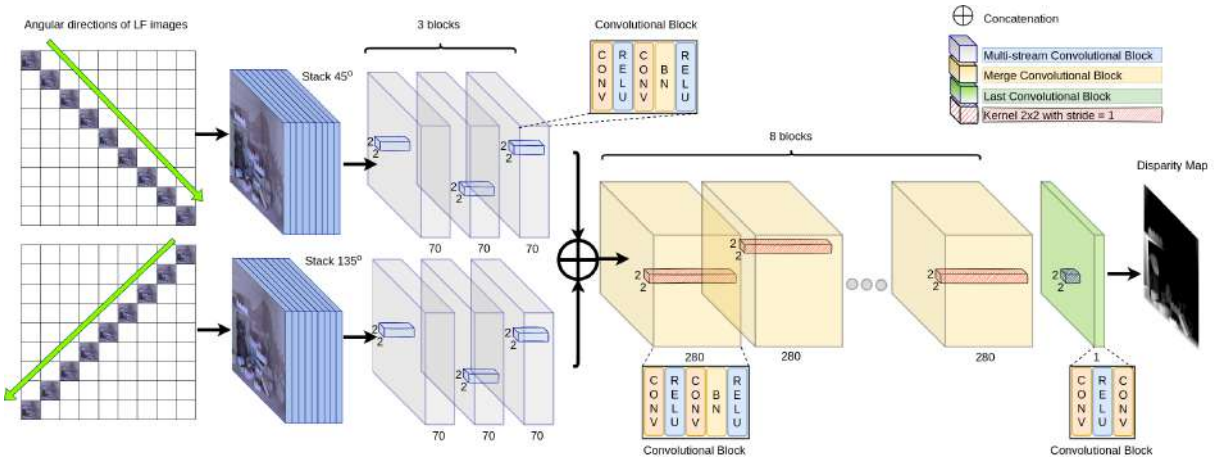


Figura 82 – EPINET-FAST. Fonte: De autoria própria.

Para cada light field são criados dois volumes usando apenas as vistas que se encontram na diagonal principal e secundária, ou seja, subaberturas que variam em um eixo de inclinação de $45°$ e de $135°$ criando dois volumes distintos por diagonal conforme mostrado na Figura 83. Essa é uma simplificação da estrutura usada na EPINET (SHIN et al., 2018), que usa além das diagonais o eixo horizontal e o eixo vertical tendo como origem a vista central. A estratégia para aumentar a quantidade de dados consiste em deslocar a vista central e pegar novas diagonais, usar rotação de imagem em todas vistas, fazer mudança de escala e *flipping*.

Para a geração do mapa de disparidade foram construídas a EPINET original (EPINET_TF_20) e três variações da EPINET-FAST básica (Figura 82). Todas foram implementadas no Tensorflow 2.4 usando a API funcional com Python 3.8. As variações

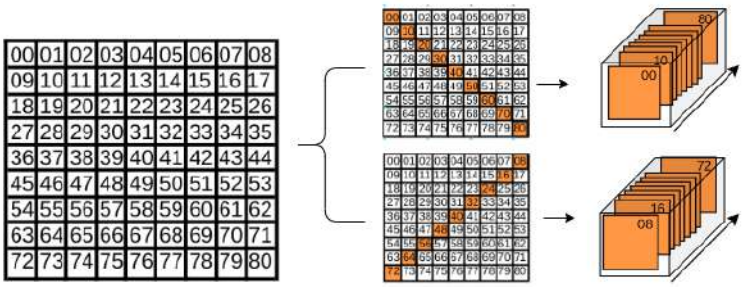


Figura 83 – Geração do volume de entrada baseado na disposição das vistas capturadas. Fonte: De autoria própria.

Tabela 7 – Efeito do número de pontos de vista no desempenho (SHIN et al., 2018).

	1- stream	2- stream	4- stream
Input views			
MSE	2.165	1.729	1.393
Bad pixel ratio (<0.07px)	7.61	5,94	3,87

da EPINET-FAST básica são as seguintes:

- EPINET-FAST-D Possui a estrutura apresentada na Figura 82.
- EPINET-FAST_0_90 - Possui a estrutura apresentada na Figura 82, mas recebe com entrada as subaberturas localizadas a $0°$ e $90°$ a partir da vista central.
- EPINET-FAST_45_135 - Possui a estrutura apresentada na Figura 82, mas com modificação na quantidade de filtros aplicados nos oito blocos finais. Nessa implementação são usados 210 filtros.
- EPINET-FAST_45_135_F - Semelhante a anterior. A única diferença é o fato da imagem de saída passar por uma abertura e um fechamento morfológico para tentar eliminar potenciais distorções. O elemento estruturante de ambas operações é em forma de cruz com raio igual a 1.

Convém ressaltar que a estrutura semelhante a EPINET-FAST_0_90 já havia sido explorada no artigo original ((SHIN et al., 2018)) e que já havia sido detectado a ineficiência de se usar apenas dois volumes com entrada as subaberturas localizadas a $0°$ e $90°$ a partir da vista central. A Tabela 7 (SHIN et al., 2018) apresenta os resultados encontrados pelos autores.

6.2.1.1 EPINET-FAST com alteração no backbone

Uma das formas de melhorar os indicadores de desempenho tanto da EPINET quanto da EPINET-FAST, em termos de qualidade do mapa de disparidade gerado, é aumentando o número de blocos convolucionais para a extração da informação de profundidade. Essa estratégia pode implicar em aumento no tempo de processamento e gerar o problema de **desaparecimento de gradiente** (*vanishing gradient*).

Em relação ao desaparecimento de gradiente, uma forma usual de evitá-lo é usando como *backbone* (estrutura base) a rede **ResNet** (do inglês *Residual Networks*) no modelo de arquitetura proposto. A ResNet usa o conceito de conexões residuais (*shotcuts*), que existem dentro dos chamados módulos e/ou blocos residuais. Um módulo/bloco residual representa uma sequência de convoluções, operações de normalização e ativações ReLU que culminam com uma conexão residual (KROHN; BEYLEVELD; BASSENS, 2020). Exemplos de blocos básicos de uma ResNet são apresentados na Figura 84.

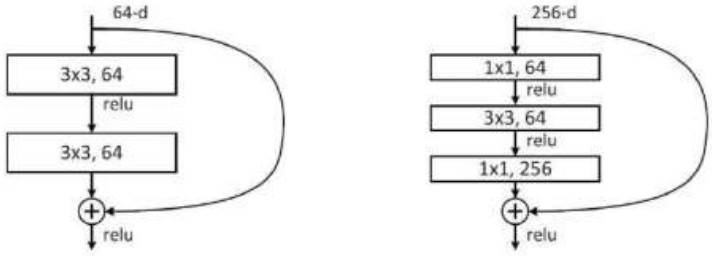


Figura 84 – Blocos de construção da ResNet. A esquerda o bloco de construção original proposto em (HE et al., 2016). A direita a variante mais comum que usa um gargalo (*bottleneck*) para reduzir o número de canais antes da convolução. As conexões diretas (*shotcuts*) permitem evitar o desaparecimento de gradiente durante o treinamento. Fonte: Szeliski (2022).

A ResNet impede a perda de gradiente através do uso dos *shotcuts* entre os blocos. Os *shotcuts* somam a entrada de cada conjunto de blocos convolucionais com sua saída. Isso proporciona a recuperação da informação de gradiente, permitindo redes mais profundas.

Para a proposta foram criados blocos residuais ajustados aos problemas de restrições impostas pela base estreita entre SAI's e para gerar dados com bom desempenho na geração dos mapas de profundidade, com tempos de processamento próximos aos encontrados na EPINET-FAST original.

As variações com *backbone* ResNet da EPINET-FAST são as seguintes:

- EPINET-RES_1_3_2_ReLU - EPINET com bloco residual da Figura 85c e estrutura apresentada na Figura 86. Essa abordagem foi criada para fins de comparação entre os testes.
- FAST-RES_1_3_2_ReLU - EPINET-FAST com bloco residual da Figura 85c e estrutura apresentada na Figura 87.

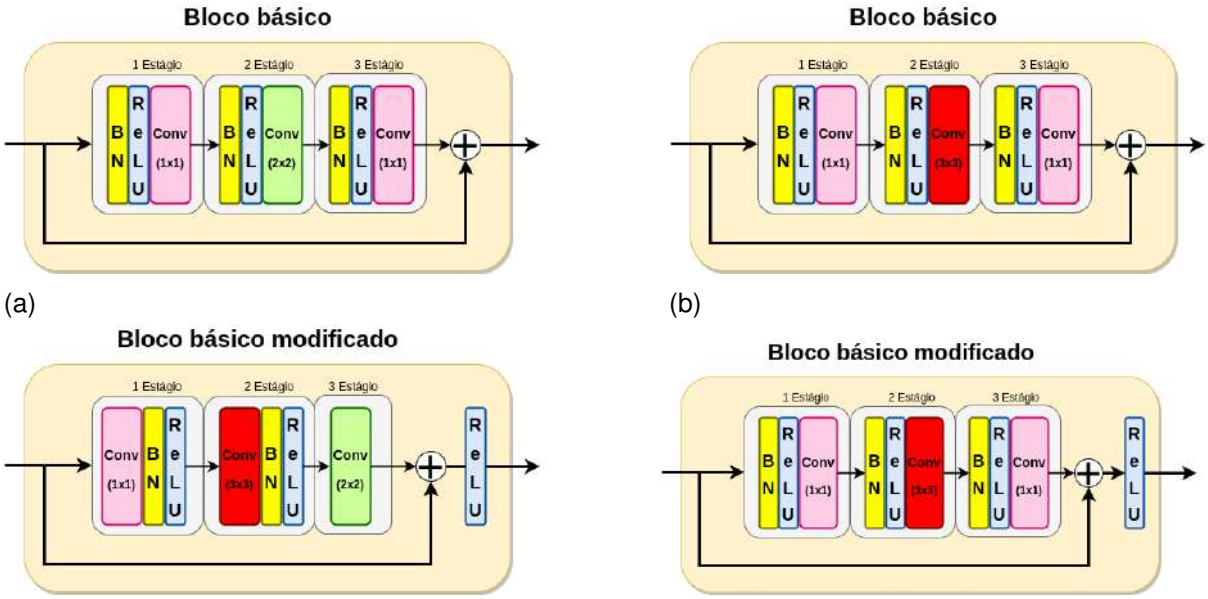


Figura 85 – Blocos residuais básicos usados no trabalho. Na Figura (a) o segundo estágio apresenta convolução com *kernel* 2x2. Na Figura (b) o segundo estágio apresenta convolução com *kernel* 3x3. A Figura (c) apresenta o bloco básico modificado, e a Figura (d) apresenta o mesmo bloco de (b) adicionado de uma saída ReLU. Fonte: De autoria própria.

- FAST-RES_1_3_1_ReLU - EPINET-FAST com bloco residual da Figura 85d e estrutura apresentada na Figura 87.

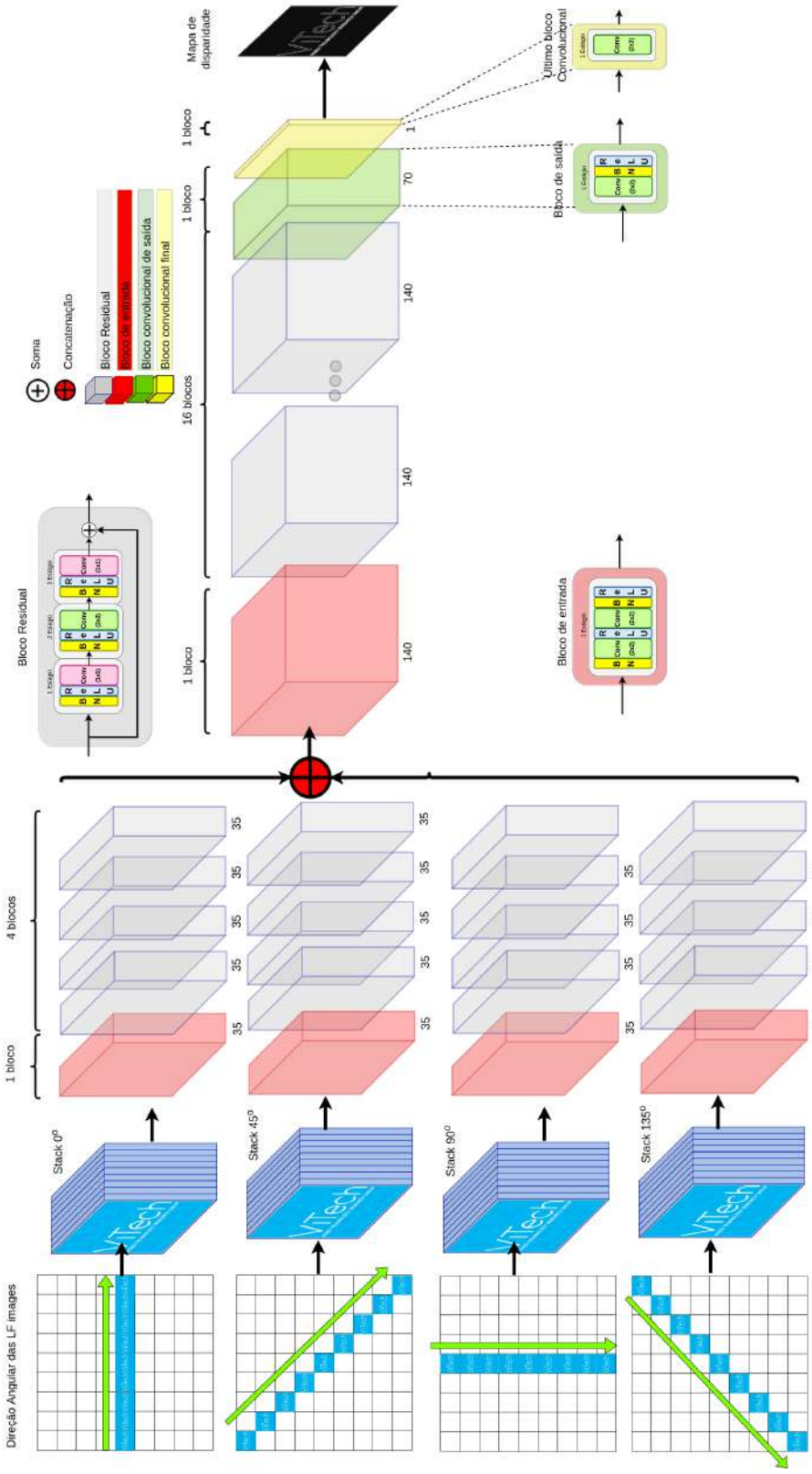


Figura 86 – EPINET com *backbone* ResNet. O bloco residual apresentado na imagem é o mesmo da Figura 85a, mas dependendo da implementação esse bloco pode assumir outra formatação. Abaixo de cada bloco está discriminado o número de filtros por camada de convolução. Fonte: De autoria própria.

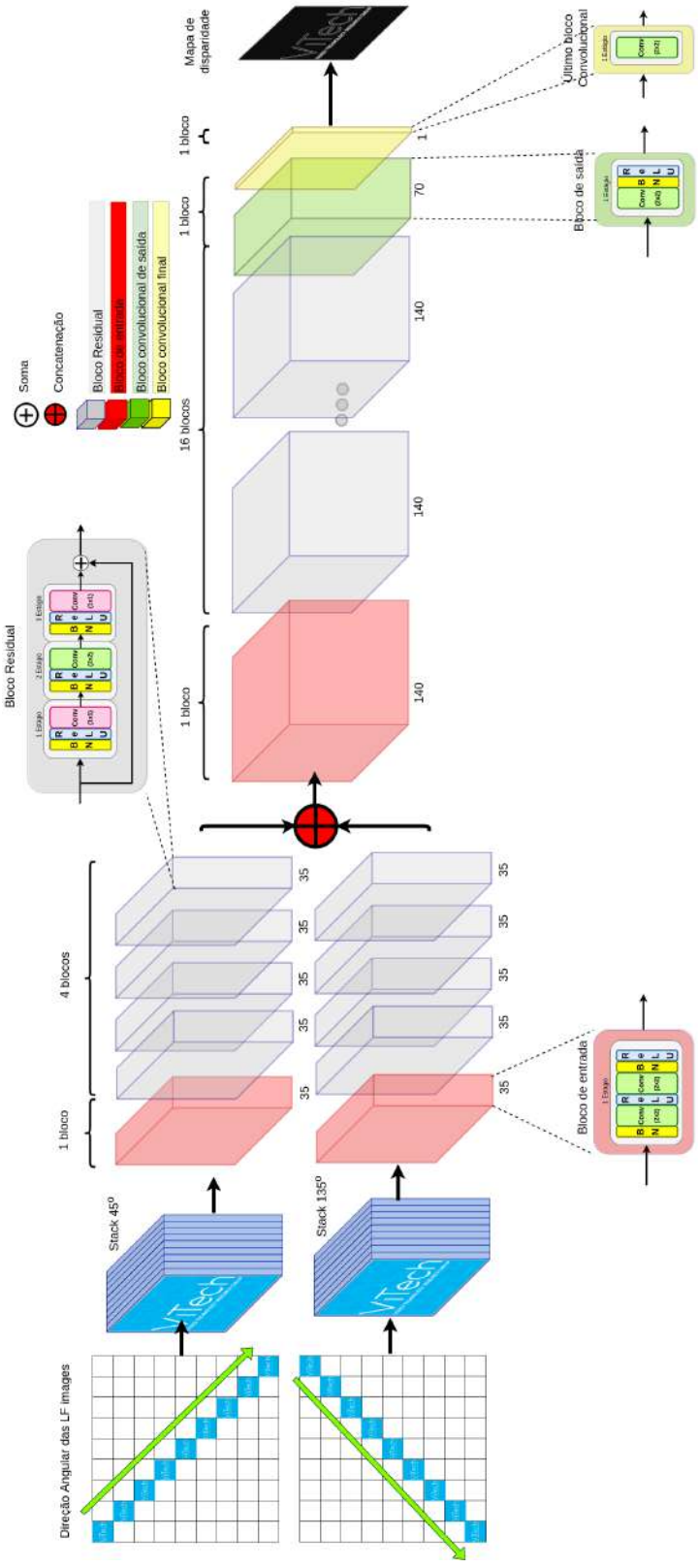


Figura 87 – EPINET-FAST com *backbone* ResNet. O bloco residual apresentado na imagem é o mesmo da Figura 85a, mas dependendo da implementação esse bloco pode assumir outra formatação. Abaixo de cada bloco está discriminado o número de filtros por camada de convolução. Fonte: De autoria própria.

6.2.2 U-EPINET

A U-Net é uma arquitetura desenvolvida para segmentação de imagens biomédicas (KROHN; BEYLEVELD; BASSENS, 2020), mas que também é usada para a geração de mapas de profundidade a partir de imagens monoculares (CANTRELL; MILLER; MORATO, 2020), e de imagens estéreo (RENTERIA-VIDALES et al., 2020). No caso de imagens LF, Heber (HEBER; YU; POCK, 2017) propõe usar a U-Net para extração de mapas de profundidade utilizando como entrada um único volume RGB EPI e fornecendo na saída um volume de disparidades.

Essa tese propõe uma arquitetura chamada U-EPINET, que é a fusão entre a entrada multifluxo da EPINET com a estrutura da U-Net. O resultado é uma rede que apresenta a extração de características multifluxo na etapa de codificação e usa apenas uma saída na etapa de decodificação. Apesar da U-EPINET ser inspirada na arquitetura apresentada em (HEBER; YU; POCK, 2017), sua entrada de dados e a saída diferem substancialmente desse artigo. Na U-EPINET as entradas são volumes de SAI's e a saída é diretamente o mapa de disparidade. A Figura 88 apresenta o modelo geral simplificado dessa fusão de arquiteturas. Nessa figura é possível observar o gargalho crescente e a forma como são concatenadas as saídas multifluxos. A seguir são detalhados os modelos e estratégias usadas.

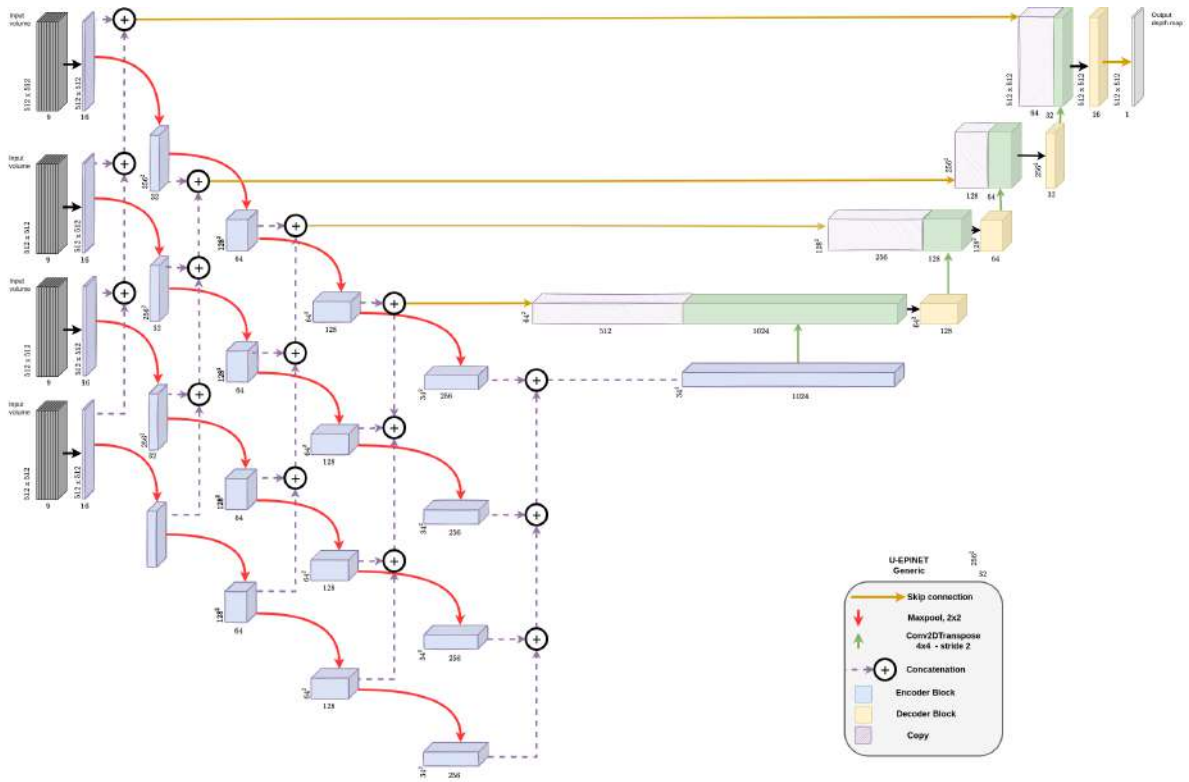


Figura 88 – Estrutura *u-shaped* genérica simplificada da U-EPINET. Fonte: De autoria própria.

6.2.2.1 U-EPINET Universal

A U-EPINET Universal é a **estrutura referencial** usada como base para a construção dos demais modelos. Essa abordagem apresenta uma fusão simples entre as arquiteturas U-Net e EPINET. A Figura 89 apresenta a U-EPINET Universal de forma detalhada. Nessa rede as entradas são os mesmos quatro volumes de entrada da EPINET (SHIN et al., 2018) e cada nível é composto por **dois blocos** com camadas convolucionais, tal como na U-Net original (RONNEBERGER; FISCHER; BROX, 2015). Convém ressaltar que cada bloco convolucional é composto por **uma camada convolucional (Conv-Relu) com *kernel*** 3**x**3. Em estudos preliminares se constatou que dependendo do tamanho do *kernel* e *stride* usado na etapa de *upsampling* podem ocorrer padrões do tipo *checkerboard artifacts* [3]. Na U-Net não é aplicado o *padding* a cada bloco, o que resulta em imagens com dimensões menores nas saídas. A fim de manter padronizado as saídas/entradas nas etapas de *enconder*/*decoder* e como o uso de preenchimento não trouxe nenhum ônus detectado, para fins de simplicidade, **se manteve o *padding* na U-EPINET**.

6.2.2.2 U-EPINET Modelo ingênuo

Os modelos U-EPINET_MODEL-A0 (Apêndice A.0.1) e U-EPINET_MODEL-A1 (Apêndice A.0.2) são modelos ditos ingênuos por reproduzirem a abordagem da U-EPINET Universal. Conforme observa-se na Figura 90, o modelo U-EPINET-MODEL-A0 possui apenas um bloco (**Conv-BN-ReLU-Conv-BN-ReLU**) por nível, tanto na etapa de *encoder* como na de *decoder*. Na etapa de *encoder* é feita uma cópia por nível do bloco convolucional de saída de cada ramo multifluxo. Essas cópias são **concatenadas entre si** gerando um bloco convolucional **quatro vezes maior**, que é usado como entrada das *skip-connections*. Em cada ramo multifluxo, no *enconder*, os blocos passam por um processo de redução de dimensionalidade/*downsampling* através do uso da operação de ***max pooling*** para dados espaciais 2D [4] com *kernel* de tamanho 2x2. Esse processo faz com que as dimensões sejam reduzidas pela metade mantendo o mesmo número de filtros convolucionais. Na etapa de *decoder* o aumento de dimensionalidade/*upsampling* é feito através da **convolução transposta bidimensional** [5] com kernel 4x4 e passo/*stride* igual a 2 para dobrar as dimensões, mantendo o número de filtros convolucionais e evitando *checkerboard artifacts*.

Na etapa de *enconder* são usadas as seguintes quantidades de filtros/camadas convolucionais por nível em ordem crescente: 16, 32, 64, 128 e 256. Já na etapa de *decoder*, os filtros por nível em ordem decrescente são: 128, 64, 32 e 16.

O modelo U-EPINET-MODEL-A1 (Figura 91) possui a mesma configuração na

[3] São padrões que lembram tabuleiros de xadrez
[4] MaxPool2D
[5] Conv2DTranspose

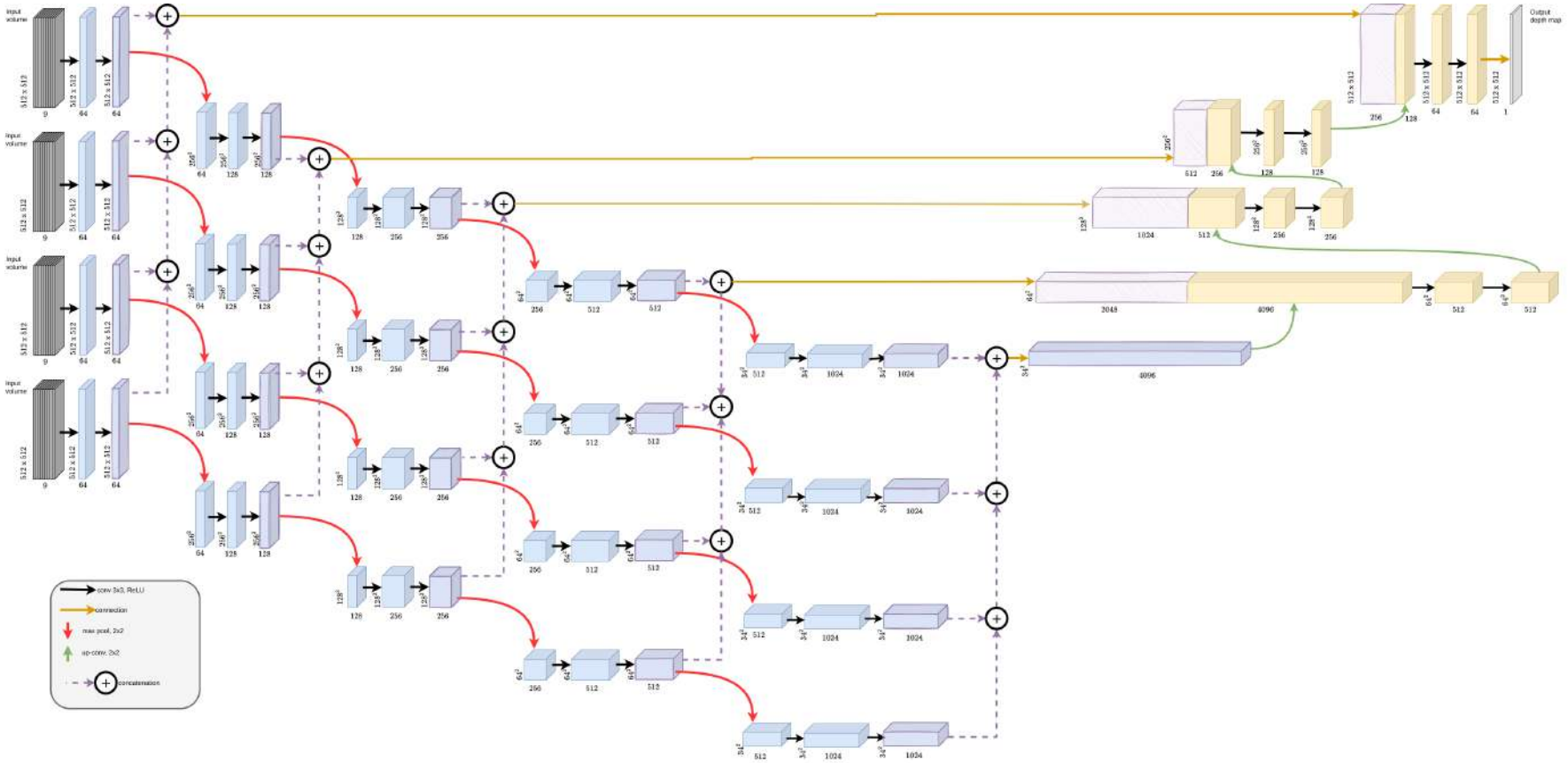


Figura 89 – U-EPINET Universal. Estrutura *u-shaped* genérica **detalhada** da U-EPINET, usada como base para o desenvolvimento dos demais modelos. Fonte: De autoria própria.

etapa de *encoder* da rede U-EPINET-MODEL-A0. Seu diferencial é na etapa de *decoder*, onde o número de características/filtros por bloco foi modificado buscando balancear o número de filtros oriundos das *skip-connections* com a quantidade de filtros usados nos blocos convolucionais. Desta forma, as convoluções por nível em ordem decrescente na etapa de *decoder* são: 512, 256, 128 e 64.

6.2.2.3 U-EPINET Modelo LinkNet

A U-EPINET Modelo LinkNet utiliza a mesma estratégia da LinkNet (CHAURASIA; CULURCIELLO, 2017) nas *skip-connections*. Na U-Net para se recuperar a informação espacial associada a cada filtro na etapa de decodificação, concatena-se a saída de cada nível de codificação (antes do *downsample*) com a entrada do primeiro bloco convolucional no nível equivalente de decodificação. Em uma U-EPINET com modelo ingênuo, as saídas dos quatro fluxos são concatenadas entre si e depois anexadas a entrada do nível de decodificação. Essa estratégia dificultou o aprendizado e não trouxe ganhos em termos de velocidade. A estratégia da LinkNet é somar a saída do *enconder* com a entrada equivalente no *decoder*, assim a informação espacial é recuperada e o processamento torna-se muito mais rápido. Na U-EPINET Modelo LinkNet se adotou uma estratégia mista, as saídas das etapas são somadas e o resultante dessa soma é concatenado com a entrada do primeiro bloco convolucional do *decoder*. Desta forma, se recupera a informação espacial e se simplifica os filtros resultantes que armazenam as disparidades. Foram criadas duas arquiteturas U-EPINET Modelo LinkNet: Arquitetura U-EPINET MODEL_B1 (Apêndice A.0.3) e Arquitetura U-EPINET MODEL_B2 (Apêndice A.0.4).

As arquiteturas U-EPINET MODEL_B1 (Figura 92) e U-EPINET MODEL_B2 (Figura 93) , apresentam um bloco convolucional simplificado (**Conv-ReLU**). Para *downsampling* as redes utilizam ***max pooling*** com *kernel* 2x2, e para *upsampling* usam camadas de **convolução transposta bidimensional** com *kernel* 8x8 e *stride* igual a 2 [6]. Na etapa de *enconder* são usadas as seguintes camadas convolucionais por nível em ordem crescente: 64, 128, 256 e 512; e na etapa de *decoder*, em ordem decrescente: 512, 256, 128 e 64. O único diferencial entre ambos modelos é a última camada convolucional antes do mapa de disparidade ser gerado. Na U-EPINET MODEL_B1 esse bloco é (**Conv-SoftMax**) e na U-EPINET MODEL_B2 o bloco permanece como (**Conv-ReLU**)

[6] Para evitar *checkerboard artifacts* o tamanho do *kernel* deve ser divisível pelo *stride*

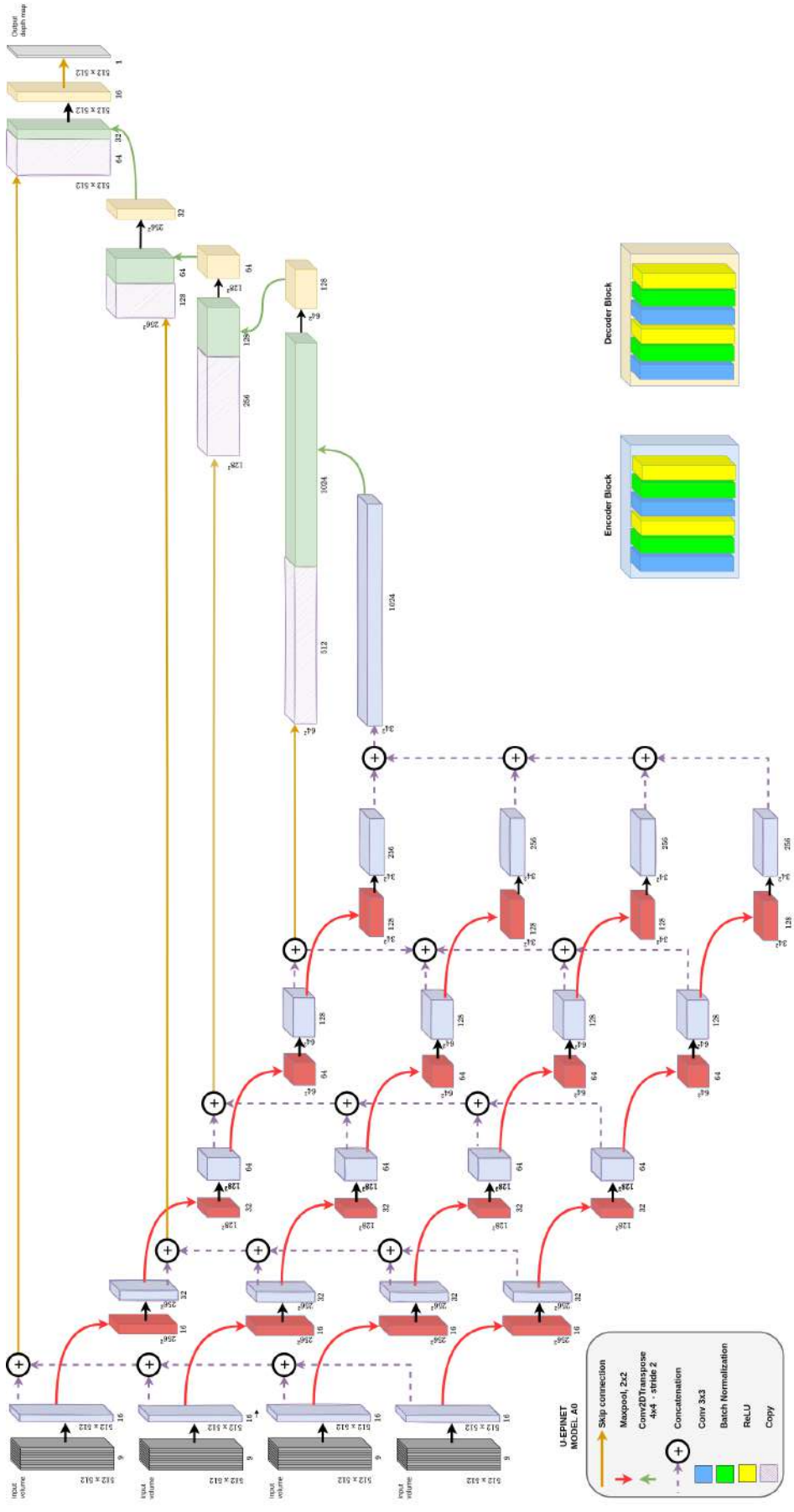


Figura 90 – Arquitetura U-EPINET MODEL_A0. Fonte: De autoria própria.

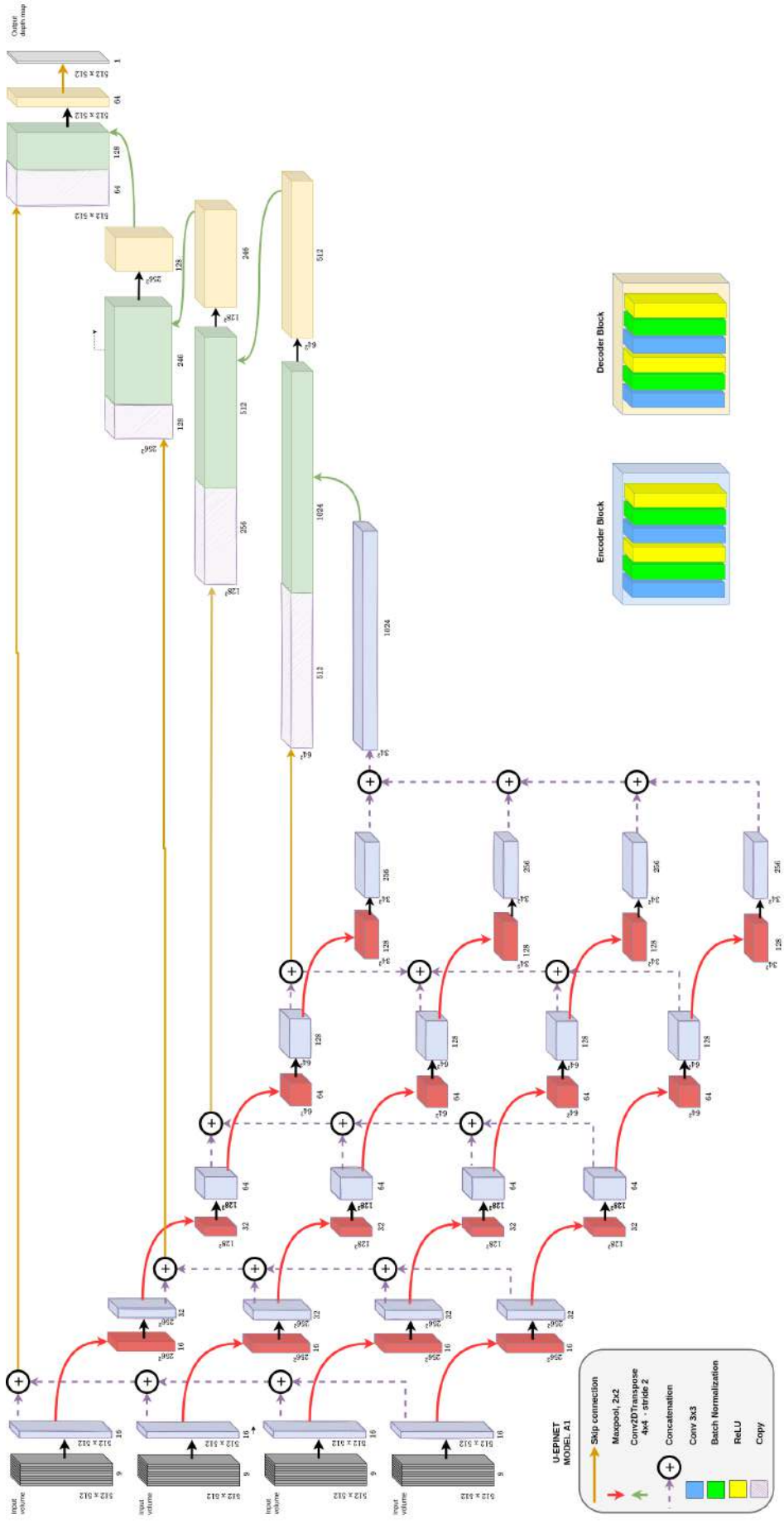


Figura 91 – Arquitetura U-EPINET MODEL_A1. Fonte: De autoria própria.

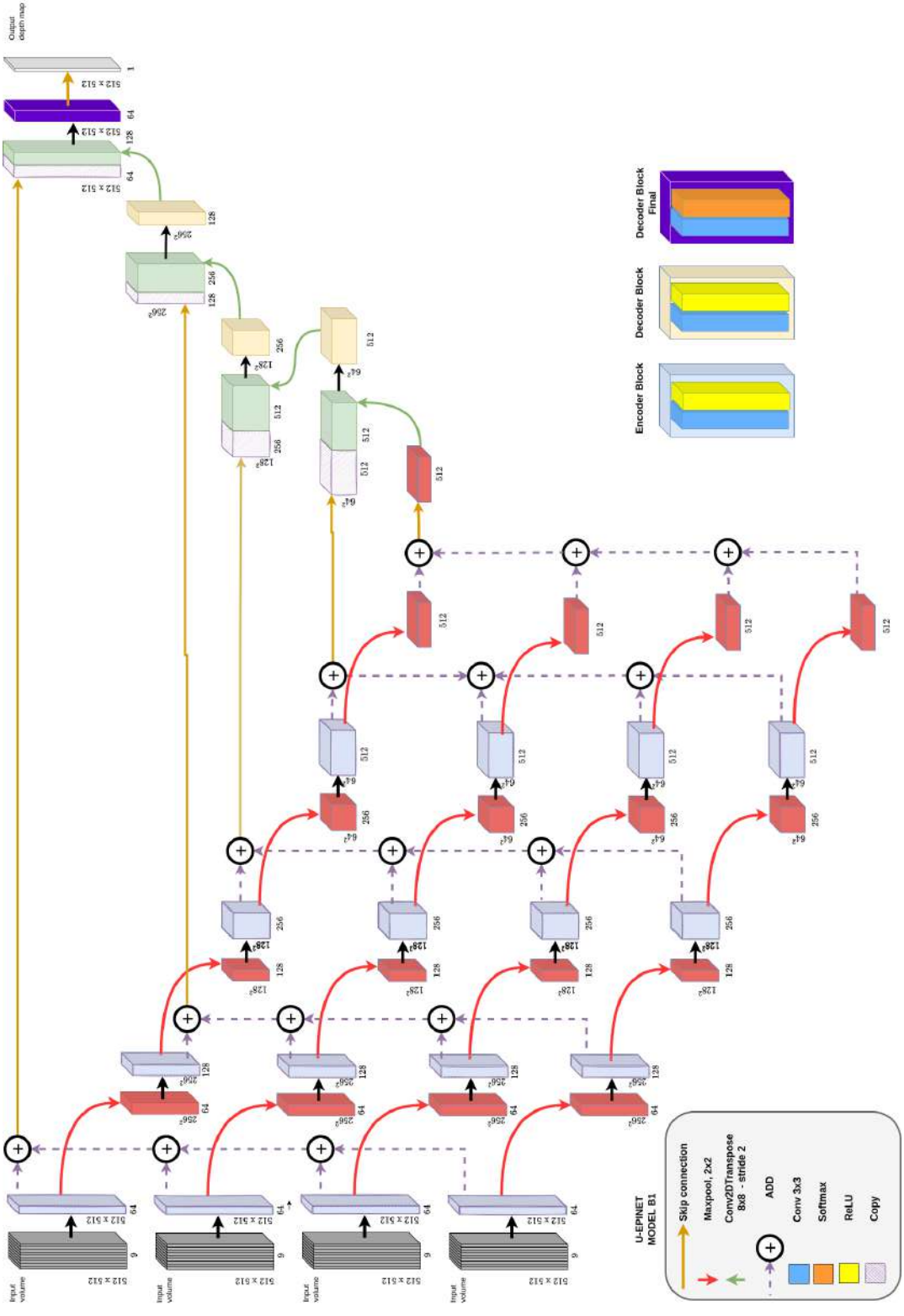


Figura 92 – Arquitetura U-EPINET MODEL_B1. Fonte: De autoria própria.

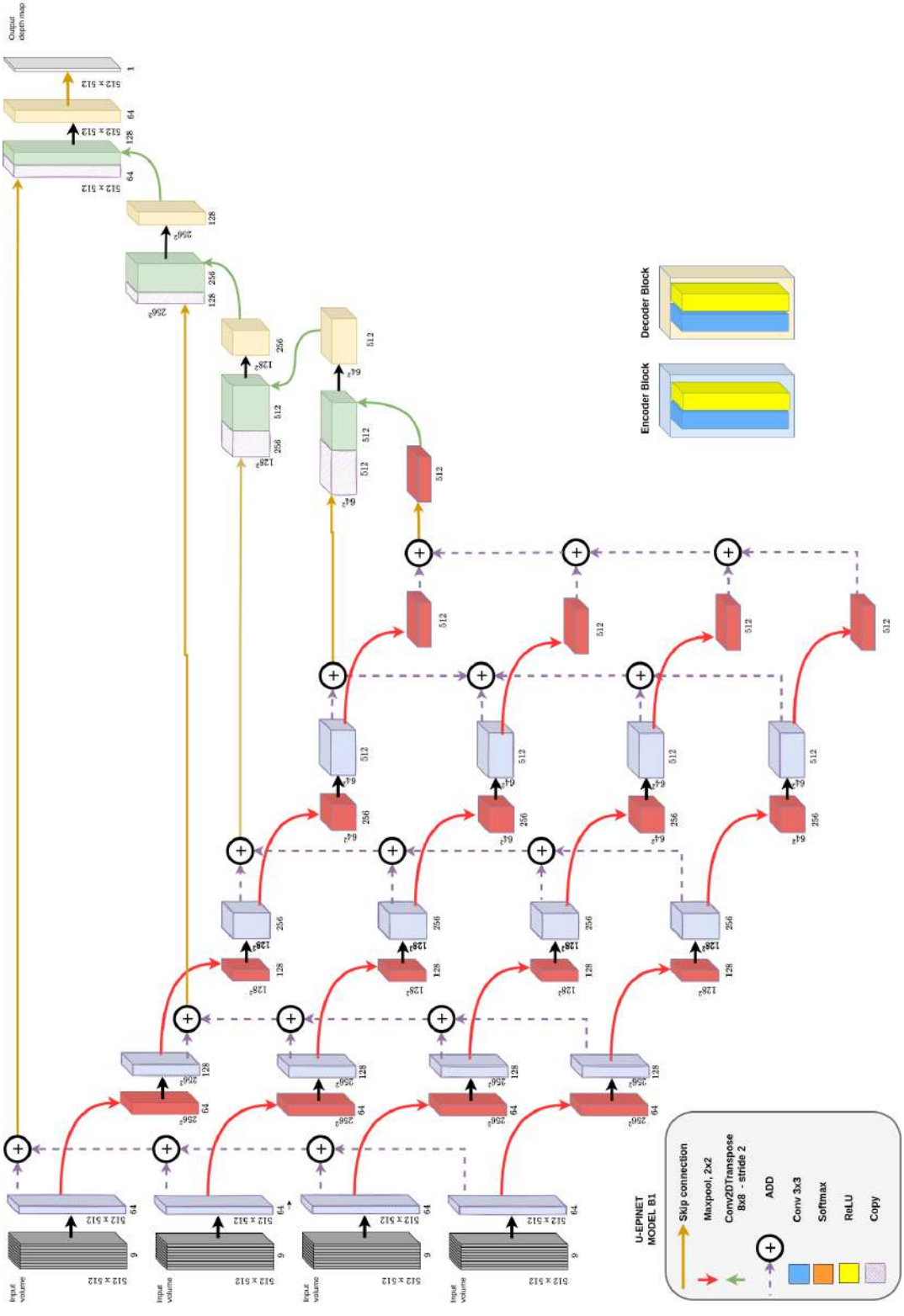


Figura 93 – Arquitetura U-EPINET MODEL_B2. Fonte: De autoria própria.

6.2.2.4 *U-EPINET Modelo com* skip-connections *convoluidas*

Esse modelo utiliza convoluções para agrupar as saídas de cada ramo dos multifluxos de entrada. Antes de concatenar com a entrada do *decoder*, as saídas do *encoder*, por nível, são concatenadas entre si e passam por uma camada de convolução para gerar um filtro, conforme o destaque na Figura 94. Isso faz com que seja passada uma combinação de filtros do *encoder* com o mesmo tamanho da entrada do *decoder*. A inspiração para essa abordagem é a propriedade associativa da convolução (Equação 30). Ao invés de apenas somar as convoluções, pode-se gerar uma única saída que agrupe as entradas e mantenha a informação espacial. Ao se aplicar a estratégia de aprendizado na *skip-conection*, se permite que a convolução associada a combinação de filtros seja inferida. Essa metodologia permite aprender também convoluções não lineares.

$$(f * h) * g = f * (h * g) \tag{30}$$

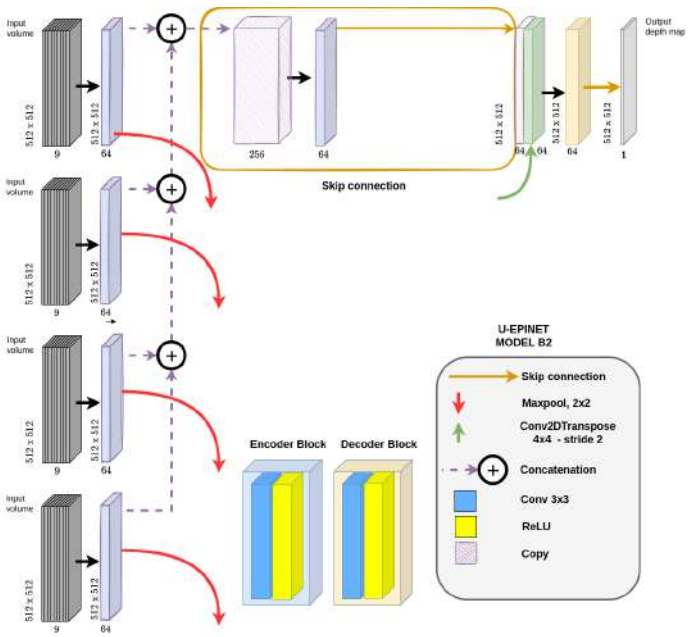


Figura 94 – *Skip-connections* convoluidas. Fonte: De autoria própria.

O modelo usado é o mesmo U-EPINET MODEL_B2, com o uso de *skip-connections* com convoluções.

6.3 Conclusão

Esse capítulo apresentou duas novas abordagens para a geração de mapas de profundidade a partir de imagens LF. A primeira abordagem explora a poda de fluxos de entrada na EPINET (SHIN et al., 2018) e propõe soluções a partir de modificações nas camadas convolucionais e filtragens morfológicas. Já a segunda abordagem é

totalmente nova, apresentando soluções híbridas entre a entrada multifluxo, redes U-Net e LinkNet. No próximo capítulo são avaliados os desempenhos dessas redes em relação a EPINET.

7 RESULTADOS EXPERIMENTAIS

Nessa etapa, a partir da idealização e modelagem propostas no capítulo anterior, foram construídas as redes neurais e realizados os testes de desempenho.

7.1 Plataforma dos experimentos

Os experimentos foram realizados em um Intel®Core™ i9-9900KF CPU @3.60GHz × 16, com uma placa de vídeo Nvidia Titan V, rodando no sistema operacional Ubuntu 20.04.4 LTS-64 bits. Os experimentos são avaliados usando a ferramenta padrão para LF, o 4D Light Field Benchmark. Esse modelo pode ser rodado localmente, ou os dados podem ser enviados para o site [1]. Optou-se por rodar localmente devido ao tempo de resposta do site. O *benchmark* tem como entrada imagens LF padrão do 4D Light Field Dataset, com tamanho de 512x512x9x9.

7.2 4D Light Field Benchmark Dataset

O *dataset* se concentra em cinco problemas/desafios que imagens LF apresentam: (i) limites de oclusão; (ii) estruturas finas; (iii) baixa textura; (iv) superfícies lisas; e (v) ruído da câmera. As cenas fornecidas são renderizadas sinteticamente em virtude da imprecisão na acurácia que ocorre ao se usar imagens reais para gerar o *ground truth*. Segundo os autores (WANG et al., 2016), para criar um *ground truth* mais acurado em uma imagem real, seria necessário o uso de algoritmos de visão computacional para realizar esse processamento, o que não é o foco do *benchmark*.

A cenas geradas são divididas em cenas estratificadas [2] e fotorrealísticas [3]. As cenas estratificadas são imagens simples dedicadas a apenas um grupo limitado dos problemas citados, permitindo desacoplar a análise de desempenho para cada problema individualmente (WANG et al., 2016). As cenas estratificadas são:

[1] `https://lightfield-analysis.uni-konstanz.de/benchmark/table?column-type=images&metric=badpix_0070`

[2] Em inglês *stratified scenes*

[3] Em inglês *fotorealistic scenes*

- *Backgammon* - Cena projetada para avaliar a interação entre estruturas finas, limites de oclusão e diferenças de disparidade;
- *Dots* - Cena projetada para avaliar o efeito do ruído da câmera na reconstrução de objetos com tamanhos variados;
- *Pyramids* - Cena projetada para avaliar o desempenho do algoritmo entre imagens com geometria convexa versus geometria côncava, e geometria arredondada versus geometria planar;
- *Stripes* - Cena projetada para avaliar a influência da textura e contraste nos limites de oclusão.

O segundo conjunto de imagens é composto por cenas sintéticas fotorrealísticas visando emular situações complexas do mundo real. Esse tipo de imagem contém desafios potencialmente significativos em combinações espaciais dos objetos na cena. Cenas fotorrealísticas permitem avaliação de desempenho em estruturas finas, áreas de oclusão complexas, superfícies planas inclinadas e superfícies não planas contínuas (WANG et al., 2016).

7.3 Métricas usadas

O conjunto de métricas usadas no 4D Light Field Benchmark (WANG et al., 2016) está dividido em três abordagens: (i) métricas de avaliação geral - aplicável a todas as cenas; (ii) métricas para cenas estratificadas; e (iii) métricas para cenas fotorrealísticas. As métricas usam o mapa de disparidade estimado pelo algoritmo chamado de *d* ou *algo*, o mapa de disparidade real *gt* (do inglês *ground-truth*) e a máscara de avaliação $\mathcal{M}$, que será associada a uma característica específica.

7.3.1 Métricas de avaliação geral

O **MSE** é erro médio quadrático sobre todos os pixels em relação a máscara de avaliação $\mathcal{M}$ multiplicado por 100 (Equação 31).

$$\mathbf{MSE}_{\mathcal{M}} = \frac{\sum\limits_{x \in \mathcal{M}} (d(x) - gt(x))^2}{|\mathcal{M}|} * 100. \tag{31}$$

Já o **BadPix(t)** fornece a porcentagem de pixels discrepantes dentro da máscara $\mathcal{M}$ com valores maior que um limiar/*threshold* t, conforme a Equação 32. Podemos interpretar de forma simplificada como $abs(gt - algo) > t$ para uma máscara específica. Os valores *default* usados no *benchmark* para t são: 0.01, 0.03, 0.07.

$$\mathbf{BadPix}_{\mathcal{M}(t)} = \frac{|\{x \in \mathcal{M} : |(d(x) - gt(x))| > t\}|}{|\mathcal{M}|}. \tag{32}$$

A métrica **Q25** apresenta o $25°$ percentil dos erros de disparidade. Ou seja, o erro máximo de disparidade absoluta dos melhores 25% pixels multiplicado por 100. Por fim, a métrica **Runtime** é o tempo de execução em segundos;

7.3.2 Métricas para cenas fotorrealísticas

Em imagens fotorrealísticas as métricas avaliam condições relacionadas a aspectos gerais dos mapas de disparidade gerados.

Para medir o desempenho do algoritmo em superfícies planas e/ou em regiões de curvas suaves se usa uma métrica chamada *Bumpiness* , ou **ondulação** em português (Equação 33). Essa medida estima a suavidade associada, mas não mede desorientação ou deslocamento na LF (WANG et al., 2016).

$$\mathbf{Bumpiness} = \frac{\sum_{x \in \mathcal{M}} min(0.05, ||H_f(x)||_F)}{|\mathcal{M}|} * 100. \tag{33}$$

A ondulação pode ser aplicada em planos irregulares (***bumpiness planes***) onde retorna a média da norma de Frobenius(F) da matriz Hessiana (H) de $(gt - algo)$ em regiões planas, multiplicada por 100; e também pode ser aplicada em superfícies com continuidades irregulares (***bumpiness planes***), onde retorna a média da norma de Frobenius da matriz Hessiana de $(gt - algo)$ em regiões não planas lisas, multiplicada por 100 (WANG et al., 2016).

O desbaste fino (***fine thinning***) retorna a porcentagem de pixels em torno de estruturas finas com $(gt - algo) > 0,15$. O cálculo do *fine thinning* é feito pela Equação 34, onde $\mathcal{M}$ é a mascara para pixel de estruturas finas. Já o engrossamento fino (***fine fattening***) retorna a porcentagem de pixels em torno de estruturas finas com $(gt - algo) < -0,15$. A Equação 35 realiza o cálculo do *fine fattening*, onde $\mathcal{M}$ é a mascara para pixels em torno de estruturas finas.

$$\mathbf{Thinning}_{\mathcal{M}(t)} = \frac{|\{x \in \mathcal{M} : gt(x) - d(x) > t\}|}{|\mathcal{M}|}. \tag{34}$$

$$\mathbf{Fattening}_{\mathcal{M}(t)} = \frac{|\{x \in \mathcal{M} : gt(x) - d(x) < t\}|}{|\mathcal{M}|}. \tag{35}$$

As outras métricas associadas a imagens fotorrealisticas são:

- **Descontinuidades** - *Discontinuities*, porcentagem de pixels nas regiões de descontinuidade com $abs(gt - algo) > 0,07$;
- **Erro angular mediano de superfícies normais em regiões planas** - *median angular error planes* ou *MAE Planes*;
- **Erro angular mediano de superfícies normais em regiões lisas e não planas** - *median angular error continuities surfaces* ou *MAE Continuities Surfaces*.

7.3.3 Métricas para cenas estratificadas

Conforme dito anteriormente, as cenas estratificadas possuem características selecionadas para explorar problemas específicos associados as imagens LF. A seguir são detalhadas as métricas por cena.

A cena ***Backgammon*** é idealizada para avaliar a interação de estruturas finas, limites de oclusão e diferenças de disparidade. Para esse fim, a cena calcula o espessamento de primeiro plano pela métrica ***Foreground Fattening*** que é definido nas bordas de oclusão em uma máscara $\mathcal{M}$ que só possui os pixels de fundo conforme a Equação 36, onde h é a soma do plano de fundo BG com o primeiro plano FG dividido por 2 (Equação 37). Assim essa métrica retorna a porcentagem de pixels em torno de estruturas finas cuja estimativa de disparidade está mais próxima do primeiro plano do que do plano de fundo.

$$\mathbf{FG\_Fattening} = \frac{|\{x \in \mathcal{M} : d(x) > h\}|}{|\mathcal{M}|}. \tag{36}$$

$$h = \frac{BG + FG}{2}. \tag{37}$$

O processo para calcular o afinamento do primeiro plano pela métrica ***Foreground Fitting*** segue um raciocínio semelhante onde é calculada a porcentagem de pixels em estruturas finas cuja estimativa de disparidade está mais próxima do plano de fundo do que do primeiro plano (Equação 38).

$$\mathbf{FG\_Fitting} = \frac{|\{x \in \mathcal{M} : d(x) < h\}|}{|\mathcal{M}|}. \tag{38}$$

Conforme dito anteriormente, a cena ***Pyramids*** é usada para avaliar o desempenho em imagens LF entre estruturas com geometria convexa versus geometria côncava; e entre geometria arredondada versus geometria planar. Para esse objetivo se usa a equação de ondulação (Equação 33) com uma máscara $\mathcal{M}$ adequada. Assim ela calcula a ondulação paralela (métrica ***Bumpiness Parallel***) que retorna a média da norma de Frobenius da matriz Hessiana de $(gt - algo)$ em uma dada região plana $\mathcal{M}$; e a ondulação inclinada (métrica ***Bumpiness Slanted***) que calcula a média da norma de Frobenius da matriz hessiana de $(gt - algo)$ em uma dada região plana $\mathcal{M}$.

A cena ***Stripes*** avalia a influência da textura e do contraste nos limites com oclusão, para isso ela usa o ***Badpix*** com máscaras $\mathcal{M}$ para cada tipo de situação, tendo o valor do *threshold* t igual a $0,07$. As métricas implementadas são:

- ***Bright Stripes*** - listras claras, porcentagem de pixels na máscara $\mathcal{M}$ sendo $abs(gt - algo) > 0,07$;
- ***Dark Stripes*** - listras escuras, porcentagem de pixels na máscara $\mathcal{M}$ sendo

$abs(gt - algo) > 0,07$;

- ***Low Texture*** - baixa textura, porcentagem de pixels na máscara $\mathcal{M}$ sendo $abs(gt - algo) > 0,07$

Já a cena ***Dots*** estuda o efeito do ruído da câmera na reconstrução de objetos de tamanhos variados. Para gerar uma aproximação do ruído térmico e do ruído de disparo de uma câmera LF foi adicionado ruído Gaussiano com variações lineares crescentes entre $0,0$ e $0,2$, seguindo a ordem de disposição dos objetos na linha principal (WANG et al., 2016). A quantificação da robustez contra ruído é calculada pelo MSE aplicado ao plano de fundo (WANG et al., 2016). Para mensurar a sensibilidade em pequenas geometrias é calculada a porcentagem de pontos detectados, onde um ponto conta como detectado se a maioria de suas estimativas de disparidade local for distinguível do plano de fundo por estar com um escore maior que 50% em relação ao ponto real indicado no *ground truth*, dado o BadPix(t) com *threshold* t de $0,4$.

7.4 Implementação e testes

Nessa seção são apresentados os cenários explorados, testes realizados e desempenho apresentado.

7.4.1 Estudo preliminar e adequação dos códigos

A primeira etapa consiste em transpôr a rede EPINET (SHIN et al., 2018)[4] para os mesmos parâmetros usados nesse trabalho, afim de avaliar o tempo de execução. O código original usa a API sequencial do Tensorflow com Keras (que não é integrado a essa API) e Python 2.x. O primeiro passo foi atualizar o código para a versão Python 3.7 e Tensorflow 2.4, que possui a API Keras integrada, de forma a trabalhar com ambientes com máxima similaridade. Cabe ressaltar que o tempo de execução varia de acordo com o hardware usado, por isso a importância de garantir o mesmo ambiente para fins de comparação. O próximo passo foi comparar os resultados que rodaram localmente com os que foram disponibilizado no site do 4D Light Field Benchmark.

As configurações são as seguintes:

- 4D Light Field Benchmark
  - EPINETFCN: EPINET usando as entradas $0°$, $90°$, $45°$, $135°$;
  - EPINETFCN9x9: modelo que usa todos os pontos de vista 9×9.
- Benchmark local
  - EPINET_TF_20: Atualização da EPINETFCN rodando localmente.

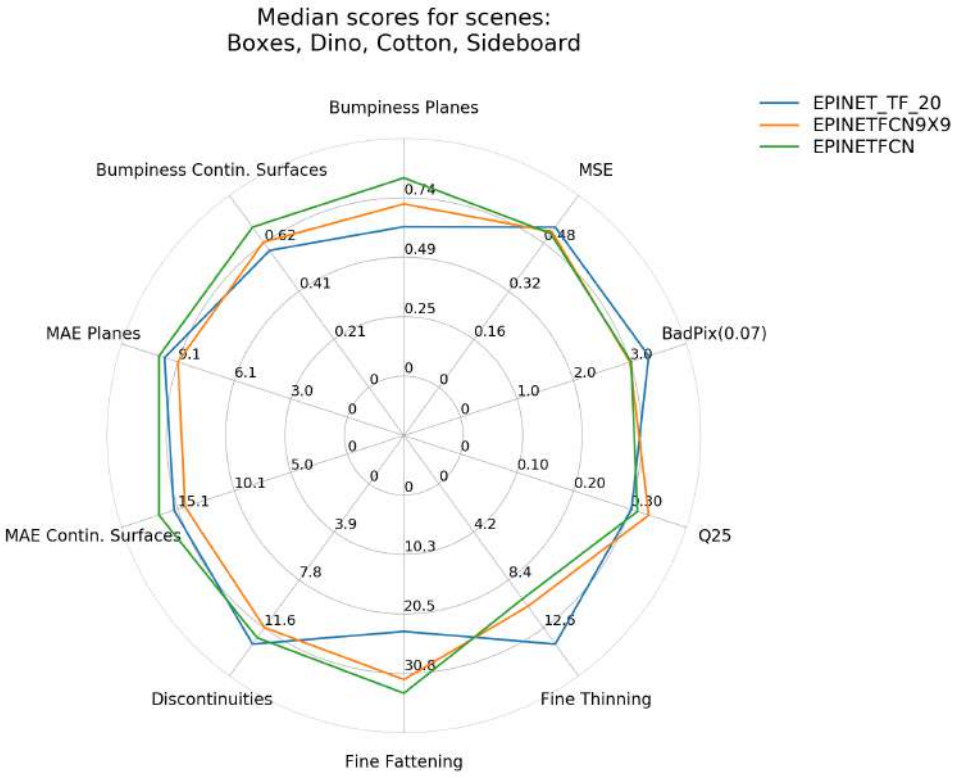


Figura 95 – Métricas de comparação entre as arquiteturas EPINET. Fonte: De autoria própria, gerado no software 4D Light Field Benchmark.

Conforme se observa na Figura 95, o comportamento para as cenas foi bastante próximo ao apresentado pelo site de *benchmark*, inclusive com desempenho melhor em alguns critérios. Sendo os três principais: MSE, MAE Planes e MAE Contin surfaces próximos do publicado. Na Figura 96 se observa que diferenças entre o apresentado no dataset e o obtido, permite usar a rede EPINET_TF_20 para fins de comparação.

7.4.2 EPINET-FAST: primeiro cenário

Para a geração do mapa de disparidade foram construídas três variações da EPINET-FAST (Figura 82). Para esse cenário se fez variações tanto na qualidade de entradas (volumes usados) quanto na quantidades de filtros utilizados nas camadas convolucionais, a fim de comparação entre os desempenhos.

As variações da EPINET-FAST são as seguintes:

- EPINET-FAST-D Possui a estrutura apresentada na Figura 82.
- EPINET-FAST_0_90 - Possui a estrutura apresentada na Figura 82, mas recebe com entrada as subaberturas localizadas a $0°$ e $90°$ a partir da vista central.
- EPINET-FAST_45_135 - Possui a estrutura apresentada na Figura 82, mas com

[4]disponibilizada em `https://github.com/chshin10/epinet`

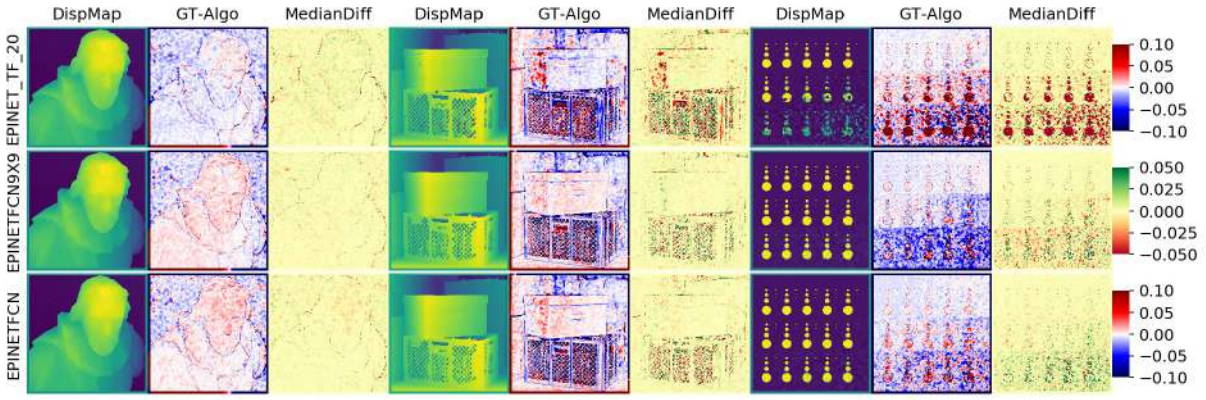


Figura 96 – Comparação entre os algoritmos EPINET. A primeira coluna ilustra os mapas de disparidade dos algoritmos. A segunda coluna representa a diferença de disparidade em relação ao *ground truth*. As áreas brancas representam estimativas altamente precisas, estimativas muito próximas nas áreas azuis e muito distantes nas áreas vermelhas. A terceira coluna apresentam o comportamento dos algoritmos em relação ao desempenho mediano. Amarelo representa desempenho médio, verde acima da média e vermelho abaixo da média. Fonte: De autoria própria, gerado no software 4D Light Field Benchmark.

modificação na quantidade de filtros aplicados nos oito blocos finais. Nessa implementação são usados 210 filtros por bloco.

- EPINET-FAST_45_135_F - Semelhante a anterior. A única diferença é o fato da imagem de saída passar por uma abertura e um fechamento morfológico para tentar eliminar potenciais distorções.

Observa-se na Figura 97 que as redes propostas apresentarem desempenho menor que a (EPINET_TF_20) em termos de MSE. O pior desempenho fica com EPINET-FAST_0_90, enquanto as outras estratégias com apenas os fluxos das vistas a $45°$ e $135°$ podem se aproximar bastante do desempenho da EPINET_TF_20.

A Figura 98 mostra que o erro médio quadrático depende muito das características da imagem. Cenas como a ***Boxes***, que possui muita informação horizontal e vertical, apresentam um bom desempenho do algoritmo EPINET-FAST_0_90. Apesar desse desempenho específico, percebe-se que o algoritmo EPINET-FAST-D também se aproxima bastante em termos de MSE. Isso sugere que os fluxos das diagonais mantêm informação vertical e horizontal.

Pela tabela 8 percebe-se que o tempo de processamento da EPINET é alto para uso em tempo real. Por outro lado, as podas de fluxos de entrada reduziram em média o tempo de processamento em $\approx 1/3$. Esse ganho de velocidade somado a pequena queda de desempenho em outras métricas pode justificar a poda de duas ramificações de entrada. Importante ressaltar que o tipo de imagem LF impacta diretamente no desempenho. Por exemplo, o erro quadrático médio (MSE) da EPINET_45_135_F é de $0,24$ para a cena ***Cotton*** (Figura 98), que é menor que da EPINET_TF_20 que possuí $0,28$. Mas para a segunda imagem **(*Boxes*)**, o MSE da EPINET-FAST_45_135_F

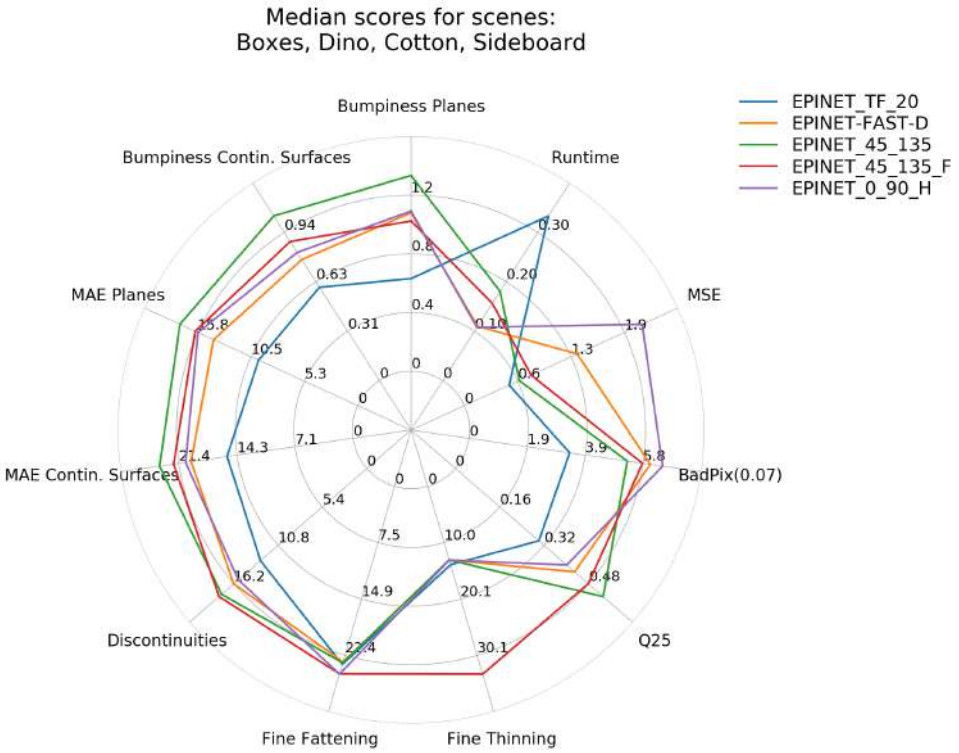


Figura 97 – Métricas de comparação entre as arquiteturas propostas. Fonte: De autoria própria, gerado no software 4D Light Field Benchmark.

passa para $9,71$ e o MSE da EPINET_TF_20 se torna o menor entre todos algoritmos avaliados, com o valor de $6,39$.

Tabela 8 – Comparação entre os tempos de processamento da EPINET (SHIN et al., 2018) e a FAST-EPINET em segundos.

Algoritmo	**EPINET_TF_20**		**EPINET_0_90_H**		**EPINET_45_135**		**EPINET_45_135_F**		**EPINET-FAST-D**	
Cena	**MSE**	**runtime**	**MSE**	**runtime**	**MSE**	**runtime**	**MSE**	**runtime**	**MSE**	**runtime**
Boxes	6,3893	13,5446	6,79043	4,72332	9,4500	7,8325	9,7144	6,1787	8,8249	4,7120
Cotton	0,2798	0,3290	0,36449	0,10658	0,3567	0,1763	0,2435	0,1516	0,2855	0,1070
Dino	0,3146	0,3281	3,30862	0,10541	0,2561	0,1758	0,6051	0,1557	1,7363	0,1024
Sideboard	0,7453	0,3246	0,90149	0,10585	0,9417	0,1797	0,9741	0,1523	0,9354	0,1096

A Figura 99 mostra a comparação entre os algoritmos citados. Pode-se perceber o desempenho próximo nos mapas de disparidades gerados.

7.4.3 EPINET-FAST: segundo cenário

Esse cenário explora o desempenho da EPINET-FAST com *backbone ResNet*. As variações criadas são as seguintes:

- EPINET-RES_1_3_2_ReLU - EPINET com bloco residual da Figura 85c e estrutura apresentada na Figura 86. Essa abordagem foi criada para fins de comparação entre os testes.
- FAST-RES_1_3_2_ReLU - EPINET-FAST com bloco residual da Figura 85c e estrutura apresentada na Figura 87.

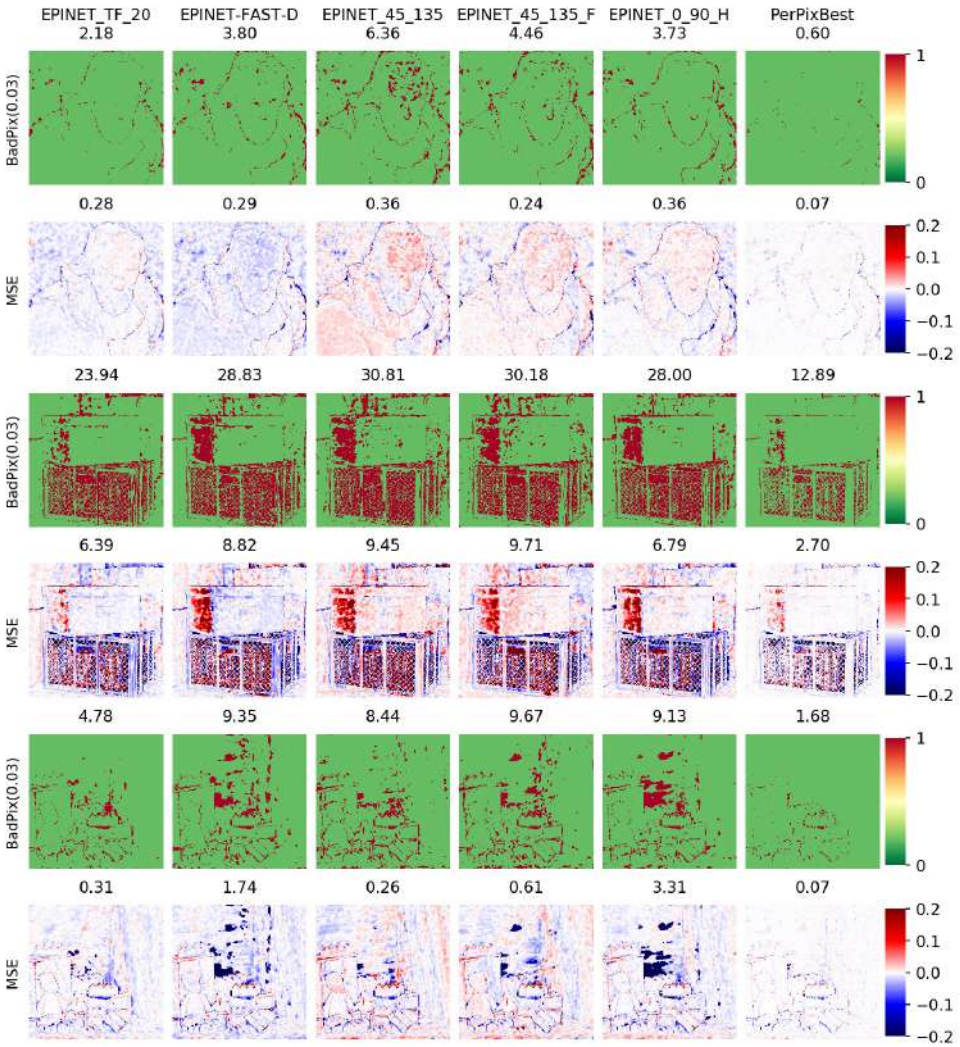


Figura 98 – Comparação do MSE entre as arquiteturas usando as cenas: ***Cotton***, ***Boxes***, ***Dino***. Fonte: De autoria própria, gerado no software 4D Light Field Benchmark.

- FAST-RES_1_3_1_ReLU - EPINET-FAST com bloco residual da Figura 85d e estrutura apresentada na Figura 87.

A Figura 100 mostra o impacto nas métricas no uso do *backbone* ResNet.

7.4.4 Discussão sobre os resultados

Em ambos cenários se fez variações tanto na qualidade de entradas (volumes usados), quanto nas quantidades de filtros utilizados nas camadas convolucionais. Foram realizados os seguintes estudos de amplitude em relação a rede EPINET: variação do número de redes convolucionais, troca do *backbone* para ResNet e alteração do número de entradas. Se constatou que os elementos que causam mais impacto na velocidade de processamento, uma vez a rede treinada, é a quantidade de camadas convolucionais e o número de filtros associados a cada camada. Ao se reduzir o número de entradas para apenas dois fluxos, se tem um ganho de velocidade de aproximadamente 3 vezes, sendo que o problema de degradação na qualidade do mapa

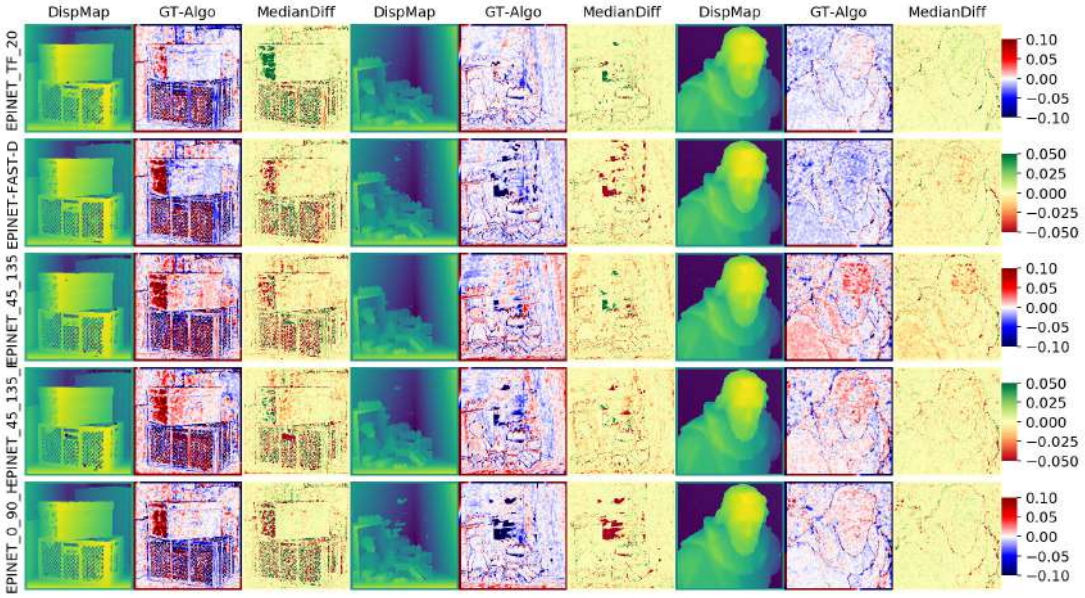


Figura 99 – Métricas de comparação entre os algoritmos EPINET-FAST. Fonte: De autoria própria, gerado no software 4D Light Field Benchmark.

de disparidade de saída pode ser tratado com acréscimos de mais camadas ou filtros convolucionais, ou com tratamento na saída com filtros tradicionais.

O uso do *backbone* ResNet apresentou melhoras de desempenho em relação a EPINET-FAST do primeiro cenário, conforme se observa pela Figura 101. O tempo de *runtime* é ligeiramente maior, mas em termos de MSE a melhora é significativa. Isso indica que a estrategia de *shotcuts* da ResNet permitem alcançar um desempenho melhor que a EPINET-FAST do primeiro cenário e manter o tempo de execução em um patamar próximo.

7.5 U-EPINET

A U-EPINET é uma fusão entre a entrada multifluxo da EPINET com a estrutura de uma U-NET. Dessa forma, ela é uma rede que realiza a extração de características multifluxo na etapa de codificação e usa apenas uma saída na etapa de decodificação. Nessa seção é apresentado o estudo realizado com a U-EPINET

7.5.1 U-EPINET-MODEL-A

Os modelos A são baseados na estrutura geral apresentada na Figura 89. Sendo sua principal característica usar *skip-connections* baseadas na concatenação entre entrada e saída. Os modelos usados estão descritos detalhadamente no Apêndice A.

Os dois modelos avaliados nessa seção são:

- A0 descrição no Apêndice A - seção A.0.1;
- A1 descrição no Apêndice A - seção A.0.2.

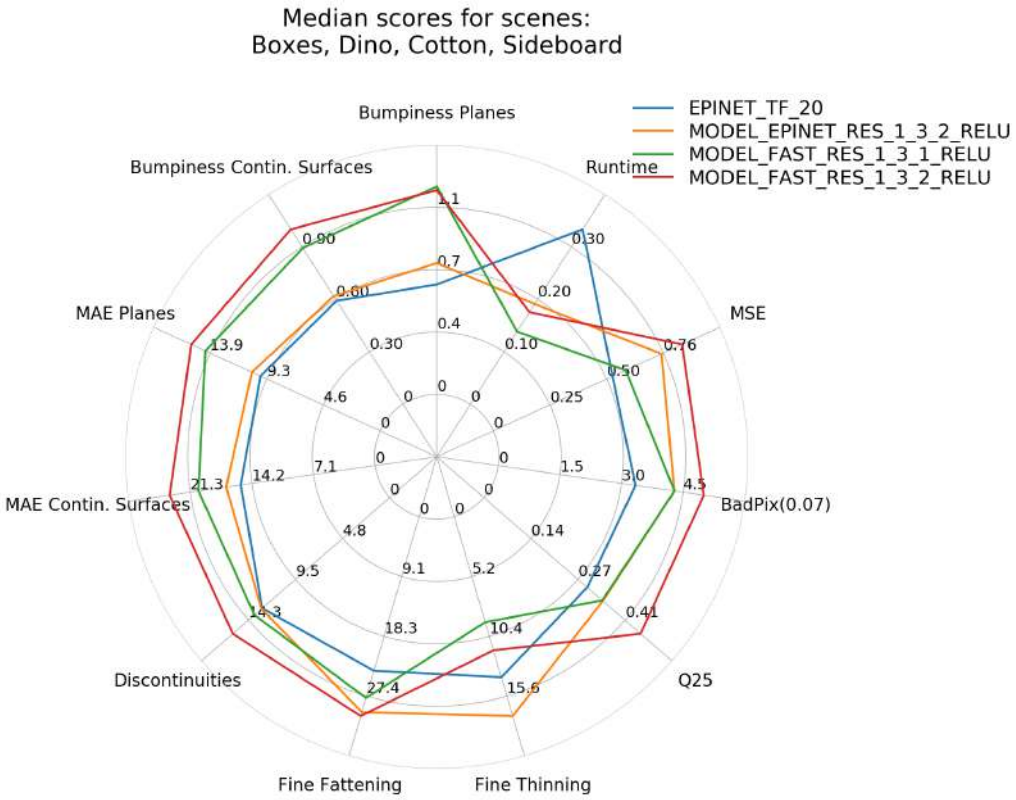


Figura 100 – Métricas de comparação entre as arquiteturas com *backbone* ResNet e a EPINET original (EPINET_TF_20). Fonte: De autoria própria, gerado no software 4D Light Field Benchmark.

A Figura 102 mostra que apesar da rede apresentar um desempenho em termos de velocidade o mais rápido que o modelo EPINET-FAST, o mesmo não ocorre com outros parâmetros. A tabela 9 demonstra a discrepância entre MSE e tempo de execução. As modificações de balanceamento impostas no MODEL_A1 se mostrou efetivo, mas não o suficiente para se aproximar da EPINET-FAST. Essa arquitetura perde muita definição em virtude do formato de concatenação dos filtros da etapa de codificação com a entrada da etapa de decodificação. O que aparenta causar a perda de detalhes. Mesmo assim o balanceamento das *skip connections* causou melhora no desempenho das redes. Para fins comparativos, a Figura 103 demonstra que os mapas de disparidade possuem pouca precisão, mas que existe potencial de melhora ao se trabalhar na questão de balanceamento e composição dos filtros.

7.5.2 U-EPINET-MODEL-B

Esses modelos trabalham a questão das *skip connections* de forma híbrida. As saídas de cada fluxo são somadas e depois concatenadas a entrada do decodificador no nível correspondente.

Os dois modelos avaliados nessa seção são:

- B1 descrição no Apêndice A - seção A.0.3;

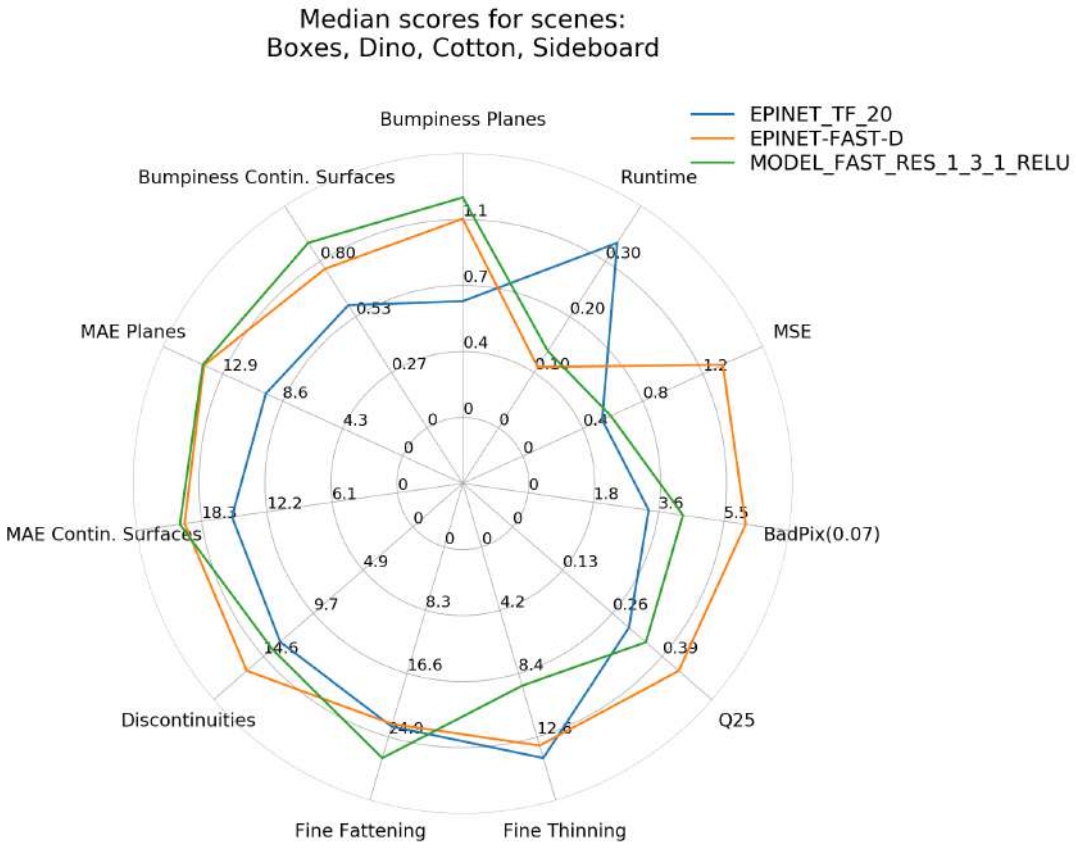


Figura 101 – Métricas de comparação entre as arquiteturas com melhores resultados. Fonte: De autoria própria, gerado no software 4D Light Field Benchmark.

- B2 descrição no Apêndice A - seção A.0.4.

A Figura 104 mostra o gráfico comparativo entre os modelos. Convém ressaltar que os valores ruins nas métricas objetivas, ocorre em virtude do ruído gerado na etapa de decodificação. Apesar desses valores desfavoráveis, o modelo MODEL_B1 possui potencial para resolver a questão ruído.

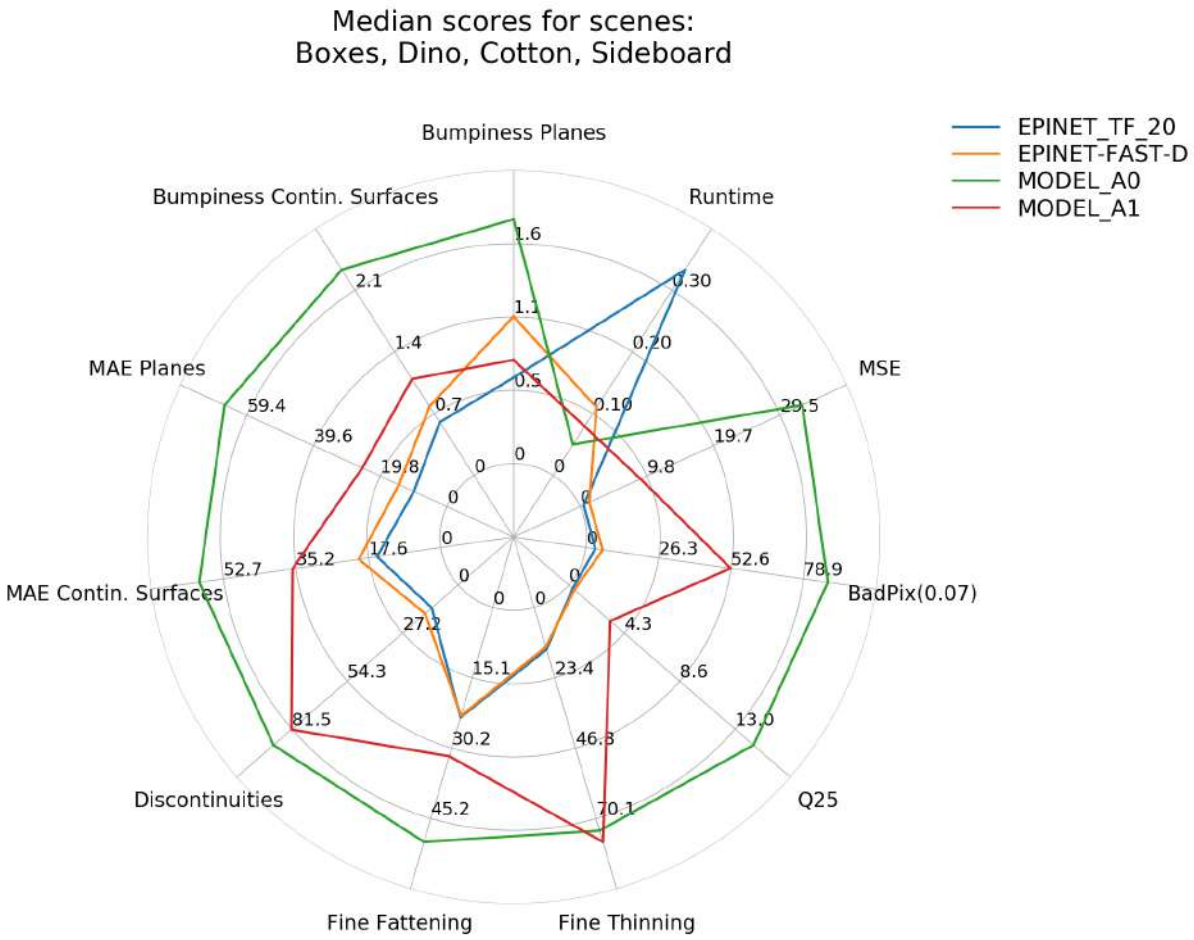


Figura 102 – Comparação entre saída e valores obtidos com MODEL_A0 e MODEL_A1. Fonte: De autoria própria, gerado no software 4D Light Field Benchmark.

7.5.3 Discussão sobre os resultados

Essa seção mostrou que a U-EPINET MODEL B possui potencial para resolver o problema de tempo de processamento. Na Figura 105 observa-se que a qualidade do mapa de disparidade é próxima da EPINET e da EPINET-FAST. O motivo das métricas se apresentarem desfavoráveis ao modelo é o ruído gerado na forma de tratamento das *skip connections*. O ajuste correto desse parâmetro deve melhorar a relação de ruído e elevar a qualidade do modelo.

7.6 Conclusão

Este capítulo apresentou dois modelos distintos para a geração de mapas de disparidade, ambos possuem potencial para aplicações que buscam balancear o tempo de processamento versus a qualidade do mapa. Um dos grandes potenciais é o modelo MODEL_B1 que possui uma boa granularidade no mapa de disparidade, devendo ser tratada a questão ruído para que o mesmo melhore seu desempenho nas métricas quantitativas.

Tabela 9 – Comparação entre os tempos de processamento da MODEL_A0 e MODEL_A1 em **segundos**.

Algoritmo	**MODEL A0**		**MODEL A1**	
Cena	**MSE**	**runtime**	**MSE**	**runtime**
backgammon	50,4538	0,0453	28,1674	0,0767
boxes	59,0208	2,3012	15,2162	3,2076
cotton	33,3946	0,0488	5,9795	0,0801
dino	14,8461	0,0500	3,1895	0,0719
dots	20,9963	0,0446	31,2299	0,0703
pyramids	6,6916	0,0442	1,2494	0,0720
sideboard	32,1372	0,0459	11,8280	0,0764
stripes	59,3751	0,0444	17,6984	0,0681

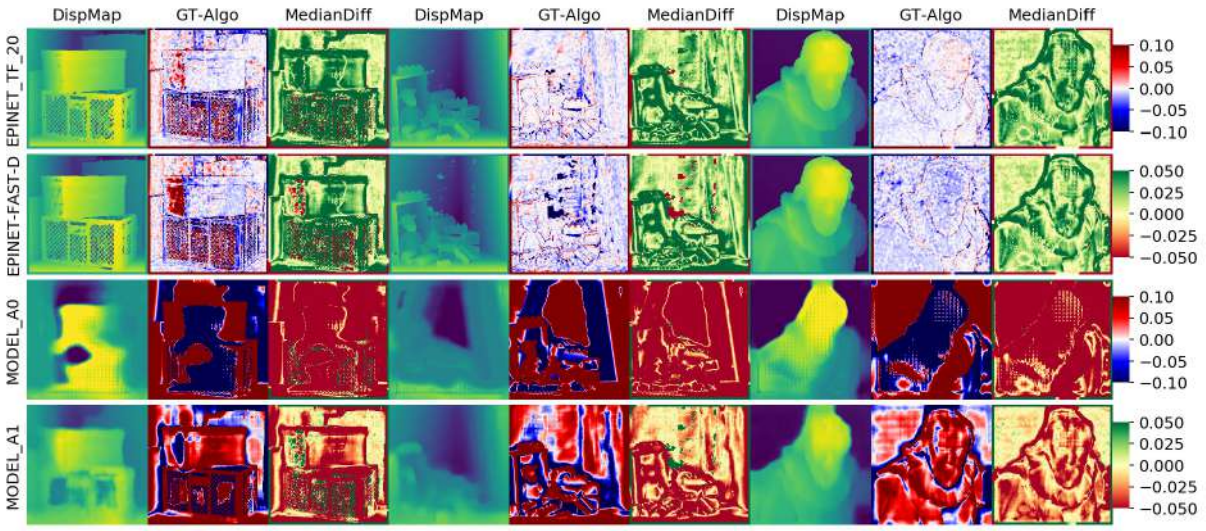


Figura 103 – Comparação entre saída e valores obtidos em MODEL_A0 versus MODEL_A1.
Fonte: De autoria própria, gerado no software 4D Light Field Benchmark.

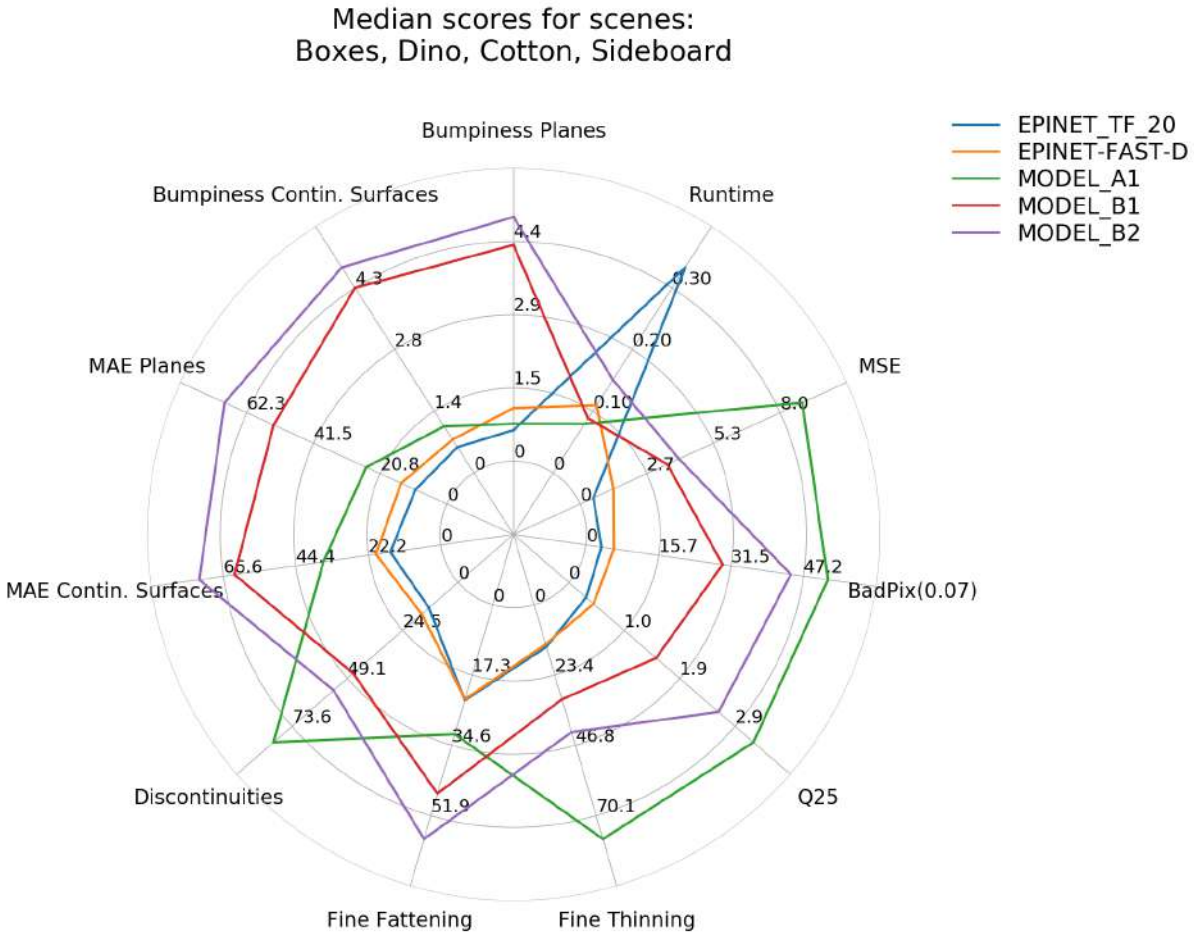


Figura 104 – Comparação entre saída e valores obtidos em MODEL_B1 e MODEL_B2. Fonte: De autoria própria, gerado no software 4D Light Field Benchmark.

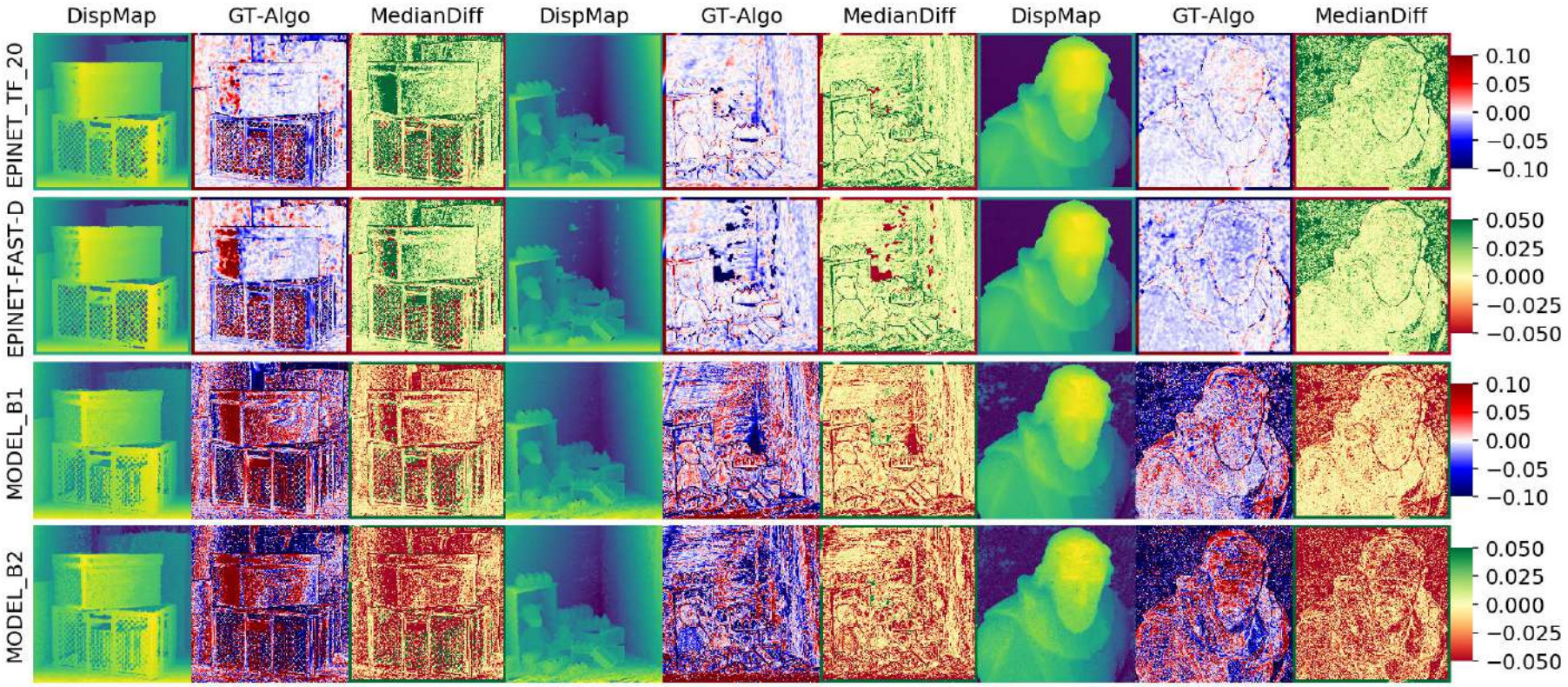


Figura 105 – Comparação entre saída e valores obtidos em MODEL_B1 versus MODEL_A1. Fonte: De autoria própria, gerado no software 4D Light Field Benchmark.

8 CONCLUSÃO

A área de ***Light Field*** se encontra em franca expansão e aberta em vários setores de desenvolvimento e pesquisa. Existem muitos trabalhos que se concentram no estudo de formas de representação, codificação e transmissão. Essa tese abordou o problema relacionado à extração de informação 3D, através da construção de mapas de profundidade, inferidas de imagens LF. Os procedimentos aqui expostos esperam simplificar o uso de informações 3D embutidas em imagens *light field*, permitindo aos pesquisadores desenvolverem estudos sem a necessidade de acessar e trabalhar diretamente na geometria óptica envolvida na geração das vistas, assim como ocorre hoje em imagens monoculares e estéreos, onde pesquisadores de processamento de imagens e visão computacional podem abstrair as informações do hardware usado na captura e operar diretamente nas imagens.

O trabalho se concentrou em três abordagens complementares:

- Estudo e **construção de um dataset** de imagens reais usando a câmera Lytro Illum®;
- Criação da **EPINET-FAST**, baseada no estudo e modificação da rede EPINET, rede de referência na área de LF para extração de mapas de profundidade;
- Proposta da **U-EPINET**, rede *u-shaped* com entrada multifluxo para extração de mapas de profundidade.

A etapa de construção do dataset evidenciou algumas limitações do hardware usado para aquisição de imagens. Um do maiores limitantes é o efeito caixa-preta[1], onde se tem acesso aos dados de entrada e aos dados de saída, mas se desconhece o funcionamento do processamento envolvido. Isso é bastante evidente nos mapas de profundidade fornecidos pelos softwares da extinta empresa Lytro ®, que fornece apenas mapas de profundidade relativos entre os objetos para fins de re-focagem. Esse processamento aparenta aplicar uma espécie de alargamento de contraste no mapa profundidade gerado, além de outros processamentos citados ao longo do texto, o que

[1] black-box effect

dificulta a extração de informação de distâncias absolutas. Mas apesar desses entraves, esse trabalho encontrou características de abertura, valores de ISO e velocidade do obturador que permitem a geração de mapas de profundidades bem definidos através de métodos geométricos externos aos softwares fornecidos com o hardware. **A construção desse dataset com distâncias absolutas bem definidas, somadas à definição desses parâmetros contribuem para projetos vindouros que trabalhem com dados reais**.

Já a **proposta da EPINET-FAST** demonstra que *subabertures images* com variações nos eixos de $45°$ e $135°$ fornecem maior diversidade de informações ângulares do que o uso de imagens apenas nos eixos vertical e horizontal. Isso pode ser explicado pelo fato que os eixos a $0°$ e $90°$, em relação a vista central, apresentam maiores concentração de disparidades apenas mesmo sentido do eixo. Por exemplo, vistas que estão no eixo horizontal tende a apresentar maior concentração de disparidades no eixo horizontal, e poucas no sentido vertical, e vistas que estão no eixo vertical ($90°$) tendem a apresentar maior concentração de disparidades no eixo vertical, e poucas no sentido horizontal. Isso se dá exatamente pela variação angular capturada por essas pilhas de vistas. O trabalho demostra que se pode abrir mão desses dois fluxos de entradas para se conseguir maior velocidade de processamento. Isso pode gerar degradações no mapa de profundidade, que podem ser corrigidas através de processamento de imagens ordinários como filtragens, operações morfológicas, *Markov Random Field*, etc. ou com o uso de mais camadas convolucionais. O problema de usar mais camadas convolucionais é o tempo de processamento voltar a se aproximar da rede EPINET original. Desta forma, deve-se procurar um balanceamento entre o objetivo a ser alcançado e a velocidade de processamento necessária ao se usar essa arquitetura. Com o uso do *backbone* ResNet, esse balanceamento foi encontrado e indica uma possibilidade de exploração em estudos futuros.

Por fim, foi **apresentada a rede U-EPINET**, que usa o mesmo conceito de quatro entradas multifluxo, mas com uma estrutura em formato de U (*u-shaped*). Essa abordagem se mostra bastante promissora, pois o próprio formato de construção da rede apresenta um processamento mais rápido sem a necessidade de acréscimo de muitas camadas convolucionais. Uma das contribuições dessa abordagem é o uso de *skip-conections* no formato usado pela rede LinkNet e não a proposta de concatenação de camadas conforme o usado na rede U-Net. Essa rede apresentou bom desempenho em algumas métricas em relação a EPINET, mas com o tempo de processamento menor que a da EPINET-FAST. O trabalho mostra que não é interessante usar todos níveis da U-Net original, pois se perde muitos detalhes, ao mesmo tempo é importante testar variações nos parâmetros utilizados para uma melhor eficiência na construção do mapa de profundidade. Outra inovação foi o uso de *skip-connections* convoluídas para buscar a correta forma de agregar a saída de cada ramo da entrada multifluxo.

A abordagem **U-EPINET** se mostra bem promissora, e com vários aspectos a serem explorados.

Desta forma, pode-se concluir que a tese apresenta dois caminhos a serem seguidos, com especial destaque para a U-EPINET e para a EPINET-FAST com *backbone* ResNet, que possuem muitos aspectos a serem explorados e potencial de uso em tempo real. Para fins de estudo, o código usado está disponível em `https://github.com/MFerrugem/`.

8.1 Trabalhos futuros

Uma das limitações desse trabalho e de outras propostas é o uso das mesmas SAI's, independente das cenas avaliadas. Em outros trabalhos são usadas outras SAI's na construção do volume de entrada, mas que se mantém fixas independente da cena capturada. Um ponto a ser explorado é a **construção de um algoritmo para a seleção dinâmica das vistas** que use informações angulares na construção do volume de entrada, pois cada cena capturada possui características únicas que impactam de formas diversas na disparidade, e dessa forma não é interessante usar sempre as mesmas vistas.

Outro ponto a ser explorado é a **busca de uma configuração de hiperparâmetros visando otimizar o funcionamento da U-EPINET**. Variações na quantidade e nas dimensões dos filtros convolucionais, e o uso de *dropout* e *batch normalization* ainda estão em aberto no uso da U-EPINET.

Por fim, pode-se sugerir a construção de um U-EPINET 3D, onde as SAI's são tratadas como volumes 3D em vez de sequências de vistas.

Ainda existe muita coisa a ser feita, mas permitir o fácil manuseio de imagens LF passa por mais propostas como aqui apresentada. Apesar de não propôr uma solução definitiva para o problema de extração de profundidade em LFs, essa tese apresenta para soluções potenciais bem estabelecidas e aponta para novos trabalhos a serem desenvolvidos.

REFERÊNCIAS

ADELSON, E. H.; BERGEN, J. R. The Plenoptic Function and the Elements of Early Vision. In: COMPUTATIONAL MODELS OF VISUAL PROCESSING, 1991. **Anais...** MIT Press, 1991. p.3–20.

Adelson, E. H.; Wang, J. Y. A. Single lens stereo with a plenoptic camera. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, [S.l.], v.14, n.2, p.99–106, 1992.

AGGARWAL, C. C. **Neural Networks and Deep Learning**: A Textbook. 1st.ed. [S.l.]: Springer Publishing Company, Incorporated, 2018.

AI, W.; XIANG, S.; YU, L. Robust depth estimation for multi-occlusion in light-field images. **Opt. Express**, [S.l.], v.27, n.17, p.24793–24807, Aug 2019.

ANG, T. **Digital Photography an Introduction**: Fully Update 5th Edition. New York, USA: DK Publishing, 2018.

BADRINARAYANAN, V.; KENDALL, A.; CIPOLLA, R. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, [S.l.], v.39, n.12, p.2481–2495, 2017.

Bajpayee, A.; Techet, A. H.; Singh, H. Real-Time Light Field Processing for Autonomous Robotics. In: IEEE/RSJ INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS (IROS), 2018., 2018. **Anais...** [S.l.: s.n.], 2018. p.4218–4225.

BISHOP, C. **Pattern Recognition and Machine Learning**. 1.ed. New York, USA: Springer-Verlag, 2006. (Information Science and Statistics).

BLACK, B. **DSLR Photography for Beginners**: The Right Way of Learning Digital SLR Photography. [S.l.]: eBookIt.com, 2015.

BOK, Y.; JEON, H.-G.; KWEON, I. S. Geometric Calibration of Micro-Lens-Based Light Field Cameras Using Line Features. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, [S.l.], v.39, n.2, p.287–300, 2017.

BROXTON, M. et al. A Low Cost Multi-Camera Array for Panoramic Light Field Video Capture. In: SIGGRAPH ASIA 2019 POSTERS, 2019, New York, NY, USA. **Anais...** Association for Computing Machinery, 2019. (SA '19).

BROXTON, M. et al. Immersive Light Field Video with a Layered Mesh Representation. **ACM Trans. Graph.**, New York, NY, USA, v.39, n.4, July 2020.

CANTRELL, K. J.; MILLER, C. D.; MORATO, C. Practical Depth Estimation with Image Segmentation and Serial U-Nets. In: INTERNATIONAL CONFERENCE ON VEHICLE TECHNOLOGY AND INTELLIGENT TRANSPORT SYSTEMS (VEHITS), 6., 2020. **Proceedings...** SCITEPRESS – Science and Technology Publications, 2020. p.406–414.

CHAURASIA, A.; CULURCIELLO, E. LinkNet: Exploiting encoder representations for efficient semantic segmentation. In: IEEE VISUAL COMMUNICATIONS AND IMAGE PROCESSING (VCIP), 2017., 2017. **Anais...** IEEE, 2017.

CHEN, J.; ZHANG, S.; LIN, Y. Attention-based Multi-Level Fusion Network for Light Field Depth Estimation. **Proceedings of the AAAI Conference on Artificial Intelligence**, [S.l.], v.35, n.2, p.1009–1017, May 2021.

CORRELL, R. **Digital SLR Photography All-in-One For Dummies**. 4th.ed. Hoboken, New Jersey: John Wiley & Sons, 2021.

DANSEREAU, D. **Light Field Toolbox for MATLAB**. `https://dgd.vision/Tools/LFToolbox/`, acessado: 20.08.2021.

DAVIES, E. R. **Computer Vision**:Principles, Algorithms, Applications, Learning. 5.ed. Londres, UK: Academic Press, 2017.

DAVIS, A.; LEVOY, M.; DURAND, F. Unstructured Light Fields. **Computer Graphics Forum**, [S.l.], v.31, n.2pt1, p.305–314, 2012.

de Faria, S. M. M. et al. Light Field Image Dataset of Skin Lesions. In: ANNUAL INTERNATIONAL CONFERENCE OF THE IEEE ENGINEERING IN MEDICINE AND BIOLOGY SOCIETY (EMBC), 2019., 2019. **Anais...** [S.l.: s.n.], 2019. p.3905–3908.

DEY, S. et al. **SigNet**: Convolutional Siamese Network for Writer Independent Offline Signature Verification.

Faraday, M. Experimental Researches in Electricity. **Philosophical Magazine**, [S.l.], v.XXVIII, n.188, p.447–452, 1846.

Flynn, J. et al. DeepView: View Synthesis With Learned Gradient Descent. In: IEEE/CVF CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR), 2019., 2019. **Anais...** [S.l.: s.n.], 2019. p.2362–2371.

GENG, J. Structured-light 3D surface imaging: a tutorial. **Adv. Opt. Photon.**, [S.l.], v.3, n.2, p.128–160, Jun 2011.

GEORGIEV, T.; YU, Z.; LUMSDAINE, A.; GOMA, S. Lytro camera technology: theory, algorithms, performance analysis. In: MULTIMEDIA CONTENT AND MOBILE DEVICES, 2013. **Anais...** SPIE, 2013. v.8667, p.86671J.

GERSHUN, A. The Light Field. **Journal of Mathematics and Physics**, [S.l.], v.18, n.1-4, p.51–151, 1939.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [S.l.]: MIT Press, 2016. http://www.deeplearningbook.org.

GORTLER, S. J.; GRZESZCZUK, R.; SZELISKI, R.; COHEN, M. F. The Lumigraph. In: ANNUAL CONFERENCE ON COMPUTER GRAPHICS AND INTERACTIVE TECHNIQUES, 23., 1996, New York, NY, USA. **Proceedings...** Association for Computing Machinery, 1996. p.43–54. (SIGGRAPH '96).

GRUNNET-JEPSEN, A. et al. **Projectors for Intel ® RealSense ™ Depth Cameras D4xx**. [S.l.]: Intel Corporation ©, 2020. Standard. (Revision 1.0).

GUO, B.; WEN, J.; HAN, Y. Deep Material Recognition in Light-Fields via Disentanglement of Spatial and Angular Information. In: COMPUTER VISION – ECCV 2020, 2020, Cham. **Anais...** Springer International Publishing, 2020. p.664–679.

HAHNE, C. **The Standard Plenoptic Camera**: Applications of a Geometrical Light Field Model. 2016. Tese (Doutorado em Ciência da Computação) — Univ. of Bedfordshire.

HAHNE, C. et al. Refocusing distance of a standard plenoptic camera. **Opt. Express**, [S.l.], v.24, n.19, p.21521–21540, Sep 2016.

HAN, L.; HUANG, X.; SHI, Z.; ZHENG, S. Depth Estimation from Light Field Geometry Using Convolutional Neural Networks. **Sensors**, [S.l.], v.21, n.18, p.6061, Sep 2021.

HAPKE, H.; NELSON, C. **Building Machine Learning Pipelines**. [S.l.]: O'Reilly Media, 2020.

HARTLEY, R.; ZISSERMAN, A. **Multiple View Geometry in Computer Vision**. 2.ed. [S.l.]: Cambridge University Press, 2004.

HAYKIN, S. S. **Neural Networks**: A Comprehensive Foundation. 2nd.ed. Upper Saddle River, NJ: Prentice Hall PTR, 1998.

HAYKIN, S. S. **Neural networks and learning machines**. 3.ed. Upper Saddle River, NJ: Pearson Education, 2009.

HE, K.; ZHANG, X.; REN, S.; SUN, J. Deep Residual Learning for Image Recognition. In: IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR), 2016., 2016. **Anais...** [S.l.: s.n.], 2016. p.770–778.

HEBER, S.; YU, W.; POCK, T. Neural EPI-Volume Networks for Shape from Light Field. In: IEEE INTERNATIONAL CONFERENCE ON COMPUTER VISION (ICCV), 2017., 2017. **Anais...** [S.l.: s.n.], 2017. p.2271–2279.

HECHT, J. **Under the Hood of Luminar's Long-Reach Lidar**. Disponível em: <https://spectrum.ieee.org/cars-that-think/transportation/self-driving/under-the-hood-of-luminars-long-reach-lidar>. Acesso em: 23/06/2021.

HEDGECOE, J. **The Book of Photography**. London ; New York: DK Publishing, 2005.

IAKUBOVSKII, P. **Segmentation Models**. [S.l.]: GitHub, 2019. `https://github.com/qubvel/segmentation_models`.

IKEUCHI, K. E. **Computer Vision**: A Reference Guide. New York, USA: Springer USA, 2014.

ILG, E. et al. FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks. In: IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR), 2017., 2017. **Anais...** [S.l.: s.n.], 2017. p.1647–1655.

ISO; IEC; JTC. **JPEG Pleno Light Field Coding Common Test Conditions V3.3**. Lausanne, Switzerland: International Organization for Standardization and Joint Photographic Experts Group and International Electrotechnical Commission, 2019. Standard. (ISO/IEC JTC 1/SC 29/WG1N84025:2019).

JEON, H.-G. et al. Accurate depth map estimation from a lenslet light field camera. In: IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR), 2015., 2015. **Anais...** [S.l.: s.n.], 2015. p.1547–1555.

JOHNSON JR., C. S. **Science for the Curious Photographer**: An Introduction to the Science of Photography. 2.ed. New York, NY: Taylor & Francis, 2017.

KEMSARAM, N.; DAS, A.; DUBBELMAN, G. A Stereo Perception Framework for Autonomous Vehicles. In: IEEE 91ST VEHICULAR TECHNOLOGY CONFERENCE (VTC2020-SPRING), 2020., 2020. **Anais...** [S.l.: s.n.], 2020. p.1–6.

KHAN, N.; KIM, M. H.; TOMPKIN, J. Edge-aware Bidirectional Diffusion for Dense Depth Estimation from Light Fields. In: BRITISH MACHINE VISION CONFERENCE (BMVC), 2021. **Anais...** [S.l.: s.n.], 2021.

KIM, C. et al. Scene Reconstruction from High Spatio-Angular Resolution Light Fields. **ACM Trans. Graph.**, New York, NY, USA, v.32, n.4, July 2013.

KIM, M. et al. **Mobile Terminal and control method for the mobile terminal**. n.US010135963B2.

KOVÁCS, Z. L. **Redes Neurais Artificiais**: Fundamentos e Aplicações. fourth.ed. São Paulo, Brasil: Editora Livraria da Física, 2006.

KROHN, J.; BEYLEVELD, G.; BASSENS, A. **Deep Learning Illustrated**: A Visual, Interactive Guide to Artificial Intelligence. 1.ed. [S.l.]: Addison-Wesley Professional, 2020. (Addison-Wesley Data and Analytics Series).

LEVOY, M.; HANRAHAN, P. Light Field Rendering. In: ANNUAL CONFERENCE ON COMPUTER GRAPHICS AND INTERACTIVE TECHNIQUES, 23., 1996, New York, NY, USA. **Proceedings...** Association for Computing Machinery, 1996. p.31–42. (SIGGRAPH '96).

LI, K.; ZHANG, J.; GAO, J.; QI, M. Self-Supervised Light Field Depth Estimation Using Epipolar Plane Images. In: INTERNATIONAL CONFERENCE ON 3D VISION (3DV), 2021., 2021, Los Alamitos, CA, USA. **Anais...** IEEE Computer Society, 2021. p.731–740.

LIN, L. et al. Unsupervised learning of light field depth estimation with spatial and angular consistencies. **Neurocomputing**, [S.l.], v.501, p.113–122, 2022.

Lippmann, G. Épreuves réversibles donnant la sensation du relief. **J. Phys. Theor. Appl.**, [S.l.], v.7, n.1, p.821–825, 1908.

LIU, L.; JIN, X.; DAI, Q. Image Formation Analysis and Light Field Information Reconstruction for Plenoptic Camera 2.0. In: ADVANCES IN MULTIMEDIA INFORMATION PROCESSING - PCM 2017 - 18TH PACIFIC-RIM CONFERENCE ON MULTIMEDIA, HARBIN, CHINA, SEPTEMBER 28-29, 2017, REVISED SELECTED PAPERS, PART I, 2017. **Anais...** Springer, 2017. p.609–618. (Lecture Notes in Computer Science, v.10735).

LIU, X.; FU, D.; WU, C.; SI, Z. The Depth Estimation Method Based on Double-Cues Fusion for Light Field Images. In: INTERNATIONAL CONFERENCE ON MODELLING, IDENTIFICATION AND CONTROL (ICMIC2019), 11., 2020, Singapore. **Proceedings...** Springer Singapore, 2020. p.719–726.

Lumsdaine, A.; Georgiev, T. The focused plenoptic camera. In: IEEE INTERNATIONAL CONFERENCE ON COMPUTATIONAL PHOTOGRAPHY (ICCP), 2009., 2009. **Anais...** [S.l.: s.n.], 2009. p.1–8.

LUO, W.; SCHWING, A. G.; URTASUN, R. Efficient Deep Learning for Stereo Matching. In: IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR), 2016., 2016. **Anais...** [S.l.: s.n.], 2016. p.5695–5703.

LYTRO. **Lytro Illum User Manual**. 2.ed. [S.l.: s.n.], 2015.

LYTRO. **Lytro Power Tools**. 1.ed. [S.l.: s.n.], 2015.

MAÎTRE, H. **From Photon to Pixel- The Digital Camera Handbook**. Hoboken, USA: John Wiley & Sons, Ltd, 2017.

MCHUGH, S. T. **Understanding photography**. San Francisco: No Starch Press, 2019.

MICHELUCCI, U. Autoencoders. In: **Applied Deep Learning with TensorFlow 2**: Learn to Implement Advanced Deep Learning Techniques with Python. Berkeley, CA: Apress, 2022. p.257–283.

NG, R. **Digital Light Field Photography**. 2006. PhD thesis — Stanford University, CA, USA.

OROZCO, R.; LOSCOS, C.; MARTIN, I.; ARTUSI, A. Chapter 3 - HDR Multiview Image Sequence Generation: Toward 3D HDR Video. In: CHALMERS, A.; CAMPISI, P.; SHIRLEY, P.; OLAIZOLA, I. G. (Ed.). **High Dynamic Range Video**. [S.l.]: Academic Press, 2017. p.61–86.

OVERBECK, R. S. et al. A System for Acquiring, Processing, and Rendering Panoramic Light Field Stills for Virtual Reality. **ACM Trans. Graph.**, New York, NY, USA, v.37, n.6, Dec. 2018.

RANGAPPA, S.; MATHARU, R.; PETZING, J.; KINNELL, P. Establishing the performance of low-cost Lytro cameras for 3D coordinate geometry measurements. **Machine Vision and Applications**, [S.l.], v.30, n.4, p.615–627, 2019.

RAY, S. F. **Scientific Photography and Applied Imaging**. Burlington, MA: Taylor & Francis, 1999.

RENTERIA-VIDALES, O. I.; CUEVAS-TELLO, J. C.; REYES-FIGUEROA, A.; RIVERA, M. ModuleNet: A Convolutional Neural Network for Stereo Vision. In: PATTERN RECOGNITION, 2020, Cham. **Anais...** Springer International Publishing, 2020. p.219–228.

ŘEŘÁBEK, M.; EBRAHIMI, T. New Light Field Image Dataset. In: INTERNATIONAL CONFERENCE ON QUALITY OF MULTIMEDIA EXPERIENCE (QOMEX), 8., 2016. **Anais...** [S.l.: s.n.], 2016.

RONNEBERGER, O.; FISCHER, P.; BROX, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. In: MEDICAL IMAGE COMPUTING AND COMPUTER-ASSISTED INTERVENTION – MICCAI 2015, 2015, Cham. **Anais...** Springer International Publishing, 2015. p.234–241.

RUSSELL, S. J.; NORVIG, P. **Artificial Intelligence**: a modern approach. 4.ed. Hoboken, USA: Pearson, 2020. (Pearson series in artificial intelligence).

SCHAMBACH, M. **Plenpy - A plenoptic processing library for Python**. `https://iiit-public.gitlab.io/plenpy/`, acessado: 20.08.2021.

SCHAMBACH, M.; PUENTE LEóN, F. Microlens array grid estimation, light field decoding, and calibration. **IEEE Transactions on Computational Imaging**, [S.l.], v.6, p.591–603, 2020.

SCHMIDHUBER, J. Deep learning in neural networks: An overview. **Neural Networks**, [S.l.], v.61, p.85–117, 2015.

SHAHNEWAZ, A.; PANDEY, A. K. Color and Depth Sensing Sensor Technologies for Robotics and Machine Vision. In: SERGIYENKO, O.; FLORES-FUENTES, W.; MERCORELLI, P. (Ed.). **Machine Vision and Navigation**. Cham: Springer International Publishing, 2020. p.59–86.

SHAPIRO, L. G.; STOCKMAN, G. C. **Computer Vision**. 1.ed. [S.l.]: Prentice Hall, 2001.

SHI, J.; JIANG, X.; GUILLEMOT, C. A Framework for Learning Depth From a Flexible Subset of Dense and Sparse Light Field Views. **IEEE Transactions on Image Processing**, [S.l.], v.28, n.12, p.5867–5880, 2019.

SHIN, C. et al. EPINET: A Fully-Convolutional Neural Network Using Epipolar Geometry for Depth from Light Field Images. In: IEEE/CVF CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, 2018., 2018. **Anais...** [S.l.: s.n.], 2018. p.4748–4757.

SILVA, A. **Light field processor**: a lytro illum imaging application. 2016. Dissertação (Mestrado em Ciência da Computação) — Instituto Universal de Lisboa.

SOLOMON, C.; BRECKON, T. **Fundamentals of Digital Image Processing**: A Practical Approach with Examples in Matlab. [S.l.]: Wiley-Blackwell, 2010.

SZELISKI, R. **Computer Vision**:Algorithms and Applications. 2.ed. [S.l.]: Springer Cham, 2022. (Texts in Computer Science).

TABORA, V. **A Concise Look At Camera Aperture**. Disponível em: <https://medium.com/hd-pro/a-concise-look-at-camera-aperture-37990accc6f4>. Acesso em: 2021-02-15.

TAYLOR, D. **Digital Photography Complete Course**: Learn Everything You Need to Know in 20 Weeks. Reissue edition.ed. New York, New York: DK, 2015.

TROTTNOW, J. et al. The Potential of Light Fields in Media Productions. In: SIGGRAPH Asia 2019 Technical Briefs, 2019, New York, NY, USA. **Anais...** Association for Computing Machinery, 2019. p.71–74. (SA '19).

WANG, T.-C.; EFROS, A.; RAMAMOORTHI, R. Occlusion-aware depth estimation using light-field cameras. In: IEEE INTERNATIONAL CONFERENCE ON COMPUTER VISION (ICCV), 2015. **Proceedings...** [S.l.: s.n.], 2015.

WANG, T.-C. et al. A 4D light-field dataset and CNN architectures for material recognition. In: EUROPEAN CONFERENCE ON COMPUTER VISION (ECCV), 2016. **Proceedings...** [S.l.: s.n.], 2016.

WILBURN, B. et al. High Performance Imaging Using Large Camera Arrays. **ACM Trans. Graph.**, New York, NY, USA, v.24, n.3, p.765–776, July 2005.

Wu, G. et al. Light Field Image Processing: An Overview. **IEEE Journal of Selected Topics in Signal Processing**, [S.l.], v.11, n.7, p.926–954, 2017.

YALÇIN, O. G. Autoencoders. In: **Applied Neural Networks with TensorFlow 2**: API Oriented Deep Learning with Python. Berkeley, CA: Apress, 2021. p.237–257.

YU, Z. et al. Line Assisted Light Field Triangulation and Stereo Matching. In: IEEE INTERNATIONAL CONFERENCE ON COMPUTER VISION, 2013., 2013. **Anais...** [S.l.: s.n.], 2013. p.2792–2799.

ZANG, S. et al. The Impact of Adverse Weather Conditions on Autonomous Vehicles: How Rain, Snow, Fog, and Hail Affect the Performance of a Self-Driving Car. **IEEE Vehicular Technology Magazine**, [S.l.], v.14, n.2, p.103–111, 2019.

ŽBONTAR, J.; LECUN, Y. Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches. **J. Mach. Learn. Res.**, [S.l.], v.17, n.1, p.2287–2318, Jan. 2016.

ZHANG, S. E. **Handbook of 3D Machine Vision**: Optical Metrology and Imaging. Boca Raton, FL: CRC Press, 2013.

ZHANG, Y.-J. **Handbook of Image Engineering**. 1.ed. [S.l.]: Springer Singapore, 2021.

ZHOU, W. et al. Learning Depth Cues from Focal Stack for Light Field Depth Estimation. In: IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING (ICIP), 2019., 2019. **Anais...** [S.l.: s.n.], 2019. p.1074–1078.

ZHOU, Z.; CHEN, X.; JENKINS, O. C. LIT: Light-Field Inference of Transparency for Refractive Object Localization. **IEEE Robotics and Automation Letters**, [S.l.], v.5, n.3, p.4548–4555, 2020.

ZHOU, Z.; SUI, Z.; JENKINS, O. C. Plenoptic Monte Carlo Object Localization for Robot Grasping Under Layered Translucency. In: IEEE/RSJ INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS (IROS), 2018., 2018. **Anais...** [S.l.: s.n.], 2018. p.1–8.

ZHU, S. et al. On the fundamental comparison between unfocused and focused light field cameras. **Appl. Opt.**, [S.l.], v.57, n.1, p.A1–A11, Jan 2018.

Z.Zhou; Chen, X.; O.C.Jenkins. LIT: Light-field Inference of Transparency for Refractive Object Localization. **CoRR**, [S.l.], v.abs/1910.00721, 2019.

Z.Zhou; Sui, Z.; C.Jenkins, O. Plenoptic Monte Carlo Object Localization for Robot Grasping Under Layered Translucency. In: IEEE/RSJ INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS (IROS), 2018., 2018. **Anais...** [S.l.: s.n.], 2018. p.1–8.

Apêndices

APÊNDICE A – Modelos U-EPINET

Modelos desenvolvidos a partir da estrutura genérica apresentada na figura 89.

A.0.1 U-EPINET-MODEL-A0

A tabela 10 apresenta os hiper-parâmetros da rede. A figura 106 apresenta exemplos da saída gerada versus o *ground-truth*. A arquitetura da rede é demostrada detalhadamente na figura 90.

Tabela 10 – Tabela de dados do MODEL_A0.

Características gerais	
Otimizador	Algoritmo Adam
Learning rate	0,0001
Skip-connections	Concatenadas
Quantidade de Filtros no bloco convolucional de saída	1
Bloco de Convolução	
Convolução	Filtros 3x3
Passo (*stride*)	1x1
Normalização	*Batch normalization*
Função de ativação	ReLU
Encoder	
Quantidade de blocos de convolução	2
Método de *downsample*	MaxPool2D de tamanho 2x2
Padding	*same*
Quantidade de filtros em cada bloco convolucional de acordo com o nível crescente	16,32,64,128,256
Decoder	
Quantidade de filtros em cada bloco convolucional de acordo com o nível decrescente	128,64,32,16
Método de *upsample*	Camada de convolução transposta (Conv2DTranspose), com filtro de tamanho 4x4, passo (*stride*)=2
Padding	*same*

Figura 106 – MODEL_A0: Comparação entre saída e valores obtidos. Acima estão o ground-truth e abaixo os *depth maps* gerados pela arquitetura. Fonte: De autoria própria

A.0.2 U-EPINET-MODEL-A1

U-EPINET baseada no Modelo anterior (MODEL-A0), com variação na quantidade de filtros nos blocos do encoder de forma a balancear valores advindos da *skip-connection* com o valor *up sampled*. A tabela 11 apresenta os hiper-parâmetros da rede. A figura 107 apresenta exemplos da saída gerada versus o *ground-truth*. A arquitetura da rede é demostrada detalhadamente na figura 91.

Tabela 11 – Tabela de dados do MODEL_A1.

Características gerais	
Otimizador	Algoritmo Adam
Learning rate	0,0001
Skip-connections	Concatenadas
Quantidade de Filtros no bloco convolucional de saída	1
Bloco de Convolução	
Convolução	Filtros 3x3
Passo (*stride*)	1x1
Normalização	*Batch normalization*
Função de ativação	ReLU
Encoder	
Quantidade de blocos de convolução	2
Método de *downsample*	MaxPool2D de tamanho 2x2
Padding	*same*
Quantidade de filtros em cada bloco convolucional de acordo com o nível crescente	16,32,64,128,256
Decoder	
Quantidade de filtros em cada bloco convolucional de acordo com o nível decrescente	512,256,128,64
Método de *upsample*	Camada de convolução transposta (Conv2DTranspose), com filtro de tamanho 4x4, passo (*stride*)=2
Padding	*same*

Figura 107 – MODEL_A1: Comparação entre saída e valores obtidos. Acima estão o *ground-truth* e abaixo os *depth maps* gerados pela arquitetura. Fonte: De autoria própria

A.0.3 U-EPINET-MODEL-B1

Esse modelo é possui *skip-connections* somadas em vez de concatenadas. A tabela 12 apresenta os hiper-parâmetros da rede. A figura 108 apresenta exemplos da saída gerada versus o *ground-truth*. A arquitetura da rede é demostrada detalhadamente na figura 92.

Tabela 12 – Tabela de dados do MODEL_B1.

Características gerais	
Otimizador	Algoritmo Adam
Learning rate	0,0001
Skip-connections	Somadas
Quantidade de Filtros no bloco convolucional de saída	1
Bloco de Convolução	
Convolução	Filtros 3x3
Passo (*stride*)	1x1
Normalização	Não utilizado
Função de ativação	ReLU/Softmax
Encoder	
Quantidade de blocos de convolução	2
Método de *downsample*	MaxPool2D de tamanho 2x2
Padding	*same*
Quantidade de filtros em cada bloco convolucional de acordo com o nível crescente	64,128,256,512
Decoder	
Quantidade de filtros em cada bloco convolucional de acordo com o nível decrescente	512,256,128,64
Método de *upsample*	Camada de convolução transposta (Conv2DTranspose), com filtro de tamanho 8x8, passo (*stride*)=2
Padding	*same*

Figura 108 – Arquitetura U-EPINET MODEL_B1: Comparação entre saída e valores obtidos. Acima estão o ground-truth e abaixo os *depth maps* gerados pela arquitetura. Fonte: De autoria própria

A.0.4 U-EPINET-MODEL-B2

Esse modelo é possui *skip-connections* somadas em vez de concatenadas. A tabela 13 apresenta os hiper-parâmetros da rede. A Figura 109 apresenta exemplos da saída gerada versus o *ground-truth*. A arquitetura da rede é demostrada detalhadamente na figura 93.

Tabela 13 – Tabela de dados do MODEL_B2.

Características gerais	
Otimizador	Algoritmo Adam
Learning rate	0,0001
Skip-connections	Somadas
Quantidade de Filtros no bloco convolucional de saída	1
Bloco de Convolução	
Convolução	Filtros 3x3
Passo (*stride*)	1x1
Normalização	Não utilizado
Função de ativação	ReLU
Encoder	
Quantidade de blocos de convolução	2
Método de *downsample*	MaxPool2D de tamanho 2x2
Padding	*same*
Quantidade de filtros em cada bloco convolucional de acordo com o nível crescente	64,128,256,512
Decoder	
Quantidade de filtros em cada bloco convolucional de acordo com o nível decrescente	512,256,128,64
Método de *upsample*	Camada de convolução transposta (Conv2DTranspose), com filtro de tamanho 8x8, passo (*stride*)=2
Padding	*same*

Figura 109 – MODEL_B2: Comparação entre saída e valores obtidos. Acima estão o ground-truth e abaixo os *depth maps* gerados pela arquitetura. Fonte: De autoria própria.

APÊNDICE B – Dataset Lytro

Para garantir encontrar imagens com boas características de refocagem, o *focus bracketing* foi ajustado para 5 disparos com 10 de *depth steps*. O *focus bracketing* faz com que a câmera tire uma série de fotos cada vez que o obturador é pressionado, incrementando o foco de acordo com os *depth steps*. Isso faz com que cenas iguais possuam variação no *depth steps*. Isso ocorre com as primeiras cenas da figura 110, apesar de capturarem a mesma imagem, cada captura possui incremento em seu *depth steps* em relação a anterior.

- ISO - 80;
- Velocidade do obturador - 1/100;
- Distância focal - 48 mm, equivalente a 15 mm devido ao fator de corte (*crop*) igual a $3,19$.

Figura 110 – Primeira parte do dataset gerado. Fonte: De autoria própria.

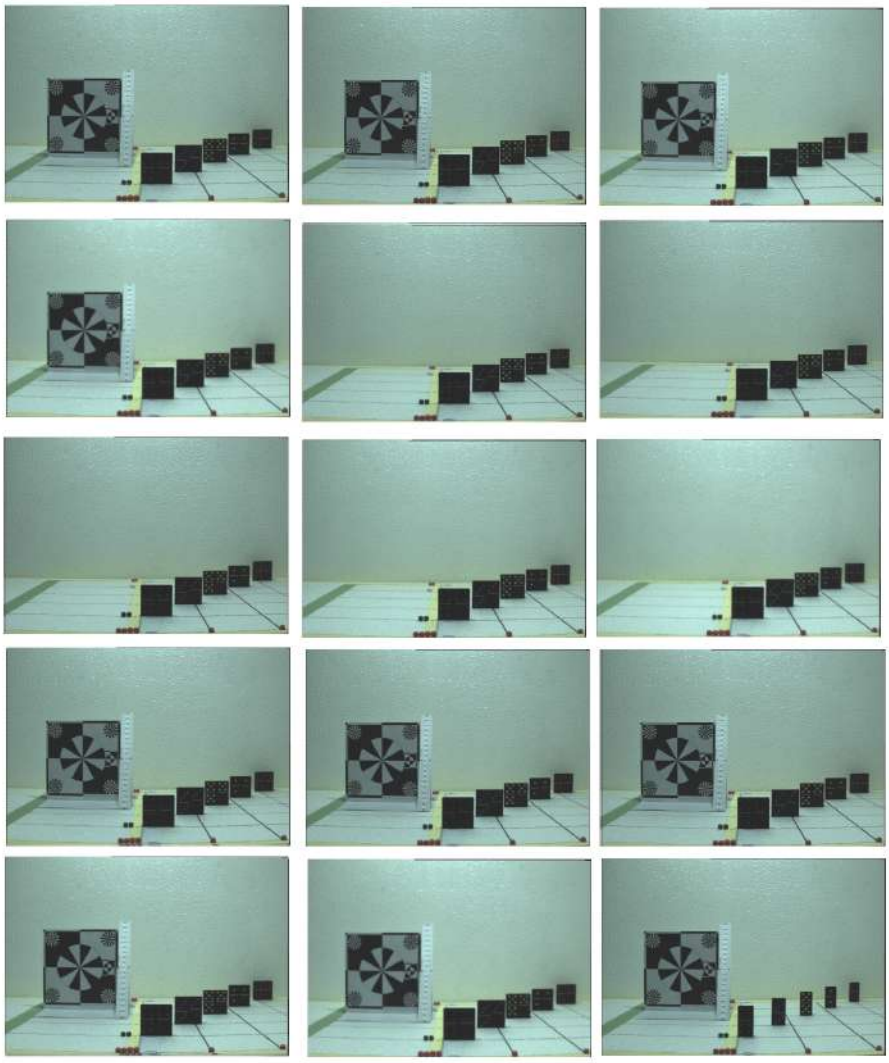


Figura 111 – Segunda parte do dataset gerado. Fonte: De autoria própria.

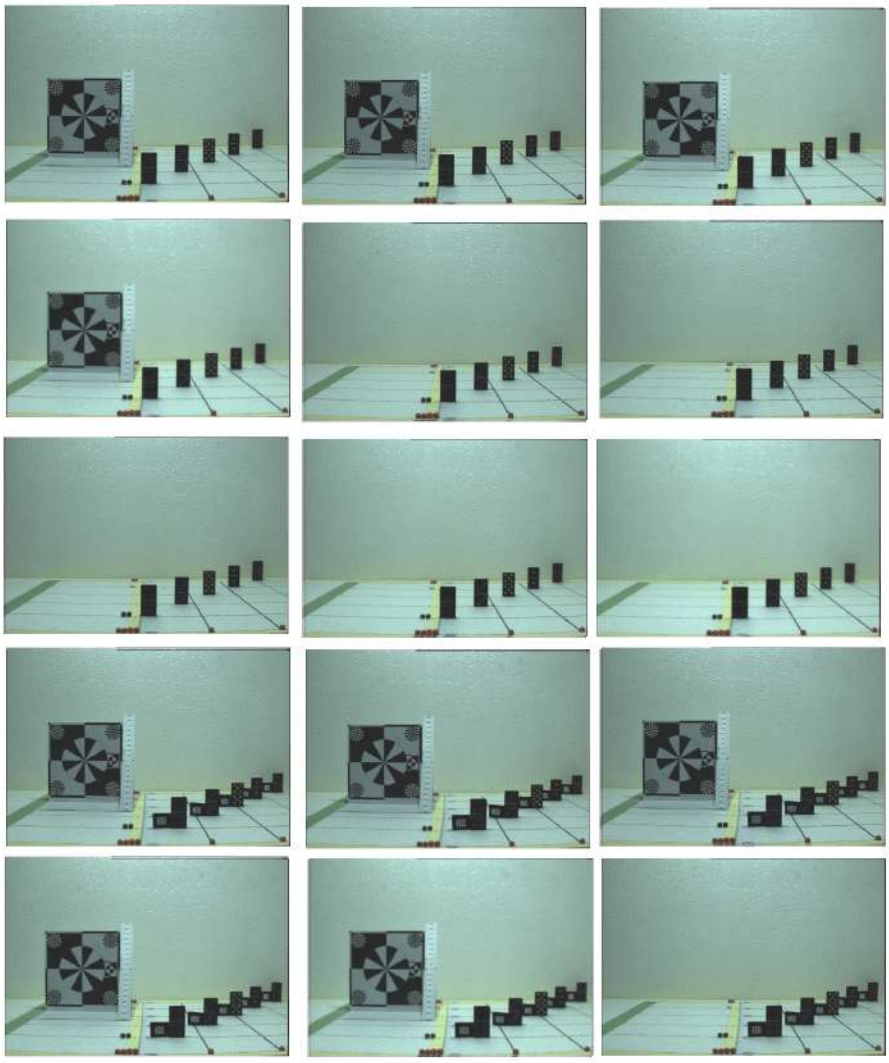


Figura 112 – Terceira parte do dataset gerado. Fonte: De autoria própria.

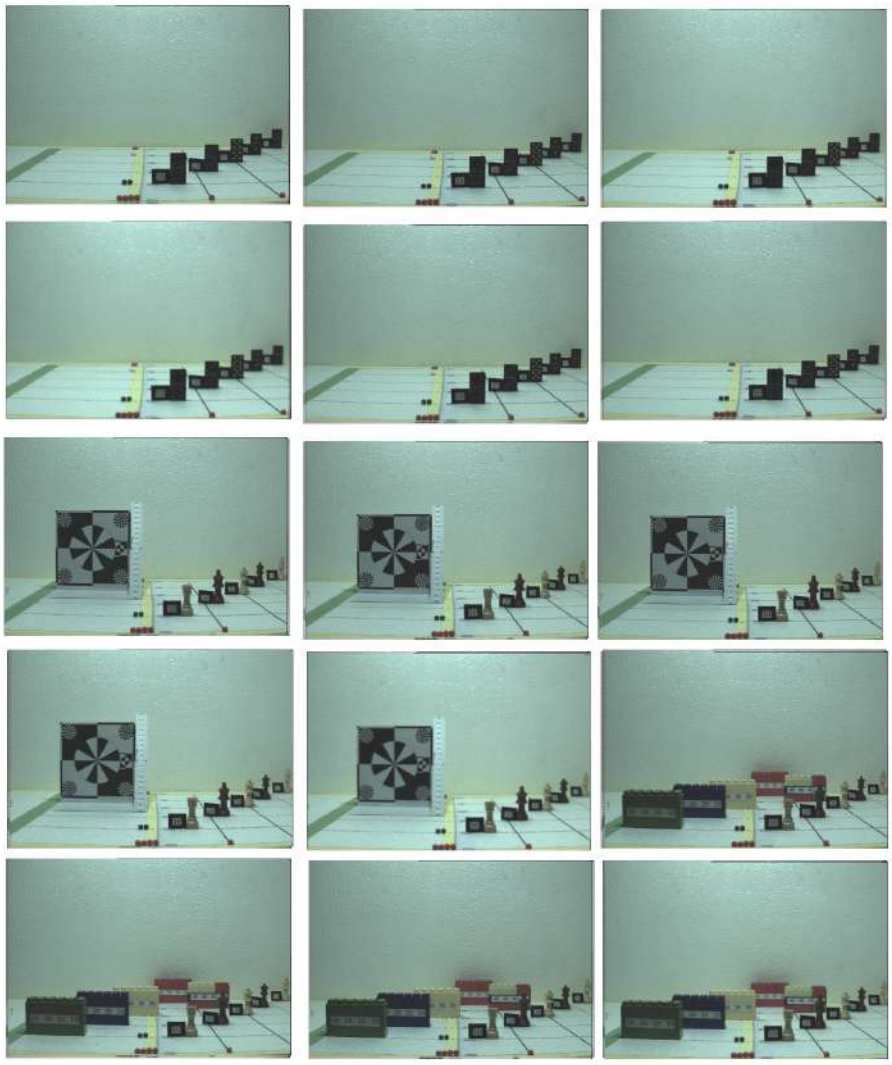


Figura 113 – Quarta parte do dataset gerado. Fonte: De autoria própria.

Figura 114 – Quinta parte do dataset gerado. Fonte: De autoria própria.

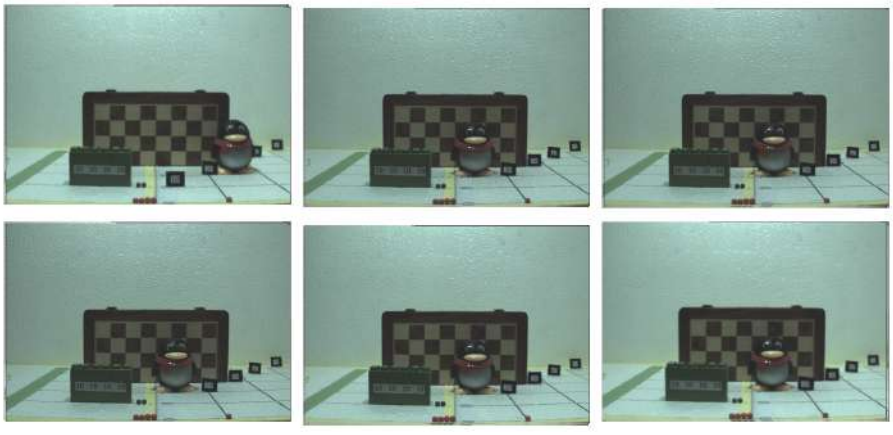


Figura 115 – Sexta parte do dataset gerado. Fonte: De autoria própria.