TRABALHO DE CONCLUSÃO DE CURSO

PRODUÇÃO DE IMAGENS ALL-IN-FOCUS BASEADO EM MAPAS DE PROFUNDIDADE

Ricardo Marra de Almeida Valladares

Brasília, julho de 2019

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASÍLIA
Faculdade de Tecnologia

TRABALHO DE CONCLUSÃO DE CURSO

PRODUÇÃO DE IMAGENS ALL-IN-FOCUS BASEADO EM MAPAS DE PROFUNDIDADE

Ricardo Marra de Almeida Valladares

Trabalho de Conclusão de Curso submetido ao Departamento de Engenharia Elétrica como requisito parcial para obtenção do grau de Engenheiro Eletricista

Banca Examinadora

Prof. Eduardo Peixoto Fernandes da Silva, Ph.D. _____
ENE/UnB
Orientador

Prof. Camilo Chang Dorea, Dr. _____
CIC/UnB
Examinador 1

Prof. Daniel Guerreiro e Silva, Dr. _____
ENE/UnB
Examinador 2

FICHA CATALOGRÁFICA

VALLADARES, RICARDO MARRA DE ALMEIDA
PRODUÇÃO DE IMAGENS ALL-IN-FOCUS BASEADO EM MAPAS DE PROFUNDIDADE [Distrito Federal] 2019.
xvi, 40 p., 210 x 297 mm (ENE/FT/UnB, Engenheiro, Engenharia Elétrica, 2019).
Trabalho de Conclusão de Curso - Universidade de Brasília, Faculdade de Tecnologia.
Departamento de Engenharia Elétrica

1. empilhamento de foco	2. pirâmide de Laplace
3. mapa de profundidade	4. Palavra chave 4
I. ENE/FT/UnB	II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

VALLADARES, R.M.A (2019). *PRODUÇÃO DE IMAGENS ALL-IN-FOCUS BASEADO EM MAPAS DE PROFUNDIDADE*. Trabalho de Conclusão de Curso, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 40 p.

CESSÃO DE DIREITOS

AUTOR: Ricardo Marra de Almeida Valladares
TÍTULO: PRODUÇÃO DE IMAGENS ALL-IN-FOCUS BASEADO EM MAPAS DE PROFUNDIDADE.
GRAU: Engenheiro Eletricista ANO: 2019

Ricardo Marra de Almeida Valladares
Depto. de Engenharia Elétrica (ENE) - FT
Universidade de Brasília (UnB)
Campus Darcy Ribeiro
CEP 70919-970 - Brasília - DF - Brasil

Dedicatória

Dedico este trabalho aos meus pais, Silvana e Sérgio, sem os quais não seria possível chegar até aqui. Dedico também aos meus amigos, por todo apoio ao longo dessa jornada.

Ricardo Marra de Almeida Valladares

Agradecimentos

Agradeço primeiramente aos meus pais, Silvana e Sérgio, por tudo que fizeram por mim ao longo dos anos. Sempre se esforçaram para me dar tudo de bom e do melhor.

Ao professor Eduardo, que me ajudou e me guiou como orientador, fazendo possível o término deste trabalho.

Agradeço aos meus colegas de curso, sem o apoio deles com certeza não teria chegado até onde cheguei. Passamos por vários altos e baixos ao longo dessa jornada, porém conseguimos completar esta viagem juntos. Vivemos histórias que para sempre lembrarei, tristes ou felizes. Obrigado principalmente àqueles mais próximos, que por muitas vezes, me carregaram como mochila para que pudesse chegar até o final.

Um agradecimento especial aos meus melhores amigos, amigos de ensino médio, que andam lado a lado comigo até hoje, Caio, Eduardo, Fernando, Pedro e Pedro. Às minhas amigas mais próximas Brenda, Isabela e Gabriela, obrigado a todos pelo apoio, conversas, choros e enfim. Obrigado até ao meu primo João Marcos. Um agradecimento repleto de carinho para minha amiga Fernanda, que me fez companhia e me apoiou durante todos estes anos de graduação.

Agradeço à Universidade de Brasília e todos os seus funcionários, por me proporcionarem uma maravilhosa experiência de universidade por todos estes anos. Agradeço também aos excelentes professores que tive ao longo dessa estrada, que sempre estiveram dispostos a ajudar os alunos, tirando dúvidas ou os motivando, sem vocês este trabalho não poderia ser realizado.

Ricardo Marra de Almeida Valladares

RESUMO

A técnica de empilhamento de foco combina diversas imagens com pouca profundidade de campo com o intuito de criar apenas uma com esta profundidade de campo ampliada. A maioria dos algoritmos já propostos considera que a entrada do algoritmo são imagens criteriosamente selecionadas. Neste trabalho, propõe-se que a entrada do algoritmo seja composta por diversas imagens com focos diferentes compondo uma pilha de focos (ou *focal stack*), gerando então uma imagem *all-in-focus* (ou completamente em foco). Propõe-se algoritmos baseados em medidores de foco e mapas de profundidade, que fazem uma seleção pós-captura das imagens que são realmente úteis para a geração da imagem *all-in-focus*. Os resultados mostram que, dado uma grande variedade de imagens com focos diferentes, os algoritmos são capazes de criar figuras *all-in-focus* de boa qualidade, podendo ser comparadas com imagens de pequena abertura de lente, que naturalmente possuem maior campo de profundidade.

Palavras-chaves: empilhamento de foco; mapa de profundidade; pirâmide de Laplace;

ABSTRACT

The focus stacking technic merges many different depth of field images with the intent of creating one final image with larger depth of field. Many focus stacking methods exists, but the majority considers the input of the algorithm is composed of carefully selected images. In this paper, we propose an input not carefully selected, but composed of various images with different focus in the focal stack, hence creating, an all-in-focus image. Based in focus measure technics and depth maps, the algorithms presets images that it thinks that are most useful to generate the all-in-focus image. The results show that the algorithms, based on a great variety of images with different focus, are capable to form an all-in-focus with good quality. These images can be compared to low aperture images, which have a naturally greater depth of field.

Key-words: focus stacking; depth maps; Laplacian pyramid;

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

Símbolos Gregos

Δ	Operador Laplaciano
Δ_m	Operador Laplaciano modificado

Siglas

DOF	*Deptf of field* (profundidade de campo)
LF	*Light-Field* (campo de luz)

1 INTRODUÇÃO

Este capítulo introduz o contexto ao qual o trabalho pertence. Além disso, apresenta-se suas principais motivações e objetivos.

1.1 CONTEXTUALIZAÇÃO

Na área da fotografia, a qualidade de uma foto está diretamente relacionada à quantidade de detalhes que ela possui, bem como sua nitidez. Um dos fatores para se obter a nitidez é o plano de foco. O plano de foco define se o objeto capturado estará ou não em foco, visto que, caso os raios de luz não conveirjam no plano de filme da imagem, isto é, caso a convergência aconteça antes ou depois, este objeto se encontrará fora de foco. Desta forma, existe apenas um plano de foco, e a nitidez do objeto diminui ou aumenta de forma proporcional com a distância dele do plano. Define-se a região onde o objeto aparenta estar nítido de profundidade de campo (*depth of field*, DOF). [1]

Fatores como a distância focal das lentes, o sensor e a distância entre o objeto e a câmera influenciam no tamanho da profundidade de campo de uma imagem, porém, para uma certa câmera e cena, se tornam fixos. Desta forma, o fator que mais influencia no tamanho da profundidade de campo é a abertura da câmera. Com todos os outros fatores fixos, a abertura da câmera determina uma profundidade de campo estreita ou larga, como pode ser visto na Figura 1.1.

(a) Abertura = f/1,4, DOF = 0,8cm (b) Abertura = f/4,0, DOF = 2,2cm (c) Abertura = f/22, DOF = 12,4cm

Figura 1.1: Influência da abertura na profundidade de campo [2].

A abertura controla a quantidade de luz que é captada pelos sensores da câmera. Quanto menos luz é captada, mais elementos da imagem ficam em foco. Um exemplo é dado pela Figura 1.1a, uma grande abertura resulta em uma pequena profundidade de campo e, a medida que se diminui a abertura, Figuras 1.1b e 1.1c, a profundidade de campo aumenta, deixando mais elementos da imagem em foco.

É verdade que a abertura define o tamanho da profundidade de campo, porém, a região em foco pode estar em qualquer lugar da imagem. Isso acontece pois, ao se tirar uma foto, é selecionado um ponto de foco na imagem, definindo onde sua profundidade de campo vai se localizar. Um

exemplo disso é visto na Figura 1.2. Na Figura 1.2a, o ponto de foco foi selecionado no canto superior esquerdo, e, portanto, os elementos daquela área ficam em foco. Já na Figura 1.2b é selecionado o canto inferior direito. Ambas as imagens foram obtidas com a mesma abertura.

(a) Ponto de foco no canto superior esquerdo. (b) Ponto de foco no canto inferior direito.

Figura 1.2: Influência do ponto de foco.

De forma geral, com exceção da técnica de *bokeh*, onde o fotógrafo busca uma profundidade de campo menor (para criar a própria estética da foto ou para deixar um objeto em perspectiva em relação ao resto da imagem), fotos com maior nitidez em toda a imagem ainda predominam. Porém, não é sempre que se consegue aumentar o tamanho da profundidade de campo apenas utilizando a abertura da lente. Por este motivo, outras maneiras de ampliar a profundidade de campo surgiram no campo de processamento de imagens.

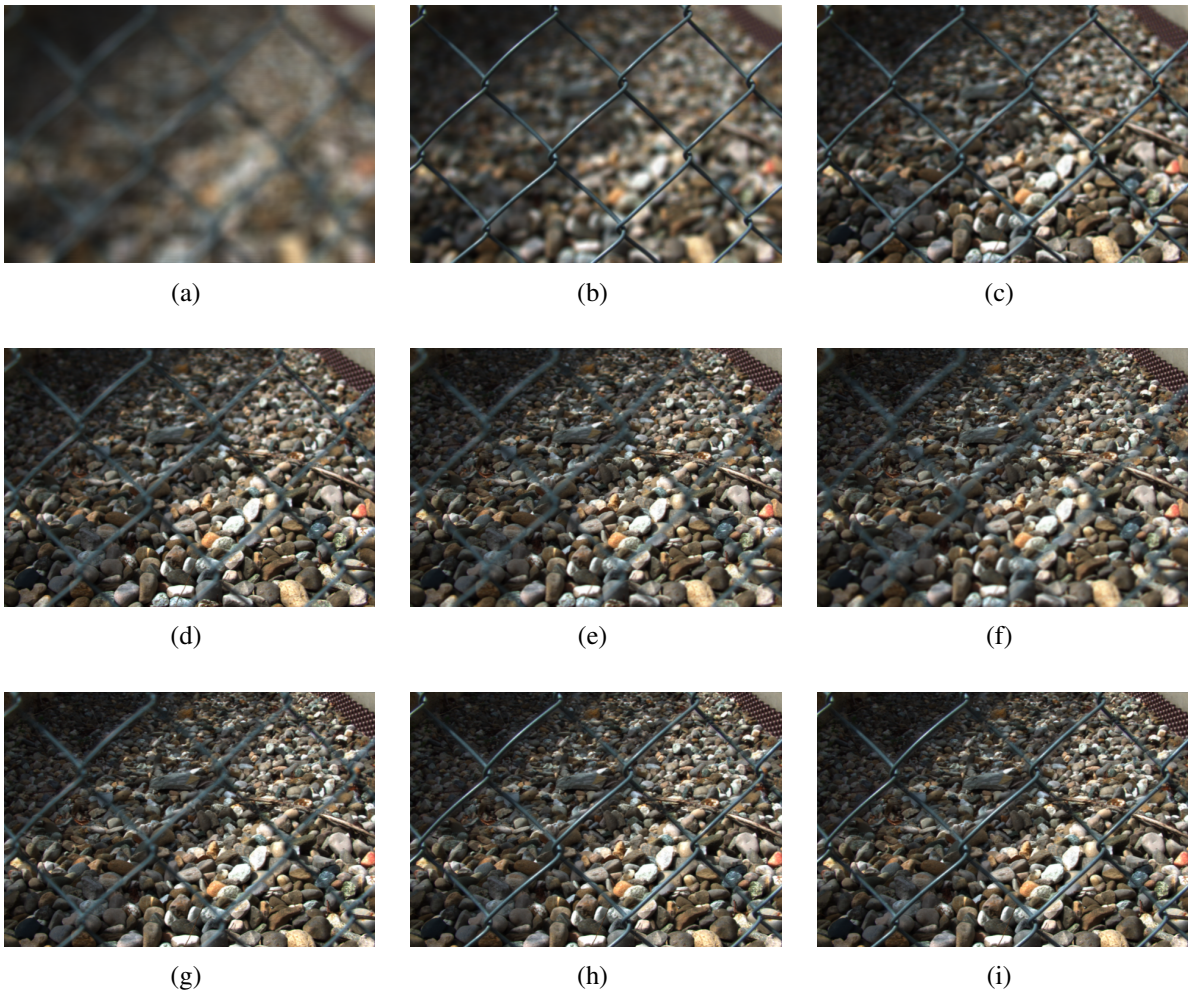
1.2 MOTIVAÇÃO

Técnicas para essa expansão consistem na fusão de várias imagens com uma menor profundidade de campo para a criação de uma imagem final com mais objetos nítidos, porém a junção pode acontecer de várias maneiras. Algoritmos desse tipo podem ser chamados de algoritmos de empilhamento de foco, e são feitos de diversas maneiras, como técnicas baseadas em redes neurais [3], na transformada de Wavelet e modelos de Markov [4]. Métodos mais consolidados consistem na decomposição multi-escala [5] [6] [7] geralmente baseadas na decomposição por pirâmides de Laplace [8].

Apesar dos trabalhos citados conseguirem realizar a fusão das imagens e criar uma imagem completamente em foco (*all-in-focus*), a maioria trabalha com imagens criteriosamente selecionadas na pilha de foco que é utilizada como entrada do algoritmo. Ao saber o que se tem na pilha, a confiança que o resultado será o desejado cresce, pois a informação necessária estará de fato na pilha de foco.

A abordagem utilizada para desenvolver uma técnica que escolhe as melhores imagens para utilização na fusão é baseada na ideia de uma pilha de foco (*focal-stack*) onde as imagens não são

criteriosamente selecionadas, ou seja, não se sabe nada sobre a informação presente na pilha. Esta técnica é discutida por Teixeira et. al. [7]. A partir dos resultados deste método, propõe-se um algoritmo que utilize menos imagens, garantindo que essas poucas imagens contenham de fato a informação necessária para que se forme uma boa imagem. Isso é feito utilizando um medidor de foco, garantindo que as imagens selecionadas estejam focadas, e mapas de profundidade, que garantem que vários planos da imagem estejam em foco.



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figura 1.3: Na primeira e segunda linha, exemplos de imagens de baixa profundidade de campo. Na terceira linha, imagens com maior profundidade de campo geradas pelo algoritmo.

1.3 DEFINIÇÃO DO PROBLEMA

Baseando-se nestas motivações apresenta-se um método que, a partir de um *focal-stack* desconhecido, obtido amostrando os pontos de foco, automatize todo o processamento. Desde a escolha até a fusão das imagens. Dessa forma, pretende-se gerar imagens com menos erros utilizando menos imagens como entrada.

Seguindo a linha do trabalho passado, continua-se baseando o método na linguagem de pro-

gramação própria da plataforma MATLAB. O MATLAB é um ambiente de computação numérica onde seus cálculos são todos otimizados para se trabalhar com matrizes, sendo uma boa plataforma para se tratar imagens.

1.4 ESTRUTURA DO TEXTO

O capítulo 2 oferece um panorama sobre todo o embasamento teórico necessário para melhor entendimento do trabalho. A apresentação do método proposto, com exemplos e diagramas explicando seu funcionamento se encontram no capítulo 3. O capítulo 4 mostra os resultados obtidos, fazendo sua análise e comparando com trabalhos antigos, para que, por fim, no capítulo 5 seja realizada uma síntese sobre o que foi discutido e propostas de futuros trabalhos.

2 REVISÃO BIBLIOGRÁFICA

Este capítulo traz uma pequena revisão de literatura abordando os temas mais importantes para compreensão deste trabalho.

2.1 A CÂMERA FOTOGRÁFICA

O princípio de funcionamento de uma câmera foi descoberto no experimento da câmera escura. Ao longo da história, aprimora-se esta ideia até chegar na câmera digital atual. Os componentes principais para compor a imagem consistem nas lentes, o diafragma, obturador e a mídia utilizada para armazenar a imagem. Antigamente utilizava-se filmes, que ao receber luz, armazena a imagem através de processos foto-químicos. Atualmente, os filmes foram trocados por sensores que armazenam essa imagem de forma digital.

2.1.1 Funcionamento

Ao se apontar a câmera para um objeto, a luz refletida deste objeto é coletada pelas lentes. As lentes fazem a luz se curvar, para que convirjam em um único ponto de foco. Caso os raios de luz convirjam sobre o sensor, a imagem formada estará em foco, como na Figura 2.1. Agora, caso os raios convirjam antes ou depois do sensor (Figura 2.2), a imagem fica fora de foco. Independentemente, a imagem é capturada pelos sensores, formando uma matriz.

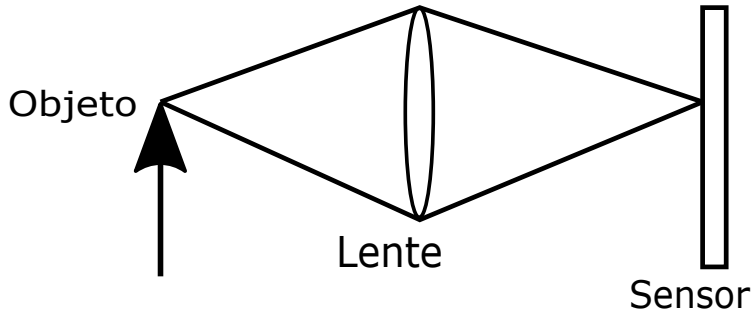
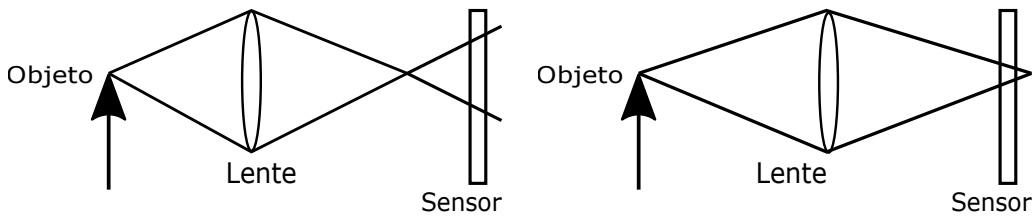


Figura 2.1: Esquemático de uma câmera convencional.



(a) Raios convergem antes do sensor. (b) Raios convergem depois do sensor.

Figura 2.2: Esquemático da câmera convencional quando a imagem fica fora de foco.

Este é o princípio de funcionamento da captura da imagem, porém, vários elementos da câmera alteram a quantidade de luz que entra na câmera e seu ponto de foco.

2.1.1.1 Velocidade de disparo

A velocidade controla o tempo que o obturador ficar aberto. O obturador é um dispositivo que abre e fecha, controlando o intervalo de tempo de exposição do sensor à luz. Como é regulada a quantidade de luz que o sensor capta no momento da fotografia, essa velocidade se relaciona com a ideia de movimento da foto. A velocidade é medida em segundos ou frações de segundos.

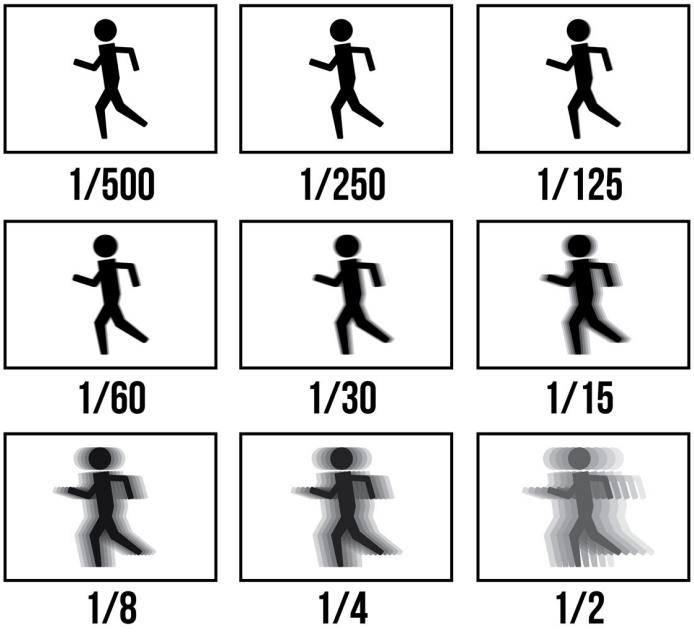


Figura 2.3: Efeito da velocidade de disparo. [9]

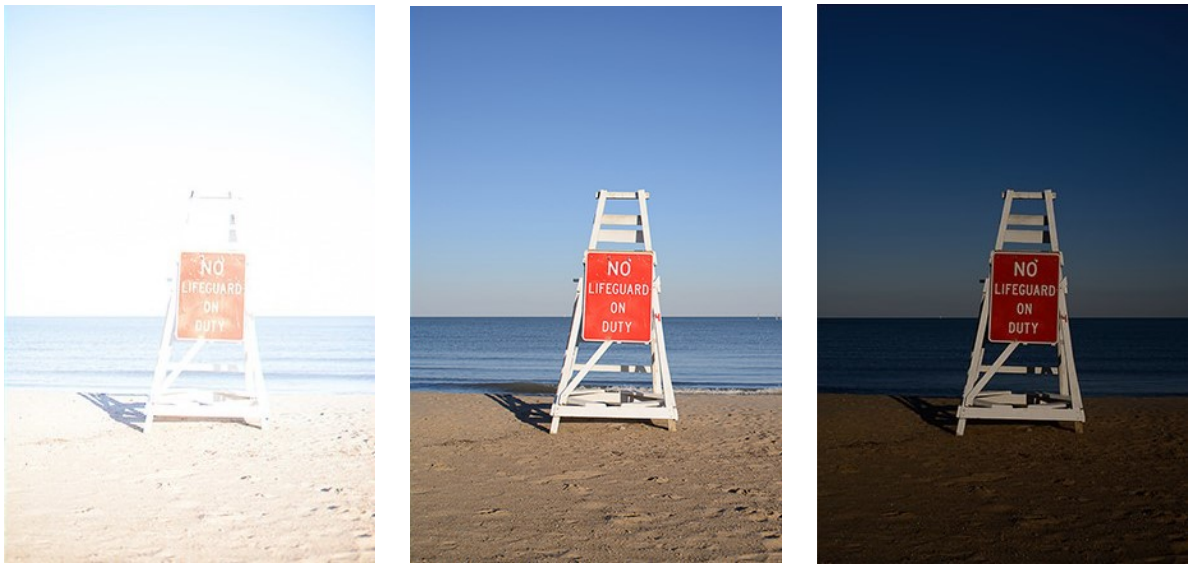
A Figura 2.3 exemplifica a quantidade de luz capturada pelo sensor. Quanto mais rápido, menos luz é captada. Ao se aumentar o tempo de exposição, tentando captar mais luz, pode acontecer o desfoque do movimento. Nestes casos, a imagem perde o foco na realização do movimento, como pode ser visto na Figura 2.4.

Figura 2.4: Consequência de um alto tempo de exposição. [10]

2.1.1.2 Abertura do Diafragma

O diafragma regula a quantidade de luz que chega ao obturador, isso acontece de acordo com sua abertura, deixando mais ou menos luz passar. A abertura do diafragma é medida em números f, escritos da forma $f/1.4$, $f/22$, quanto menor o número, maior a abertura.

A abertura do diafragma é normalmente chamada apenas de abertura e ela é diretamente relacionada com a profundidade de campo da imagem. Quanto menor a abertura, menos luz é captada pelo sensor, fazendo com que mais elementos da imagem estejam em foco, ou seja, aumentando o profundidade de campo. Este efeito pode ser visualizado na Figura 1.1.



(a) Grande abertura. (b) Média abertura. (c) Pequena abertura.

Figura 2.5: Influência da abertura na captação de luz. Modificado de [11].

Mesmo sendo possível aumentar a profundidade de campo da foto diminuindo sua abertura,

menos luz é capturada pela câmera. E isso pode causar efeitos indesejáveis na foto. Este efeito pode ser observado na Figura 2.5.

Portanto, mesmo que seja possível, não é sempre que se consegue aumentar a profundidade de campo por meio da abertura. Como se observa na Figura 2.5c, pequenas aberturas resultam em fotos mais escuras, devido à pouca passagem de luz.

2.2 GERAÇÃO DO *FOCAL-STACK*

O algoritmo neste funciona por meio de um *focal-stack* que pode ser gerado por meio de câmeras convencionais, ou simulado por meio de dados do campo de luz (*Light-Field*, LF) da imagem, capturado por câmeras plenóticas.

2.2.1 Por meio de câmeras convencionais

Idealmente, o *focal-stack* deve ser criado por meio de câmeras convencionais. Montando a cena desejada para a confecção do *focal-stack*, capturam-se as fotos variando a posição do foco na imagem, gerando assim uma pilha com diversas imagens com focos em diferentes lugares.

Apesar de ser possível utilizando apenas uma câmera convencional, o processo se torna complicado devido a fatores como manualmente selecionar a posição do foco, ou enquadrar corretamente a cena. Com isso em mente, para facilitar a aquisição, seria necessário uma câmera com controle via *software* onde fosse possível automatizar a seleção do foco e a captura da imagem, bem como um tripé para manter o enquadramento da cena sempre igual.

Embora algumas câmeras ofereçam algum nível de controle via *software*, câmeras com estas características não são muito acessíveis. Por este motivo, pode ser gerado um *focal-stack* simulado, por meio das imagens capturadas por uma câmera plenótica.

2.2.2 Câmeras plenóticas

Câmeras convencionais capturam apenas a intensidade de luz vista por seus sensores, resultando em uma fotografia de duas dimensões $I(x, y)$. Câmeras plenóticas, por outro lado, capturam todo o campo de luz mensurando a quantidade de luz propagando ao longo de cada raio que passa pelo sensor.

A configuração básica de uma câmera plenótica se baseia em uma lente principal, como de uma câmera comum, um conjunto de microlentes e um conjunto de sensores. Um esquemático pode ser visto na Figura 2.6.

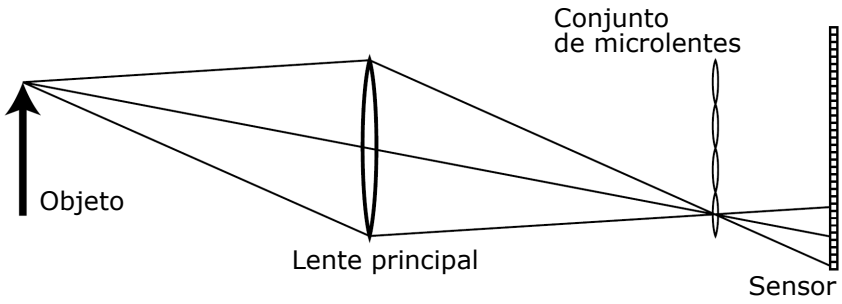


Figura 2.6: Esquemático de uma câmera plenótica. Adaptado de [12]

Como pode ser visto na Figura 2.6, os raios de luz que partem do objeto convergem em um único ponto no plano focal do conjunto de microlentes, a partir deste ponto, as microlentes no local separam esses raios baseando-se na direção, criando uma imagem focada de abertura da lente principal no conjunto de pixels debaixo das microlentes. Microscopicamente é possível observar as subimagens capturadas pelas microlentes, nestas imagens se captura a estrutura de luz da cena, revelando, por exemplo, a profundidade de objetos.

Os dados adquiridos pela câmera plenótica consideram os dois planos do campo de luz LF, onde $LF(u, v, s, t)$ representa a luz propagada ao longo do raio que converge no plano da lente principal em (u, v) e no plano das microlentes em (s, t).

2.2.3 Imagens de Sub-Abertura

Formam-se imagens extraindo o mesmo pixel de baixo de cada microlente. A extração é feita fixando (u, v) e considerando todo (s, t). Todos os raios que passam por estes pixels são originados da mesma sub-abertura da lente principal, portanto a imagem extraída resultante é a fotografia convencional, caso esta fosse tirada com aquela sub-abertura. Escolher um pixel diferente corresponde a escolher outra sub-abertura. A soma de todas essas sub-aberturas é a abertura original da lente.

Utilizando as imagens de sub-abertura é possível obter representações da mesma cena com diferentes regiões em foco. Essas representações são obtidas com a utilização de sensores sintéticos.

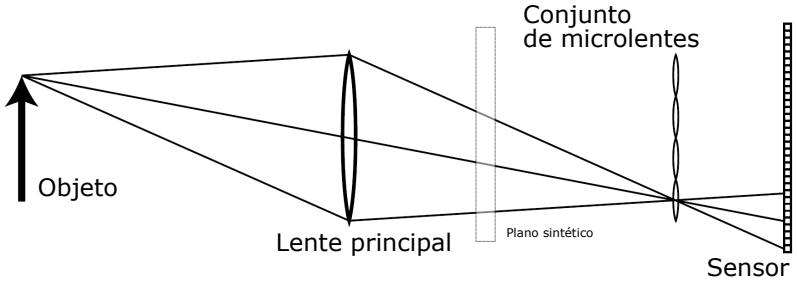


Figura 2.7: Exemplo do plano sintético. Adaptado de [12].

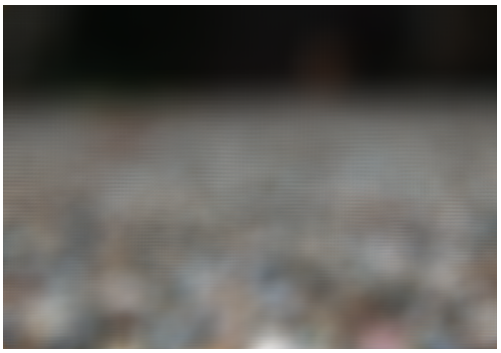
Simula-se que os raios de luz estejam convergindo com os sensores em uma diferente região, fazendo com que o foco da imagem mude. Um exemplo deste plano sintético é mostrado na Figura 2.7.

A partir disso, as imagens utilizadas para compor o *focal-stack* são obtidas por um algoritmo de deslocamento e soma. A Equação 2.1 move este plano sintético, fazendo com que sejam geradas as imagens com diferentes focos.

$$I_n(s,t) = \frac{1}{M} \sum_{u,v} LF\left(u, v, s + u\left(1 - \frac{1}{\alpha_n}\right), t + v\left(1 - \frac{1}{\alpha_n}\right)\right) \tag{2.1}$$

Na Equação 2.1 α_n é um parâmetro que indica o local onde este plano sintético se encontra, *M* é o número de imagens de sub-abertura e I_n é a imagem gerada com foco em α_n. Variou-se α_n em passos de 0.007, começando em 0.2 e terminando em 2, gerando assim, 258 imagens.

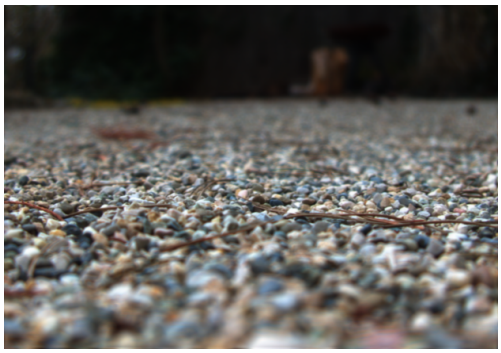
Alguns exemplos dessas imagens são mostrados na Figura 2.8. A Figura 2.8b tem seu foco próximo da câmera, e, conforme se aumenta o valor de α_n, seu foco vai se distanciando, como é visto na Figura 2.8c e depois na Figura 2.8d. A Figura 2.8a não contém foco em parte alguma da imagem, esse tipo de imagem pode ser tratada como ruído.



(a) $\alpha_1 = 0.2$

(b) $\alpha_{53} = 0.564$

(c) $\alpha_{95} = 0.858$



(d) $\alpha_{196} = 1.565$

Figura 2.8: Exemplos de imagens geradas a partir do *Light-Field*.

2.3 CODIFICAÇÃO POR PIRÂMIDE DE LAPLACE

Uma das características de uma imagem é ter pixels vizinhos que são altamente correlatos. Portanto, representar uma imagem em termos dos valores dos pixels se torna uma maneira ineficiente, já que toda informação codificada é redundante. Encontrar uma representação que descorreciona os pixels é o primeiro problema para criar um código comprimido.

Pelo uso de codificação preditiva, os pixels são codificados sequencialmente em uma matriz de erros de predição. Para codificação de cada pixel, seu valor é predito de pixels já codificados. Esse valor predito, que representa informação redundante, é subtraído do valor do pixel atual e apenas a diferença (erro de predição) é codificada. Esse processo é dito causal, visto que só valores já codificados são usados para prever cada valor de pixel, e, por isso, facilita a decodificação.

Outra abordagem é a codificação não-causal, que tipicamente envolve transformadas. Ao contrário da causal, elas não codificam o pixel sequencialmente, e sim em blocos, baseando-se em uma vizinhança simétrica centrada em cada pixel. Técnicas não-causais produzem predições mais precisas, logo, realizam melhor compressão.

A pirâmide de Laplace é um técnica que combina os dois casos. Usa-se uma média ponderada local para prever os valores de cada pixel, utilizando uma função de pesos unimodal tipo Gaussiana no próprio pixel. A predição dos pixels é obtida pela convolução dessa função com a imagem. O resultado é uma imagem filtrada por um filtro passa-baixas, que então é subtraída da original.

Seja $g_0(ij)$ a imagem original e $g_1(ij)$ a imagem filtrada, o erro de predição $L_0(ij)$ é dado por:

$$L_0(ij) = g_0(ij) - g_1(ij) \tag{2.2}$$

Ao contrário de codificar g_0, se codifica L_0 e g_1, gerando uma compressão que tem os benefícios de ambos tipos, já que L_0 é altamente descorrelata e g_1 é uma imagem filtrada. Mais dados podem ser comprimidos repetindo este processo várias vezes. Fazendo isso são obtidas matrizes L_0, L_1, ..., L_n. Nesta implementação cada camada é menor que a antiga por uma escala de 1/2. Imaginando essas matrizes empilhadas, formam uma pirâmide, como visto na Figura. 2.9.

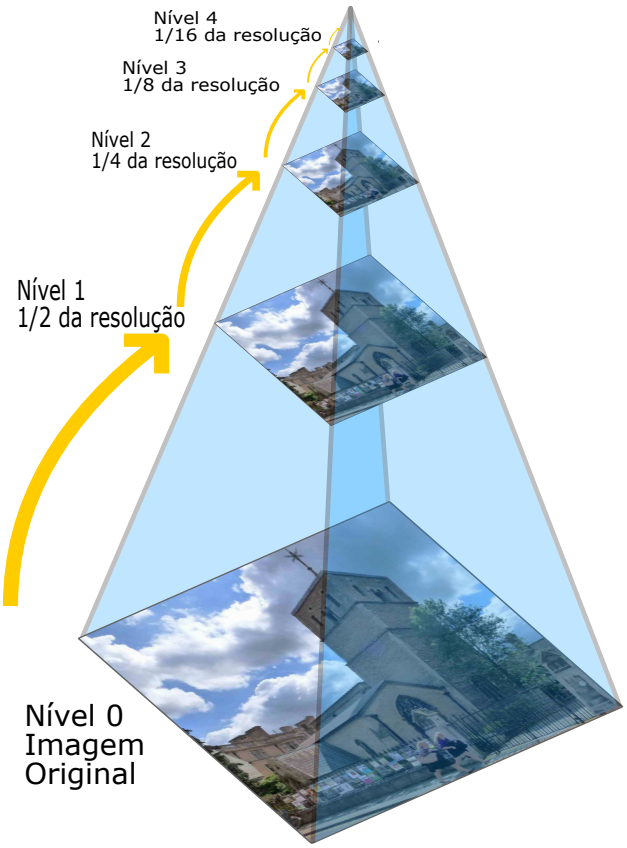


Figura 2.9: Pirâmide de Laplace. [13]

2.3.1 Pirâmide Gaussiana

O primeiro passo para se gerar uma pirâmide de Laplace é aplicar um filtro passa baixas na imagem original g_0 para se obter a imagem g_1. É dito que g_1 é uma versão reduzida de g_0, já que tanto resolução e densidade de amostras diminuem. O processo de filtragem é equivalente a uma convolução com uma função de um conjunto de funções locais de pesos simétricos. Uma função importante deste conjunto se parece com a distribuição de probabilidade Gaussiana, portanto, chama-se a sequência de imagens g_0, g_1, ... g_n de pirâmide Gaussiana.

2.3.1.1 Geração da pirâmide Gaussiana

A geração começa partindo de g_0, base da pirâmide Gaussiana, uma imagem de I linhas e J colunas. O primeiro nível da pirâmide contém a imagem g_1, uma versão filtrada em baixas frequências de g_0. Cada valor neste primeiro nível é calculado como uma média ponderada dos valores do nível 0 em uma janela de tamanho 5×5. Os valores do nível 2 são calculados baseados nos valores do nível 1 e assim por diante. Este processo é feito pela operação dada pela Equação 2.3.

$$g_l(i,j) = \sum_{m=-2}^{2} \sum_{n=-2}^{2} w(m,n) g_{l-1}(2i+m, 2j+n) \tag{2.3}$$

Onde $0 < l < N$ são os níveis da pirâmide, e $0 < i \leq I_l$ e $0 < j \leq J_l$ são as dimensões da matriz no nível l da pirâmide. A função *reduz*, que gera um nível da pirâmide a partir do nível anterior, é dada pela Equação 2.3. As imagens reduzidas são mostradas na Figura 2.10.

$$g_k = reduz(g_{k-1}) \tag{2.4}$$

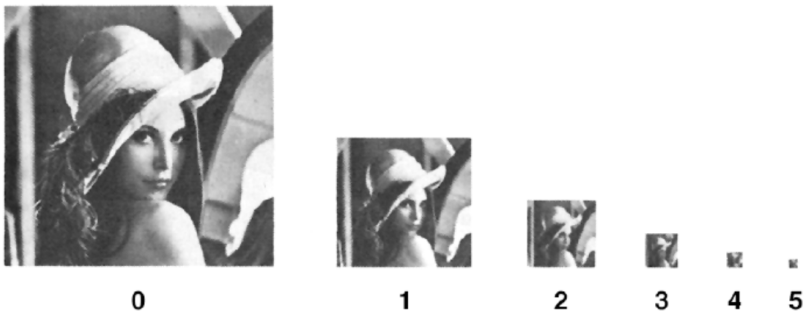


Figura 2.10: Pirâmide Gaussiana. [8]

2.3.1.2 Interpolação da pirâmide Gaussiana

Define-se a função *expande* como reverso da função *reduz*. Essa função expande uma matriz $(M+1) \times (N+1)$ para uma matriz $(2M+1) \times (2N+1)$ interpolando novos os valores de cada ponto da matriz por meio dos valores existentes. Portanto, aplicando a função em uma matriz g_l da pirâmide Gaussiana, resulta na matriz $g_{l,1}$, de mesmo tamanho de g_{l-1}.

Sendo $g_{l,n}$ o resultado de expandir g_l n vezes, então $g_{l,0} = g_l$ e $g_{l,n} = expande(g_{l,n-1})$.

A função expande é obtida pela Equação 2.5.

$$g_{l,n}(i,j) = 4 \sum_{m=-2}^{2} \sum_{n=-2}^{2} w(m,n) g_{l,n-1} \left(\frac{i-m}{2}, \frac{j-n}{2} \right) \tag{2.5}$$

Onde $0 < l \leq N$, $n \geq 0$, $0 \leq i < I_{l-n}$ e $0 \leq j < J_{l-n}$. Além disso, apenas os termos inteiros de $\frac{i-m}{2}$ e $\frac{j-n}{2}$ são incluídos na soma. Se expandir l vezes uma imagem g_l, obtém-se $g_{l,l}$, que tem o mesmo tamanho que a imagem original g_0.

2.3.2 Pirâmide de Laplace

A razão por trás da construção da imagem reduzida g_1 é para que esta sirva como predição dos valores de pixels da imagem original g_0. Para obter a representação comprimida, codifica-se

a imagem de erro, que é obtida quando se subtrai g_1 de g_0. Esta se torna a base da pirâmide de Laplace. O próximo nível é gerado codificando g_1 da mesma maneira.

2.3.2.1 Geração da pirâmide de Laplace

A pirâmide de Laplace é dada pela sequência de imagens de erro L_0, L_1, ..., L_N. Cada uma é a diferença entre dois níveis da pirâmide Gaussiana, ou seja, $L_l = g_l - expande(g_{l+1})$. Portanto, para $0 \leq l < N$, L_l é calculada pela Equação 2.6.

$$L_l = g_l - g_{l+1,1} \tag{2.6}$$

Como não existe imagem g_{N+1} para servir como imagem de predição para g_N, $L_N = g_N$. Um exemplo das imagens geradas por este procedimento é dado na Figura 2.11.

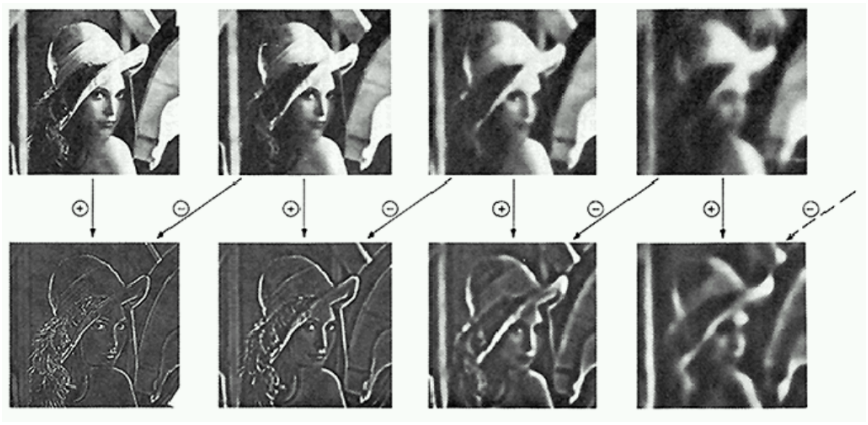


Figura 2.11: Geração da Pirâmide de Laplace. [8]

2.3.2.2 Decodificação

Uma maneira de se recuperar a imagem é apenas expandindo e somando todos os níveis da pirâmide. Porém, uma maneira mais eficiente é expandir L_N uma vez e somar L_{N-1}, e então expandir essa imagem e adicionar L_{N-2}, e repetir o procedimento até se chegar no nível 0 e a imagem g_0 ser recuperada. Esse procedimento é o reverso da geração da pirâmide de Laplace. Da Equação 2.6, se tem que $L_N = g_N$, então, para $l = N-1, N-2, ..., 0$, se dá a Equação 2.7.

$$g_l = L_l + expande(g_{l+1}) \tag{2.7}$$

Na Figura 2.12 observa-se todos os procedimentos, tanto para geração, quanto para recuperação das imagens.

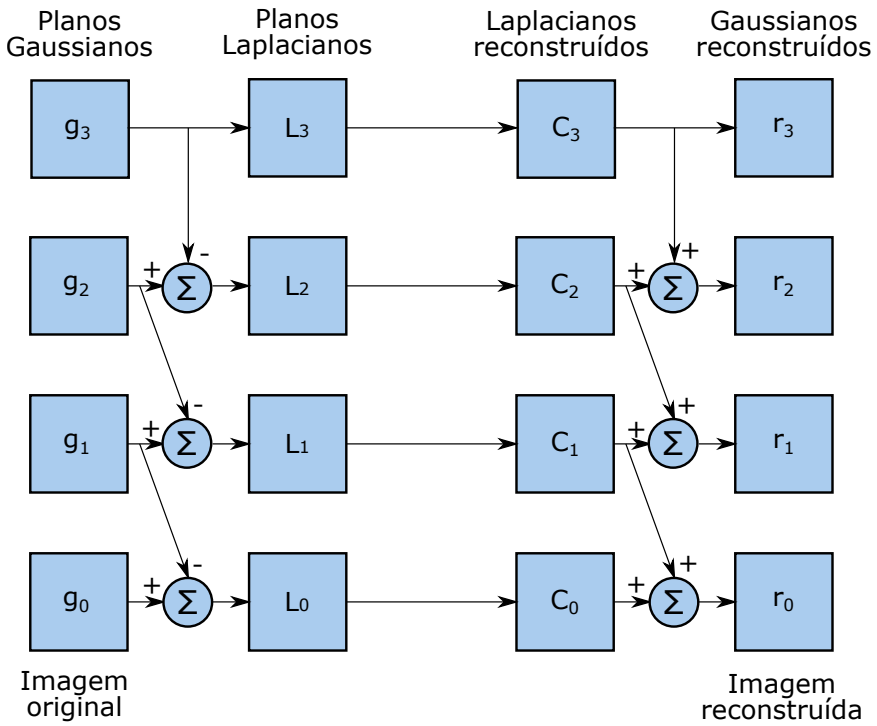


Figura 2.12: Esquemático de todo o processo. Modificado de [8].

2.4 MAPAS DE PROFUNDIDADE

A estimação de profundidade (*depth*), ou estimação de mapas de profundidade, de uma cena é aplicada em diversas áreas, como robótica, digitalização de ambientes e a própria fotografia. Um mapa de profundidade carrega informação dos níveis de profundidade de uma imagem, detalhando o quão distante certo objeto está da câmera.

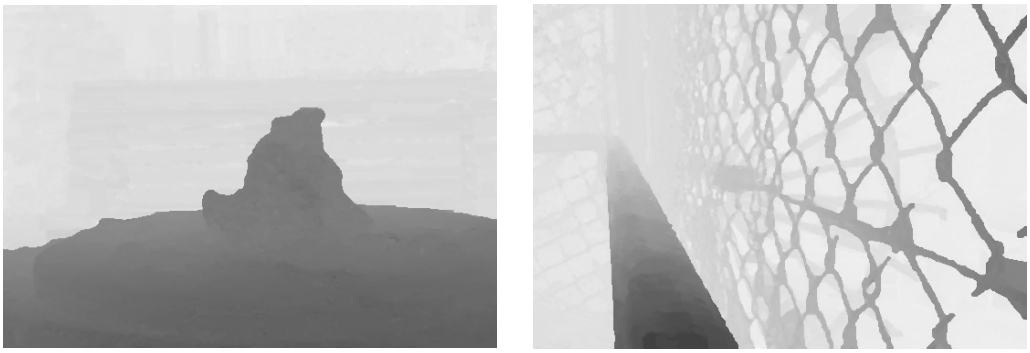
Métodos convencionais de estimação deste mapa requerem vários pontos de vista da cena ou múltiplas capturas, podendo ser inviável a depender da situação. Entretando, vários algoritmos propostos na literatura geram mapas de profundidade a partir de um *focal-stack*, uma vez que o plano de foco está relacionado com esta distância. Assim, como o nosso algoritmo usa como entrada esse mesmo *focal-stack*, é pensado em como o mapa de profundidade poderia auxiliar no algoritmo de fusão de imagens.

Tao et. al [14] propõe um algoritmo para se obter este mapa a partir das figuras de sub-abertura obtidas pelo LF de uma câmera plenótica.

O algoritmo consiste em deslocar cada imagem de sub-abertura por vários valores, a depender de cada profundidade. Pixels em profundidade ótima não mudam muito de uma imagem de sub-abertura para outra. Por outro lado, pixels fora dessa profundidade ótima variam muito. É estimado então a profundidade de cada pixel ao se analisar qual sua profundidade ótima. São feitas duas medidas desta profundidade ótima, ambas são utilizadas de entrada em um algoritmo de minimização, tentando gerar uma saída melhor que as duas medidas.

A Figura 2.13 mostra exemplos de mapas de profundidade. A figura se torna mais clara quanto

mais longe da câmera.



(a) (b)

Figura 2.13: Exemplos de mapas de profundidade.

2.5 MEDIÇÃO DE FOCO

Na literatura, algoritmos de medição de foco são utilizados para recuperar informações do mundo real 3D a partir de uma foto 2D [15] [16]. Para o problema estabelecido neste trabalho, se utiliza dessa ferramenta para verificar o quanto uma imagem está em foco e atribuir uma pontuação para cada figura. Alta para figuras em foco, baixa para figuras fora de foco.

Entretanto, existem vários algoritmos para se medir o grau de foco, tanto de uma imagem como de um pixel. Em [17], Said et. al faz uma análise de vários grupos de métodos, avaliando quais são os melhores. Dividem-se estes métodos em seis grupos.

1. Operadores baseados em gradientes. Este tipo de operador é baseado no gradiente ou na primeira derivada da imagem. Estes algoritmos assumem que a imagem em foco apresenta mais bordas nítidas que desfocadas. Portanto, o gradiente é utilizado para se medir o foco.

2. Operadores baseados em laplacianos. Similar ao primeiro grupo, estes algoritmos buscam medir a quantidade de bordas presente na imagem, porém, utilizam da segunda derivada ou Laplaciano.

3. Operadores baseados em wavelet. Estes se baseiam na vantagem que os coeficientes da transformada discreta de wavelet tem de detalhar o conteúdo espacial e frequencial da imagem. Podendo assim ser utilizados para se medir o nível de foco.

4. Operadores baseados em estatística. Utilizam de várias estatísticas da imagem como descritores de textura para computar o nível de foco.

5. Operadores baseados em DCT. Similar ao grupo baseado em wavelet, porém utilizam a transformada discreta do cosseno para medir o nível de foco da imagem a partir de seu conteúdo na frequência.

6. Operadores diversos. Este grupo engloba os algoritmos que não são baseados em nenhum algoritmo previamente mencionado.

Em seu trabalho, a avaliação é feita baseada na performance geral do algoritmo bem como em sua robustez. Como todos os testes realizados são em aplicações para retirar informações tridimensionais da imagem, a escolha do método para este trabalho foi feita baseada na performance geral do algoritmo. Após avaliações dos algoritmos, decidiu-se pela utilização do método do Laplaciano Diagonal [15].

2.5.1 Laplaciano Diagonal

Este algoritmo é proposto por Thelen et. al [15] como uma modificação do método proposto por Nayar et al. [18].

Nayar et. al. propõe que o nível de foco é medido a partir de uma definição alternativa do laplaciano, onde se considera pixels vizinhos na horizontal e na vertical

$$\Delta_m I = |I * \mathcal{L}_X| + |I * \mathcal{L}_Y| \tag{2.8}$$

Onde $\mathcal{L}_X = [-1 \quad 2 \quad -1]$ e $\mathcal{L}_Y = \mathcal{L}_X^T$.

Porém, esta modificação considera apenas os vizinhos na horizontal e vertical. Thelen et. al propõe uma segunda modificação, considerando também pixels vizinhos da diagonal.

$$\Delta_m I = |I * \mathcal{L}_X| + |I * \mathcal{L}_Y| + |I * \mathcal{L}_{X1}| + |I * \mathcal{L}_{X2}| \tag{2.9}$$

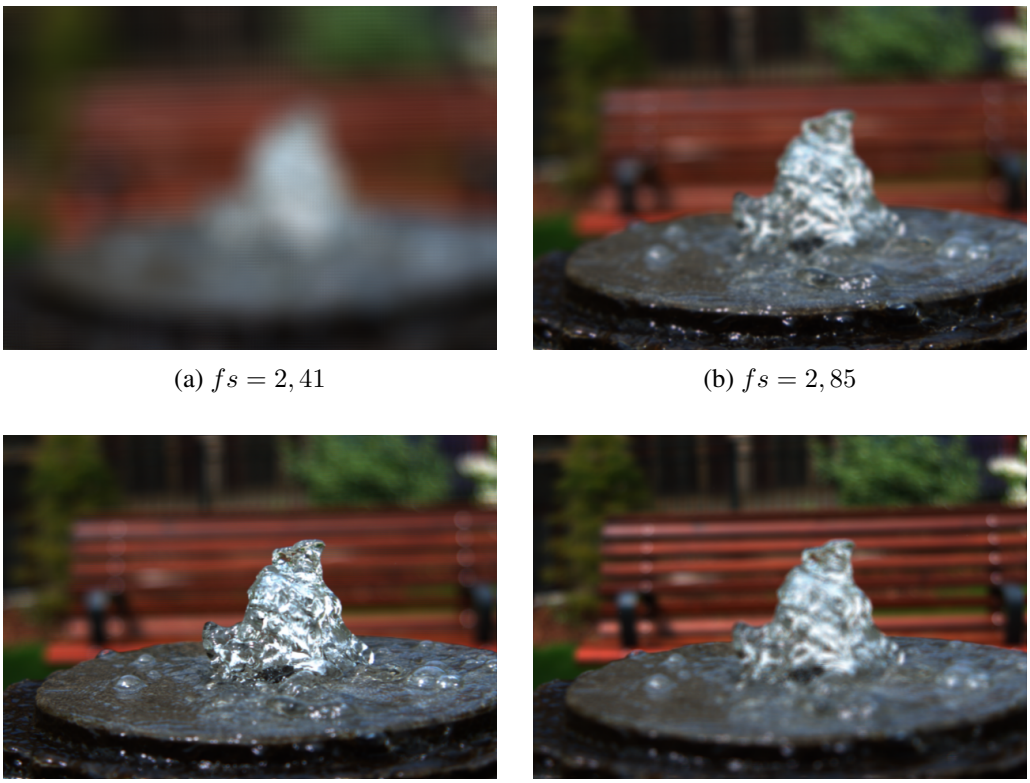
$\mathcal{L}_X$ e $\mathcal{L}_Y$ continuam iguais para este caso, já $\mathcal{L}_{X1}$ e $\mathcal{L}_{X2}$ são definidas como:

$$\mathcal{L}_{X1} = \frac{1}{\sqrt{2}} \begin{bmatrix} 0 & 0 & 1 \\ 0 & -2 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \mathcal{L}_{X2} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.10}$$

A partir da Equação 2.9 o foco é medido de acordo com a Equação 2.11.

$$\phi(x, y) = \sum_{(i,j) \in \Omega(x,y)} \Delta_m I(i, j) \tag{2.11}$$

Onde $\Omega(x, y)$ é a vizinhança ao redor do pixel, Δ_m é o operador laplaciano modificado e I é a imagem.



(a) $fs = 2,41$ (b) $fs = 2,85$

(c) $fs = 8,89$ (d) $fs = 2,93$

Figura 2.14: Exemplos de imagens e suas pontuações de foco.

Na Figura 2.14 são mostrados exemplos de imagens e suas pontuações fs para cada imagem. A Figura 2.14a é a que possui menos nitidez, e seu valor de fs é baixo. À medida que se tem mais elementos em foco na imagem, o valor de fs vai aumentando, como nas Figuras 2.14b e 2.14c. Na Figura 2.14d perde-se foco de alguns elementos, e, portanto, diminui-se fs novamente.

3 MÉTODO PROPOSTO

Este capítulo aborda todos os procedimentos para geração das imagens *all-in-focus*, desde o algoritmo de escolha das imagens ao algoritmo de fusão, bem como o banco de dados utilizado para teste do método.

3.1 BANCO DE DADOS

As imagens utilizadas para testar o algoritmo fazem parte do banco de imagens JPEG Pleno LF Image [19]. Utilizou-se dos arquivos de *Light Field* para gerar o *focal-stack*. As imagens utilizadas foram *Bikes*, *Chain-link Fence 2*, *Danger de Mort*, *Fountain & Bench*, *Friends 3*, *Gravel Garden*, *Palais du Luxembourg* e *Spear Fence 2*. Os mapas de profundidade fazem parte do mesmo banco de dados.

3.2 ALGORITMO DE FUSÃO

O método utilizado para fusão das imagens é baseado no algoritmo de empilhamento de foco de Bilcu. et. al. [5], porém com modificações propostas por Natália et. al. [7]. Utilizando como entrada as N imagens do *focal-stack* realiza-se a decomposição em pirâmide de Laplace de cada uma das I_n imagens de entrada, gerando L_n^k pirâmides, uma para cada imagem de entrada, em que k é o nível de resolução da pirâmide.

Após a decomposição em pirâmides para cada imagem, é construído um mapa de ação A_n^k obtido pela aplicação do operador $p(\cdot)$ nas pirâmides L_n^k. A modificação proposta por Natália et. al. [7] dita que o mapa de ação não é apenas a aplicação operador $p(\cdot)$ em cada pixel da pirâmide, mas sim, a soma dos resultados do operador quando este é aplicado aos pixels vizinhos.

$$A_n^k(i,j) = \sum_{i_n=-\delta}^{\delta} \sum_{j_n=-\delta}^{\delta} p\big(L_n^k(i+i_n, j+j_n)\big) \tag{3.1}$$

O operador $p(\cdot)$ utilizado é $p(x) = x^2$. Foi utilizado este operador de acordo com Bilcu et. al. [5]. Na Eq 3.1 $i = 1, 2, ..., I_k$ e $j = 1, 2, ..., J_k$ onde I_k e J_k são as dimensões de L_n^k e δ define o tamanho da vizinhança a ser considerada. Nas bordas da imagem, pixels fora dos limites são considerados nulos.

Após a confecção do mapa de ação, se compõe um mapa de decisão para definir qual pixel irá compor a pirâmide de saída da imagem final. O mapa de decisão de cada nível da pirâmide é

dado pelos pixels máximos entre os mapas de ação.

$$D_k(i,j) = \underset{n}{\operatorname{argmax}} \left(A_1^k(i,j), ..., A_n^k(i,j) \right) \tag{3.2}$$

Por fim, a imagem é reconstruída. É gerada uma pirâmide de saída a partir do mapa de decisão para cada nível e a imagem final é formada por essas pirâmides.

$$L_O^k(i,j) = L_S^k(i,j), S = D_k(i,j) \tag{3.3}$$

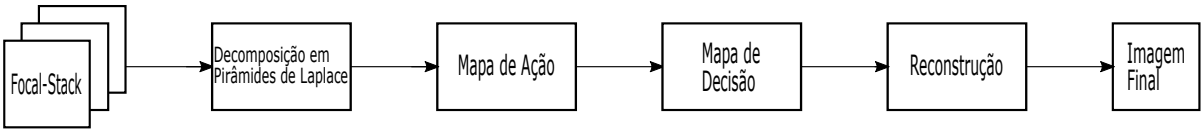


Figura 3.1: Diagrama de blocos do Algoritmo de Fusão.

A fusão de imagens coloridas só é feita fazendo um mapeamento para sua representação em YCbCr, aplicando a decomposição em pirâmide de Laplace nos três canais de cor, porém, o mapa de decisão é aplicado apenas na camada Y. Após a confecção do mapa de decisão, se utiliza das três camada para gerar a imagem final. Para teste do algoritmo, fez-se decomposições em até seis níveis.

3.3 ALGORITMOS DE SELEÇÃO DE IMAGENS DE ENTRADA

É possível utilizar vários critérios para escolher as imagens de entrada do algoritmo de fusão. Um critério pode ser, inclusive, utilizar todas as imagens geradas pela Equação 2.1. Porém como várias imagens podem conter pouca ou nenhuma relevância para a imagem final, propõe-se métodos para filtrar essa entrada e escolher imagens que contribuam mais para o resultado final desejado, diminuindo ruído. Para isso, propõe-se a utilização de três algoritmos de filtragem diferentes.

3.3.1 Algoritmo proposto 1

O primeiro algoritmo é proposto por Natália et. al. [7]. Utilizando todas as imagens geradas pela Eq. 2.1, cria-se um mapa de decisão. A partir deste mapa de decisão, propõe-se um parâmetro $score_n$ que indica o número de pixels do mapa de decisão D_k que é originado de cada imagem.

$$score_n = sum(D_k(i,j) = n), n = 1, 2, ..., N_{in} \tag{3.4}$$

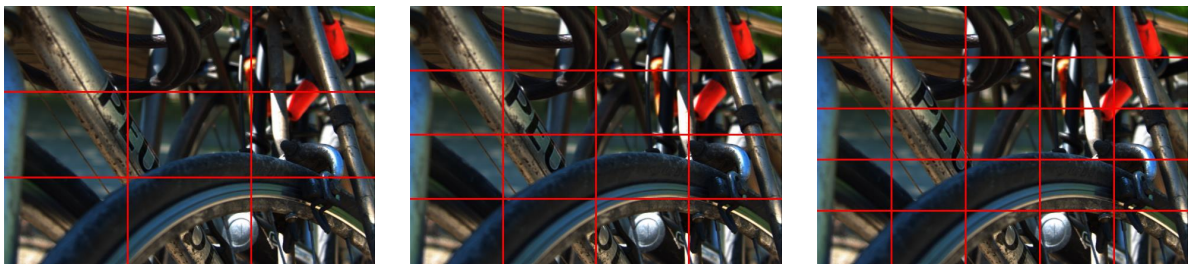
A partir deste parâmetro, é feita a escolha de N_{final} imagens com os maiores valores de $scores$

para serem utilizadas como entrada do algoritmo de fusão.

Figura 3.2: Diagrama de blocos do Algoritmo 1.

3.3.2 Algoritmo proposto 2

Este método propõe a validação de cada entrada por meio de um *medidor de foco*. Medindo o foco das imagens de entrada, acarreta em uma garantia de que a entrada utilizada contribuirá mais para a geração de um melhor resultado, ou seja uma melhor imagem final. Para isso é utilizado um algoritmo que mede o quanto a imagem está focada por meio de uma pontuação. Uma alta pontuação indica uma imagem com pixels mais nítidos e uma baixa pontuação indica poucos pixels nítidos.



(a) Grade 3x3 (b) Grade 4x4 (c) Grade 5x5

Figura 3.3: Exemplos das imagens com as grades aplicadas.

Utilizando todo o *focal-stack*, divide-se cada uma das imagens em uma grade de tamanho arbitrário $Q \times Q$, criando, para cada imagem I_n, Q^2 blocos da imagem original, a depender de Q, como visto na Figura 3.3. Para cada bloco de I_n, é aplicado o algoritmo de medição de foco, a saída deste algoritmo retorna um parâmetro denominado de *subScore*, dizendo o quanto aquele bloco de imagem está focada ou não.

Com os valores de *subScores* de todas as imagens do *focal-stack*, escolhe-se um número qualquer de N imagens, por bloco, com os maiores valores de *subScores* para utilização no algoritmo de fusão. Para evitar imagens repetidas, é feita uma filtragem para retirar índices escolhidos duas ou mais vezes a partir do método.

Um exemplo pode ser visto na confecção da Figura 3.5, por meio das imagens da Figura 3.4. Para este caso, utiliza-se $Q = 5$ e $N = 1$. Para cada bloco que a imagem é separada, é feita a escolha das N melhores imagens para tal bloco a partir de seu *subScore*. Essa escolha pode ser vista na Tabela 3.1, onde cada célula é um bloco da imagem, e seu conteúdo corresponde a imagem escolhida para aquele bloco. Como $N = 1$, utiliza-se uma imagem de cada bloco,

portanto, para o primeiro bloco, a imagem 115 tem maior *subScore*, logo ela será utilizada para formação da imagem final. O mesmo pode ser dito da imagem 116, que tem o maior foco medido no bloco 3 e assim por diante. As nove imagens escolhidas para este podem ser observadas na Figura 3.4 e a imagem final na Figura 3.5.

Imagem escolhida por bloco				
115	115	116	116	115
258	258	258	258	115
241	240	241	240	241
115	115	115	115	115
70	64	60	61	70

Tabela 3.1: Imagens com maior *subScore* em cada bloco.

(a) Imagem 60 do *focal-stack*. (b) Imagem 61 do *focal-stack*. (c) Imagem 64 do *focal-stack*.

(d) Imagem 70 do *focal-stack*. (e) Imagem 115 do *focal-stack*. (f) Imagem 116 do *focal-stack*.

(g) Imagem 240 do *focal-stack*. (h) Imagem 241 do *focal-stack*. (i) Imagem 258 do *focal-stack*.

Figura 3.4: Imagens escolhidas pelo algoritmo 2.

Figura 3.5: Imagem final formada pelas imagens da Figura 3.4.

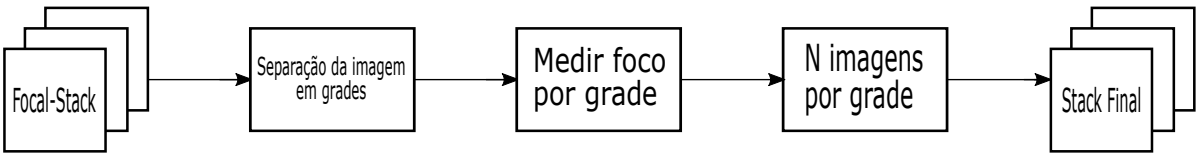


Figura 3.6: Diagrama de blocos do Algoritmo 2.

Assumindo que não se repita nenhuma escolha, este método cria um *stack* final com no máximo $Q^2 \cdot N$ imagens.

3.3.3 Algoritmo proposto 3

Continuando com a validação de cada entrada, este método propõe a validação por meio do uso de um mapa de profundidade, além do algoritmo de medição de foco. Como discutido em 2.4, um mapa de profundidade de uma imagem registra o quão próximo ou afastado do observador cada pixel se encontra. Por meio desta informação, a escolha de imagens com uma grande variação de profundidade é feita com mais garantia que se contribui mais para o algoritmo de fusão. Esse detalhe de profundidade não é notado pelo algoritmo de medição de foco apenas. Um exemplo deste mapa pode ser visto na Figura 3.7, neste caso, quanto mais perto do observador, mais escuro.

Figura 3.7: Exemplo de um mapa de profundidade.

Por questões de clareza, o processo do algoritmo será divido em duas etapas.

3.3.3.1 Geração das máscaras

Na primeira etapa do algoritmo, cria-se máscaras que serão aplicadas às imagens do *focal-stack*.

Utilizando o mapa de profundidade, calcula-se seu histograma. O histograma de uma imagem é a representação da distribuição dos valores de pixels da imagem, ou seja, a quantidade de vezes que dado valor de pixel é encontrado na imagem. Um exemplo de histograma é visto na Figura 3.8.

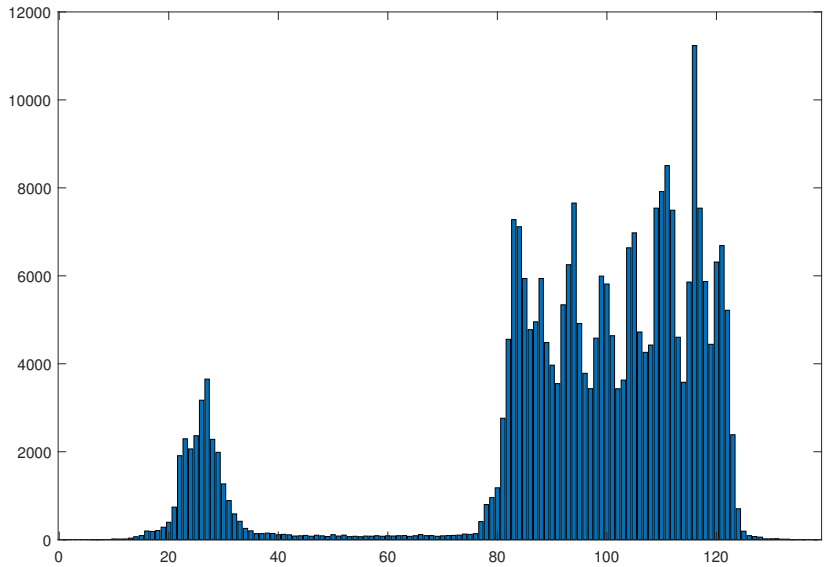


Figura 3.8: Histograma da Figura 3.7.

Define-se a energia total do histograma E_{total} como a soma de todos os valores de pixels que se encontram na imagem. Onde p é o valor do pixel, L é o maior valor de pixel encontrado na imagem e $H(p)$ a quantidade de vezes que este pixel se repete na imagem.

$$E_{total} = \sum_{p=0}^{L} H(p) \tag{3.5}$$

De maneira análoga, define-se a energia parcial $E_{parcial}$ de um histograma como a energia até certo valor de pixel K, podendo ser qualquer valor de pixel encontrado na imagem.

$$E_{parcial} = \sum_{p=0}^{K} H(p) \tag{3.6}$$

A partir das Equações 3.5 e 3.6, cria-se um limiar de energia, ou seja, a razão entre a energia parcial e a energia total para certo valor de K. O limiar inicial de energia é dado pelo valor M de máscaras a serem criadas, este limiar é dado pelo inverso de M. Com estes conceitos definidos, a geração das máscaras é procedida da seguinte maneira.

Para cada valor de K do histograma, calcula-se sua energia parcial (Equação 3.6) e o limiar (razão entre as Equações 3.6 e 3.5) para aquele valor de K. Caso este limiar seja maior ou igual que o limiar definido pelas M máscaras, uma lista de M intervalos é atualizada com o valor de K e novo limiar é calculado a partir a soma do limiar antigo e o limiar inicial.

Dessa forma, entre o primeiro e segundo elemento da lista contém um intervalo de valores pixels que corresponde a $\frac{1}{M}\%$ da energia do histograma. Assim como do segundo pro terceiro elemento e assim por diante. Baseando-se nestes intervalos, são criadas M máscaras verificando o mapa de profundidade. Verifica-se cada pixel do mapa, caso o valor deste pixel esteja contido no intervalo, o mesmo pixel, na máscara, recebe valor 1, caso contrário, 0.

(a) Pixels entre 0 e 49. (b) Pixels entre 50 e 84. (c) Pixels entre 85 e 90.

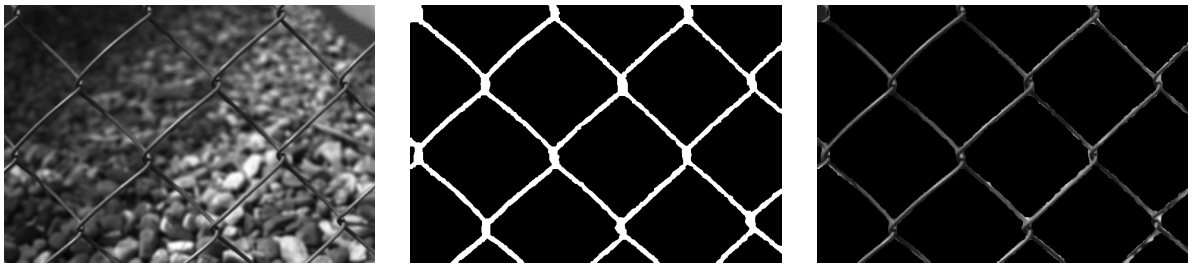
Figura 3.9: Exemplos de máscaras geradas.

Na figura 3.9, o pixel em branco é o valor 1 e o pixel em preto o valor 0. Por exemplo, na Figura 3.9a todos os pixels em branco tinham valores entre 0 e 49 no mapa de profundidade.

3.3.3.2 Escolha das imagens

Na segunda etapa, aplica-se as máscaras e se faz a escolha das melhores imagens.

Com as máscaras criadas, aplica-se cada uma delas em cada uma das imagens do *focal-stack*. O resultado dessa operação é uma imagem com pixels apenas onde se tinha valor 1 na máscara. Um exemplo disso é a imagem 3.10c.



(a) Imagem original (b) Máscara (c) Imagem filtrada pela máscara

Figura 3.10: Exemplos de uma imagem filtrada pela máscara.

A partir das imagens filtradas calcula-se, por meio de um medidor de foco, uma pontuação de foco *depthScore*. Por meio deste valor, escolhe-se as N melhores imagens, com as maiores pontuações, para cada máscara utilizada na filtragem das figuras iniciais.

Um exemplo de imagem gerada é visto na Figura 3.12, formada pelas imagens da Figura 3.11. Este caso tem $M = 10$ e $N = 1$. Em cada imagem filtrada pela máscara, se mede seu *depthScore* e é feita a escolha das N imagens. As imagens escolhidas para este caso são vistas na Tabela 3.2. Dessa forma, é possível ver que para a primeira máscara, a imagem 180 do *focal-stack* possui maior *depthScore* e, portanto, será utilizada para formação da imagem final. O mesmo pode ser dito da imagem 77 para a segunda máscara, e assim por diante. Neste caso $N = 1$, mas caso se aumentasse este valor, seriam escolhidas mais imagens por máscara. As imagens listadas na Tabela 3.2 podem ser vistas na Figura 3.11.

Imagem escolhida por máscara				
180	77	115	116	142
54	47	36	180	10

Tabela 3.2: Imagens com maior *depthScore* em cada máscara.

(a) Imagem 10 do *focal-stack*.

(b) Imagem 36 do *focal-stack*.

(c) Imagem 47 do *focal-stack*.

(d) Imagem 54 do *focal-stack*.

(e) Imagem 77 do *focal-stack*.

(f) Imagem 115 do *focal-stack*.

(g) Imagem 116 do *focal-stack*.

(h) Imagem 142 do *focal-stack*.

(i) Imagem 180 do *focal-stack*.

Figura 3.11: Imagens escolhidas pelo algoritmo 3.

Figura 3.12: Imagem final formada pelas imagens da Figura 3.11.

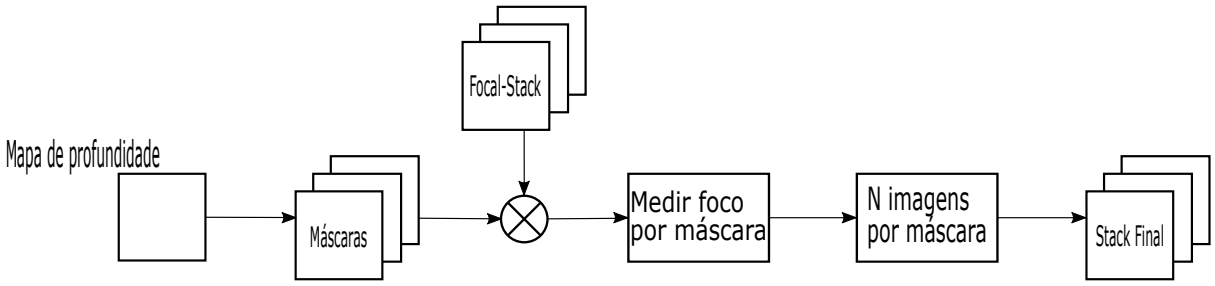


Figura 3.13: Diagrama de blocos do Algoritmo 3.

Assumindo que não se repita nenhuma escolha, este método cria um *stack* final de no máximo $N \cdot M$ imagens.

4 RESULTADOS

Primeiramente, apresenta-se os resultados de cada algoritmo proposto seguida de uma breve análise sobre cada um. O algoritmo proposto por Natália et. al. [7] será usado como parâmetro de comparação ao final do capítulo. Utiliza-se como métrica para avaliação de ambos os algoritmos, principalmente a qualidade da imagem final e o *score* atribuído pelo algoritmo de medição de foco.

4.1 ALGORITMO 2

Como discutido na seção 3.3.2, este método divide a imagem em grades $Q \times Q$. O foco de cada subimagem é medido e é feita a escolha das N com os maiores valores medidos, por subimagem. Criando um *stack* final com no máximo $Q^2 \cdot N$.

Para este algoritmo é feita avaliação do número de subimagens e número de imagens escolhidas, portanto, utilizou-se $Q = 2, 3, 4$ e 5 e $N = 1, 2, 3$ e 4. Dessa forma, para a fusão, se dispõe de 2 a 100 imagens, caso não se repita nenhuma escolha. Baseado nestes números, crima-se 16 imagens finais, uma para cada caso.

Na Figura 4.1 é possível observar todas as imagens geradas com base no banco de dados *Palais de Luxembourg*. Para este caso, consegue-se observar a diferença entre as imagens conforme se aumenta o valor de Q. Comparando as Figuras 4.1a, 4.1e e 4.1i, é possível perceber a diferença no detalhe da cerca que está à frente da imagem. Também observa-se um limite neste aumento de Q, já que a diferença entre as Figuras 4.1i e 4.1m não é tão grande como nas outras.

Outro exemplo deste caso é visto nas Figuras 4.2a, 4.2b e 4.2c. Neste caso, percebe-se, que, com o aumento de Q, a região inferior da imagem se torna mais nítida.

Observando as figuras, não é tão nítida a diferença no aumento de N. Esta diferença pode ser vista ao olhar para o *score* de cada imagem final. A partir da Tabela 4.1, percebe-se que com o aumento de N, o valor de *score* também é elevado, mesmo que pouco.

Outro ponto a ser discutido é a qualidade da foto em função da quantidade de imagens utilizadas para fusão. Com $Q = 4$ e $N = 1$, utiliza-se 10 imagens. Aumentando o número de imagens a partir deste ponto, não se gera uma diferença notável no resultado final, com Q ou N maiores. Essa diferença também é vista comparando as Tabelas 4.1 e 4.2. Percebe-se que não acontece um aumento significativo nos *scores* ao aumentar o número de imagens utilizadas.

(a) $Q = 2, N = 1$ (b) $Q = 2, N = 2$ (c) $Q = 2, N = 3$ (d) $Q = 2, N = 4$

(e) $Q = 3, N = 1$ (f) $Q = 3, N = 2$ (g) $Q = 3, N = 3$ (h) $Q = 3, N = 4$

(i) $Q = 4, N = 1$ (j) $Q = 4, N = 2$ (k) $Q = 4, N = 3$ (l) $Q = 4, N = 4$

(m) $Q = 5, N = 1$ (n) $Q = 5, N = 2$ (o) $Q = 5, N = 3$ (p) $Q = 5, N = 4$

Figura 4.1: Imagens geradas a partir do banco *Palais du Luxembourg*.

Q \ N	1	2	3	4
2	17,50	17,54	17,55	17,56
3	19,13	19,15	19,16	19,19
4	19,54	19,55	19,56	19,57
5	19,55	19,56	19,56	19,57

Tabela 4.1: Valores de *scores* para cada imagem gerada.

Q \ N	1	2	3	4
2	3	7	9	12
3	6	14	20	24
4	10	17	23	30
5	9	19	25	34

Tabela 4.2: Número de imagens para cada imagem gerada.

(a) $Q = 3$ (b) $Q = 4$ (c) $Q = 5$

Figura 4.2: Influência de Q na imagem final.

Baseado nestes resultados, a influência da variação de Q é mais nítida que a variação de N. Isso pode se dar ao fato de que, apesar de se poder escolher mais imagens, as imagens que tem mais influência no resultado final já são escolhidas previamente. Como é mostrado na Tabela 4.1, apesar de se dispor de 100 imagens no caso $Q = 5$ e $N = 4$, menos de 40% dessas imagens são escolhidas. Portanto, o aumento do número de imagens no algoritmo de fusão se torna apenas um ajuste fino para uma pequena melhora em seus *scores*.

O método em questão consegue escolher figuras evitando redundância, já que, mesmo com poucas imagens, cria-se um resultado nítido e que consegue aumentar a profundidade de campo. Porém, apesar de funcionar em alguns casos, em outros o algoritmo não consegue capturar toda a informação desejada. Por exemplo, quando uma imagem tem sua informação concentrada em um plano da foto, o algoritmo não leva em consideração os outros planos.

(a) (b) (c)

Figura 4.3: Imagens onde a informação necessária não foi utilizada.

Nas Figuras 4.3a, 4.3b, a maior parte da informação se encontra no fundo da foto, e a cerca que se encontra na frente não é levada em consideração, fazendo com que a imagem final não tenha foco nestes locais. O contrário também acontece, como na Figura 4.3c, onde a informação está principalmente na frente, fazendo com que o algoritmo ignore o banco ao fundo da foto.

Todas as melhores imagens serão dispostas ao final do capítulo, para comparação com os outros métodos.

4.2 ALGORITMO 3

Este algoritmo, como discutido em 3.3.3, tem como propósito a utilização de mapas de profundidade para escolher as imagens. Tentando melhorar os resultados apresentados em 4.1, são criadas máscaras a partir dos vários níveis de profundidade das imagens, tentando assim, buscar informações que não foram utilizadas no outro método.

Utilizando M máscaras, se mede o foco da imagem com cada máscara aplicada e se escolhe as N melhores imagens para cada máscara. Dessa maneira, são utilizadas no máximo $M \cdot N$ imagens como entrada do algoritmo de fusão. Para avaliação deste método, varia-se o número de máscaras em $M = 10, 15$ e o número de imagens escolhidas em $N = 1, 2, 3$ e 4, criando assim, uma faixa entre 10 e 60 imagens de entrada. O número M foi limitado em 15 pois, ao se fazer 20 máscaras, percebe-se que muitas delas não contém mais informação, às vezes contendo apenas um valor de pixel.

(a) $M = 10, N = 1$ (b) $M = 10, N = 2$ (c) $M = 10, N = 3$ (d) $M = 10, N = 4$

(e) $M = 15, N = 1$ (f) $M = 15, N = 2$ (g) $M = 15, N = 3$ (h) $M = 15, N = 4$

Figura 4.4: Imagens geradas a partir do banco *Palais du Luxembourg*.

Na Figura 4.4 são mostradas todas as imagens geradas a partir do banco de dados *Palais de Luxembourg*. Dessa vez, não se percebe muita diferença entre as imagens finais, todas já tem um grande nível de nitidez, inclusive na cerca que compõe a imagem.

A diferença entre as imagens se dá para o valor de *score* atribuído para cada, contemplado na Tabela 4.3. Os valores de *scores* não sofrem tanta alteração, mostrando que, apesar de aumentar o número de máscaras, a diferença não é tão grande. Ao se aumentar o número de imagens N, para $M = 10$, houve um pequeno crescimento em questão do *score*, porém, para $M = 15$, houve uma redução. Essa redução pode ter acontecido por escolher imagens que contém informações que não acrescentam no resultado final, como ruído.

Avaliando as imagens e observando a Tabela 4.4, pode-se discutir a qualidade dos resultados em função do número de imagens utilizadas para cada figura final. Para $M = 10$ e $N = 1$ já é feita uma imagem que obtém grande nitidez em todo local da foto, com o uso de apenas 8

imagens.

M \ N	1	2	3	4
10	19,46	19,48	19,52	19,56
15	19,56	19,61	19,61	19,59

Tabela 4.3: Valores de *scores* para cada imagem gerada.

M \ N	1	2	3	4
10	8	14	21	27
15	8	17	26	31

Tabela 4.4: Número de imagens utilizadas para gerar cada imagem.

Neste caso, não houve grande diferença entre as imagens, tanto na variação de M quanto na variação de N, em nenhum dos bancos de fotos utilizados, a maioria das imagens geradas são muito parecidas e só se percebe a diferença no *score*. Porém, o maior problema encontrado no último algoritmo foi melhorado. As Figuras 4.5a, 4.5b e 4.5c mostram como o uso do mapa de profundidade causa uma melhora no resultado final. A cerca nas duas primeiras tornam-se muito mais nítidas e na última, elementos do fundo da foto encontram-se mais em foco.

(a) (b) (c)

Figura 4.5: Exemplo da melhora das imagens geradas.

O método consegue gerar imagens com a maior parte em foco, aumentando sua profundidade de campo. Como não houve muita variação com os valores de M, geram-se imagens bem nítidas com uma pequena quantidade de imagens de entrada, e, como no outro caso, os valores de N se tornam ajuste fino na geração da imagem. Além disso, ainda consegue corrigir as imagens onde se tem muita variação de profundidade.

4.3 COMPARAÇÃO COM ALGORITMO 1

Por fim, será feita uma comparação com o primeiro algoritmo, onde se mostra todas as melhores imagens de cada banco de dados, bem como seus *scores*.

(a)

(b)

(c)

(d)

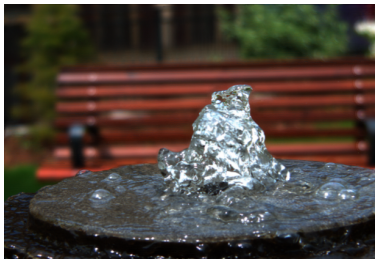
(e)

(f)

(g)

(h)

(i)



(j)

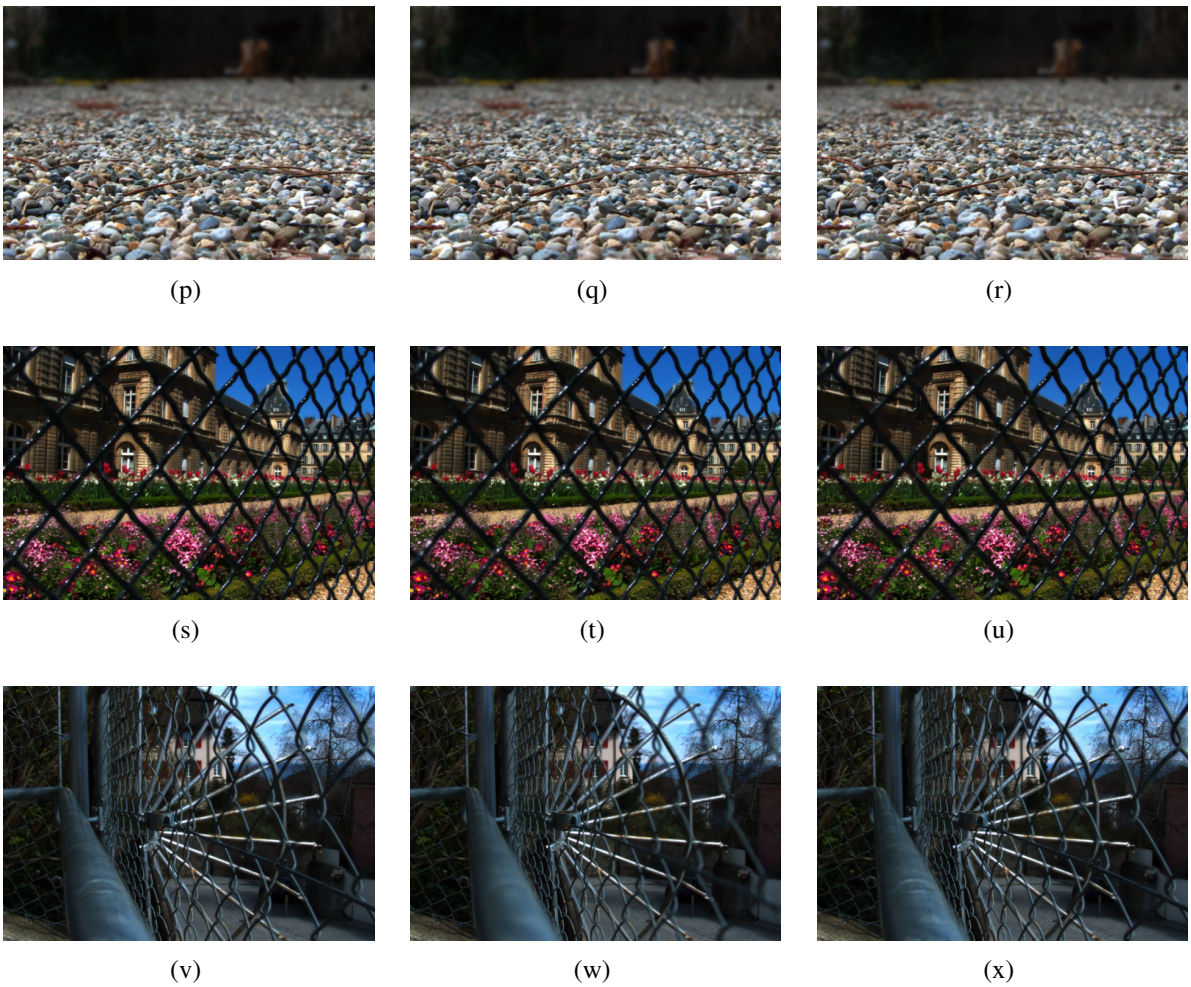
(k)

(l)

(m)

(n)

(o)



(p) (q) (r)

(s) (t) (u)

(v) (w) (x)

Figura 4.6: Melhores imagens de cada algoritmo. Primeira coluna, algoritmo 1. Segundo coluna, algoritmo 2. Terceira coluna, algoritmo 3.

A melhor imagem foi escolhida baseando-se no *score* dado. Os valores destes *scores* podem ser vistos na Tabela 4.5.

Scores			
Banco de Imagens	Algoritmo 1	Algoritmo 2	Algoritmo 3
Bikes	10,03	9,39	9,54
Danger de Mort	10,28	9,96	9,95
Chain-link Fence 2	19,05	17,72	18,89
Fountain Bench	11,05	9,54	9,40
Friends 3	7,84	7,30	7,53
Gravel Garden	17,04	15,86	16,11
Palais du Luxembourg	20,47	19,57	19,61
Spear Fence 2	15,09	13,56	14,40

Tabela 4.5: Valores dos maiores *scores* gerados de cada algoritmo.

Número de imagens			
Banco de Imagens	Algoritmo 1	Algoritmo 2	Algoritmo 3
Bikes	50	44	11
Danger de Mort	50	37	23
Chain-link Fence 2	50	10	41
Fountain & Bench	50	30	36
Friends 3	50	23	35
Gravel Garden	50	9	39
Palais du Luxembourg	50	30	17
Spear Fence 2	50	15	47

Tabela 4.6: Número de imagens para gerar cada imagem de cada algoritmo.

Na Tabela 4.6 se encontra a quantidade de imagens utilizadas para gerar cada resultado. Como o critério para decidir as melhores imagens foi o *score*, essa tabela é baseada nestes valores. Apesar dos *scores* do algoritmo 1 serem maiores que o restante, as imagens construídas pelos outros algoritmos não deixam a desejar, visto que, utilizando menos imagens, foi possível obter resultados que se equiparam visualmente com os do algoritmo 1. Imagens como estas podem ser observadas na Figura 4.7.

As imagens apresentadas na Figura 4.7 mostram como o resultado do método 3 é comparável diretamente com os resultados já obtidos do método 1. Porém, utiliza-se uma quantidade bem menor de imagens para chegar a um resultado significativamente parecido com outro método que utiliza um número maior de imagens.

Apesar do método 2 deixar a desejar em alguns aspectos já mencionados na Seção 4.1, também foi possível gerar resultados satisfatórios com menos imagens que o algoritmo 1. A única exceção são as imagens em que existem muita informação em diferentes profundidades da imagem.

Como foi observado, a diferença entre os *scores* dos 3 algoritmos é nítida. Porém, apesar de ser um bom meio de validação dessas imagens, o critério visual ainda é predominantemente mais importante. Isto se deve ao fato de que, para uma imagem qualquer, o medidor de foco pode atribuir um alto valor de *score*, porém a imagem pode conter falhas devido a fusão. Portanto, o critério de decisão de sucesso dos algoritmos é feito pelas imagens resultantes de cada um.

(a) 50 imagens utilizadas.

(b) 9 imagens utilizadas.

(c) 50 imagens utilizadas.

(d) 7 imagens utilizadas.

(e) 50 imagens utilizadas.

(f) 7 imagens utilizadas.

(g) 50 imagens utilizadas.

(h) 9 imagens utilizadas.

Figura 4.7: Imagens comparando o algoritmo 1 (esquerda) com o algoritmo 3 (direita).

5 CONCLUSÃO

A principal motivação deste trabalho foi apresentar uma nova maneira de se determinar quais seriam as melhores imagens para se criar uma imagem com grande profundidade de campo. Estipulou-se então, dois métodos. O primeiro separando a imagem em várias subimagens, determinando qual delas teria sua área mais em foco, e assim, fazendo a escolha. Outro método, baseado em mapas de profundidade, separa a imagem em níveis de profundidade baseados em seu histograma, e então com um medidor de foco, se escolhe as imagens que são mais relevantes.

Os resultados apresentados foram comparados com outro método que já gerava boas imagens. Algumas figuras geradas pelos métodos propostos não conseguiram superar o algoritmo já existente, ao se utilizar uma das métricas. Porém, visualizando as imagens geradas por ambos os métodos, o método proposto consegue chegar bem próximo escolhendo uma quantidade considerável menor de amostras, mostrando que escolher as imagens baseando-se no mapa de profundidade delas é uma maneira promissora para se realizar fusão de imagens.

Em suma, a partir dos resultados obtidos, o trabalho desenvolvido sugere um novo caminho para a geração de imagens de grande profundidade de campo, introduzindo técnicas para desenvolvimento de futuros trabalhos. Os mapas de profundidade, caso inseridos em outras partes do algoritmo, demonstra ser bastante promissora.

5.1 DESENVOLVIMENTO FUTURO

Como já mencionado, existe a possibilidade de introduzir os mapas de profundidade em outros locais do algoritmo. Um dos locais onde pode ser integrado é no mapa de ação do algoritmo de fusão. Escolhendo pixels vizinhos que correspondem ao mesmo nível de profundidade pode resultar em imagens melhores. Outra proposta seria fazer uma junção de ambos os algoritmos propostos, tentando assim, utilizar menos imagens ainda para a fusão.

Além disso, propõe-se a utilização dos algoritmos em pilhas de foco geradas manualmente por uma câmera convencional, sem utilização da pilha artificial confeccionada a partir do *Light Field* da imagem.

REFERÊNCIAS BIBLIOGRÁFICAS

1 JOYNER, H.; MONAGHAN, K. *Focus on Photography*. Davis Publications, Incorporated, 2006. (The Davis Studio Series). ISBN 9780871927446. Disponível em: <https://books.google.com.br/books?id=aHuWPQAACAAJ>.

2 Disponível em: <https://en.wikipedia.org/wiki/Depth_of_field>.

3 LI, S.; KWOK, J. T.; WANG, Y. Multifocus image fusion using artificial neural networks. *Pattern Recognition Letters*, v. 23, n. 8, p. 985–997, jun 2002.

4 LIAO, Z.; HU, S.; TANG, Y. Region-based multifocus image fusion based on hough transform and wavelet domain hidden markov models. *2005 International Conference on Machine Learning and Cybernetics*, v. 9, n. 9, p. 5490–5495, aug 2005.

5 BILCU, R. C.; ALENIUS, S.; VEHVILAINEN, M. A novel method for multi-focus image fusion. *16th IEEE International Conference on Image Processing (ICIP)*, v. 1, p. 1525–1528, 2009.

6 BOGONI, L.; HANSEN, M.; BURT, P. Image enhancement using pattern-selective color image fusion. *Proceedings 10th International Conference on Image Analysis and Processing*, p. 44–49, sep 1999.

7 TEIXEIRA, N.; PEIXOTO, E. Produção de imagens all-in-focus a partir de uma grande amostra de imagens com diferentes focos. 2018.

8 BURT, P.; ADELSON, E. H. The laplacian pyramid as a compact image code. *IEEE Transactions on Communications*, v. 31, n. 4, p. 532–540, apr 1983.

9 Disponível em: <https://www.creativelive.com/photography-guides/what-is-shutter-speed>.

10 Disponível em: <https://digital-photography-school.com/fun-shutter-speed-motion-blur/>.

11 Disponível em: <https://photographylife.com/what-is-aperture-in-photography>.

12 NG, R.; LEVOY, M.; BRéDIF, M.; DUVAL, G.; HOROWITZ, M.; HANRAHAN, P. Light field photography with a hand-held plenoptic camera. *Stanford University Computer Science Tech Report CSTR 2005-02*, apr 2005.

13 Disponível em: <https://en.wikipedia.org/wiki/Pyramid_(image_processing)>.

14 TAO, M. W.; HADAP, S.; MALIK, J.; RAMAMOORTHI, R. Depth from combining defocus and correspondence using light-field cameras. In: . [s.n.], 2013. Disponível em: <http://graphics.berkeley.edu/papers/Tao-DFC-2013-12/>.

15 THELEN, A.; FREY, S.; HIRSCH, S.; HERING, P. Improvements in shape-from-focous for holographic reconstructions with regard to focus operators, neighborhood-size, and height value interpolation. *IEEE Transactions on Image Processing*, v. 18, n. 1, p. 151–157, nov 2009.

16 AHMAD, M. B.; CHOI, T. S. Application of three dimensional shape from image focus in lcd/tft displays manufacturing. *IEEE Transactions on Consumer Electronics*, v. 53, n. 1, p. 1–4, apr 2007.

17 PERTUZ, S.; PUIG, D.; GARCIA, M. A. Analysis of focus measure operators for shape-from-focus. *Pattern Recognition Letters*, v. 46, n. 5, p. 1415–1432, may 2013.

18 NAYAR, S. K.; NAKAGAWA, Y. Shape from focus. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 16, n. 8, p. 824–831, aug 1994.

19 ŘEřáBEK, M.; EBRAHIMI, T. New light field image dataset. *8th International Workshop on Quality of Multimedia Experience (QoMEX), Lisbon, Portugal*, jan 2016. Disponível em: <http://mmspg.epfl.ch/EPFL-light-field-image-dataset>.