Aplicação de estereoscopia em imagens digitais

Trabalho de Conclusão de Curso

Engenharia da Computação

Renan de Freitas Leite
Orientador: Prof. Bruno José Torres Fernandes

Universidade de Pernambuco
Escola Politécnica de Pernambuco
Graduação em Engenharia de Computação

RENAN DE FREITAS LEITE

APLICAÇÃO DE ESTEREOSCOPIA EM IMAGENS DIGITAIS

Monografia apresentada como requisito parcial para obtenção do diploma de Bacharel em Engenharia de Computação pela Escola Politécnica de Pernambuco – Universidade de Pernambuco.

Recife, Junho de 2014.

De acordo

Recife

____/_____/_____

_____
Orientador da Monografia

(Na versão final esta página deve ser substituída pela folha de aprovação digitalizada.)

Resumo

A estereoscopia é uma técnica usada pelo sistema visual humano para a concepção de imagens tridimensionais a partir do processamento inconsciente do cérebro das informações das imagens bidimensionais capturadas em perspectivas diferentes por cada olho. Na área de visão computacional, esta mesma técnica pode analogamente ser empregada para encontrar as distâncias entre os objetos e um par de câmeras digitais do mesmo modelo posicionadas lado a lado de maneira similar aos olhos humanos. Dessa forma será necessário conhecer as características ópticas das câmeras, tais como a distância focal e o ângulo de convergência, assim como a resolução, o tamanho do sensor e a distância entre elas. Será necessário saber como manipular a imagem em uma linguagem de programação através de uma matriz com pares de coordenadas para cada pixel.

Uma das etapas da estereoscopia é parear as duas projeções de um mesmo ponto em cada câmera, porém isso pode ser uma tarefa complexa porque haverá uma abordagem estatística das diversas características de cada região das duas imagens. Essas abordagens além de serem complexas têm um custo computacional elevado e chances de erro. Para simplificar esse procedimento, será utilizado um laser semicondutor para marcar um mesmo ponto nas duas projeções e o critério de pareamento será simplificado para dois fatores exatos: a cor do laser e a altura das projeções.

Neste trabalho, será abordada a aplicação de estereoscopia em imagens digitais, seguindo o modelo binocular humano, para mapear ou “escanear” em três dimensões o cenário capturado por duas câmeras. Pode-se dizer que as câmeras estarão imitando os olhos humanos, porque o sistema visual humano usa estereovisão binocular, que após a sua concepção, possibilita uma pessoa ter a referência em profundidade daquilo que é visto. Isso é possível através da disparidade entre as imagens, podendo ser obtidos resultados com um nível de precisão aceitável. Existem outros métodos que usam outros tipos de sensores e a partir da resposta deles é possível mapear distâncias. Os sensores mais usados atualmente são: o sensor infravermelho que emite uma luz infravermelha para que o seu tempo de reflexão ou sua densidade revele a distância dos objetos ou elementos do cenário; o sensor de ultrassom que usa a mesma ideia do sensor de

infravermelho, usando o sinal ultrassônico. As câmeras destinadas a captar as imagens devem ser posicionadas lado a lado em uma distância não muito longa, pois a sensibilidade do sistema estará em distâncias próximas às câmeras. As imagens adquiridas pelas câmeras necessitam ser enviadas a um computador onde serão processadas através dos cálculos para a aquisição de uma imagem de profundidade.

Abstract

Stereoscopy is a technique used by the human system for designing three-dimensional images from the unconscious brain processing of information from two-dimensional images captures from different perspectives for each eye. In computer vision, this same idea can similarly be used to find the distances between objects and a pair of digital cameras, with the same model, positioned side by side in a similar way to the human eyes. Thus it will be necessary to know the optical characteristics of the camera, such as focal length and the angle of convergence, as well as the resolution, sensor sizes and the distance between them. It will need to know how to manipulate the image in a programming language by a matrix with pair of coordinates for each pixel.

One of the stages of stereoscopy is to pair the two projection of the same point in each camera, but it can be a complex task because there is a statistical approach to the different characteristics of each region of each images. These approaches besides being complex have a high computational cost and chances of error. To simplify this procedure, a semiconductor laser is used to mark the same point in the two projections and the criteria for pairing is simplified for two exact factors: the laser color and the height of the projections.

In this paper will address the stereoscopic application on digital images, following the human binocular model to map or to scan in three dimensions the scene captured by two digital cameras. It can be said that the cameras are going to imitate the human eyes, because the human visual system use binocular stereovision, that after its conception, it enables a person to get reference deph in what is seen. This is possible by disparity between the images with a acceptable accuracy level. Other methods use other types of sensors to map distances. Most sensors used today are: infrared sensor that emits an infrared light to its reflection time or reflection density reveals the distances of objects or elements of the scenario; the ultrasonic sensor that uses the same idea of the infrared but this use the ultrasonic signal. Cameras designed to capture the images should be positioned side by side in a not very long distance, because the system sensitivity will be in close distances to the digital

cameras. The images acquired by the cameras need to be sent to computer which will be processed through the calculations for the acquisition of a depth image.

Sumário

Capítulo 1 Introdução **1**

Objetivos 2

Estrutura da monografia 3

Capítulo 2 Imagens digitais **5**

2.1 Aquisição de imagens 5

2.2 Captação de cores 8

2.3 Modelos de câmeras 10

2.4 Conceito de imagem digital 13

Capítulo 3 Estereoscopia **16**

3.1 Princípios de triangulação 17

3.2 Pareamento dos pontos 22

Capítulo 4 Laser semicondutor **23**

4.1 Introdução ao laser e aos semicondutores 23

4.2 Princípio de funcionamento de um laser 24

4.3 Materiais utilizados 28

Capítulo 5 Modelo proposto **30**

Capítulo 6 Resultados experimentais **35**

Capítulo 7 Conclusão **39**

Apendice A **40**

Introdução ao MATLAB 40

MATLAB como ferramenta de processamento de imagens digitais 40

Representação de uma imagem digital no MATLAB 41

Imagens digitais como matrizes 42

Capturando as imagens das câmeras digitais com o MATLAB 44

Visualizando as imagens com o MATLAB 45

Programação eficiente no MATLAB 46

Pré alocação de matrizes 46

Vetorização 47

Programação em GPU no MATLAB 49

Apêndice B Código da aplicação em MATLAB 52

Referências 55

Índice de Figuras

Figura 1. (a) diagrama esquemático do globo ocular humano, (b) perspertiva microscópica da retina humana composta por 1 cone, 2 células bipolares, 9 bastonetes e 3 axônios de células ganglionares do nervo óptico. [1].....6

Figura 2. (a) sensor CCD, (b) sensor CCD junto a um circuito de uma câmera digital. [1].....7

Figura 3. O sensor CMOS. [1] 7

Figura 4. As diversas resoluções possíveis para uma mesma imagem em medição de *Megapixel.* [5].....8

Figura 5. Imagem colorida resultante da junção das informações dos três sensores vermelho, verde e azul por meio de um divisor de feixes. [5] 9

Figura 6. O filtro de Bayer em um sensor de câmera digital. [1].....10

Figura 7. Modificação da distância focal para obter o *zoom* centralizado de uma mesma imagem.....11

Figura 8. (a) captura de uma imagem em uma câmera *pinhole*, (b) captura de uma imagem após a luz passar por uma lente convexa. 11

Figura 9. Afastamento de um objeto e seu efeito de redução de tamanho na imagem capturada. [1].....12

Figura 10. (a) Esquema de uma captação vista de cima, de objetos em posições diferentes, (b) Imagem da captação. [1].....12

Figura 11. (a) Exemplo de um imagem monocromática, (b) Uma visão minuciosa de uma pequena área da imagem, (c) Os valores de cada *pixel* da imagem. [5] 14

Figura 12. Valores em hexadecimal e decimal de algumas cores no modelo RGB. [1] 14

Figura 13. Imagem decomposta em 3 componentes. [5].....15

Figura 14. Triangulação para encontrar a distância D. [1].....18

Figura 15. Visão de cima da situação de visão estereoscópica em conjunto com a ideia de ângulo de convergência e distância focal. [1] 19

Figura 16. Perda de precisão do deslocamento y das imagens conforme se aumenta a distância x dos pontos a serem emparelhados com a estereoscopia. 21

Figura 17. Dopagem de um semicondutor.....24

Figura 18. (a) Junção heterogênea dos semicondutores AlGaAs (mais pesado) e GaAs (mais leve). (b) Diagrama de da banda de energia do esquema da junção heterogênea (observa-se o “gargalo” de elétrons). (c) Gráfico que mostra a situação de emissão estimulada através da relação oferta-demanda de energia. [20] 25

Figura 19. O laser semicondutor possui uma cavidade óptica para refletir a luz para fora do semicondutor como um feixe. [20] 27

Figura 20. (a) Comparação da potencia de luz emitida por um laser e por um LED em relação a corrente que passa pelos semicondutores. (b) Comparação dos comprimentos de onda presentes na luz do LED e do laser com a distribuição de intensidade para cada comprimento de onda. [20].....28

Figura 21. Esquema de um feixe de luz passando por uma lente cilíndrica. 31

Figura 22. Efeito esbranquiçado do laser. 31

Figura 23. Câmeras digitais (webcams) utilizadas na aplicação. 32

Figura 24. Modelo proposto. 33

Figura 25. (a) Imagem de entrada. (b) Imagem binarizada pela faixa de cor vermelha do laser. 35

Figura 26. Cenário dos experimentos. 36

Figura 27. Resultado da aplicação. (a) A imagem capturada pela câmera direita. (b) A imagem de profundidade. (c) A representação da imagem de profundidade em pseudo-cores para uma melhor visualização. (d) Matrizes de seções relacionadas à imagem de profundidade com as distâncias, em centímetros, dos pontos. 37

Figura 28. Visão matricial de uma imagem (a) em muitas das bibliotecas de imagens (b) na toolbox de processamento de imagens do MATLAB. 43

Figura 29. Representação matemática para a Figura 28 (a). 43

Figura 30. Representação matemática da Figura 28 (b). 44

Figura 31. Uma abordagem superficial da arquitetura de uma CPU e de uma GPU. 50

Índice de Tabelas

Tabela 1. Emissão de comprimentos de onda de vários tipos de lasers semicondutores. [19] 29

Tabela de Símbolos e Siglas

CCD	*Charge coupled device*	Dispositivo de carga acoplada
CMOS	*Complementary metal oxide semicondutor*	Semicondutor metal-óxido complementar
Pixel	*Picture e Element*	Elemento de figura
RGB	*Red, green, blue*	Vermelho, verde, azul
CMY	*Cyan, magenta, yellow*	Ciano, magenta, amarelo
CPU	*Central Processor Unit*	Unidade central de processamento
GPU	*Graphic Processor Unit*	Unidade de processamento gráfico
f	Distância focal.	
f(x, y)	Ponto em coordenadas cartesianas de uma imagem.	
x	Coordenada horizontal de uma imagem.	
y	Coordenada vertical de uma imagem.	
P	Ponto de interesse da triangulação.	
X	Base do triângulo.	
D	Altura do triângulo P, Ce, Cd.	
E	Vértice esquerdo.	
C	Vértice direito.	
α	Ângulo entre X e PE.	
β	Ângulo entre X e PC.	
θ	Ângulo entre D e PE.	
ϒ	Ângulo entre D e PC.	
x'	Distância em X de E até D.	
x''	Distância em X de C até D.	
α1 e α2	Ângulos alternos internos equivalentes a α.	

β1 e β2 Ângulos alternos internos equivalentes a β.

a Distância do centro do sensor esquerdo até onde o ponto de interesse se formará.

a’ Distância de onde o ponto de interesse se formará ate o limite direito do sensor esquerdo.

b Distância do centro do sensor direito até onde o ponto de interesse se formará.

b’ Distância de onde o ponto de interesse se formará ate o limite direito do sensor direito.

s Reta que passa pelos sensores.

r Reta que passa pelos ângulos de convergência dos sensores.

l Largura total das imagens captadas pelos sensores.

Xe Equivalente a l – a'.

Xd Equivalente a b’.

Capítulo 1

Introdução

A visão é um dos principais meios de orientação humana e pode-se dizer também uma das mais complexas de serem estudadas e aplicadas em uma máquina. Dentre suas funcionalidades, a de identificar pessoas e objetos e a de possibilitar a orientação espacial chamam bastante atenção na área de visão computacional. De acordo com a teoria evolucionária, o ser humano se tornou bípede para ter uma ampliação do campo de visão proporcionada pelo aumento da altura dos olhos em relação ao chão. A precisão do sentido visual humano em situações normais é o suficiente para haver a orientação espacial e assim possibilitar, por exemplo, que uma pessoa ande sabendo as possíveis colisões em objetos em seu caminho ou a noção de velocidade do deslocamento dos objetos em relação a esta pessoa.

A visão computacional [1-2] estuda e relata sistemas de visão artificial implementados por *software* ou *hardware*. Isso é possível através do uso de um conjunto de algoritmos que tenta extrair diversas informações a partir de imagens digitais. Porém, devido à complexidade do sistema visual humano, por ser bastante sua implementação em máquinas se trona uma tarefa difícil. A maioria das aplicações se limita a classificar um único tipo de objeto ou faixa de cores em um cenário.

Ao usar a ideia de visão binocular humana, a estereoscopia [3] é o processo fundamental para a concepção de profundidade daquilo que é visto por uma pessoa. É possível haver essa concepção de profundidade com o sistema monocular por causa de alguns fatores como, por exemplo, noção de tamanho relativo e distância focal, porém é bastante limitado e impreciso quando comparado ao sistema binocular.

Para um ser humano [4], fazer uma análise de uma imagem parece ser uma tarefa simples. Isso porque o sistema visual humano tem o cérebro para processar paralelamente tudo que é visto. Ou seja, para que um cálculo aplicado à visão computacional corresponda analogamente ao cérebro humano é necessário que

cada elemento da imagem seja calculado em paralelo. Um algoritmo sequencial deverá ser executado muito mais rapidamente que um algoritmo paralelo para que se tenha o mesmo desempenho, se tratando de um sistema com várias entradas independentes tais como imagens digitais.

Os problemas de localização e classificação de objetos [5-6], detecção de possíveis obstáculos e mapeamento do ambiente [7] são problemas encontrados na robótica que podem ser resolvidos através da aplicação da visão computacional. Um sistema ideal para a segurança e um bom funcionamento do robô é aquele que tenha uma taxa de acerto elevada e uma rápida velocidade de resposta.

A ideia da estereoscopia no sistema visual humano é de emparelhar ou parear os elementos das duas imagens captadas pelo olho esquerdo e direito, havendo assim um deslocamento no sentido do eixo dos olhos. Com esse deslocamento, é possível achar a profundidade através do princípio de triangulação já que as duas projeções adquiridas pelas imagens do olho esquerdo e direito, e os pontos do cenário formam um triângulo. Na visão computacional, de forma análoga ao sistema visual humano, os olhos serão as câmeras digitais e o cérebro será o computador.

Um problema encontrado no pareamento dos elementos das imagens é que muito provavelmente estes elementos não estarão morfologicamente iguais nas suas projeções. O deslocamento e a distância focal irão alterar o formato e o posicionamento dos elementos nas projeções. Porém, para que os cálculos sejam os melhores possíveis é necessário que cada *pixel* da primeira imagem seja exatamente pareado com seu correspondente na segunda imagem. Existem diversas técnicas de custo computacional elevado com a finalidade de emparelhar os elementos das imagens.

Objetivos

Este trabalho tem o objetivo de desenvolver um método capaz de gerar uma imagem de profundidade de um cenário usando apenas duas câmeras digitais comuns. As imagens capturadas pelas câmeras terão suas informações processadas para ser obtida a profundidade de cada elemento pareado e assim ter

esse valor de profundidade armazenado na informação da imagem de profundidade, que é o resultado para avaliação da eficiência do algoritmo.

Para haver a exatidão no pareamento dos elementos da imagem, será usado um laser semicondutor, mais conhecido como apontador, porém com uma modificação de exposição em linha. Isso porque o desenvolvimento de diversos cálculos de profundidade para rápida montagem de uma imagem de profundidade é o objetivo principal deste trabalho.

O algoritmo de montagem será em MATLAB e deverá conter algumas das técnicas de otimização providas pela ferramenta, tais como a pré-alocação de matrizes, a vetorização e a programação em GPU, que serão de fundamental importância para seu aumento de desempenho.

Estrutura da monografia

No Capítulo 2, será feito um estudo abordando o processamento de imagens digitais, que é o assunto essencial para o desenvolvimento da aplicação.

O Capítulo 3, abordará os conceitos de estereoscopia, assim como a geometria envolvida a respeito da triangulação que permite um par de sensores adquirirem a informação de distância dos objetos. A busca dos elementos na imagem também será discutida nesse capítulo. Após a localização dos elementos, o mapeamento por profundidade será feito de uma maneira simplificada através de uma fórmula resultante dos cálculos.

No Capítulo 4, será feito um estudo em relação ao laser semicondutor, abordando desde a eletrônica à óptica. Isso será de importância para a aplicação, pois o objetivo é usar o laser para facilitar a busca dos pontos da imagem a serem mapeados.

No Capítulo 5, será proposta uma implementação para demonstrar uma funcionalidade para os cálculos desenvolvidos bem como o funcionamento da aplicação utilizando o MATLAB.

No Capítulo 6, os resultados experimentais serão mostrados para que seja possível ver na prática o assunto abordado e os objetivos alcançados.

As considerações finais e os trabalhos futuros serão apresentados no Capítulo 7.

No Apêndice A, será abordado o processamento de imagens digitais no MATLAB, tais como o uso de uma programação eficiente dessa ferramenta que melhora o desempenho da aplicação.

No Apêndice B, será mostrado o código em MATLAB da aplicação.

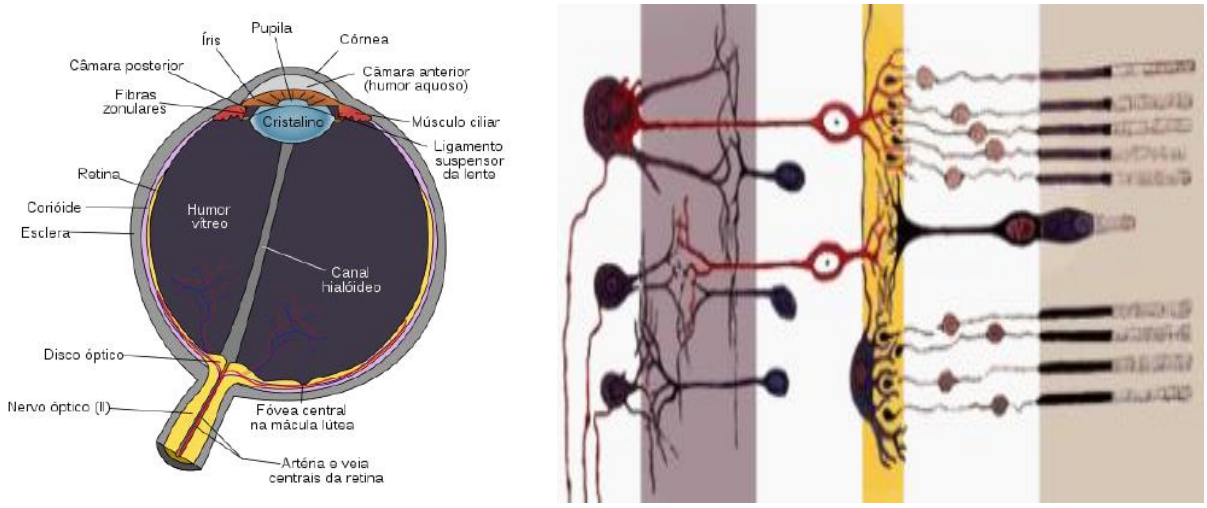
Capítulo 2

Imagens digitais

A visão computacional [5] existe em analogia ao sistema visual humano, e seguindo este raciocínio é cabível um estudo da capacidade humana em visualizar e discernir o cenário ao seu redor. Os olhos atuam como sensores, onde a imagem é projetada e transformada em impulsos elétricos, e a informação proveniente destes chega ao cérebro para ser interpretada.

2.1 Aquisição de imagens

A Figura 1(a) mostra um diagrama esquemático de um olho ou globo ocular de um ser humano (sistema visual saudável) com suas subestruturas internas, dentre elas tem a retina, onde a luz é captada e convertida em sinal nervoso, e o cristalino, onde a imagem é focalizada. O sensoriamento ocular consiste na atuação de dois tipos de células fundamentais: os bastonetes, responsáveis por captar a luminosidade; e os cones, capazes de diferenciar as diversas cores existentes na imagem, e o diagrama esquemático de uma estrutura composta por estas duas células é mostrada na Figura 1(b). Existem várias dessas estruturas espalhadas homogeneamente pela retina, onde todas as informações são preparadas para serem enviadas ao cérebro por meio do nervo óptico.

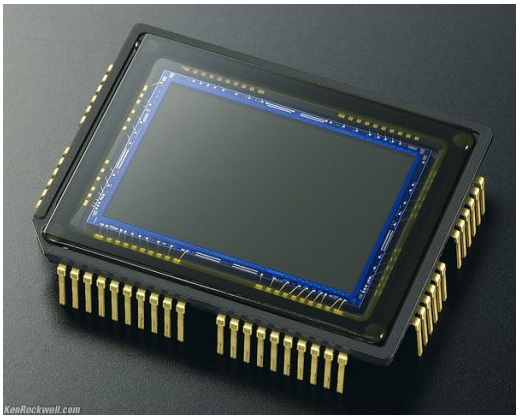


(a) (b)

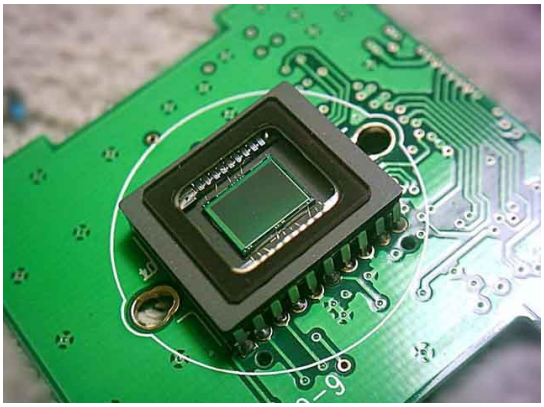
Figura 1. (a) diagrama esquemático do globo ocular humano, (b) perspertiva microscópica da retina humana composta por 1 cone, 2 células bipolares, 9 bastonetes e 3 axônios de células ganglionares do nervo óptico. [1]

Quando se trata de imagens digitais [4-5], a ideia é a mesma. A maioria das câmeras digitais funciona por meio de pequenos sensores retangulares dispostos em homogeneamente lado a lado. Um tipo de sensor mais usado é o dispositivo de carga acoplada (CCD – *charge coupled device*). Outra tecnologia também usada é a de semicondutor metal-óxido complementar (CMOS - *complementary metal oxide semicondutor*). Esses sensores fazem a conversão da energia luminosa em carga elétrica e quando milhares deles são agrupados em uma matriz são capazes de capturar uma imagem digital. E assim, cada microssensor captura um elemento mínimo da imagem (*pixel-Picture Element*).

A imagem digital consiste em toda a informação capturada pelos sensores, onde cada um deles responde, ao mesmo tempo, um valor que corresponde à carga acumulada de energia luminosa incidida. Este valor irá variar de acordo com o brilho (intensidade) e a cor (frequência) da luz capturada. Em um CCD, por exemplo, a carga é conduzida através do circuito eletrônico e armazenada em um tipo de *buffer* para poder haver, posteriormente, a sua leitura. Um conversor analógico-digital relaciona cada valor de quantidade de carga obtido a um valor digital ou binário. A Figura 2(a) mostra um sensor CCD enquanto a Figura 2(b) apresenta um CCD junto a um circuito de uma câmera digital.



(a)



(b)

Figura 2. (a) sensor CCD, (b) sensor CCD junto a um circuito de uma câmera digital. [1]

Para não necessitar de um conversor analógico digital, os dispositivos CMOS consistem de uma matriz de pequenos sensores, onde cada um destes possui micro transistores que fazem a leitura da carga diretamente para valores digitais. Por não precisar de um conversor, o tempo de resposta deste dispositivo é bem menor quando comparado ao CCD e o seu consumo de energia elétrica também é minimizado. Porém a informação é mais suscetível a ruídos, já que cada sensor vai ter uma leitura individual, ocorrendo assim uma aleatoriedade nos erros de captação de cada elemento ou *pixel* da imagem. A Figura 3 mostra um sensor CMOS.

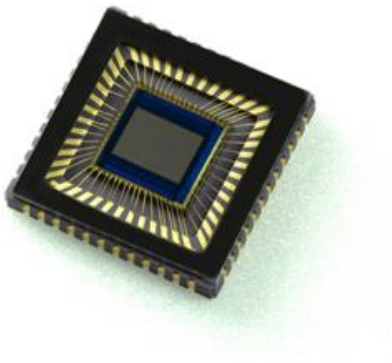


Figura 3. O sensor CMOS. [1]

Quanto mais sensores existirem em um dispositivo, maior será a resolução da câmera. A Figura 4 apresenta as diferentes resoluções possíveis para uma mesma imagem. O valor de 1 *megapixel* representa a quantidade de elementos ou *pixels* da

imagem digital, ou seja, 1 milhão de *pixels*. Pode se dizer também que este valor corresponde à área da imagem, já que uma imagem digital com 1600 *pixels* de comprimento e 900 *pixels* de altura tem aproximadamente 1,4 *megapixels* (1600 x 900 = 1440000).

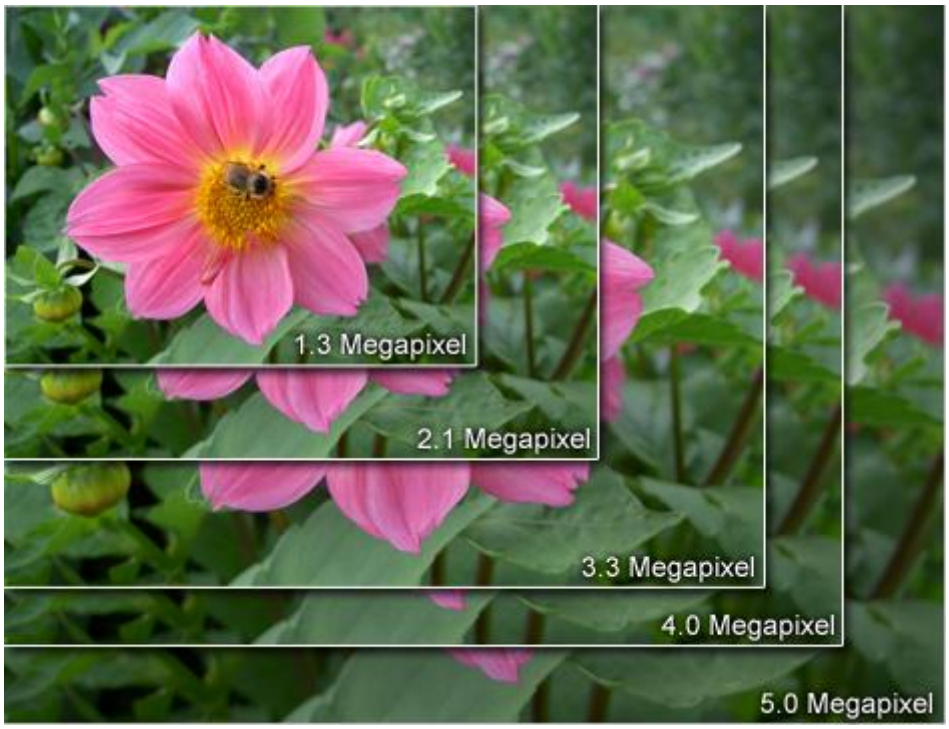


Figura 4. As diversas resoluções possíveis para uma mesma imagem em medição de *Megapixel.* [5]

2.2 Captação de cores

Os sensores das câmeras digitais [5], tanto o CCD e o CMOS, convertem a intensidade de luz para um valor que possa ser, posteriormente, processado. Esta funcionalidade é similar a dos bastonetes da retina do olho humano, porém só é possível enxergar em preto e branco. Para ser possível a aquisição ou captação de cores, o ser humano tem os cones na retina dos olhos. Então, para existir esta funcionalidade em uma câmera, é necessário usar um artifício que utiliza a ideia de dividir ou filtrar a captação em três cores primárias para cada elemento ou *pixel* da imagem. Estes três valores resultantes são combinados para obter-se uma cor de espectro.

Para cada pixel deverá existir três sensores, e cada um destes terá um filtro para determinada faixa de cor primária. Neste caso, terá que existir um divisor de feixe que terá a função de dividir por igual a intensidade de luz para cada um dos três sensores conforme é apresentado na Figura 5. Dessa forma, os dispositivos de sensoriamento, que usam essa ideia, tem melhor qualidade, contudo são mais densos e mais caros.

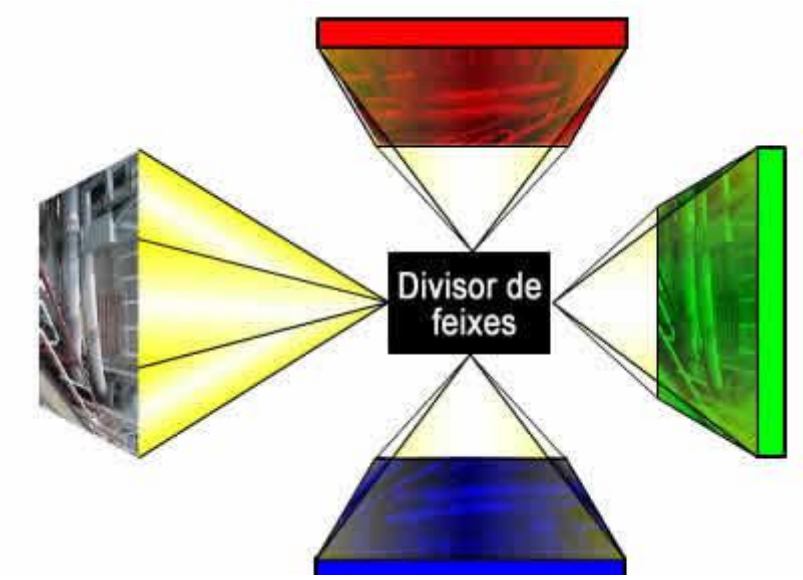


Figura 5. Imagem colorida resultante da junção das informações dos três sensores vermelho, verde e azul por meio de um divisor de feixes. [5]

Outra possibilidade para captação de cores, e também a mais comum, é utilizar a interpolação de *pixels*. Neste método, cada elemento terá um único sensor com apenas um filtro de determinada cor primária, interpolando ou alternando as cores dos filtros ao longo da imagem digital. A cor de espectro de cada *pixel* é encontrada por meio de uma estimativa entre o valor adquirido no sensor correspondente e nos seus vizinhos.

O filtro de Bayer, apresentado na Figura 6, consiste na ideia de enfileirar alternadamente filtros vermelhos e verdes, e filtros azuis e verdes, mudando de um padrão a outro para cada linha horizontal ou vertical da imagem. Uma maior quantidade do filtro de verde explica-se pelo fato de ser a cor que o olho humano é mais sensível, assim a imagem digital adquirida terá mais valor significativo em sua coloração.

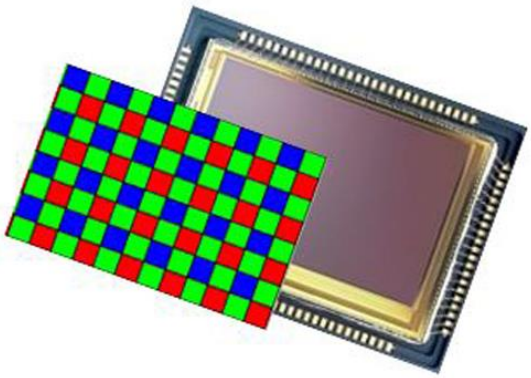


Figura 6. O filtro de Bayer em um sensor de câmera digital. [1]

2.3 Modelos de câmeras

A imagem digital é adquirida através da projeção de uma imagem real em um plano imaginário, chamado de plano de imagem da câmera, onde se encontram os sensores. Essa projeção tem como resultado uma matriz a qual se dá o nome de matriz de projeção da câmera. Cada câmera tem sua matriz de projeção característica, sendo definida pelo seu tipo de modelo e seu processo de fabricação.

A variação da matriz de projeção de uma câmera está diretamente relacionada à distância focal desta. Existem outros fatores, tais como variações térmicas ou mecânicas, que podem causar alterações nessa matriz.

A regulagem da distância focal é um recurso necessário nas câmeras, pois essa distância da lente entre o sensor da câmera pode detalhar a imagem digital dependendo da profundidade ou distância entre os pontos dispostos na imagem real e a lente. Isso porque a imagem tende a corresponder uma vista mais ampla quando a lente é aproximada do sensor, enquanto o aumento da distância focal torna menor [5] a área captada pelo sensor ocorrendo assim um *zoom* centralizado na imagem adquirida, assim como é mostrado na Figura 7.

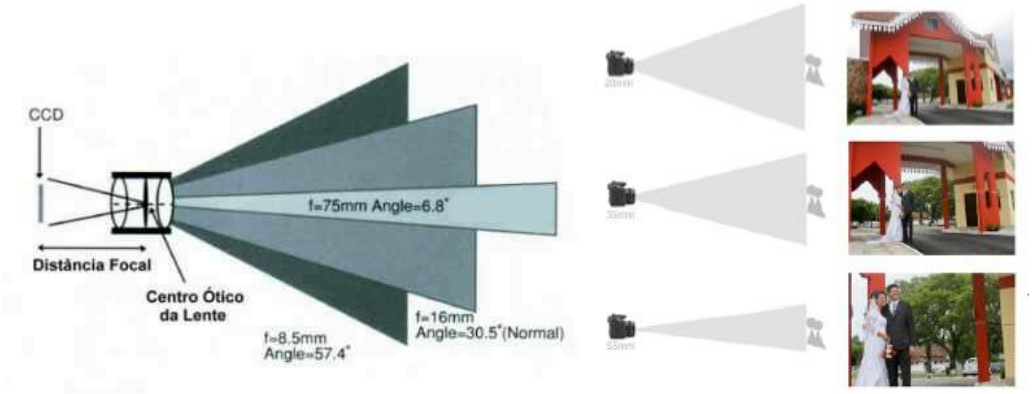
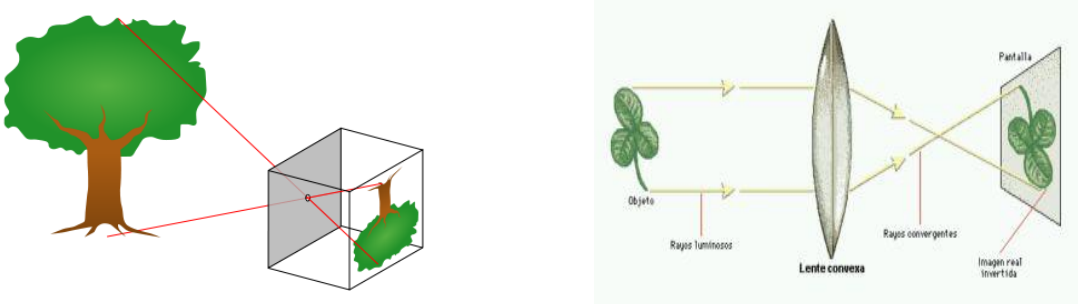


Figura 7. Modificação da distância focal para obter o *zoom* centralizado de uma mesma imagem.

Dentre os principais tipos de câmeras existem estes: as de foco fixo, as de foco automático, e as *pinhole*. A diferença de cada um desses tipos é a maneira em que a imagem se projeta no sensor da câmera. No modelo *pinhole*, a luz passa por um caminho estreito antes de ser captada, conforme mostra a Figura 8(a). Para existir uma regulagem na distância focal da imagem projetada e evitar certas distorções, a luz é desviada por lentes convergentes antes de atingir o sensor, conforme é apresentado na Figura 8(b).



(a) (b)

Figura 8. (a) captura de uma imagem em uma câmera *pinhole*, (b) captura de uma imagem após a luz passar por uma lente convexa.

Sabendo que plano de imagem da câmera tem tamanho fixo, o ângulo formado entre os pontos da extremidade da imagem adquirida e o ponto de passagem da luz é diretamente relacionado à distância focal conforme também mostra a Figura 7. Esse ângulo é chamado de ângulo de convergência da imagem e

é de fundamental importância na calibragem das câmeras usadas nas aplicações de visão computacional.

Conforme um objeto se movimenta paralelamente em relação ao plano de imagem da câmera, o deslocamento será correspondido por uma mudança de posição de todos os pontos deste objeto imagem adquirida. Quando este se movimenta perpendicular em relação ao plano de imagem da câmera haverá uma ampliação ou redução do seu tamanho no sensor, e é essa peculiaridade associada à noção de tamanho relativo que torna possível o cálculo de profundidade em uma visão monocular como mostrado na Figura 9. Essa ampliação ou redução ocorre por causa da convergência da luz emitida pela imagem real até o sensor. Os feixes de luz se aproximam ou se afastam do centro da imagem do objeto projetado no plano de imagem da câmera, e isso é mostrado na Figura 10.

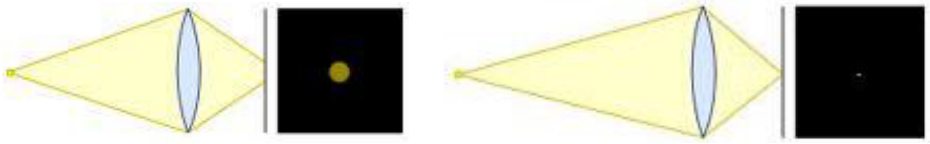
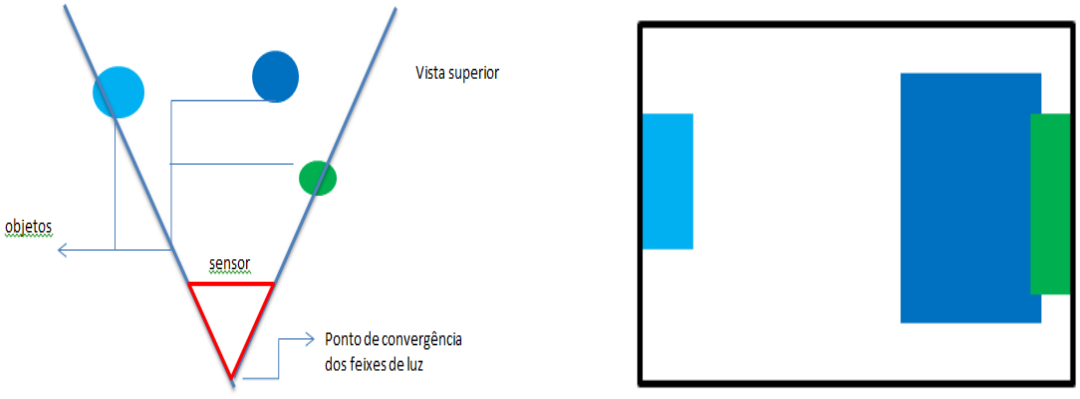


Figura 9. Afastamento de um objeto e seu efeito de redução de tamanho na imagem capturada. [1]



(a) (b)

Figura 10. (a) Esquema de uma captação vista de cima, de objetos em posições diferentes, (b) Imagem da captação. [1]

2.4 Conceito de imagem digital

Uma imagem digital é uma matriz bidimensional de *pixels*. De certa forma, cada *pixel* de uma imagem é independente e pode representar qualquer informação visual, ou seja, uma imagem pode ser uma matriz completamente aleatória. Segundo Gonzalez [5], “Uma imagem digital é uma função f(x,y) discretizada tanto em coordenadas espaciais quanto em brilho”. As coordenadas x e y são positivas, pelo conceito matemático de que uma matriz não tem índice negativo, e a origem parte do ponto superior esquerdo da imagem, assim como é mostrado na Figura 11(a).

Uma imagem é binarizada quando cada *pixel* que a compõe só possa representar duas informações ou cores diferentes, a cor preta e a cor branca. Na maioria das API’s (*Application Programming Interface*) ou ferramentas de programação que aborda processamento de imagens digitas, no que se refere às imagens binarizadas, a cor preta é representada pelo valor zero, e a cor branca é atribuído o valor 1.

Uma imagem binarizada pode ser resultado da extração de alguma característica de uma imagem colorida ou de tons de cinza. Geralmente, após a filtragem ou extração de características de uma imagem, é necessário utilizar uma técnica de limiarização para segmentar uma imagem em dois grupos distintos, tendo como resultado uma imagem binarizada.

Já em uma imagem de tons de cinza [5], como a imagem da Figura 11(a), uma das representações é a sua matriz possui valores decimais de 0 à 255, conforme a Figura 11(c) mostra, representados por números binários, ou seja, 1 *byte*. As diferentes intensidades, partindo do escuro em direção ao claro, são representadas com um valor crescente conforme mostra a Figura 11(b), em que o valor 255 representa a cor branca e o zero representa a cor preta.

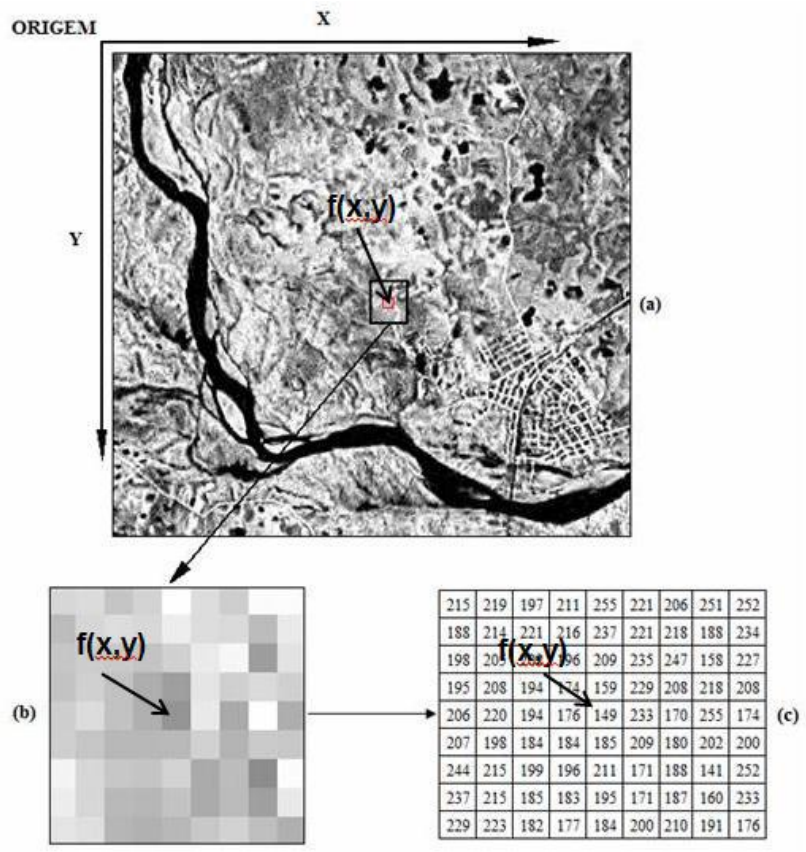


Figura 11. (a) Exemplo de um imagem monocromática, (b) Uma visão minuciosa de uma pequena área da imagem, (c) Os valores de cada *pixel* da imagem. [5]

Aumentando a quantidade de informação, as imagens coloridas necessitam de valores relacionados à cor de espectro de cada *pixel*. Dentre os modelos de representação existem o ciano, magenta e amarelo (CMY – *cyan*, *magenta*, *yellow*), e o vermelho, verde, azul (RGB – *red*, *green*, *blue*). No modelo RGB, a cor de um *pixel* é representada pelas três cores primárias que dão nome ao mesmo. A Figura 12 mostra algumas cores com seus valores que correspondem ao modelo RGB.

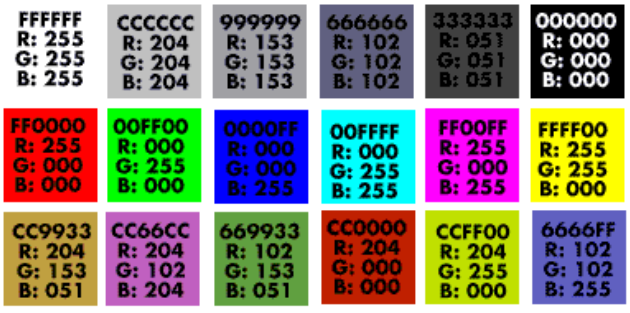


Figura 12. Valores em hexadecimal e decimal de algumas cores no modelo RGB. [1]

Estes modelos funcionam porque os 3 sinais emitidos pelos *pixels*, quando muito próximos, são visualmente sobrepostos causando assim um somatório dos comprimentos de ondas. Então um sinal vai interferir nos demais, e assim uma nova cor será mostrada por essa mistura aditiva. Em certos casos pode ser útil decompor uma imagem em cores primárias para facilitar a identificação de elementos ou pontos de interesse, ou então para ter a finalidade de simplificar ou diminuir a quantidade de informação a ser posteriormente analisada. A Figura 13 mostra o modelo RGB aplicado a uma imagem colorida usando a essa ideia.

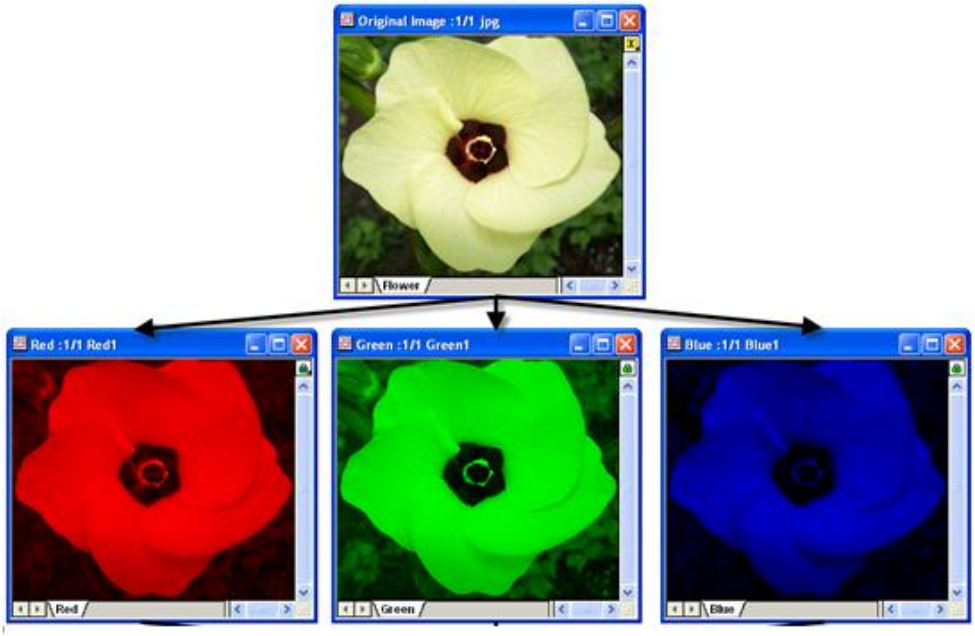


Figura 13. Imagem decomposta em 3 componentes. [5]

Capítulo 3

Estereoscopia

A estereoscopia [3] (do grego stereós = sólido, firme; do latim copia = traslado, reprodução) é uma técnica que visa analisar duas ou mais imagens digitais obtidas em perspectivas diferentes, e assim obter informações do espaço tridimensional. O sistema visual humano é binocular e utiliza essa técnica naturalmente por meio dos olhos, que capturam o par estereoscópico, e do cérebro, que processa essas imagens bidimensionais e as conceitua em informações tridimensionais.

No momento em que uma foto é capturada por uma câmera digital, a imagem resultante é uma representação bidimensional de um espaço tridimensional. Logicamente, alguns fatores dessa imagem podem causar a percepção de profundidade, tais como: iluminação, perspectiva, oclusão, sombra, gradiente da textura, dentre outros.

A iluminação pode causar a noção de profundidade através da reflexão. A variação de iluminação ao longo de uma superfície fornece informações de formato e de curvatura.

A perspectiva se baseia na ideia de que o tamanho dos objetos da imagem digital diminui à medida que o seu correspondente real se afasta do ponto de captura. As distâncias entre os objetos também diminuem à medida que estes se afastam.

A oclusão ou interposição ou interrupção de contorno acontece quando um objeto se encontra a frente de outro, ocultando informações relativas a este.

A sombra adiciona a informação de posição relativa de um objeto a outro na imagem digital, pois por meio disso é possível saber se os objetos estão encostados ou afastados, ou se um deles está à frente do outro e vice-versa.

O gradiente da textura se refere a características ou padrões que podem classificar e distinguir as diversas regiões ou conjunto de elementos da imagem

digital. Assim, quanto mais distante o objeto, mais densa fica sua textura e, por fim, menos nítida.

Para existir a visão estereoscópica, é necessário poder extrair informações de duas ou mais perspectivas diferentes. O sistema visual humano é binocular e assim o cérebro recebe duas perspectivas do que é visto. Então um olho vê uma imagem ligeiramente diferente do outro, onde a diferença é compatível à profundidade de cada ponto da imagem. Essa diferença ou deslocamento entre as imagens recebe o nome de disparidade.

Por fim, a disparidade entre duas imagens capturadas em perspectivas diferentes, por câmeras digitais, pode ser usada no cálculo para achar a distância de um determinado objeto ou marcador. E assim quanto mais perto o objeto estiver das câmeras, maior será a disparidade.

Existem duas abordagens na estereoscopia: a triangulação e o pareamento de pontos do par estereoscópico.

3.1 Princípios de triangulação

A estereoscopia se trata da aplicação de vários cálculos de triangulação para cada um dos diferentes pontos pareados por duas imagens. A distância do objeto P em relação ao segmento EC, onde os pontos E e C correspondem respectivamente às câmeras esquerda e direita, é representada por D. O ângulo α é formado através do encontro dos segmentos PE e EC, e o ângulo β corresponde ao encontro dos segmentos PC e EC. A Figura 14 é uma visualização desse cenário.

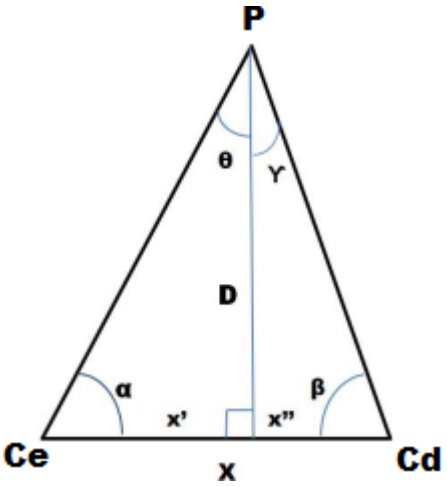


Figura 14. Triangulação para encontrar a distância D. [1]

De início, a Lei dos Senos é formulada pelas equações (1) e (2):

$$D/\text{sen}(\beta) = x''/\text{sen}(\gamma) \therefore x'' = D.\text{sen}(\gamma)/\text{sen}(\beta), \quad (1)$$

$$D/\text{sen}(\alpha) = x''/\text{sen}(\theta) \therefore x'' = D.\text{sen}(\theta)/\text{sen}(\alpha), \quad (2)$$

$$X = x' + x'' = (D.\text{sen}(\gamma)/\text{sen}(\beta))+(D.\text{sen}(\theta)/\text{sen}(\alpha)). \quad (3)$$

Já que a soma dos ângulos internos de um triângulo é 180°, seguem as equações (4) e (5):

$$\alpha + \theta + 90° = 180° \therefore \theta = 90° - \alpha, \quad (4)$$

$$\beta + \gamma + 90° = 180° \therefore \gamma = 90° - \beta. \quad (5)$$

A partir das equações (3), (4) e (5) é possível obter a seguinte expressão:

$$X = (D.\text{sen}(90°- \alpha)/\text{sen}(\alpha)) + (D.\text{sen}(90°- \beta)/\text{sen}(\beta)). \quad (6)$$

E utilizando a relação seno-cosseno,

$$\text{sen}(90°- \omega) = \cos(\omega), \quad (7)$$

na equação 6 se obtém:

$$X = (D.\cos(\alpha)/\text{sen}(\alpha)) + (D.\cos(\beta)/\text{sen}(\beta)). \quad (8)$$

Usando a relação trigonométrica,

$$\cos(\omega)/\text{sen}(\omega) = 1/\text{tg}(\omega), \quad (9)$$

na equação 8 é possível obter:

$$X = (D/\text{tg}(\alpha)) + (D/\text{tg}(\beta)). \quad (10)$$

Pondo em evidência o valor de D:

$$D=X / [(1/tg(\alpha)) + (1/tg(\beta))]. \quad (11)$$

Por meio dessa equação final, é possível encontrar a distância de um objeto em relação ao observador apenas conhecendo os ângulos formados entre os dois pontos de observação e o objeto.

Porém, numa imagem digital as informações relativas aos ângulos de projeção não são diretamente obtidos. Então se faz necessário algumas adaptações para poder utilizar a fórmula de triangulação em um par de imagens estereoscópicas.

O desenho mostrado na Figura 15 corresponde à situação visualizada na Figura 14 em conjunto com o modelo de câmera mostrado na Figura 10.

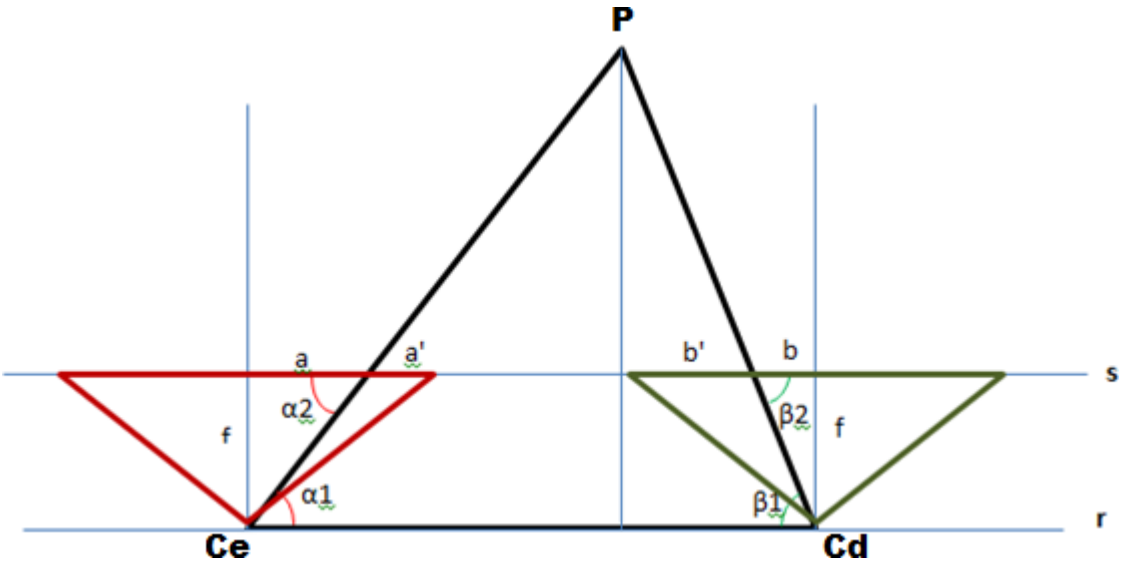


Figura 15. Visão de cima da situação de visão estereoscópica em conjunto com a ideia de ângulo de convergência e distância focal. [1]

Sabendo-se que a reta r é paralela a reta s, o ângulo α1 possui grau de abertura igual ao ângulo α2 por que são alternos internos entre si, assim como os ângulos β1 e β2.

Então o valor da tangente de α1 é obtida pela fórmula:

$$tg(\alpha) = f/a, \quad (12)$$

e da mesma forma:

$$tg(\beta) = f/b. \quad (13)$$

Incluindo essa ideia no cálculo da distância:

$$D = X/ [(1/f/a) + (1/f/b)] = X/[(a/f)+(b/f)] = X/[(1/f).(a+b)] = f.X/(a+b). \quad (14)$$

Onde os valores de a e b podem ser obtidas pelas imagens digitais capturadas em unidades de *pixels*. Os valores referem-se à distância em *pixels* do ponto da imagem até o eixo vertical localizado no centro da imagem assim como mostra a Figura 15.

É possível uma simplificação ao usar o valor absoluto dos pontos a e b, ou seja a coordenada horizontal, na equação 14, que serão representados por a' e b', respectivamente. Sabendo-se do valor total da largura da imagem em *pixels* I e que este valor é igual nas duas imagens (câmera esquerda e direita), no caso de serem utilizadas câmeras de modelo igual, o valor de a será substituído por I/2 – a' e o valor de b será substituído por I/2 – b' resultando em:

$$D = f.X/[(I/2)-a'+(I/2)-b'] = f.X/(I - a' - b') \quad (15)$$

O termo I – a' representa a distância do ponto até a lateral esquerda da imagem da câmera direita em *pixels*, assim como o termo b' representa a distância do ponto até a margem esquerda da imagem da câmera esquerda. Para simplificar mais ainda a fórmula, I – a' será substituído por Xd e b' será substituído por Xe e isto é apresentado na equação a seguir:

$$D = f.X/(Xd - Xe) \quad (16)$$

Relembrando que f é o valor da distância focal das duas câmeras (valor igual para as duas câmeras, quando do mesmo modelo) que permanecerá fixo durante todo o processo da aplicação e o valor de X é ajustável (afastamento das câmeras) e influencia na precisão do cálculo da distância: quando o valor de X for alto, a precisão é melhor para calcular longas distâncias.

Assim o valor de fX é a máxima distância possível a ser calculada. Por exemplo, para um valor fX igual a 1 metro.*pixel* o cálculo não corresponderá a distância maiores que 1 *metro*, porque não existe variação ou deslocamento menor que 1 *pixel*. Então não é conveniente calcular a distância perto do limite, pois a variação de 1 para 2 *pixels* causa uma diferença de metade da distância (no caso do exemplo anterior, a variação de 1 *pixel* corresponde a 1 metro e a variação de 2

pixels resulta em 50 centímetros, ou seja, uma variação de 50% no cálculo da distância).

A seguir é exibido um gráfico na Figura 16, que mostra variação ou deslocamento de *pixels* em relação à distância. É possível notar que enquanto a variação dos *pixels* for alta mais precisa será a medição.

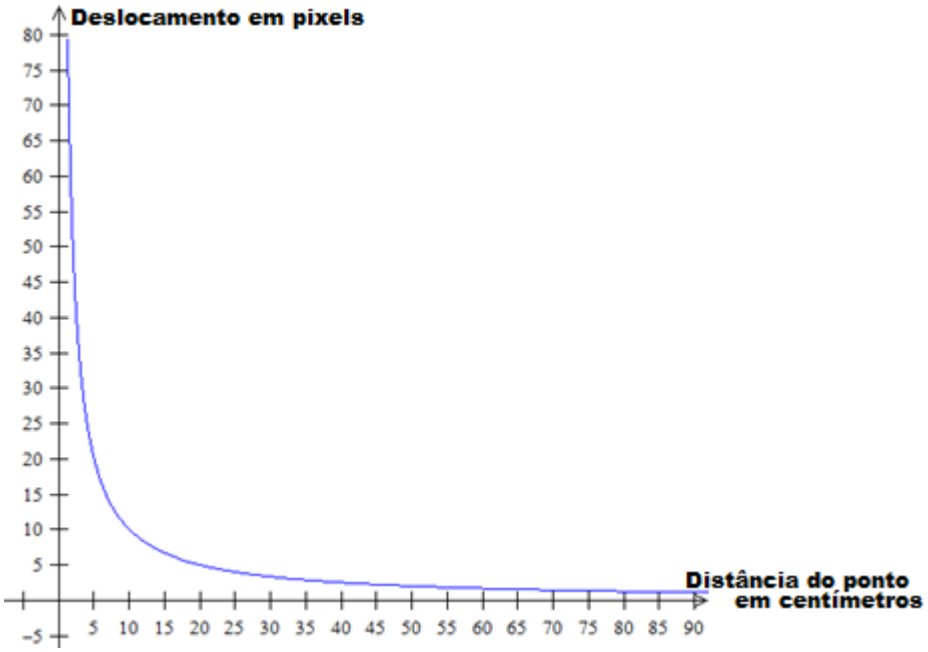


Figura 16. Perda de precisão do deslocamento y das imagens conforme se aumenta a distância x dos pontos a serem emparelhados com a estereoscopia.

Um fator importante na precisão do cálculo da distância é a resolução das câmeras digitais. Uma câmera com alta resolução será mais precisa, pois terá uma variação ou deslocamento de uma maior quantidade de *pixels*. Por exemplo, um par de câmeras digitais capturando imagens com 1000 *pixels* de comprimento terá o fator Xd-Xe variando de 0 até 999. A variação de 0 *pixels* ocorre quando a distância está tão alta que não causa deslocamento nas duas imagens e a variação 999 pixels acontece quando a distância está tão próxima que o ponto se desloca da borda esquerda da câmera direita para a borda direita da câmera esquerda.

Assim um par de câmeras digitais com resolução de 1600 *pixels* horizontais tem melhor precisão quando comparado a um com resolução de 800 *pixels*, pois a primeira tem um maior deslocamento em *pixels* por proximidade.

3.2 Pareamento dos pontos

O cálculo da distância simplificado [8-12] consiste de operações matemáticas de subtração e divisão, sem comparações ou iterações, e assim é uma tarefa de custo computacional desprezível. Outra tarefa que dificulta o cálculo da distância é o pareamento dos pontos do par estereoscópico de imagens. O uso de medidas de similaridade, tais como Minkowski, Hamming, Tanimoto e Entropia Cruzada [5], para parear os pontos pode se tornar uma tarefa complexa (alto custo computacional) e passível de erros (falsos pares). Porém, para um objeto ou uma marca de uma cor ou uma característica única, essa tarefa se torna bastante simplificada.

Abordando um cenário onde o par de câmeras digitais captura duas imagens com diversos objetos sem nenhum padrão e espalhados de maneira aleatória (sem uniformidade) o problema de pareamento se torna mais complexo. Além de que, existirão pontos que uma câmera captará e a outra não, por causa do ângulo de convergência, e assim haverá um aumento do número de informação irrelevante para o pareamento dos pontos.

Para simplificar as medidas de similaridade existe a possibilidade de filtrar a informação da imagem por determinadas faixas de cor (delimitar as faixas manualmente ou por *clustering*) ou por variação de *pixels* vizinhos da imagem. Neste último caso, os pontos seriam as bordas de algum objeto, ou então as linhas da imagem.

Depois de encontrar os pontos a serem pareados, também existem duas possíveis abordagens que podem ser usadas para correlacioná-los. Uma abordagem consiste em encontrar um ponto em uma das imagens e depois procurá-lo na outra imagem. A outra se fundamenta em achar os pontos em ambas as imagens e depois tentar correlaciona-los. No caso da primeira abordagem, o custo computacional é elevado pois o espaço de busca é superdimensionado (no caso de uma imagem de 10 megapixels, seriam 10 milhões de pontos em cada busca!). Já na segunda abordagem, pode existir mais de um par para um único ponto (ambiguidade) e com isso podem vir a ocorrer erros, porém a complexidade ou o custo computacional é reduzido.

Capítulo 4

Laser semicondutor

Laser ou *Light Amplification by Simulated Emission of Radiation* [13-20] que em português significa Amplificação da Luz por Emissão Estimulada de Radiação é um mecanismo que gera e amplifica um feixe de luz monocromático de alta intensidade. A luz emitida por um laser é bem definida e corresponde a um único valor de frequência e comprimento de onda.

Estes mecanismos que se tratam de amplificadores de luz em funcionamento através de emissão estimulada podem ser de vários tipos, porém o tipo que vai ser abordado nesse trabalho será o laser semicondutor.

4.1 Introdução ao laser e aos semicondutores

Semicondutor é um material de estrutura cristalina e de condutividade elétrica intermediária entre condutores e isolantes. Os elementos semicondutores podem ser dopados quimicamente para serem capazes de controlar e transmitir uma corrente elétrica. A dopagem é um procedimento de adição minuciosa de impurezas a um elemento semicondutor para torná-lo mais condutor, com um teor de uma parte de impureza por milhão.

Os semicondutores são usados na fabricação dos diodos que são os emissores de luz dos LED's (*Light Emitting Diode*). O funcionamento do laser semicondutor é parecido com funcionamento do diodo, já que a diferença está na geração de fótons que, para o caso do diodo, a emissão é espontânea enquanto que no laser é auto-estimulada. Então é comum utilizar o termo laser diodo para se tratar do laser semicondutor.

Diodo é um componente eletrônico composto geralmente por semicondutores dopados de maneira que ocorra uma junção com dois tipos de semicondutores: de tipo-P e de tipo-N. O semicondutor de tipo-P é resultado do processo de dopagem de um semicondutor com um composto trivalente, ou seja, com 3 elétrons na

camada de valência. Os compostos trivalentes mais usados na fabricação de semicondutores tipo-P são o boro e o gálio.

Já os semicondutores de tipo-N são resultados da dopagem em que se adiciona um composto pentavalente, isto é, com 5 elétrons na camada de valência, ao semicondutor. A junção dos semicondutores dos tipos P e N tem a característica de haver oferta (Tipo-P) e demanda (Tipo-N) de portadores de carga conforme o material é estimulado por uma diferencia de potencial elétrico, conforme mostra a Figura 17.

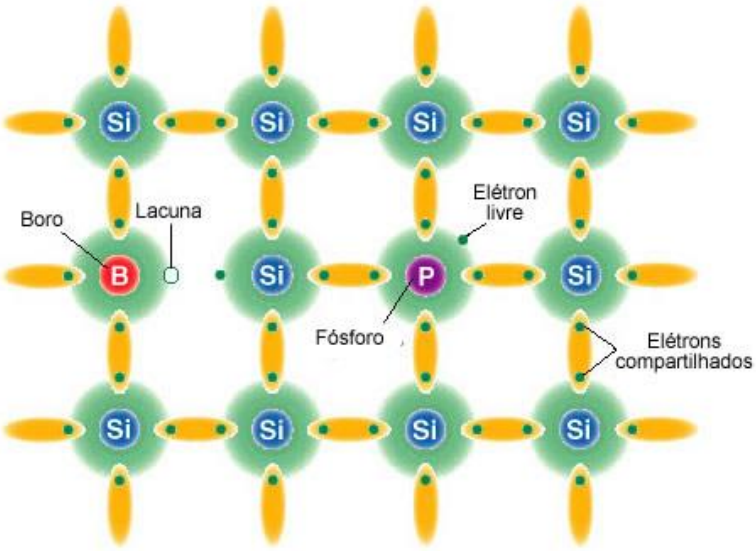


Figura 17. Dopagem de um semicondutor.

No caso do diodo emissor de luz, os semicondutores mais utilizados são arsenieto de gálio (GaAs), arsenieto de alumínio (AlAs) e arsenieto de gálio-alumínio (AlGaAs) que são compostos químicos sintéticos.

4.2 Princípio de funcionamento de um laser

No diodos LED a junção dos semicondutores P-N é homogênea, assim a emissão de luz é espontânea e se existir um fornecimento de energia haverá uma resposta referente ao estímulo desta energia e nada mais. Já nos lasers semicondutores a junção utilizada é heterogênea, havendo assim uma emissão auto-estimulada por causa do confinamento da luz num trecho dessa junção.

Os lasers semicondutores mais comuns têm uma heterojunção dupla (DH), cujo diagrama da banda de energia é mostrado na Figura 18. Como já foi dito, os semicondutores mais comumente utilizados são AlGaAs com Eg = 2 eV e GaAs com Eg = 1,4 eV. Neste dispositivo, a quantidade de semicondutor p-GaAs é bastante pequena: uma fina camada de normalmente 0,1-0,2 μm de espessura.

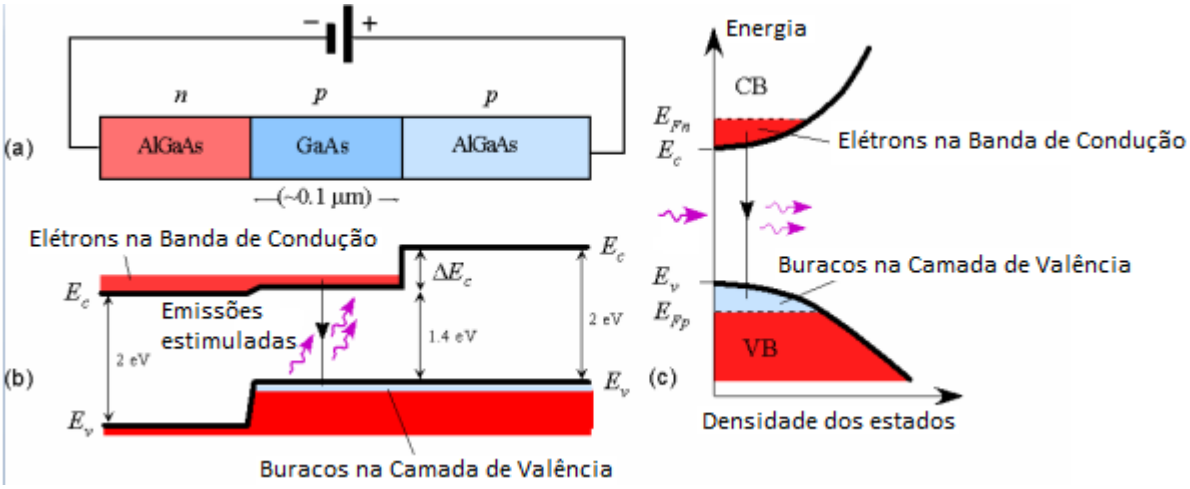


Figura 18. (a) Junção heterogênea dos semicondutores AlGaAs (mais pesado) e GaAs (mais leve). (b) Diagrama de da banda de energia do esquema da junção heterogênea (observa-se o “gargalo” de elétrons). (c) Gráfico que mostra a situação de emissão estimulada através da relação oferta-demanda de energia. [20]

A emissão estimulada acontece no semicondutor p-GaAs e ambos os semicondutores do tipo P (p-GaAs e p-AlGaAs) são profundamente dopados, ou seja, eles possuem alta demanda de portadores de carga. Aplicando uma diferença de potencial elétrico cada vez maior neste dispositivo de maneira em que ocorra uma polarização direta, a energia de condução (Ec) do semicondutor n-AlGaAs vai se aproximando da Ec do p-GaAs. Neste caso, é provocada uma passagem de elétrons da banda de condução do n-AlGaAs para a do p-GaAs. Esta movimentação de elétrons fica limitada a região de condução do semicondutor p-GaAs, pois existe uma barreira causada pela diferença de energia de condução entre o p-GaAs e p-AlGaAs. Essa diferença ocorre por que há uma mudança na banda de GAP (energia necessária para o elétron transite da banda de valência para a banda de condução) entre o p-AlGaAs e o p-GaAs.

A fina camada de p-GaAs é profundamente dopada. Dessa forma, a sua banda de valência está repleta de “buracos”, ou seja, o material tem uma grande demanda de portadores de carga. Isso quer dizer que todos os estados eletrônicos acima do Efp estão vazios na camada. Por causa da aplicação de uma diferença de potencial elétrico, uma grande concentração de elétrons do n-AlGaAs entra na banda de condução da camada p-GaAs como é mostrado na Figura 18. Em consequência disso, existe uma grande concentração de elétrons na banda de condução e estados eletrônicos vazios (“buracos”) no topo da sua banda de valência. Isso possibilita que, um fóton que entra com energia maior que a Eg possa estimular um elétron de condução na camada p-GaAs a decair da banda de condução para a banda de valência. Esse fenômeno é chamado de lasing recombination ou photon-stimulated eléctron-hole recombination. Com isso haverá uma série de emissões estimuladas causando uma intensa amplificação óptica dos fótons. Quanto maior for a corrente, maior será a quantidade de elétrons a serem estimulados no gargalo provido pela junção p(GaAs)-p(AlGaAs) e assim maior será a intensidade de luz resultante da emissão. Por fim o dispositivo funciona como um amplificador óptico semicondutor que amplifica o sinal óptico que atravessa a fina camada p-GaAs.

Na construção de um laser semicondutor a fina camada de emissão é condicionada em uma cavidade óptica, que se trata de espelhos seletivos que apenas refletem o comprimento de onda relativo à cor de espectro desejada. A cavidade óptica reflete os fótons coerentes para frente e para trás o que os leva a interferências construtivas dentro da cavidade, ou seja, haverá uma condução da luz para uma única direção sem grandes perdas. A Figura 19 mostra essa ideia de incorporar uma cavidade óptica na fina camada ativa do laser. Assim existe uma oscilação eletromagnética de alta energia na cavidade óptica, em que uma parte dessa energia é lançada para fora como radiação de saída e outra parte é refletida. No caso da Figura 19, ao lado da cavidade óptica tem um refletor especial, chamado de refletor de Bragg distribuído (BRD), onde um lado reflete somente comprimentos de onda específicos de volta à cavidade. O BDR é uma estrutura ondulada uniformemente que é colocada em um semicondutor que reflete somente um comprimento de onda específico relacionado com a ondulação do refletor. Essa relação se refere à cor emitida pelo laser onde a frequência dessa cor corresponderá

à periodicidade da ondulação do refletor. Assim, somente irá ser refletido o comprimento de onda desejável, amplificando dessa forma a intensidade de luz.

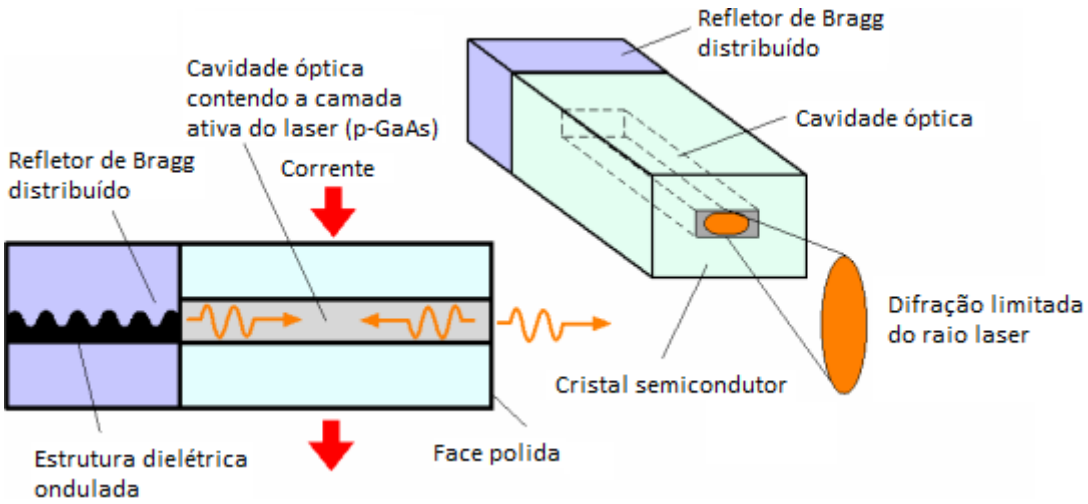


Figura 19. O laser semicondutor possui uma cavidade óptica para refletir a luz para fora do semicondutor como um feixe. [20]

Existem muitas vantagens de se usar heterojunção dupla. Os semicondutores com banda de GAP maior tem tendência a ter um menor índice de refração, assim o AlGaAs tem um índice de refração menor do que o GaAs. A mudança entre os índices funciona como um guia para luz na cavidade óptica p-GaAs e assim reduz as perdas e aumenta a concentração de luz ou fótons. Este aumento considerável na concentração dos fótons implica no aumento da taxa de emissões estimuladas e na eficiência do laser.

A corrente aplicada no laser semicondutor tem que exceder um limiar (Tth), para haver a quantidade mínima de emissões estimuladas e acumular as oscilações ópticas necessárias dentro da cavidade óptica. A Figura 20 apresenta esse comportamento. Abaixo dessa corrente existe uma energia óptica fraca, que é causada somente pela combinação dos elétrons e “buracos” na região da junção n-AlGaAs e p-GaAs, e nessa situação o comportamento do laser se parece com o de um LED.

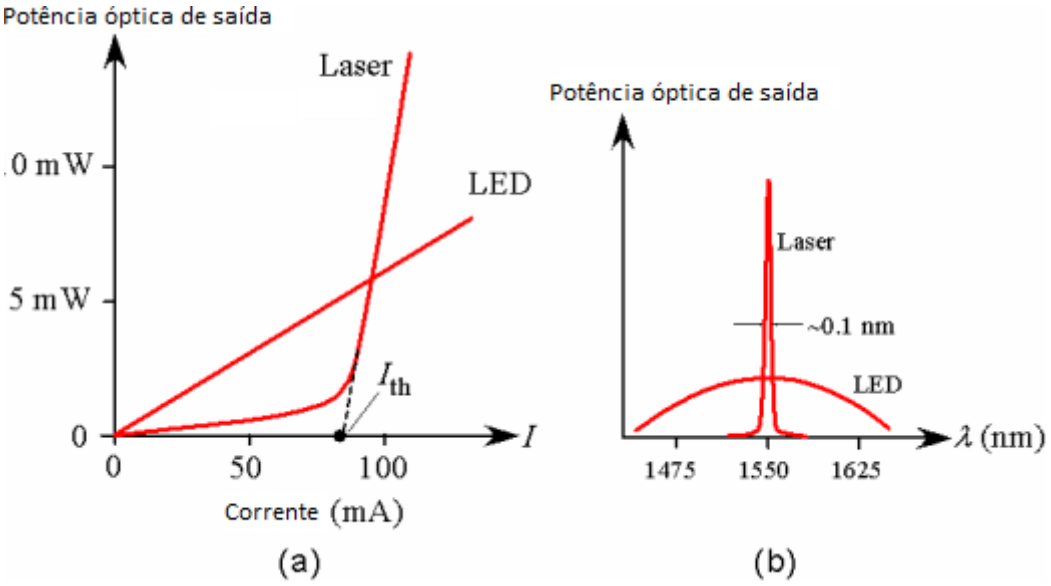


Figura 20. (a) Comparação da potencia de luz emitida por um laser e por um LED em relação a corrente que passa pelos semicondutores. (b) Comparação dos comprimentos de onda presentes na luz do LED e do laser com a distribuição de intensidade para cada comprimento de onda. [20]

4.3 Materiais utilizados

Os materiais utilizados na fabricação do dispositivo laser dependem da zona do espectro ou cor da luz que se pretende emitir. São 4 os principais tipos de materiais utilizados: Arseneto de Gálio; Fosfeto de Índio; Seleneto de Zinco; Nitreto de Gálio. Os lasers que tem AsGa como semicondutor operam no espectro eletromagnético 635 nm – 870nm (vermelho podendo chegar a obter o infravermelho).

Os leitores de disco compacto (CD) utilizam o AlGaAs como semicondutor do laser. Diferentemente dos CDs, os DVDs utilizam o AlGaInP como material da camada ativa do laser. Neste caso a capacidade de escrita do dispositivo depende do comprimento de onda do laser. Os lasers baseados no fosfeto de índio operam no infravermelho numa frequência de aproximadamente 1.55 μm. O nitreto de gálio pode operar na zona do azul e ultravioleta e a tecnologia *Blu-ray* se baseia em lasers desse material. Já o seleneto de zinco emite luz na zona do azul e do verde (460 nm – 520 nm). A Tabela 1 relaciona cada material aos respectivos comprimentos de onda e aplicações.

Tabela 1. Emissão de comprimentos de onda de vários tipos de lasers semicondutores. [20]

Material do laser (Camada ativa / Restante)	Comprimentos de onda	Aplicação
InGaN / GaN, SiC	380, 405, 450, 470 nm	Armazenamento de dados
AlGaInP / GaAs	635, 650, 670 nm	Apontadores laser, leitores de DVD
AlGaAs / GaAs	720-850 nm	Leitores de CD, impressoras laser
InGaAs / GaAs	900-1100 nm	Amplificadores ópticos, espectroscópio
InGaAs / InP	1000-1650 nm	Dispositivos de fibra-óptica

Capítulo 5

Modelo proposto

Neste trabalho é proposta a implementação de uma aplicação capaz de mapear um par estereoscópico de imagens digitais e gerar uma imagem monocromática que represente a profundidade ou o deslocamento de cada *pixel*, chamada de imagem de profundidade ou *deph image* [5].

O equipamento necessário para que a aplicação funcione consiste de duas câmeras digitais do tipo webcam com resolução de 600x480 *pixels*, um laser semicondutor ou apontador laser vermelho e um computador para executar a aplicação no ambiente MATLAB [20-26].

O apontador laser vermelho será de fundamental importância na aplicação deste trabalho. O feixe de luz vermelha irá incidir sobre os objetos e terá a função de marcar os pontos a serem pareados na medida em as imagens são capturadas pelas duas câmeras digitais [20].

Um apontador emite um feixe vermelho que faz uma marcação circular bem pequena ao incidir nos objetos não-reflexivos e opacos [13-20]. Para parear mais pontos por captura, o apontador ira emitir o laser passando por uma peça oca de acrílico cilíndrica e assim a marcação do laser passará a ser uma espécie de linha. Isso se deve pelo fato de que a peça irá servir de lente cilíndrica e já que o diâmetro do feixe é bem pequeno a ampliação do feixe será bem maior nos eixos paralelos do que nos meridianos da peça conforme mostra a Figura 21.

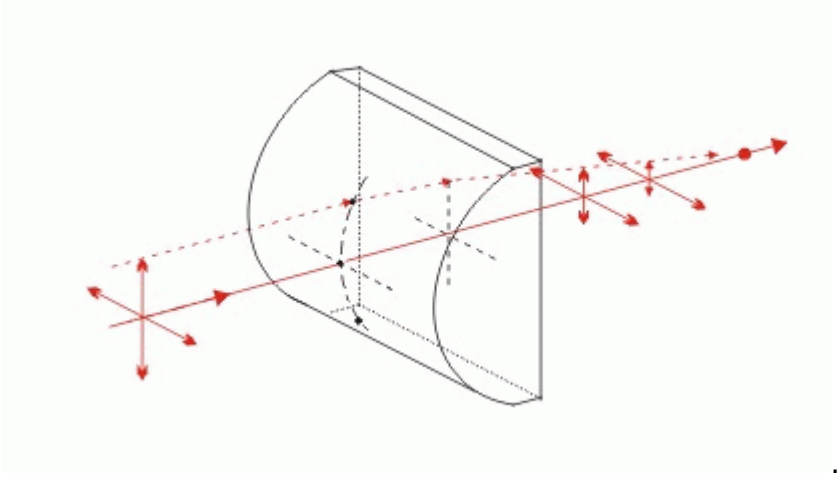


Figura 21. Esquema de um feixe de luz passando por uma lente cilíndrica.

Existe um ajuste automático de contraste das câmeras digitais, via *hardware*, onde a região do centro da marcação circular do laser se torna esbranquiçado nas imagens digitais. Isso é devido à alta intensidade de luz do laser no centro do feixe, como já foi visto anteriormente. Porém isso deixa de ser um problema já que a peça de acrílico absorve esse excesso de intensidade e assim a marca fica totalmente vermelha em uma iluminação normal. A Figura 22 mostra esse efeito causado por este ajuste das câmeras digitais comuns.

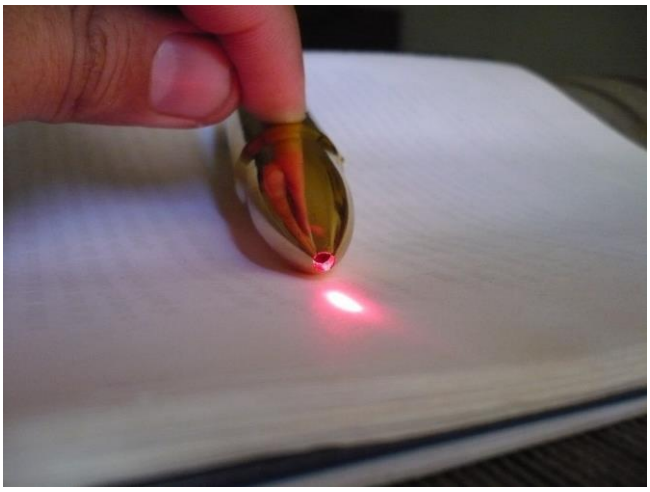


Figura 22. Efeito esbranquiçado do laser.

Por fim, o pareamento dos pontos será bastante simplificado, pois se baseará na marcação vermelha do laser nos objetos do cenário, evitando assim possíveis ambiguidades ou falsos pares.

As câmeras deverão ser posicionadas lado a lado, conforme mostra a Figura 23, de forma que seus planos de captura se mantenham o mais alinhado possível. O par de câmeras é conectado ao computador por duas entradas USB (*universal serial bus*), um para cada câmera e dessa forma o computador irá receber as imagens das câmeras para poder processá-los. As câmeras que são usadas na aplicação tem uma matriz de projeção fixa, ou seja, não terá distância focal ajustável por dispositivo ou *hardware*.

Figura 23. Câmeras digitais (webcams) utilizadas na aplicação.

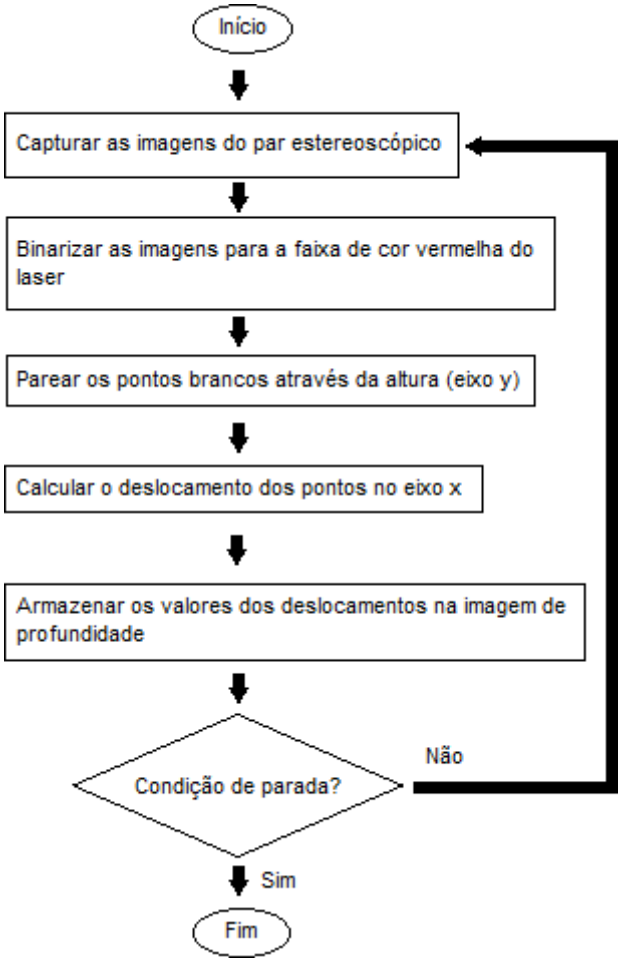


Figura 24. Modelo proposto.

Após capturar o par estereoscópico de imagens das câmeras, e pré-alocar a imagem de profundidade a aplicação irá segmentar as duas imagens pela cor vermelha do laser através de um limiar fixo, através da seguinte lógica:

```
R - G > limiarVermelho &&  R - B > limiarVermelho
```

onde R, G e B são matrizes que correspondem respectivamente aos componentes vermelho, verde e azul da imagem e limiarVermelho é um valor de 0 à 255. Dessa forma, somente serão verdadeiros (brancos) os *pixels* que tiverem um valor de vermelho bem maior do que o valor de verde e o azul. Neste caso irá ser feita a vetorização para aumento de desempenho da aplicação.

Nas duas imagens, os *pixels* cujos valores correspondem à inequação acima terão valor booliano 1 ou verdadeiro (*true*) e o restante dos *pixels* terão valor 0 ou

falso (*false*). Assim as imagens resultantes dessa operação serão imagens binarizadas.

Com as imagens binarizadas, irão ser pareados os *pixels* da imagem da câmera esquerda e direita se baseando na posição y da imagem, ou seja, se os *pixels* estiverem na mesma altura um será o par do outro. Isso funciona porque o deslocamento será somente na horizontal, pois as câmeras estarão alinhadas lado a lado.

Após se obter os pares, uma imagem será escolhida como referência das coordenadas (x,y) para o *pixel* da imagem de profundidade, onde terá um valor de intensidade correspondente ao valor de deslocamento horizontal de cada *pixel* do par estereoscópico. Esse valor de deslocamento é o termo (Xd-Xe) da fórmula 16 do capítulo 3. A imagem será normalizada, ou seja, os valores serão ajustados de forma proporcional ao valor máximo que um *pixel* pode ter, que no caso será 255. Essa normalização é feita no MATLAB multiplicando a matriz por 255 e dividindo cada elemento pelo valor máximo da matriz antes da multiplicação.

A cada captura das câmeras irá ser feita uma mudança no posicionamento da marcação vermelha do laser, assim novos *pixels* serão pareados e adicionados na imagem de profundidade. Após um determinado número de capturas a imagem de profundidade gerada será analisada para se observar a eficiência da ideia de aplicar estereoscopia em imagens digitais.

Capítulo 6

Resultados experimentais

O cálculo da distância é feito através do deslocamento Xe - Xd e do fator f.x de acordo com a equação 16. Para encontrar o valor de f.x é necessário fazer uma calibragem das câmeras. Essa calibragem consiste em projetar um feixe de laser em um plano, cuja distância é conhecida. Com o valor da distância e do deslocamento é possível achar o valor de f.x, simplesmente com uma multiplicação. É importante salientar que as câmeras devem estar muito bem alinhadas e que a distância para teste esteja em uma área próxima às câmeras por questões vistas no Capítulo 3.

Com o valor de f.x, é possível fazer experimentos para encontrar distâncias desconhecidas dos diversos objetos de um cenário. A estratégia usada para isolar os pontos a serem mapeados se baseia em detectar a cor vermelha do laser. Dada a imagem de entrada, Figura 25(a), após a passagem pela função de binarização por um limiar de faixa de cor vermelha resulta na imagem binarizada mostrada na Figura 25(b). A partir da binarização das duas imagens do par estereoscópico, será feito o pareamento dos pontos através da altura.

(a)

(b)

Figura 25. (a) Imagem de entrada. (b) Imagem binarizada pela faixa de cor vermelha do laser.

Após serem realizados os experimentos, os resultados gerados foram considerados bastante aceitáveis. Os objetos do cenário localizados mais perto das câmeras são marcados com uma intensidade alta (esbranquiçada) e os objetos mais distantes são marcados com uma intensidade baixa (escurecida). O cenário do experimento é apresentado na Figura 26.

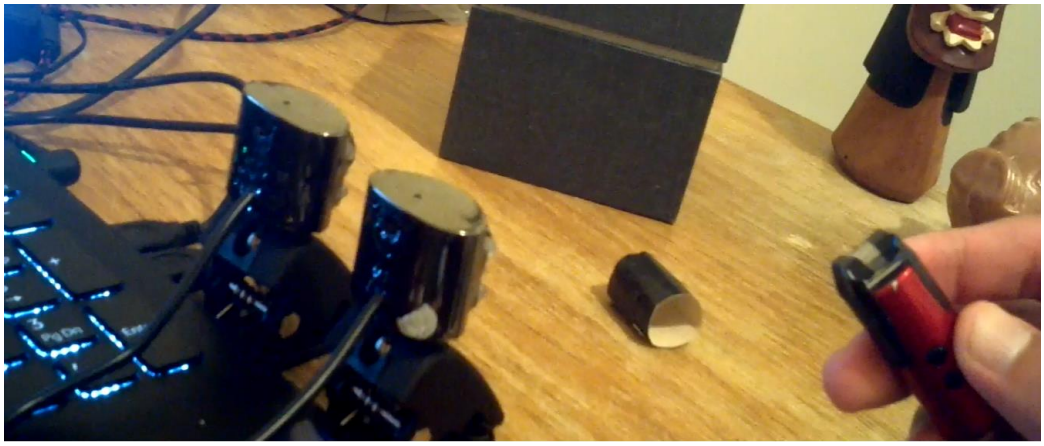


Figura 26. Cenário dos experimentos.

A Figura 27 mostra um resultado da aplicação. A Figura 27(a) mostra a imagem capturada pela câmera direta e o laser sendo projetado no objeto para que ocorra o cálculo das distâncias dos pontos. Já Figura 27(b) mostra a imagem de profundidade resultante da aplicação, onde os seus valores são os deslocamentos providos pela mudança de perspectiva das câmeras esquerda e direita.

A Figura 27 (c) apresenta a imagem de profundidade em pseudo-cores para uma melhor visualização. Isso se deve pelo fato de que o olho humano é mais sensível a cores que a intensidade de luz, por conter mais cones do que bastonetes, assunto abordado no Capítulo 1. Já a Figura 27 (d) mostra algumas matrizes de seções relacionadas à imagem de profundidade, porém os valores representam a distância, em centímetros, de cada ponto. Esses valores são encontrados através da divisão do valor f.x, encontrado na calibragem, pelo valor de deslocamento dos pontos. A imagem de profundidade é pré-alocada em uma matriz de zeros, dessa forma os pontos que tem valor zero são aqueles que não foram detectados pela aplicação.

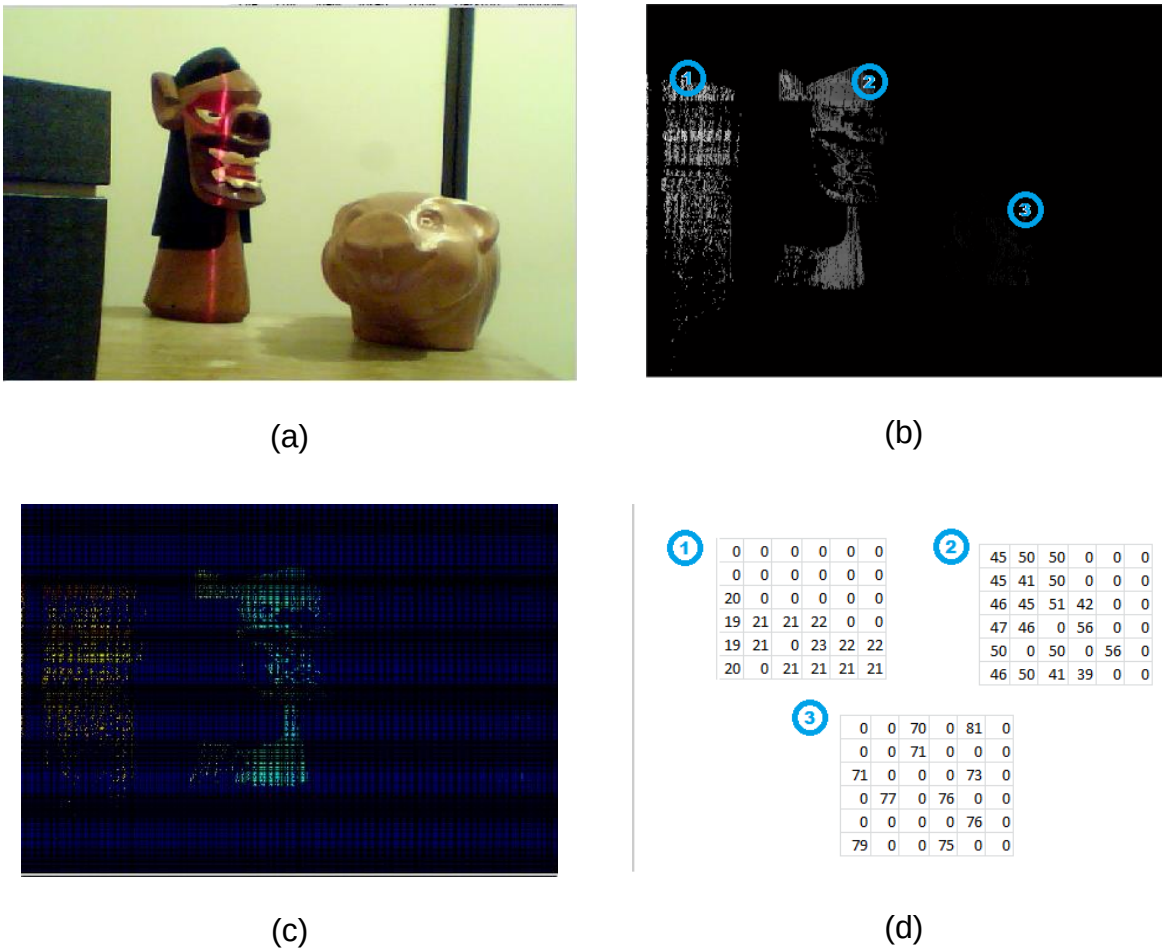


Figura 27. Resultado da aplicação. (a) A imagem capturada pela câmera direita. (b) A imagem de profundidade. (c) A representação da imagem de profundidade em pseudo-cores para uma melhor visualização. (d) Matrizes de seções relacionadas à imagem de profundidade com as distâncias, em centímetros, dos pontos.

A aplicação possui um resultado aceitável para esta situação, porém quando o cenário possui objetos de coloração vermelha bem intensa isso não acontece. Isso porque a binarização, a depender do limiar declarado, pode considerar os *pixels* correspondentes aos objetos vermelhos como *pixels* correspondentes à marcação do laser. Além disso, imagens com objetos reflexivos (igual ao objeto localizado na direita no experimento), escuros ou transparentes podem desviar, absorver ou distorcer a marcação do laser, provocando erros no pareamento ou nos cálculos.

A aplicação foi executada a uma velocidade de aproximadamente 40 quadros por segundo. Isso se deve pelo uso de boas práticas de programação do MATLAB,

tais como, a vetorização e a pré-alocação. A programação em GPU pode fornecer um aumento ainda maior de desempenho, já que os cálculos são feitos individualmente em cada *pixel.*

A aplicação que consiste no uso de estereoscopia em imagens digitais se baseou nos conceitos mostrados neste trabalho. Com um equipamento de baixo custo, com a captura de imagens limitadas a 600x480 *pixels* e um apontador laser simples de cor vermelha, foi possível se obter bons resultados que correspondem com a realidade.

Capítulo 7

Conclusão

A estereoscopia aplicada em imagens digitais pode ser utilizada para mapear a profundidade dos objetos que estejam no campo de visão dos dispositivos de captura (câmeras digitais). Neste caso, as câmeras podem servir como sensores de distância para vários pontos do cenário. A precisão da aplicação melhora conforme os objetos estejam mais próximos do par estereoscópico, sendo assim o cenário ideal para aplicação. No caso de cenários muitos distantes a aplicação da estereoscopia se torna ineficiente.

A exposição à alta claridade, a aparição de elementos de mesma faixa de cor do laser, a diminuição da precisão em função da alta profundidade dos elementos e a baixa resolução das câmeras são alguns dos fatores negativos para sua utilização em cenários complexos após serem analisados os resultados.

As boas práticas de programação no MATLAB deixaram a execução do código bastante veloz. Dessa forma foi possível manter a taxa de execução de cada captura a aproximadamente 30 frames por segundo. A ideia de usar o laser semicondutor ou apontador vermelho para marcar os pontos a serem mapeados foi outro fator positivo, pois diminuiu a complexidade de emparelhar os pontos das imagens.

Apendice A

Introdução ao MATLAB

O MATLAB ou *Matrix Laboratory* [21] é uma ferramenta para cálculos matemáticos e matriciais, o qual pode ser categorizado como uma linguagem de programação. As variáveis são os objetos capazes de representar e armazenar um valor ou expressão no caso da programação comum. No caso do MATLAB, em particular, todas as variáveis são tratadas como matrizes [28] e assim possui uma característica especial: a dimensão é ajustada automaticamente. Essa característica simplifica bastante a implementação de algoritmos matriciais [29] e também aplicações de processamento de imagens digitais [25], já que estas são interpretadas como matrizes, conforme já foi dito no Capítulo 1 deste trabalho. Outra característica que facilita a programação em MATLAB é o seu extenso conjunto de rotinas de representação gráfica disponíveis em sua *toolbox* de processamento de imagens.

No MATLAB é possível modularizar a aplicação em subprogramas ou sub-rotinas, com declarações de funções ou estruturas para implementar algoritmos mais complexos. Todos esses módulos são escritos em arquivos de extensão .m e são chamados de *scripts*.

Neste capítulo irá ser abordado os conceitos básicos de utilização do MATLAB, orientados ao processamento digital de imagens. E após isso irão ser mostradas as ferramentas disponíveis e as técnicas de programação que ajudam no desempenho da aplicação. Uma dessas ferramentas tem relação ao uso da unidade de processamento gráfica (GPU) e assim irá ser feita também uma abordagem neste *hardware*.

MATLAB como ferramenta de processamento de imagens digitais

O MATLAB é uma linguagem de alto desempenho para computação técnica [31]. Essa ferramenta integra computação, visualização e programação em um

ambiente de fácil uso, onde os problemas e as soluções são expressos em uma representação ou notação matemática bastante familiar.

O MATLAB é também um sistema interativo do qual os elementos básicos de dados ou variáveis são *arrays* [24] (vetores) que não precisam de dimensionamento como já foi dito anteriormente. Isso permite a formulação de soluções para representações de matrizes relativas às imagens digitais de forma rápida e simplificada, quando comparada à formulação desse mesmo problema em linguagens de programação como C++, JAVA, e outras.

A ferramenta MATLAB foi criada originalmente para prover fácil acesso aos pacotes de desenvolvimento de cálculos matriciais LINPACK (*Linear System Package*) e EISPACK (*Eigen System Package*). Atualmente o MATLAB incorporou as bibliotecas LAPACK (*Linear Algebra Package*) e BLAS (*Basic Linear Algebra Subprograms*) que constituem de algoritmos extremamente rápidos de computação matricial.

A toolbox de processamento de imagens é uma coleção de funções do MATLAB (chamados de arquivos .m) que estendem a capacidade do ambiente MATLAB para a solução de processamento digital de imagens.

Representação de uma imagem digital no MATLAB

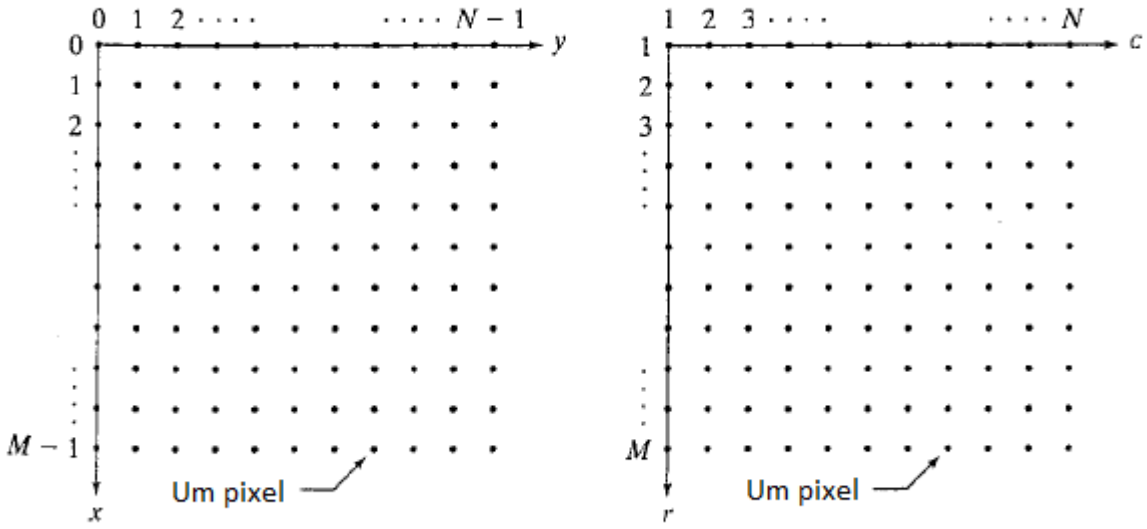
Uma imagem pode ser definida como uma função bidimensional, f(x,y), onde x e y são coordenadas espaciais, e o valor de f para cada par de coordenadas (x,y) é chamado de intensidade da imagem para esse ponto (x,y). Para imagens monocromáticas é usado o termo “tons de cinza” em referência ao aspecto visual sem cor. Já as imagens coloridas são formadas por combinações de imagens bidimensionais. Por exemplo, no sistema RGB, a imagem colorida consiste de três componentes de imagem (vermelho, verde, e azul). Por esta razão, muitas das técnicas de desenvolvimento para imagens monocromáticas podem ser estendidas para as imagens coloridas através de um processamento para as três componentes de imagem de maneira individual.

Uma imagem pode ser contínua se referindo às coordenadas x e y e também ao valor da função f(x,y). Converter uma imagem para sua forma digital requer que as coordenadas, assim como seu valor resultante da função f(x,y), sejam digitalizadas. O ato de digitalizar os valores das coordenadas é chamado de amostragem ou *sampling*. Já o ato de digitalizar os valores da função f(x,y) é denominado de quantização ou *quantization*. Então quando os valores das coordenadas (x,y) e da função f(x,y) são finitos em quantidades discretas, ou seja, representam amostras da imagem real, essa imagem pode ser chamada de imagem digital.

Imagens digitais como matrizes

O resultado da amostragem e quantização é uma matriz de números reais. Existem duas maneiras principais de representar imagens digitais. Assumindo-se que uma imagem f(x,y) é quantizada então a imagem resultante tem M linhas e N colunas. Assim a imagem tem tamanho M X N. Os valores das coordenadas (x,y) tem quantidades discretas que representam os *pixels* da imagem digital. Para facilitar a notação, são usados valores inteiros para essas coordenadas discretas. Em muitas linguagens de programação, os vetores ou matrizes têm como menor índice o valor zero, sendo assim a coordenada (0,0) a origem da função f(x,y). A próxima coordenada seguindo pela primeira da imagem é (0,1). É importante manter a ideia de que a notação (0,1) tem o significado de segunda amostra ao longo da primeira linha da imagem. Isso não significa que esses valores representem os valores das coordenadas em unidades de medida real, em metros ou centímetros por exemplo. A Figura 28 (a) mostra essa ideia. Nota-se que os valores de x vão desde 0 até M - 1, e y desde 0 até N – 1, em incrementos inteiros.

A ideia das coordenadas usadas no *toolbox* do MATLAB para denominar matrizes é diferente da ideia exposta no parágrafo anterior em 2 aspectos. Primeiro, ao invés de usar (x,y), o *toolbox* usa a notação (r,c) para indicar linhas e colunas. E o outro aspecto é que a origem do sistema de coordenadas é (1,1), e assim o valor de r vai de 1 à M e de c vai de 1 à N em incrementos inteiros. Essa ideia é apresentada na Figura 28 (b).



(a) (b)

Figura 28. Visão matricial de uma imagem (a) em muitas das bibliotecas de imagens (b) na toolbox de processamento de imagens do MATLAB.

A documentação do MATLAB referencia as coordenadas como coordenadas do *pixel*, como já foi dito anteriormente. Menos frequentemente, o *toolbox* também empregar outras ideias de coordenadas chamadas coordenadas espaciais, o qual – usa o x para se referir às colunas e y para se referir às linhas.

O sistema de coordenadas na Figura 29 mostra a seguinte representação para uma função de imagem digitalizada.

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix}$$

Figura 29. Representação matemática para a Figura 28 (a).

O lado direito dessa equação é uma imagem digital pela definição. Cada elemento dessa matriz representa um *pixel*. Essa imagem digital pode ser representada facilmente como uma matriz em MATLAB como mostra a Figura 30.

$$f = \begin{bmatrix} f(1,1) & f(1,2) & \cdots & f(1,N) \\ f(2,1) & f(2,2) & \cdots & f(2,N) \\ \vdots & \vdots & & \vdots \\ f(M,1) & f(M,2) & \cdots & f(M,N) \end{bmatrix}$$

Figura 30. Representação matemática da Figura 28 (b).

Obviamente as duas representações tem a mesma ideia, tendo só a diferença no deslocamento da origem de (0,0) para (1,1). A notação f(p,q) se trata de um *pixel* localizado na linha p e coluna q. Por exemplo, f(6,2) é o *pixel* na sexta linha e segunda coluna da matriz f. Uma matriz ou imagem 1 x 1 é um escalar.

Capturando as imagens das câmeras digitais com o MATLAB

As imagens das câmeras são capturadas no ambiente MATLAB através da função *getsnapshot*, a qual tem sintaxe: getsnapshot(OBJ). OBJ é um objeto do tipo *video input* e necessita corresponder a somente um único dispositivo, ou seja, é uma matriz 1x1 de objetos do tipo *vídeo input*. Essa função retorna imediatamente um *frame*. Um *frame* nada mais é do que um quadro do vídeo da câmera digital, ou seja, a função *getsnapshot* captura exatamente a imagem corrente da câmera. O *frame* retornado por essa função corresponde a um tipo de dados nativo usando o espaço de cores especificado pela propriedade *ReturnedColorSpac*e do objeto *vídeo input*. Por fim, o *frame* capturado irá ser representado por uma matriz, onde as dimensões e os tipos dos elementos dessa matriz são determinados pelas propriedades do objeto *video input*.

Os objetos do tipo *video input* são obtidos através da função *videoinput*, a qual tem a sintaxe: videoinput(ADAPTORNAME, DEVICEID, FORMAT). ADAPTORNAME é uma string ou conjunto de caracteres que representa a especificação do nome do adaptador usado na comunicação com o dispositivo de vídeo, ou seja, as câmeras digitais. O DEVICEID é um número inteiro positivo que identifica o dispositivo por onde serão obtidas as imagens. A função *imaqhwinfo* pode ser usada para saber quais dispositivos estão conectados no adaptador e quais são os DEVICEIDs desses dispositivos. A sintaxe dessa função é:

imaqhwinfo(ADAPTORNAME). Por fim, o FORMAT é uma string ou conjunto de caracteres que especifica o formato de vídeo suportado pela câmera. A função *imaqhwinfo* explica quais são os formatos suportados pelos dispositivos.

Segue um exemplo de uso dessas funções para a captura de uma imagem ou *frame* de uma câmera digital:

```
info = imaqhwinfo('winvideo')
info.DeviceInfo.SupportedFormats
vid1 = videoinput('winvideo', 1, 'YUY2_640x480');
vid1.ReturnedColorspace = 'rgb';
start(vid1)
frame = getsnapshot(vid1);
```

A função start serve somente para inicializar a captura de frames dos dispositivos pelo adaptador.

Visualizando as imagens com o MATLAB

As imagens das câmeras são mostradas no MATLAB usando a função *imshow*, a qual tem a sintaxe: imshow(f). Onde f é uma matriz que representa uma imagem, podendo ser no caso o *frame* capturado pela função *getsnapshot*. Essa função pode mostrar uma imagem colorida para uma matriz RGB, uma imagem em tons de cinza para uma matriz monocromática ou uma imagem preta e branca para uma matriz binarizada, ou seja, que todos os seus elementos são valores boolianos ou zeros e uns.

A função *pixval* é usada frequentemente para mostrar aos valores de cada *pixel* individualmente. Essa função também mostra as coordenadas em que o cursor está em relação à imagem mostrada. Quando o cursor é movido por cima da imagem através do mouse, as coordenadas e os valores de intensidade do *pixel* são apresentados na tela. Essas informações são mostradas na parte de baixo da janela da figura. No caso de imagens coloridas, as coordenadas assim como os componentes vermelho, verde e azul são apresentados.

No caso de alguma outra imagem ser visualizada usando *imshow*, o MATLAB põe essa imagem por cima da que já estava sendo mostrada. Para manter a

primeira imagem e visualizar a segunda imagem pode-se usar a função *figure*. Segue um exemplo do uso dessa função:

```
imshow(f),figure, imshow(g)
```

Programação eficiente no MATLAB

Na aplicação proposta nesse trabalho deve capturar as imagens, processá-las para encontrar a marcação do laser, e emparelhar os pontos para calcular a distância. Essa sequência irá ser repetida várias vezes, durante o processo de gravação. Então é importante que a aplicação execute essa sequência de uma maneira que seja tão rápida quanto à captura das câmeras. Uma velocidade de processamento de 30 *frames* por segundo (ou 30 FPS) irá ser adotada como objetivo da aplicação e causará uma impressão de tempo real.

Existem diversas boas práticas no MATLAB no que se trata de programação eficiente. Neste trabalho irão ser abordadas as seguintes práticas de programação: Pré-alocação de matrizes, vetorização e programação em GPU. A abordagem irá ser feita partindo do ponto de que se tenha uma noção de programação básica do MATLAB.

Pré alocação de matrizes

As matrizes do MATLAB são habilitadas para dinamicamente aumentar as linhas e colunas. Por exemplo,

```
>> a = 3

a =

     3

>> a(2,6) = 7

a =

     3     0     0     0     0     0

     0     0     0     0     0     7
```

O MATLAB automaticamente ajusta a matriz. Internamente, a matriz precisa ser realocada na memória com um tamanho maior. Se uma matriz for ajustada várias vezes, como em um loop por exemplo, essa tarefa poderá ser significante no tempo de execução. Para evitar realocações frequentes, pré-alocar a matriz com a função zeros pode ser considerada como uma boa prática. Considera-se o código:

```
a(1) = 1;
b(1) = 0;
for k = 2:8000
     a(k) = 0.99803 * a(k - 1) - 0.06279 * b(k - 1);
     b(k) = 0.06279 * a(k - 1) + 0.99803 * b(k - 1);
end
```

Esse código pode em uma máquina demorar 0,47 segundos para rodar por exemplo. Depois de todas as repetições, ambas as matrizes tem o dimensionamento de 1 linha e 8000 colunas. Então o ato de pré-alocar, criando matrizes vazias com 8000 colunas é uma boa prática:

```
a = zeros(1,8000); %pré alocação
b = zeros(1,8000);
a(1) = 1;
b(1) = 0;
for k = 2:8000
a(k) = 0.99803 * a(k - 1) - 0.06279 * b(k - 1);
b(k) = 0.06279 * a(k - 1) + 0.99803 * b(k - 1);
     end
```

Com esta modificação, o código pode demorar 0.14 segundos na mesma máquina do exemplo anterior, ou seja, aproximadamente 3 vezes mais rápido. Pré-alocar é frequentemente fácil de se fazer, neste caso foi somente necessário determinar justamente um pré-alocação ideal para o problema.

No caso da aplicação, será uma boa prática a pré-alocação da imagem de profundidade, cujos valores dos *pixels* representam a profundidade ou o grau de disparidade das imagens do par estereoscópico, que é o resultado da aplicação.

Vetorização

Uma computação é vetorizada quando se tem uma vantagem ou eficiente em usar operações entre vetores ou matrizes. Uma variedade de situações na programação pode ser vetorizada, e frequentemente melhora a velocidade de

execução em 10 vezes ou até muito mais. Vetorização é uma das mais efetivas técnicas para escrever um código realmente eficiente em MATLAB.

Muitas funções padrões do MATLAB são vetorizadas, ou seja, elas podem operar em um vetor ou matriz como se a função fosse aplicada individualmente para cada elemento do vetor ou da matriz. Por exemplo:

```
>> sqrt([1,4;9,16])
ans =
     1     2
     3     4

>> abs([0,1,2,-5,-6,-7])
ans =
     0     1     2     5     6     7
```

Considerando a seguinte função:

```
function d = minDistance(x,y,z)
% achar a minima distância do ponto até a origem
nPoints = length(x);
d = zeros(nPoints,1); % pre alocação
for k = 1:nPoints % calcular a distância para cada ponto
d(k) = sqrt(x(k)^2 + y(k)^2 + z(k)^2);
end
     d = min(d); % pegar a distância mínima
```

Para cada ponto, a distancia para a origem é computada e armazenada em d, Para aumento de desempenho, o vetor d é pré-alocado. A distância mínima é então encontrada com a variável min. Para vetorizar este código é só trocar a iteração ou repetição pelas operações entre vetores:

```
function d = minDistance(x,y,z)
% achar a minima distância do ponto até a origem
d = sqrt(x.^2 + y.^2 + z.^2); % calcular a distância para cada
ponto
     d = min(d); % pegar a distância mínima
```

Cada elemento dos vetores x, y e z são elevados ao quadrado pelo operador .^ . Os operadores de multiplicação e divisão entre os elementos das matrizes são, respectivamente, .* e ./ . O quadrado dos componentes são adicionados entre si pelo operador + . Finalmente, a raiz quadrada do vetor soma é computada para cada elemento utilizando a função sqrt.

A primeira versão do distanciaMinima demora em uma máquina 0.73 segundos para executar em vetores de 50000 colunas. A versão vetorizada nessa mesma máquina demora menos de 0.04 segundos para executar, ou seja, mais do que 18 vezes mais rápida. Isso se deve pelo motivo de que são evitadas as sucessivas comparações para condição de saída das iterações, as quais exigem muito mais recurso computacional do que uma operação vetorizada.

Existe a vetorização também para operadores relacionais tais como >, >=, <, <=, ¬= e == , conforme o exemplo:

```
function [ binario ] = binarizarEmVermelho(
R,G,B,limiarVermelho )
    binario = 255* uint8((double(R) - double(G)) >
limiarVermelho).*uint8((double(R) - double(B)) >
limiarVermelho);
end
```

onde R, G e B são matrizes das componentes RGB de uma imagem colorida e limiarVermelho um número escalar, podendo ser também uma matriz.

Programação em GPU no MATLAB

Máquinas *multicore* (com vários núcleos de processamento) e com tecnologia de *hyper-threading* (um processador simula em ser mais de um, executando simultaneamente mais de um processo) tem entusiasmado as pessoas por acelerar aplicações de computação intensa. A unidade de processamento gráfico (*graphics processing unit*) ou GPU é um tipo de *hardware* que promete um alto desempenho computacional para tarefas com processos independentes que possam ser executados em paralelo.

Originalmente usada para acelerar as tarefas relacionadas ao gráfico do sistema, as GPU's são cada vez mais utilizadas em cálculos científicos. Ao contrário da unidade de processamento central (*central processing unit*) ou CPU, que possui não mais do que 8 núcleos, a GPU tem uma matriz de processadores massivamente paralelos, bem como uma memória dedicada de alta velocidade. Os processadores da GPU são subdimensionados em relação aos da CPU, ou seja, possuem um poder computacional (desempenho) individual bem menor. A Figura 31 mostra uma comparação entre uma CPU e uma GPU.

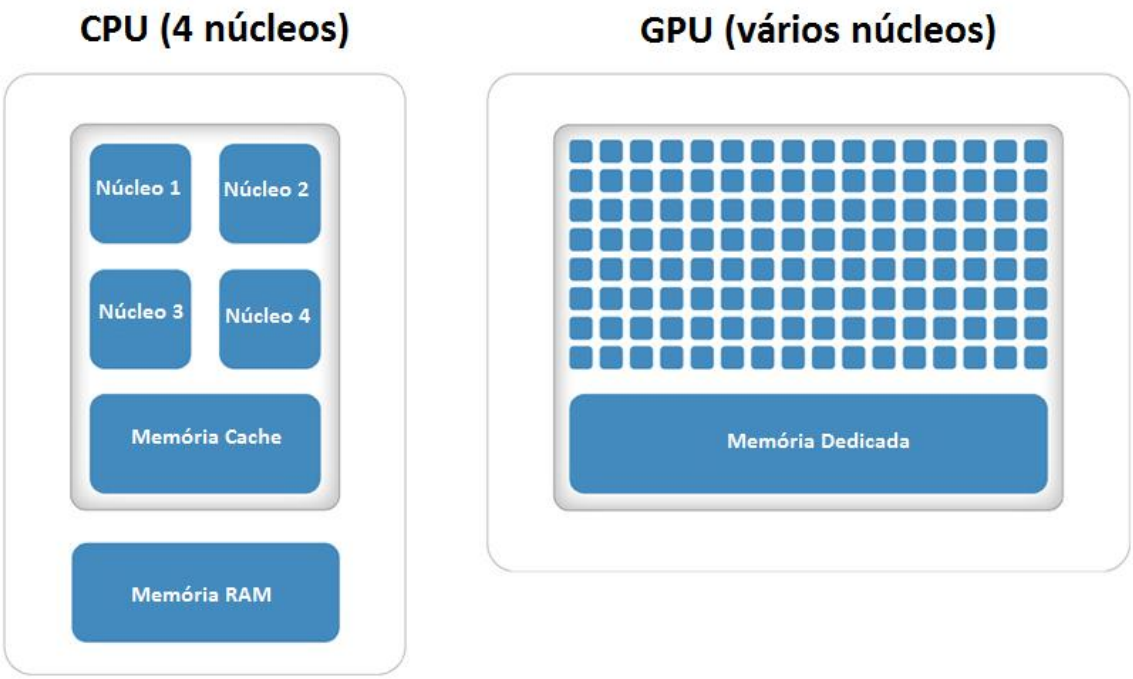


Figura 31. Uma abordagem superficial da arquitetura de uma CPU e de uma GPU.

O grande aumento do rendimento ao utilizar a GPU, no entanto, tem um custo. Em primeiro lugar, o acesso à memória se torna um gargalo para os cálculos. Os dados devem ser enviados desde a CPU para a GPU antes de se fazerem os cálculos e, em seguida serem lidos os resultados depois enviá-los de volta para CPU. Isso porque uma GPU é conectada na CPU via barramento *PCI Express* e a memória de acesso é mais lenta do que a de uma CPU tradicional. Isso significa que o aumento do desempenho computacional é limitado pelo montante de dados transferidos proveniente do algoritmo da aplicação a ser feita. Em segundo lugar, a programação para GPU em C ou Fortran, linguagens de programação em que são feitos os pacotes LAPACK e BLAS do MATLAB, requer um diferente modelo a ser seguido que pode se tornar complicado e demorado.

No caso do processamento das imagens digitais, a programação em GPU pode ser de grande utilidade para aumentar o desempenho da aplicação. Cada *pixel* da imagem pode ser processado de forma paralela, precisando de apenas uma operação a ser feita ao mesmo tempo para vários elementos da imagem.

Existe então a função gpuArray, que tem como parâmetro de entrada uma matriz e o resultado é a matriz carregada na GPU. Todas as operações feitas

usando vetorização irão ser processadas em paralelo usando a GPU para obter rapidamente as respostas.

Apêndice B

Código da aplicação em MATLAB

```
dephImage = zeros(480,640);

vid1 = videoinput('winvideo', 2, 'YUY2_640x480');
vid1.ReturnedColorspace = 'rgb';
start(vid1)
preview(vid1);

vid2 = videoinput('winvideo', 3, 'YUY2_640x480');
vid2.ReturnedColorspace = 'rgb';
start(vid2)
preview(vid2);


resolucao=240;
qualidade=240;
limiarBrilho= 230;
limiarVermelho = 40;
limiarMovimentacao = 50;

k=13200;
f=1;

fprintf('Iniciando busca do laser');



maxZ = 1;

for j=1:500

    tic;

    data1 = getsnapshot(vid1); %captura a imagem RGB da camera
1
    data2 = getsnapshot(vid2); %captura a imagem RGB da camera
2


    R = data1(:,:,1);
    G = data1(:,:,2);
    B = data1(:,:,3);
```

```
    binarioVermelho =  binarizarEmVermelho(
R,G,B,limiarVermelho );

    pontos1 =
buscaDaLinhaDeLaser(binarioVermelho,1,length(binarioVermelho(:
,1)));

    R = data2(:,:,1);
    G = data2(:,:,2);
    B = data2(:,:,3);

    binarioVermelho =  binarizarEmVermelho(
R,G,B,limiarVermelho );

    pontos2 =
buscaDaLinhaDeLaser(binarioVermelho,1,length(binarioVermelho(:
,1)));

    pontos = aplicarEstereoscopia(pontos1,pontos2,k);

    while pontos.size>0
        xyz=pontos.pop;
        if maxZ<xyz(3)
            maxZ = xyz(3);
        end
        dephImage(xyz(2),xyz(1)) = xyz(3);
    end
    dephImageNormalized = 255*dephImage / maxZ;
    figure(4), subplot(1,2,1),
imshow(uint8(dephImageNormalized));
    figure(4), subplot(1,2,2), imshow(data1);

    toc;
end

stoppreview(vid1)
closepreview(vid1)
delete(vid1)

stoppreview(vid2)
closepreview(vid2)
delete(vid2)
```

```
function [f_y] = buscaDaLinhaDeLaser(
binarioVermelho,limitadorDeBuscaxMin,limitadorDeBuscaxMax)
        resolucao = size(binarioVermelho);
        altura = resolucao(1);

        f_y = -1*ones(1,altura);
        for i=1:altura
            for j=limitadorDeBuscaxMin:limitadorDeBuscaxMax
                if(binarioVermelho(i,j) == 255)
                    f_y(i) = j;
                    break;
                end
            end
        end
end

function [ binario ] = binarizarEmVermelho(
R,G,B,limiarVermelho )
    binario = 255* uint8((double(R) - double(G)) >
limiarVermelho).*uint8((double(R) - double(B)) >
limiarVermelho);
end
```

Referências

[1] ESTEVES, G. R. P. **Estereoscopia no cálculo de distância e controle de plataforma robótica.** Monografia apresentada como requisito parcial para obtenção do diploma de Bacharel em Engenharia Eletrônica pela Escola Politécnica de Pernambuco - Universidade de Pernambuco; 2011.

[2] FRANÇA, J. A. **Desenvolvimento de Algoritmos de Visão Estereoscópica para Aplicações em Robótica Móve**l. UFSC, Florianópolis - SC, Brasil

[3] BERTOZZI, M.; BROGGI, A. **GOLD: a parallel real-time stereo vision system for generic obstacle and lane detection**. IEEE Transactions on Image Processing, 1997. URL http://citeseer.nj.nec.com/bertozzi98gold.html.

[4] RAPOSO, A. B.; SZENBERG, F., GATTASS, M.; CELES, W. **Visão Estereoscópica, Realidade Virtual, Realidade Aumentada e Colaboração**.

[5] GONZALEZ, R. C.; WOODS, R. E. **Processamanto de Imagens Digitais.** Editora Edgard Blücher.

[6] LUNAZZI, J. J.; SERRA, R. L.; MAGALHAES, D. S. F. **Estereoscópio com tela holográfica para ver tomografias**. Anais do encontro latino-americano “LatinDisplay 2008”, p. 86-88, Brasil; 2008

[7] SHIBATA, S.; YAMAMOTO, T. **Human-Robot Interface with Instruction of Neck Movement Using Laser Pointer**. IEEE/SICE International Symposium on System Integration (SII); 2011

[8] WADA, T.; TAKAHASHI, M.; KAGAWA, K.; OHTA, J.**Laser Pointer as a Mouse**. SICE Annual Conference 2007; Set 2007

[9] TAKAHASHI, Y.; YASHIGE, M. **Hand System of Robotic Manipulator with Human Interface using Laser Pointer**. Conference of the IEEE Industrial Electronics Society; 2001

[10] XIE, W.-F.; LI, Z.; TU, X.-W.; PERRON, C. **Switching Control of Image-Based Visual Servoing With Laser Pointer in Robotic Manufacturing Systems**. IEEE Transactions on Industrial Electronics, vol 56, no 2; Fev 2009

[11] HU, J.-S.;WANG, J.-J. **Surface Coordinate Measurement Using Un-calibrated Cameras and Lase Pointer Equipped Robot Manipulator.** International Conference on Advanced Intelligent Mechatronics; Jul 2011

[12] YASHIGE, M.; TAKAHASHI, Y. **Robotic manipulator operated by human interface with positioning control using laser pointer**. Industrial Electronics Society; 2000.

[13] FUKUDA, Y.; KURIHARA, Y.; KOBAYASHI, K.; WATANABE, K. **Development of electric wheelchair interface based on laser pointer**. ICROS-SICE International Joint Conference, Japão; 2009

[14] FERNÁNDEZ, F.; CHÁVEZ, F. **Evolutionary Learning of a Laser Pointer detection Fuzzy System for an Environment Control System**. IEEE International Conference on Fuzzy Systems, Taiwan; 2011

[15] ZHANG, L.; SHI, Y.; CHEN, B. **NALP: Navigating Assistant for Large Display Presentation using Laser Pointer.** IEEE Computer Society First International Conference on Advances in Computer-Human Interaction; 2008

[16] CHENG, Y.; XIN, D.; ZHANG, J. **Research on the Key Technology of the Emission System for Semiconductor Laser Range Finder**. IEEE International Conference on Optoelectronics and Microelectronics; 2012

[17] HUI, R.; CAPONIO, N.; BENEDETTO, S.; MONTROSSET, I. **Linewidth of a Semiconductor Laser Operating Near Threshold**. IEEE Photonics Technology Letters, vol 4, no8; Ago 1992

[18] TREUSCH, H.-G.; OVTCHINNIKOV, A.; HE, X.; KANSKAR, M.; MOTT, J.;YANG, S. **High-Brightness Semiconductor Laser Sources for Materials Processing: Stacking, Beam Shaping, an Bars**. IEEE Journal of Selected topics in Quantum Electronics, vol 6, no 4; Ago 2000

[19] LV, Z.; ZHANG, Z. **Build 3D laser scanner based on binocular stereo vision**. In Intelligent Computation Technology and Automation (ICICTA), 2011 International Conf. on, vol. 1, p. 600-603; 2011

[20] GRILO, M. P. R.; FREIRE, R. **Laser Semicondutor** Monografia – Universidade de São Paulo, São Paulo (SP); 2010

[21] MILOSAVLJEVIC, I. Z.; JABRI, M. A. **Automatic Array Alignment in Parallel Matlab Scripts**. 2006

[22] EDELSMAN, A.; CHOY, R. **Parallel MATLAB: Doing It Right.** Proceedings of the IEEE, vol 93, no 2; Fev 2005

[23] MOYA, V.; GONZALEZ, C.; ROCA, J.; FERNANDEZ, A.; ESPASA, R. **Shader Performance Analysis on a Modern GPU Architecture**. Proceedings of the 38th Annual IEEE/ACM Internacional Symposium on Microarchitecture; 2005

[24] MENON, V.; TREFETHEN, A. E. **MultiMATLAB: Integrating MATLAB with Hight-Performance Parallel Computing**. Proceedings of the ACM/IEEE; 1997

[25] GUILFOOS, B.; GARDINER, J.; CHAVES, J. C.; NEHRBASS, J.; AHALT, S.; KRISHNAMURTHY, A.; UNPINGCO, J.; CHALKER, A.; HUMPHREY, L.; SAMSI, S. **Applications in Parallel MATLAB**. IEEE/HPCMP Users Group Conference; 2006

[26] KRISHNAMURTHY, A.; NEHRBASS, J.; CHAVES, J. C.; SAMSI, S. **Survey of Parallel MATLAB Techniques and Applications to Signal and Image Processing**. IEEE; 2007

[27] BLISS, N. T.; KEPNER, J.; KIM, H.; REUTHER, A. **PMATLAB: Parallel MATLAB Library for signal Processing Applications.** Proceding of IEEE: Acoustic, Speech and Signal Processing, vol 4; 2007

[28] TANWER, A.; SINGH, R.; REEL, P. S. **Real Time Low Level Parallel Image Processing For Active Vision Systems**. IEEE International Advance Computing Conference; 2009

[29] GORYAWALA, M.; GUILLEN, M. R.; BHATT, R.; MCGORON, A.; ADJOUADI, M. **A Compatarative Study on the Performance of the Parallel and Distributing Computing Operation in MatLab.** IEE International Conference on Advanced Information Networking and Applications; 2010

[30] BAUER, S.; KOHLER, S.; DOLL, K.; BRUNSMANN, U. **FPGA-GPU Architecture for Kernel SVM Pedestrian Detection**. IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Estados Unidos; 2010

[31] WANG, G. **Power Analysis and Optimizations for GPU Architecture using a Power Simulator**. 3th International Conference on Advance Computer Theory and Enginnering; 2010

[32] BRANDAO, D.; ZAMITH, M.; CLUA, E.; MONTENEGRO, A. **Performance Evaluation of Optimized Implementations of Finite Difference Method for Wave Propagation Problems on GPU Architecture.** 22nd International Symposium on Computer Architeture and High Performance Computing Worshops; 2010

[33] FRESSE, V.; HOUZET, D.; GRAVIER,C. **GPU architecture evaluation for multispectral and hyperspectral image analysis**. Design and Architectures for Signal and Image Processing (DASIP), 2010 Conference on. IEEE, 2010.