Exploração da Visão Estéreo

Rodrigo Ferreira Guimarães
rodrigofegui@unb.br

Departamento de Ciência da Comptutação
Universidade de Brasília
Campus Darcy Ribeiro, Asa Norte
Brasília-DF, CEP 70910-900, Brazil,

Resumo

A compreensão do mundo através da visão é um conceito intuitivo à humanidade, mas ao possibilitar isso às máquinas têm-se um extenso campo de pesquisa, a visão computacional. Ao se tratar da compreensão 3D do mundo é preciso recorrer as técnicas de visão estéreo e cálculo dos mapas de disparidade e de profundidade para os objetos de estudo para imagens correspondentes retificadas ou não. Este projeto visa a exploração e o entendimento sobre esses conceitos, com implementação em Python com uso da biblioteca *OpenCV*.

1 Introdução

A percepção sobre o mundo é feita através dos sentidos, dentre eles existe a visão. Com ela a humanidade pode distinguir formas, padrões, luzes e sombras de estruturas 3D que a cerca. Ao possibilitar esta mesma interpretação ao computador têm-se a denominada *visão computacional* [5]. Esta área possui diversas aplicações, como: *autentificação visual*, *geração de panorâmicas*, *captura de movimentos*, *vigilância*, *modelagem 3D a partir de imagens* e tantas outras mais. Para os casos cujo objetivo é entender o mundo a partir das múltiplas imagens é preciso usar *visão estéreo*.

1.1 Geometria

No sistema de coordenadas euclidianas há um problema da representatividade da origem e dos demais pontos, uma vez que o primeiro é um ponto distinto e os demais são geometricamente idênticos. Como solução têm-se as *coordenadas homogêneas*, onde a origem é removida do plano e acrescida de uma dimensão [3]. Nestas coordenadas, têm-se as equivalências de representação: pontos 2D são representados como $\mathbf{X} = (x, y)$ no espaço euclidiano são representados como $\tilde{\mathbf{X}} = \tilde{\mathbf{w}}(x, y, 1)$; linha 2D de $\mathbf{L} = ax + by + c$ paraa $\tilde{\mathbf{L}} = (a, b, c)$; e ponto 3D de $\mathbf{X} = (x, y, z)$ para $\tilde{\mathbf{X}} = \tilde{\mathbf{w}}(x, y, z, 1)$.

Em coordenadas homogêneas, as transformações geométricas são multiplicação de matrizes, como: a *translação* ($\mathbf{x}'_{trans} = \begin{bmatrix} \mathbf{I} \mid \mathbf{t} \end{bmatrix} \tilde{\mathbf{x}}$) e a *rotação com translação* ($\mathbf{x}'_{rot+trans} = \begin{bmatrix} \mathbf{R} \mid \mathbf{t} \end{bmatrix} \tilde{\mathbf{x}}$), onde $\mathbf{R}\mathbf{R}^T = \mathbf{I}$ e $|\mathbf{R}| = 1$.

Além disso, um produto vetorial $\mathbf{v}_\times = \mathbf{a} \times \mathbf{b}$ é equivalente à multiplicação de matrizes $\mathbf{v}_\times = [\mathbf{a}]_\times \mathbf{b}$, sendo $[\mathbf{a}]_\times$ a forma matricial do operador de produto vetorial [5].

1.2 Câmeras

Para a aquisição de imagens, uma câmera é posicionada no mundo com um ponto de referência . Dessa forma, por semelhança de triângulos, a projeção $\mathbf{P}$ de pontos 3D do mundo em pontos 2D da imagem corresponde a $(x,y,z) \rightarrow (f\frac{X}{Z}, f\frac{Y}{Z}, f) = (f\frac{X}{Z}, f\frac{Y}{Z})$, ou seja, $\mathbf{PX} = \mathbf{x}$.

A projeção $\mathbf{P}$ é a combinação de três transformações de coordenadas: *matriz de intrínsecos* $\mathbf{K}$ (posicionamento do ponto focal e do ponto principal, ângulo entre eixos e outros), matriz de projeção $[\mathbf{I} \mid \mathbf{0}]$ e *matriz de extrínsecos* $[\mathbf{R} \mid \mathbf{t}]$ (relação entre as coordenadas da câmera e os *pixels*) [2].

1.3 Visão Estéreo

A partir de duas ou mais imagens de um mesmo local (objeto ou até uma pessoa), é possível extrair, identificar e triangular suas características nas imagens e obter uma modelagem 3D; os problemas de *correspondência* [2].

A priori a busca pelas correspondências ocorreria por toda a imagem; com uso das *linhas epipolares* reduz-se a uma busca linear. Sabe-se que o ponto real $\mathbf{p}$ é representado pelos pontos $\mathbf{x_0}$ e $\mathbf{x_1}$ das imagens. Ao reprojetar linearmente o ponto $\mathbf{p}$ no infinito, $\mathbf{p}_\infty$, de modo a continuar sendo projetado no ponto $\mathbf{x_0}$ é possível determinar a linha epipolar a partir da diferença de projeção em relação a $\mathbf{x_1}$ [5].

Com a aquisição das linhas epipolares de imagens correspondentes é possível reduzir a busca de correspondência numa linha horizontal, para alguns conjuntos de imagem [5], sendo denominado de *retificação*.

Além disso, a partir de duas imagens correspondentes a quantidade de movimentação que é percebida é denominada *disparidade*. Essa grandeza informa a distância entre os *pixels* correspondentes das imagens: $d_{(x,y)} = |x - x'| + |y - y'|$. Ao se calcular a disparidade para todos os *pixels* da imagem obtém-se seu *mapa de disparidade* . No cenário de imagens correspondentes retificadas onde uma está certamente à direita da outra, a busca é ainda reduzida à esquerda do *pixel* base e vice-versa.

Com a aquisição do mapa de disparidade e em pose dos dados das câmeras (distância focal da imagem base f e distância do mundo entre as câmeras b) é possível calcular o *mapa de profundidade*, uma vez que $Z_{xy} = (f \cdot b)/d_{xy}$.

2 Desafio

Ao utilizar pares de imagens correspondentes deve-se buscar o entendimento e a exploração sobre a visão estéreo ao: extrair mapas de disparidade e de profundidade; utilizar os dados de calibração das câmeras e medição de objetos em 3D. Para tanto, três requisitos devem ser atendidos:

1. **Estimativa de mapa de profundidade a partir de imagens estéreo retificadas**: manipulando, pelo menos, os conjuntos de imagens *"Jadeplant"* e *"Playtable"* da base de imagens de *Middleburry* de 2014 [1] da configuração perfeita para computar os mapas de disparidade e de profundidade; a métrica BAD2.0 deve ser empregada;

2. **Câmeras estéreo com convergência**: manipulando, pelo menos, os conjuntos de imagens *"Morpheus"* não retificado da base de imagens produzida por Furukawa e Ponce (indisponível) para computar o mapa de disparidade;

3. **Paralelepípedo**: manipulando os mapas obtidos com o requisito anterior para computar a menor caixa que comporta os cliques de mouse fornecidos.

3 Metodologia

A implementação foi realizada em Python, ver. 3.7.2, utilizando *OpenCV* e está disponível online no GitHub [4].

Com o intuito de intensificar a caracterização dos *pixels* das imagens de entradas, estas são convertidas para a escala de cinza e têm seu contraste e brilho ajustados seguidos pela equalização de histograma, conforme demonstrado na Figura 1.

(a) (b) (c)

Figura 1: Evolução do pré-processamento das imagens base, onde esta (a) é convertida para escala de cinza (b) e é ajustada (c).

Com o caráter exploratório como principal objetivo, cinco algoritmos foram implementados:

1. **OpenCV (OCV)**: método do algoritmo SBGM (*Semi-Global Block Matching*) implementados pela biblioteca, gerando dois mapas de disparidades tendo ambas as imagens como base e pós-processando-os por intermédio da filtragem WLS (*Weighted Least Squares*). Para tanto, uma exploração sobre a parametrização também foi realizada: *uniquenessRatio* (porcentagem de vantagem entre as duas melhores correspondências para sua confirmação), *speckleWindowSize* (tamanho máximo das regiões de disparidade suavizadas), *speckleRange* (variação máxima entre os componentes conectados), *wls_lambda* (quantidade de regularização durante o filtro) e *wls_sigma_color* (sensibilidade às bordas durante a filtragmem);

2. **Busca linear (BL)**: em cada linha retificada, considera-se como referência **R** (imagem da esquerda) um vetor de tamanho máximo N e como janela de busca (imagem da direita) de tamanho máximo M. Com isso desliza-se uma janela do tamanho da referência obtida e a diferença entre janela deslizante **W** e a referência é computada, seguida pela busca da janela de menor diferença. A janela encontrada é validada considerando uma variação $|V|/pixel$;

3. **Correlação cruzada (CC)**: semelhante à *Busca linear* até o cálculo da diferença, visto que é computado "o ângulo" entre a janela deslizante e a referência: $(\mathbf{R} \cdot \mathbf{W})/(|\mathbf{R}| \cdot |\mathbf{W}|)$. A janela encontrada é validada considerando uma variação menor ou igual a 30^o $(0,15)$;

4. **Janela deslizante (JD)**: semelhante à *Busca linear*, considera uma vizinhança 2D quadrada de tamanho fixo;

5. **Janela deslizante adaptativa (JDA)**: semelhante à *Janela deslizante*, considera uma vizinhança 2D de tamanho variável controlada pela janela de busca.

Ainda sob esta perspectiva, foram explorados diversos tamanhos ímpares para os blocos de verificação com $N \in [3,17]$. Além disso, tanto os mapas de disparidade e de profundidade foram normalizados para comparação também.

4 Resultados

Considerando o aspecto exploratório deste projeto sobre visão estéreo, as primeiras investigações manipularam as imagens redimensionadas, devido à limitação do tempo de processamento.

A avaliação sobre os parâmetros do algoritmo *OpenCV* se deu como: *wls_sigma_color* $\in [80,7980] \implies 80$, *wls_lambda* $\in [0.5,1.5] \implies 0.5$, uma vez que os erros foram diretamente proporcionais às suas variações; *speckleRange* $\in [1,2] \implies 1$, *speckleWindowSize* $\in [50,200] \implies 200$, *uniquenessRatio* $\in [5,15] \implies 10$, uma vez que não demonstraram influência direta frente os demais parâmetros. Com esta análise realizada, têm-se os resultados por requisitos.

4.1 Requisito 1

Devido à pouca fluência sobre a biblioteca *NumPy* este foi o requisito mais desafiador, já que a abordagem de manipulação das imagens comumente utilizada nas linguagens C/C++ foi utilizada: uso de laços *for*, o que tem baixíssima performance em Python.

Uma vez utilizando a biblioteca, os algoritmos foram avaliados tanto no original quanto normalizados, com exceção da *Janela Deslizante Adaptativa* por ser um algoritmo de baixa performance temporal. A Tabela 1 traz os desempenhos sobre os dados originais, encontrando os melhores resultados para ambas as imagens base com a *Janela Deslizante* de tamanho 15 e os piores com a *OpenCV* independente do tamanho da janela. Para os dados normalizados, a Tabela 2, encontrando os melhores resultados para ambas as imagens com a *OpenCV* variando o tamanho entre $9-11$ e os piores com a *Busca Linear* para "Jadeplant" e com a *Correlação Cruzada* para "Playtable" ambas com janelas de tamanho 5.

		Tamanho do bloco, N						
Algoritmo	Imagem	3	5	7	9	11	13	15
OCV	Jadeplant	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	Playtable	0.0	0.0	0.0	0.0	0.0	0.0	0.0
BL	Jadeplant	0.62	0.81	0.72	0.88	1.41	0.59	0.55
	Playtable	1.84	0.75	0.78	1.53	1.37	1.46	1.39
CC	Jadeplant	1.01	1.08	0.95	0.9	0.89	0.88	0.86
	Playtable	3.41	5	1.32	0.93	0.79	0.72	0.68
JD	Jadeplant	2.33	3.29	4.18	5.04	5.89	6.74	7.56
	Playtable	31.56	40.74	45.12	47.84	49.55	50.76	51.66
JDA	Jadeplant	-	2.62	-	-	-	-	-
	Playtable	-	-	-	-	-	-	-

Tabela 1: Desempenho percentual dos acertos dos algoritmos com base na métrica BAD2.0.

A partir da Figura 2 fica claro o motivo da *OpenCV* ficar com os piores resultados: os valores ficaram em um *range* muito superior ao esperado ($[-30000,10000]$ frente ao $[27,600]$,

		Tamanho do bloco, N						
Algoritmo	Imagem	3	5	7	9	11	13	15
OCV	Jadeplant	8.47	9.15	12.47	46.66	52.13	51.31	50.22
	Playtable	4.65	4.66	4.68	55.85	54.92	54.85	53.77
BL	Jadeplant	4.47	4.09	3.93	4.04	4.12	4.25	4.36
	Playtable	4.53	4.05	4.13	5.25	5.11	4.87	4.75
CC	Jadeplant	3.75	3.64	3.7	3.82	3.9	3.93	3.98
	Playtable	5.29	8.86	17.3	29.53	38.64	39.47	36.86
JD	Jadeplant	4.13	4.07	4.04	4	3.97	3.95	3.94
	Playtable	8.07	7.68	7.71	7.76	7.84	7.91	7.96
JDA	Jadeplant	-	-	-	-	-	-	-
	Playtable	-	-	-	-	-	-	-

Tabela 2: Desempenho percentual dos acertos dos algoritmos normalizados com base na métrica BAD2.0.

para a "Jadeplant" e $[0, 4000+]$ frente ao $[27, 270]$, para a "Playtable"); o mesmo não pode ser dito sobre os melhores. Para estes últimos, a grande quantidade de detalhes da "Jadeplant" não foi bem administrada pelos algoritmos, uma vez que nenhum teve um aproveitamento acima de 10%, mas como a "Playtable" possui regiões mais constantes os algoritmos tiveram melhores desempenhos. Ao passo que ao normalizar os resultados no *range* $[0, 255]$, como ilustrados na Figura 3, o algoritmo da *OpenCV* detêm os melhores resultados por possuir os blocos correspondentes mais bem definidos e na mesma faixa do *groun truth* normalizado.

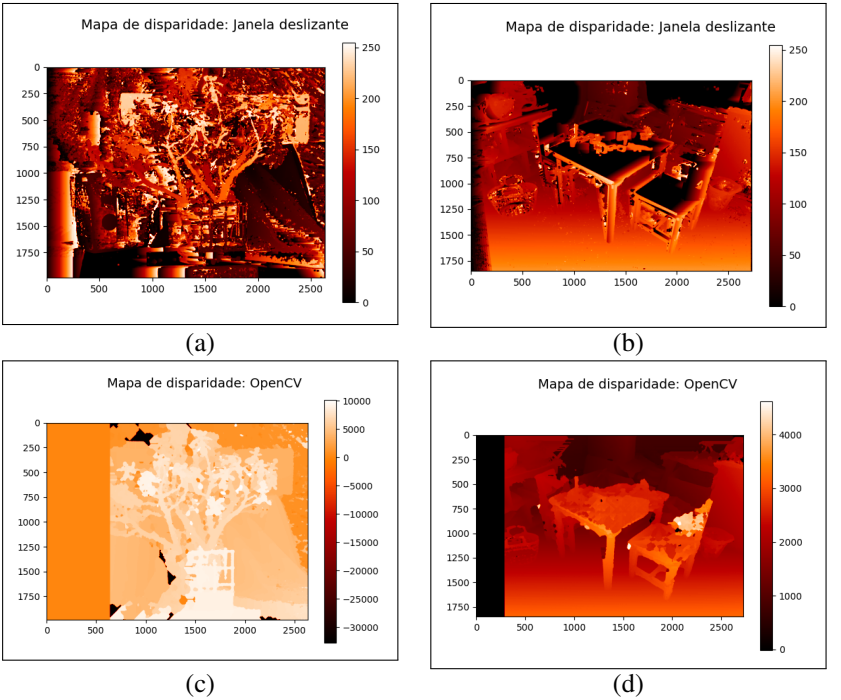


Figura 2: Mapas de disparidade dos melhores e piores resultados, onde (a) e (c) são o melhor e o pior, respectivamente, para "Jadeplant" e (b) e (d) para "Playtable".

Na análise sobre os mapas de profundidade, os erros nos cálculos dos mapas de dispa-

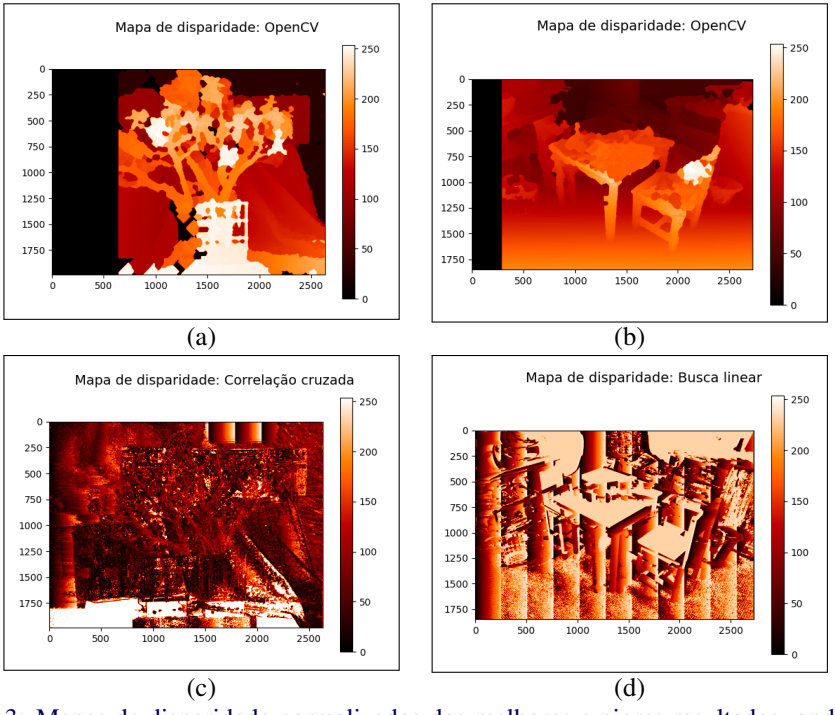


Figura 3: Mapas de disparidade normalizados dos melhores e piores resultados, onde (a) e (c) são o melhor e o pior, respectivamente, para "Jadeplant" e (b) e (d) para "Playtable".

ridade tornam-se mais evidentes, já que a Figura 4 demonstram-os como imagens de valor único e de baixíssima profundidade, ou seja, simplesmente errados. Com base nos dados das câmeras, a variação de profundidade deveria estar entre $4,63$ ($d_{xy} = 600$) e $102,99$ ($d_{xy} = 27$) metros para a "Jadeplant" e entre $1,67$ ($d_{xy} = 270$) e $16,65$ ($d_{xy} = 27$) para a "Playtable".

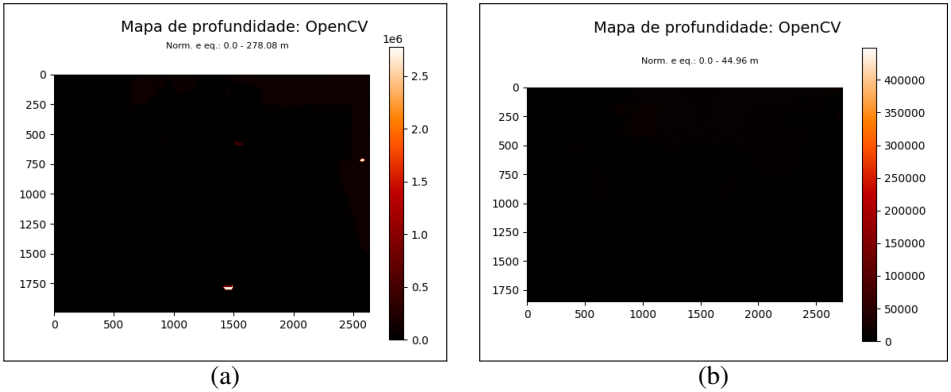
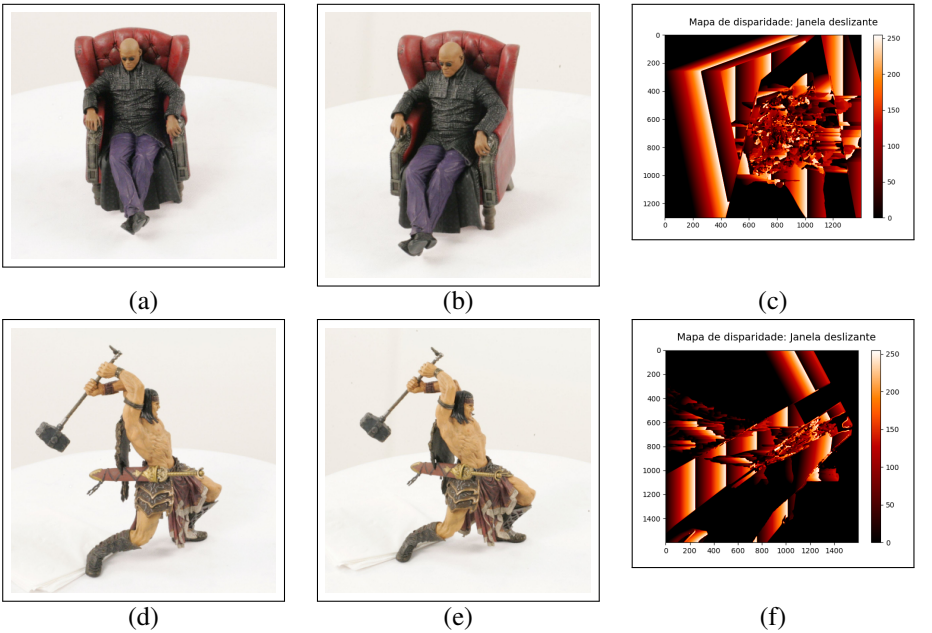


Figura 4: Mapas de profundidade para os mapas de disparidade com as melhores taxas de acertos, onde (a) é o melhor para a "Jadeplant" e (b) para "Playtable".

4.2 Requisito 2

Com base nos resultados do Requisito 4.1 e considerando que: (a) não há uma base para comparação e (b) as imagens não estão necessariamente retificadas; o algoritmo *Janela Deslizante* com blocos de tamanho 15 foi selecionado para este requisito. Dessa forma, a exploração aconteceu sobre o tamanho da janela de busca $M \in [100, 800]$, por também ser desconhecida para esta base.

Com isso, os mapas de disparidade não apresentaram bons resultados visuais, uma vez que não é possível vislumbrar uma similaridade com as imagens originais, como demonstrado na Figura 5. Com o intuito de melhorar o desempenho, as imagens correspondes tiveram o mesmo pré-processamento das imagens do primeiro requisito, sendo realizada uma busca pelos ponto-chaves e suas correspondências entre as imagens, com a finalidade de identificar as linhas epipolares, seguida pela retificação das imagens considerando as suas próprias homografias.



(a) (b) (c)

(d) (e) (f)

Figura 5: Mapas de disparidade para o Requisito 4.2, onde (a) e (d) são as imagens base, (b) e (e) são as imagens correspondentes e (c) e (f) os mapas de disparidade com uma busca de até 400 *pixels* calculados para o "Morpheus" e "Warrior", respectivamente. Fonte das imagens (a), (b), (d) e (e) para a base de imagens desativada supracitada.

4.3 Requisito 3

Devido à falta de clareza e distinção dos resultados do Requisito 4.2 não houve tentativa de desenvolvimento para este requisito.

5 Conclusão

Os algoritmos implementados se provaram desafiadores seja pela dificuldade da programação seja por fatores intrínsecos. Para o Requisito 4.1, o algoritmo *OpenCV* apresentou os melhores resultados visuais, por ter conseguido conectar os blocos relacionados, mas os valores de disparidade destoam do *ground truth*, o que justifica o seu baixo desempenho. Entretanto, ao normalizar os mapas de disparidade a taxa de acertos aumentou, justamente pela compactação das escalas distintas numa comum o que aumenta o número de boas "colisões" de disparidade, esta pode ser uma boa técnica a ser utilizada para quando os limites de disparidade são previamente conhecidos, como no caso em questão, pois poderiam ser escalonados à faixa conhecida.

Ao se deparar com o Requisito 4.2, entretanto, houve a constatação da fragilidade dos algoritmos desenvolvidos, sem considerar as influências do pré-processamento das imagens, devido a falta de solução para o problema apresentado.

Referências

[1] Middleburry College. 2014 stereo datasets with ground truth, 2015. URL https://vision.middlebury.edu/stereo/data/scenes2014/.

[2] R Fergus, D. Forsyth. Computer Vision - Lecture 1 (Part 2), unknown. Slides usados na exposição teórica, baseado em produções dos supracitados.

[3] Hongbo Li, David Hestenes, and Alyn Rockwood. *Generalized Homogeneous Coordinates for Computational Geometry*, pages 27–59. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001. ISBN 978-3-662-04621-0. doi: 10.1007/978-3-662-04621-0_2. URL https://doi.org/10.1007/978-3-662-04621-0_2.

[4] Autoria própria. Pvc: Projeto 1 - visão estéreo, 2020. URL https://github.com/rodrigofegui/pvc/tree/main/Projetos/Projeto_1.

[5] Richard Szeliski. *Computer vision: Algorithms and Applications*. Springer, London, New York, 2nd edition edition, 2011. URL http://dx.doi.org/10.1007/978-1-84882-935-0.