3DRT – Sistema Embarcado para Reconstrução 3D em Tempo Real

Luis E. O. Fernandez, Elizabeth V. C. Avila, Luiz M. G. Gonçalves
Universidade Federal do Rio Grande do Norte
Avenida Senador Salgado Filho, 3000, CEP 59.078-970, Natal, RN, Brasil

***Resumo*—We present current efforts towards the development of an embedded system for 3D reconstruction of a scene in real time. Algorithms for system calibration, rectification, and 3D reconstruction of a point cloud using the *NVIDIA Jetson TK1* based on a stereo disparity map given by the *Stereolabs ZED* 3D camera are currently developed. They are implemented using the stereo camera *SDK*, in *C++*, and the *FreeGLUT* library. We have built a tool for visualization of the reconstructed data in a monitor and, as a partial contribution of this work, we have done an analysis to measure time processing for each step of the whole process to certify that the real-time requirement is met. Partial results indicate feasibility of a more complex, efficient system (with precision and real-time operation) thus allowing its use in robotics vision applications.**

***Resumo*—Apresentamos trabalho correntes para o desenvolvimento de um sistema embarcado para reconstrução 3D de uma cena em tempo real. Foram desenvolvidos algoritmos para calibração, retificação, e reconstrução de uma nuvem de pontos usando um mapa de disparidade estéreo dado pela câmera *Stereolabs ZED*, implementados com a placa *NVIDIA Jetson TK1*. As implementações usam o *SDK* da câmera estéreo, em *C++*, e a biblioteca *FreeGLUT*. Construímos também uma ferramenta para visualização dos dados reconstruídos em um monitor e, como contribuição parcial do trabalho, introduzimos uma análise para medir os tempos de processamento de cada etapa do processo para certificar o atendimento do requisito de tempo real. Resultados parciais indicam a factibilidade de um sistema mais complexo eficiente (com boa precisão e operando em tempo real), propiciando seu uso em aplicações de visão robótica.**

***Index Terms*—visão em tempo real; reconstrução 3D; disparidade estéreo; nuvens de pontos.**

I. Introdução

As técnicas de reconstrução 3D referem-se à criação de modelos tridimensionais de um ambiente usando um conjunto de imagens e técnicas de visão artificial [1, 2]. Nos últimos anos, tem ocorrido uma demanda maior para o uso de informação 3D devido ao incremento de aplicações na linha de visão artificial, tais como vigilância, navegação autônoma, reconhecimento e identificação de pessoas e objetos, entre outras. Em aplicações típicas da área de robótica, por exemplo, misturando visão artificial e controle robótico, a construção de mapas 3D é uma tarefa essencial para facilitar a navegação de robôs, para que sejam completamente autônomos.

Há no mercado ferramentas de hardware e de software desenvolvidas com o intuito de se conseguir a reconstrução 3D que seja de boa qualidade, de forma eficiente, tais como sistemas baseados no Kinect (www.xbox.com) e na Bumblebee (www.ptgrey.com), usando suas interfaces de programação específicas (SDK) ou usando bibliotecas como a PCL [3]. Essas ferramentas citadas usam placas de captura dedicadas (Bumblebee) ou computadores com recursos específicos, normalmente as placas de processamento de vídeo dedicadas (GPU). Entretanto, os requisitos de baixo consumo e de tempo real de algumas aplicações citadas acima exigem o uso de processamento embarcado, que, atualmente, podem ser encontrados em sistemas menores, portáteis, e de menor consumo energético. Neste trabalho, para a captura de imagens, usamos a câmera estéreo ZED da *Stereolabs*, uma placa *NVIDIA Jetson TK1* para o processamento gráfico e um monitor comum para a visualização. Os algoritmos foram implementado usando o *SDK* da própria câmera, em linguagem *C++*, e *FreeGLUT*.

No artigo, apresentamos os resultados parciais dos esforços correntes no sentido de desenvolver o sistema embarcado para a captura de imagens estéreo, o seu processamento e a reconstrução 3D em tempo real. Este trabalho é a primeira etapa de um projeto maior que busca desenvolver um sistema para construção de mapas 3D em tempo real, online.

II. Trabalhos relacionados

O principal desafio abordado neste trabalho é permitir que a reconstrução 3D seja factível como um mínimo de esforço computacional, ou seja, atendendo aos requisitos de tempo real e de precisão inerentes aos sistemas robóticos, usando o mínimo de hardware possível (no sentido de volume e peso), e online. Encontramos alguns trabalhos na literatura sobre o mesmo tema. No trabalho de Gomez [4], é realizada uma análise da reconstrução 3D em tempo real usando imagens estéreo e GPU, com simulação dos requisitos de tempo real sendo determinado pelo uso de vídeos. Heng [5] propõe a criação de fotos realistas em tempo real, e em 3D, usando micro-aeronaves. Em outro trabalho encontrado na literatura [6], é feita a estimação, em tempo real, da trajetória de um robô e a reconstrução usando uma câmera 3D.

No presente trabalho, introduzimos um sistema embarcado para reconstrução 3D em tempo real, com operação em modo contínuo, isto é, que fornece um mapa 3D do ambiente a cada intervalo de tempo determinado pela restrição de temo real (em 33 ms). Comparado com outros trabalhos relacionados acima citados, o sistema aqui desenvolvido tem algumas vantagens. Uma delas é que a visualização da reconstrução 3D ocorre com um atraso mínimo, imperceptível à visão humana. Além disso, a distância para geração da reconstrução 3D é ate vinte metros, tendo mais alcance que com o uso do Kinect, por exemplo.

III. O Sistema Estéreo 3D

O sistema embarcado desenvolvido é um sistema de tempo real do tipo não crítico (ou *soft*), uma vez que tem o tempo como parâmetro fundamental mas uma falha pode ser aceitável. O não cumprimento de uma tarefa em um determinado intervalo de tempo não provoca danos irreversíveis à plataforma.

O hardware do sistema é composto por uma placa *NVIDIA Jetson TK1* [7]. Nesta plataforma de processamento gráfico, são implementadas todas as etapas para a reconstrução 3D, tendo ela processamento paralelo, acelerado por *GPU*, e sendo de baixo consumo energético se comparada à placas tradicionais. Suas principais características são: GPU NVIDIA Kepler GPU com 192 CUDA Cores; CPU NVIDIA 4-Plus-1^{TM} Quad-Core ARM® Cortex^{TM}-A15; memória com capacidade de armazenamento de 16 GB e memória RAM com 2 GB; 1 Porta HDMI e 1 USB 3.0 tipo A. Exige fonte de energia de 12 VDC, consumindo 5 Watts.

Para captura de dados, optamos pela câmera, *ZED-Stereolabs* [8], que, basicamente, determina a profundidade usando reconstrução estéreo. Esta câmera produz um vídeo de alta resolução *side-by-side* em *USB 3.0* que contém dois fluxos de vídeo, esquerda e direita, sincronizados, e gera um mapa de disparidade do ambiente em tempo real, que é convertido em um mapa de profundidade utilizando a unidade de processamento gráfico (GPU) da placa *Jetson TK1*. As principais características da ZED são: separação entre as lentes (linha de base) de 120 mm; campo de visão de 110°; resolução (com GPU) de 720p(2560x720), 60 frames/seg; faixa de profundidade de 1 - 20 m; e conector USB 3.0. O consumo de energia da ZED é de (5V) 380mA. A implementação em software realizada neste trabalho segue as etapas mostradas na Figura 1, descritas a seguir.

A. Calibração e retificação da câmera estéreo

Nesta etapa, calcula-se as relações geométricas entre as duas câmeras e a cena (parâmetros intrínsecos e extrínsecos). Esses parâmetros são usados posteriormente para para eliminar as distorções radial e geométricas que ocorrem na captura de imagens da câmera. A câmera estéreo tem parâmetros de calibração de fábrica (Figura 2). No início do processo, a câmera é automaticamente calibrada, sendo isso realizado só uma vez.

B. Captura de imagens

Neste trabalho, duas imagens da cena são obtidas pela câmera estéreo, sincronizadas, sendo uma a imagem da esquerda e a outra, deslocada, a imagem da direita. As duas câmeras têm características similares, possuindo uma linha de base (distância entre os centros óticos de cada câmera) de 12 cm, como dito acima. Após a captura, ocorrem as etapas de calibração e retificação. O resultado das etapas de captura, calibração e retificação são imagens sem distorção e alinhadas horizontalmente (pixels correspondentes com mesma coordenada Y). Imagens resultantes da etapa de calibração são mostradas na Figura 3.

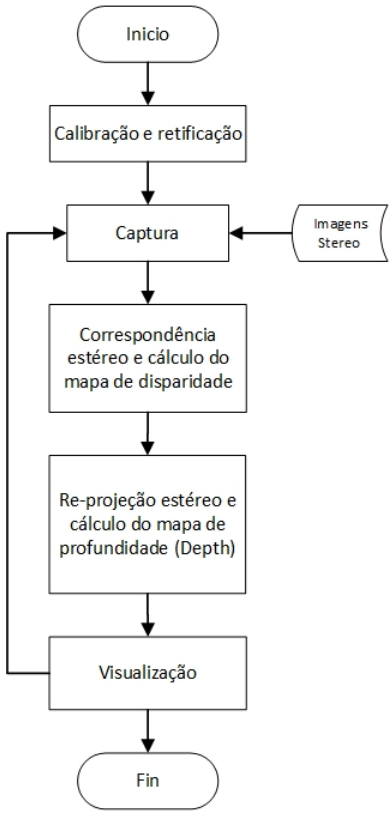

Figura 1. Diagrama de funcionamento do algoritmo para reconstrução 3D em tempo real.

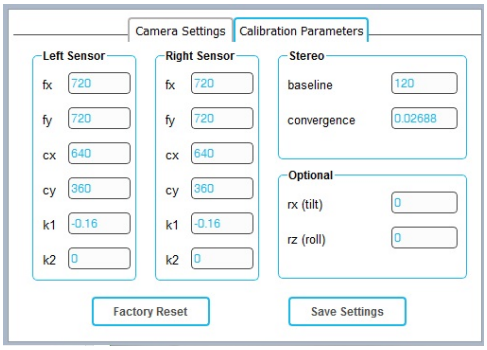

Figura 2. Parâmetros de calibração da câmera.

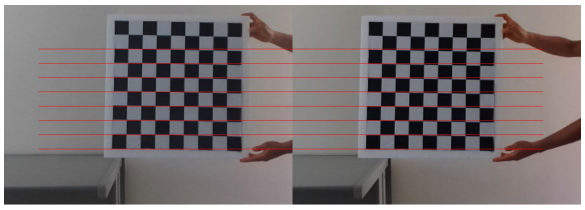

Figura 3. Resultado da calibração e retificação.

C. Correspondência Estéreo (Mapa de Disparidade)

Com as imagens retificadas, a etapa seguinte é o realce de características (*features*) e a consequente determinação de pontos homólogos entre as imagens esquerda e direita (também chamado de correspondência ou *matching*). Uma

Figura 4. Mapa de disparidade de uma imagem estéreo.

vez determinado o casamento para todos os pontos de uma imagem com relação à outra, calcula-se a diferença entre a localização (coordenadas) de um ponto em uma imagem e o seu correspondente na outra (x^l, x^r). Este cálculo também é feito para todos os pontos de uma imagem com relação à outra, sendo que o processo resulta na determinação do mapa de disparidade, como mostrado na Figura 4. Regiões mais claras estão mais próximas, com maior disparidade, com a disparidade dada simplemente por $d = x^l - x^r$.

D. Reprojeção e cálculo do mapa de profundidade

Conhecida a composição geométrica das câmeras, pode-se transformar o mapa de disparidade em distâncias usando **triangulação***(triângulos semelhantes)*. Este processo é chamado reprojeção e a saída é um mapa de profundidade. O algoritmo para o cálculo da profundidade a partir da disparidade baseia-se no esquema ilustrado na Figura 5. Na Figura, ***P*** é um ponto objeto (no mundo físico), ***(O_l, O_r)*** são os centros de projeção esquerdo e direito, ***T*** é a distância entre os centros de projeção, ***Z*** é a profundidade, ***f*** é distância focal da câmera, c^{left}, c^{right} são os pontos principais dos planos imagem; e (x^l, x^r) são as posições horizontais dos pontos do gerador de imagens esquerdo e direito. A partir da Figura 5 pode-se verificar que a profundidade ***Z*** é inversamente proporcional à disparidade ***d***, conforme a Equação 1 [9].

$$Z = \frac{fT}{x^l - x^r} \tag{1}$$

Usando as coordenadas ***(x,y)*** dos pontos de uma imagem e seu mapa de disparidade, por triangulação, pode-se obter as coordenadas 3D ***(x,y,z)*** dos pontos na cena. O resultado obtido deste processo é uma nuvem de pontos 3D. Um exemplo pode ser visto na Figura 6. Nesta figura, pode-se ver que os pontos mais próximos são de cor branca (disparidade maior), enquanto que os que estão mais longe são de cor preta (disparidade menor, ou próxima de zero).

E. Visualização

Uma vez obtida a nuvem de pontos, com a informação de coordenadas 3D e cor dos pontos, sentimos a necessário de criar um aplicativo de visualização 3D neste trabalho, uma vez que a SDK da câmera não provê esta ferramenta. Para isto, usamos o *FreeGLUT* [10] que é uma alternativa

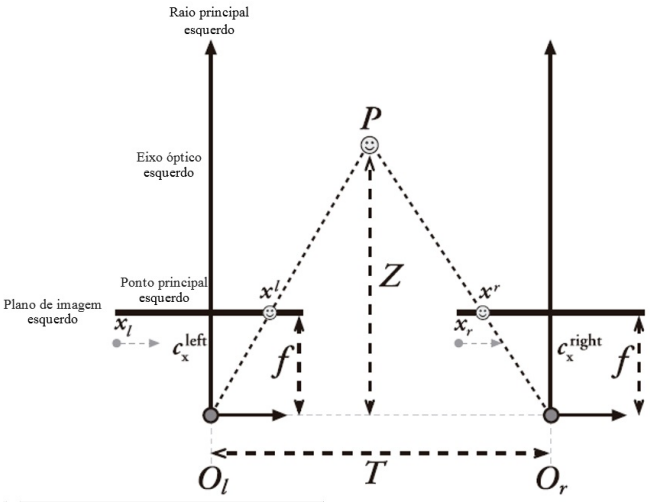

Figura 5. Relação entre disparidade e profundidade em imagens estéreo.

Figura 6. Reprojeção e cálculo do mapa de profundidade de uma imagem estéreo.

de software livre para a biblioteca *GLUT* do *OpenGL*. A biblioteca *FreeGLUT* realiza todas as tarefas específicas do sistema necessários para a criação de janelas, inicializando contextos *OpenGL* e manipulação de eventos de entrada. Um exemplo da visualização de uma reconstrução 3D obtida é mostrado na Figura 7.

IV. Análise de resultados

Visando verificar a execução em tempo real do sistema em desenvolvimento, realizamos dez testes do algoritmo implementado. Para cada execução, são capturados 60 quadros. É medido o tempo das etapas de calibração e retificação, cálculo da disparidade, profundidade e visualização. Além disso, foram calculados os tempos totais de cada teste. Todas as medições realizadas são apresentadas na Tabela I.

É possível ver a distribuição das medições na Figura 8, onde a linha vermelha recortada representa o valor médio dos tempos de processamento. Deste grafico, também foi calculado o desvio padrão dos tempos totais de processamento para cada teste. É possível ver que existe uma diferença padrão entre cada medida de tempo de *633,25 ms*. A câmera estéreo, idealmente, processa sessenta frames por segundo, com resolução de HD-720 (2560 $\times$ 720). Esta medida faz

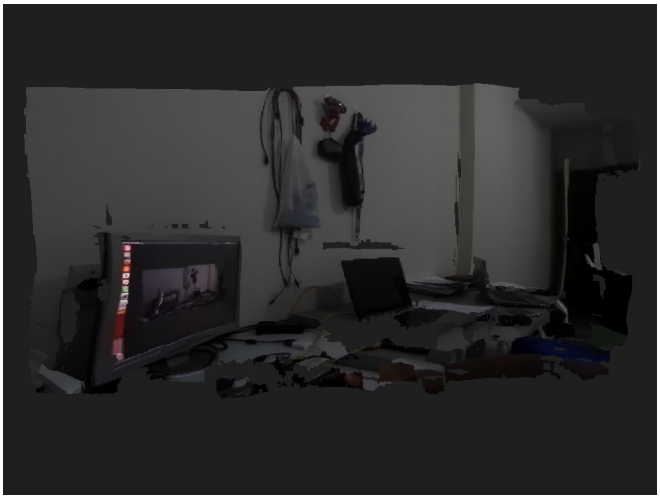

Figura 7. Visualização de uma nuvem de pontos 3D (obtida a partir de imagens estéreo) em uma janela do *FreeGLUT*.

Tabela I
TEMPOS DE PROCESSAMENTO, EM MILISSEGUNDOS, DAS ETAPAS DO ALGORITMO PARA RECONSTRUÇÃO 3D EM TEMPO REAL.

Teste	Calibração	Outras Etapas	Total
1	2414,44	2070,02	4485,46
2	2396,7	2067,26	4465,96
3	2424,06	3459,87	5886,93
4	2363,17	2590,35	4957,52
5	2400,95	2815,26	5221,21
6	2439,52	1553,88	3999,4
7	2509,07	1374,93	3891
8	2446,45	2791,65	5246,1
9	2468,36	2118,85	4596,21
10	2474,27	2207,6	4691,87

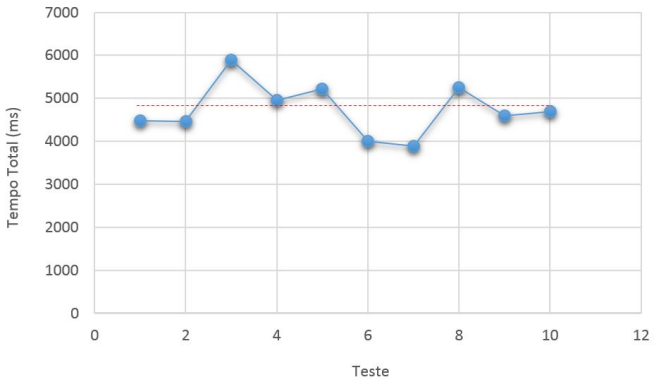

Figura 8. Grafico da distribuição dos tempos de processamento para a reconstrução 3D em tempo real.

referencia ao tempo que a câmera demora para capturar as imagens sem fazer nenhum tipo de processamento com elas. A partir dos testes realizados, foi possível notar que o sistema implementado demora em média *4772,91 ms* para processar sessenta frames (aproximadamente 13 frames por segundo). Visto de outra forma, o sistema demora *79.54 ms* desde a captura de um frame estéreo até gerar a reconstrução 3D e visualiza-la na tela.

Pode-se notar que, para aplicações de robótica, a visualização não seria necessário, acelerando o processo. Ainda, já existem trabalhos relacionados como a reconstrução 3D em tempo real, porém, o sistema aqui implementado ainda não pode ser comparado com estes outros, uma vez que as ferramentas de hardware e software aqui usadas são novas e existe pouca pesquisa sobre elas.

V. CONCLUSÕES

Neste trabalho foi introduzido um sistema embarcado funcional, que gera a reconstrução 3D de uma cena em tempo real. Para o desenvolvimento usou-se plataformas de hardware e software já existentes (câmera ZED e placa Jetson TK). A partir da análise dos resultados, pode-se concluir que o tempo de processamento dos frames não apresenta um atraso elevado, uma vez que a reconstrução 3D pode ser visualizada de forma correta e continua na tela, o que é mais importante. Além disso, pode-se afirmar que a etapa mais lenta do algoritmo para geração da reconstrução 3D é a calibração e retificação, com um tempo médio de *2,43 seg*. Porém, este processo só é realizado uma vez, no início do processo.

Uma característica do sistema implementado é o alcance maior, de até vinte metros, em comparação a outros sistemas, por exemplo os que usam *kinect*, e outros métodos de captura de imagens. Este trabalho forneceu uma ideia geral do processo e pudemos computar o tempo gasto para gerar a reconstrução 3D de uma cena. As analises ralizadas serão de muita utilidade para os trabalhos futuros que buscam desenvolver um sistema aéreo para construção de mapas 3D em tempo real usando o mesmo arcabouço.

REFERÊNCIAS

[1] E. Trucco and A. Verri, *Introductory Techniques for 3-D Computer Vision*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1998.

[2] R. Szelisky, *Algorithms and Applications*. London, UK: Springer-Verlag London, 2011.

[3] P. C. Lybrary. (2014) Pcl tutorial at ias 2014. [Online]. Available: https://pointclouds.org/media/ias2014.html

[4] H. D. Gomez-Balderas J.E., "A 3d reconstruction from real-time stereoscopic images using gpu," 2016.

[5] F. F. Lionel Heng, Gim Hee Lee and M. Pollefeys, "Real-time photo-realistic 3d mapping for micro aerial vehicles," pp. 4012–4019, Sep. 2011.

[6] T. N. Kazunori Ohno and S. Tadokoro, "Real-time robot trajectory estimation and 3d map construction using 3d camera," pp. 5279–5285, Oct. 2006.

[7] N. Corporation. (2016) Embedded Systems modules & dev kits. [Online]. Available: http://www.nvidia.com/object/embedded-systems-dev-kits-modules.htm/

[8] Stereolabs. (2016) Product tech specs. [Online]. Available: https://www.stereolabs.com/zed/specs/

[9] G. Bradski and A. Kaebler, *Learning OpenCV-Computer Visipn with the OpenCV Library*, 1st ed., O'Reilly, Ed. United States of America: O'Reilly Media, Inc., 2008.

[10] FreeGLUT. (2016) Freeglut. [Online]. Available: http://freeglut.sourceforge.net/