DETECÇÃO DE OBSTÁCULOS A PARTIR DO KINECT NO PLANEJAMENTO DE TRAJETÓRIAS DO RAPIDLY-EXPLORING RANDOM TREE STAR (RRT*)

Tarcisio Santos Santana, Gildeberto de Souza Cardoso, José Valentim dos Santos Filho
Universidade Federal do Recôncavo da Bahia, Centro de Ciências Exatas e Tecnológicas, Cruz das Almas – BA.
tarcisio18sant@gmail.com, gildeberto@ufrb.edu.br, valentim@ufrb.edu.br

***Resumo* - Este trabalho tem como um dos grandes enfoques a realização e avaliação de um planejamento de trajetórias para robôs não-holonômicos por meio das informações de sensores de profundidade em uma visão "bird's eyes", viabilizada pelo uso do Kinect, para importar informações cruciais de um determinado ambiente em auxilio à navegação utilizada na locomoção de um robô neste local. Para isso se utiliza da facilidade encontrada na utilização do software Matlab e da eficiência encontrada no algoritmo Rapidly-exploring Random Tree Star (RRT*), que busca aperfeiçoar o plano de percurso de maneira a evitar colisões, além disto, esse algoritmo é comumente utilizado para planejar trajetórias para sistemas não-holonômicos.**

***Palavras-chave* – Robôs não-holonômicos, planejamento de trajetórias, RRT*, Kinect.**

DETECTION OF OBSTACLES FROM KINECT IN RAPIDLY-EXPLORING RANDOM TREE STAR'S (RRT*) TRAJECTORY PLANNING

***Abstract* - This work has as one of its major approaches to implementation and evaluation of a trajectory planning for nonholonomics robot through the depth of sensor information on a "bird's eyes" vision, made possible by Kinect to import crucial information from a particular environment aid in navigation used in locomotion of a robot at this location. It is used to the ease of use found in Matlab and efficiency found in Rapidly-exploring Random Tree Star (RRT*) algorithm which seeks to optimize the route plan in order to avoid collisions moreover this algorithm is generally used for trajectory planning for nonholonimc systems.**

***Keywords* - Nonholonomics robot, Trajectory planning, RRT*, Kinect.**

I. INTRODUÇÃO

A robótica móvel está em grande evolução devido as crescentes descobertas em relação ao número de aplicações destes em campos como entretenimento, pequenas tarefas domésticas, indústrias e principalmente em instalações militares, pois, desde a Segunda Guerra Mundial as potências militares têm investido e requisitado muitos avanços tecnológicos para sua própria evolução.

Seguindo uma regra geral da evolução tecnológica a robótica móvel tem priorizado cada vez mais a autonomia dos robôs móveis em relação ao ambiente, exigindo desta forma uma melhor abrangência da programação e eficiência dos sistemas de navegação.

Esses sistemas são amplamente conhecidos como planejamento de trajetórias e visa capacitar o robô para atuar em algumas áreas, conhecidas ou desconhecidas, com a finalidade de alcançar objetivos com uma mínima intervenção humana.

No entanto, um dos primeiros passos geralmente tomados neste campo é o reconhecimento de área através do processamento de imagens devido à grande quantidade de dados que este método é capaz de transferir ao robô em relação a sua localização em uma área previamente analisada.

Atualmente é comum se utilizar no processamento de imagens a visão adquirida pelo Microsoft Kinect, pois este dispositivo consta com um sensor de profundidade e uma câmara RGB e isso o torna capaz de associar a detecção de bordas comum com uma análise da distância das bordas em relação a sua câmara RGB.

Com esta motivação se iniciou este trabalho que consiste em aplicar o conhecimento do processamento de imagens no campo da robótica com a finalidade de se adquirir dados suficientes para identificação de obstáculos através de suas respectivas bordas, e então, através do ambiente MATLAB, utilizar estes dados na realização do planejamento de trajetória através do RRT*.

Tendo em vista esse objetivo, este trabalho consiste em verificar a utilidade de uma visão "bird's eye view", neste caso do Microsoft Kinect e o seu sensor de profundidade, em auxiliar o navegador gerado pela RRT*, além de certa maneira avaliar este algoritmo ao decorrer do processo no intuito de examinar sua utilidade na evolução do trabalho decorrente da robótica móvel e suas futuras aplicaç.

Este artigo está organizado da seguinte forma: a Seção 2 apresenta um embasamento teórico sobre o Microsoft Kinect e sobre o tratamento de suas informações de distância, a Seção 3 teoriza sobre planejamento de trajetórias para robôs móveis, enquanto a Seção 4 apresenta os resultados experimentais e, por fim, a Seção 5 descreve as conclusões.

XIV CEEL - ISSN 2178-8308
03 a 07 de Outubro de 2016
Universidade Federal de Uberlândia - UFU
Uberlândia - Minas Gerais - Brasil

II. MICROSOFT KINECT

O Microsoft Kinect é um dispositivo de controle baseado em sensores de movimentos e comandos de voz criado pela Microsoft em parceria com a *Primesense* e moldado para o console Xbox 360, o dispositivo é constituído por uma câmera RGB, um emissor e um sensor IRs (Infravermelho), sendo todos estes com resolução de 640x480, além de um conjunto de microfones para a captura de som.

Com a popularização do Kinect a Microsoft decidiu lançar o Kinect for Windows, especialmente para pesquisas e desenvolvedores, que segundo Luis Guilherme Pegas (2014) focou-se na utilização deste dispositivo para utilização em computadores aumentando a resolução da câmera RGB para até 1920x1080 pixels, além de novas funções e conexão USB.

Com a criação do Kinect for Windows a Microsoft trouxe a tona drivers oficiais do Windows e uma framework denominada Software Development Kit (SDK), a qual juntamente com o sensor pode fornecer aos desenvolvedores com a fundação para criar e programar aplicações interativas que respondem a movimentos, gestos e comandos de voz das pessoas, resultando em interações de computadores naturais. (MICROSOFT KINECT FOR WINDOWS, 2015).

Uma grande vantagem do Kinect é a obtenção da profundidade, podendo ser adquirida ao mesmo tempo em que o áudio e a imagem colorida são obtidos, o que nos retorna as informações de distância de acordo ao referencial do sensor do Kinect, porém essa distância não vem em escalas convencionais de forma que se tem que trata-las para utilizar esses dados.

De acordo com F.S. Fikke & B.K. Gardiner (2013) o emissor IR através de uma de uma fonte de luz IR ilumina uma transparência com um padrão fixo de pontos, projetando um padrão de manchas para o objeto de interesse, com isso a câmara IR captura o padrão transformado e gera um mapa tridimensional, porém se gera um desvio entre o que se foi capturado e o padrão, o que pode ser considerado uma disparidade.

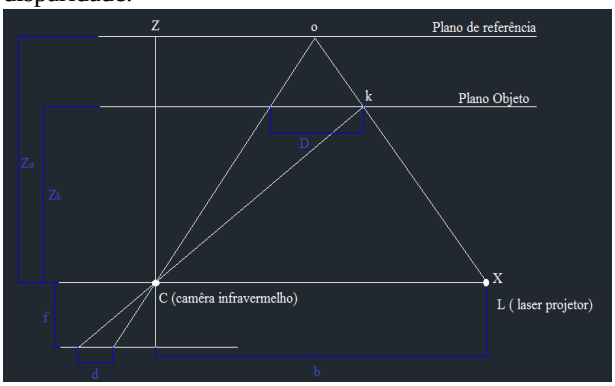

Fig. 1. Relação entre a profundidade e disparidade.

Além disso, observamos por Khoshelham, K., & Elberink, S. O. (2012) que a Figura 2 demonstra a relação de um objeto no ponto K para o sensor em relação a um plano de referência e a disparidade d. As coordenadas 3D tem em sua origem a vista da câmara IR, o eixo Z sendo ortogonal ao plano da imagem para o objeto, o eixo X perpendicular ao Z e, além disso, um eixo Y ortogonal a X e a Z formando-se um sistema de coordenadas.

Ao assumir um objeto no plano de referência a uma distância Zo em relação ao sensor e uma mancha sobre o objeto é detectada, e ocorrer algum deslocamento do objeto a mancha vai ser deslocada na direção X, isto pode ser medido como disparidade d em pixels correspondente a um ponto k no espaço objeto e por semelhança de triângulo obtemos:

$$\frac{D}{b} = \frac{Zo - Zk}{Zo} \quad (1)$$

E

$$\frac{d}{f} = \frac{D}{Zk} \quad (2)$$

Nota-se que Zk é a distância do ponto k no espaço objeto, b é a distância entre o transmissor e o receptor IR (em metros), f é a distância focal de a câmara IR (em pixels) e D é o deslocamento do ponto k no espaço objeto, enquanto d é a disparidade observada no espaço imagem.

III. PLANEJAMENTO DE TRAJETÓRIAS

Em robótica móvel é necessária à utilização de um sistema de navegação que guia um robô em uma área analisada por um período de tempo em um caminho com viável com um ponto inicial e final que devem ser percorridos pelo robô.

Sendo essa, segundo Sônia Cristina B. de Souza (2008) a função primordial de execução, gerando-se assim pequenos obstáculos a serem resolvidos pela navegação como:

-A localização do robô na área;
-Planejamento de um caminho admissível;
-Geração e execução de um percurso;
-Restrições cinemáticas dos robôs móveis.

Para uma melhor eficiência na resolução destes obstáculos, esta é basicamente dividida em três etapas:

- Definição do espaço de configuração, ou seja, posição, orientação e ângulos de articulação dos robôs em seus diversos estados;
- Aplicação de técnicas sobre este espaço de configurações com o intuito de determinação a validade dos estados para a busca;
- Busca no espaço de configuração que nada mais é que o planejamento de caminhos que consiste na aplicação de algum algoritmo para alcançar a posição e orientação desejada.

Várias técnicas são utilizadas para resolver tais problemas, muitas sem sucesso pelas suas restrições, e tais técnicas segundo Guilherme de L. Ottoni (2000) são baseado em três métodos gerais: roadmap, decomposição em células e campo potencial.

A. Rapidly-exploring Random Tree Star (RRT)*

O RRT* é um algoritmo principalmente utilizado para planejamento de trajetória de robôs móveis e é uma extensão do RRT, o qual segundo Karaman, Sertac et al. (2011) é um algoritmo baseado em amostragem e portanto probabilisticamente completo, sendo assim tendem a encontrar uma solução de acordo ao aumento do número de amostras, além de serem capazes de encontrar um plano de movimento viável de maneira rápida.

Apesar da praticidade do RRT, que baseia seu caminho através de uma árvore probabilística, se percebeu a necessidade de aperfeiçoar este processo através de extensões.

Uma dessas é justamente o RRT*, que tem como principais vantagens segundo Karaman and Frazolli (2011) converge para uma solução assintoticamente ótima. Tornando assim a possibilidade de uma trajetória mais cometida em relação à da RRT para robótica móvel.

Todos esses pontos tornam o RRT* uma ferramenta muito utilizada em diversas áreas até mesmo fora da robótica, devido ao seu processamento e suas atualizações que visam um planejamento de movimento mais prático no quesito de encontrar uma solução viável de maneira rápida e eficiente.

A RRT* garante uma otimização assintótica e pode-se utilizar em sistemas não-holonômicos, além disto, a dificuldade dos planejadores de trajetórias A*, campos potencias e D* para dinâmicas complexas e restrições diferenciais dos veículos fez com que convergisse para escolha da RRT*.

O algoritmo da RRT* é demonstrado abaixo (Nasir et al. 2013):

Algorithm 1 RRT*

```
1: função ALGORITMO(τ = (V;E))←RRT*(zinit)
2:   τ ← InitializeTree()
3:   τ ← InsertNode(φ, zini, τ )
4:   para i ←1 até n faça
5:       zrand ← Sample(i)
6:       znearest ← Nearest(τ, zrand)
7:       (xnew, unew, Tnew) ← Steer(znearest, zrand)
8:       se ObstacleFree(xnew) então
9:           ZNear ← Near(τ, znew, |V|)
10:          zmin ← ChooseParent(Znear, znearest, znew, xnew)
11:          τ ← InsertNode(zmin, znew, τ)
12:          τ ← Rewire(τ, Znear, zmin, znew)
13:      fim se
14:    fim para
15:    Retorna τ
16: fim função
```

O qual considera que X é a configuração no espaço no qual X*obs* é a região na qual contém obstáculos, X*free* = X \ X*obs* é a região sem obstáculos e X*goal* é a região chegada. O RRT* tem como objetivo calcular uma entrada $u : [0:T] \in U$ que leva a um caminho $x(t) \in X_{free}$, o qual começa em $x(0) = x_{init}$ e vai para $x(T) \in X_{goal}$ seguindo as restrições do sistema. E onde:

• *Sampling*: é uma amostra aleatória *zrand* ∈ X*free*.

• *Distance*: a função que retorna o custo do caminho entre dois estados, assumindo que a região entre eles pertençam a X*free*. O custo é determinado pela distância Euclidiana.

• *Nearest Neighbor*: A função *Nearest*(τ, *zrand*), através do calculo da distância Euclidiana, retorna o nó mais próximo de τ = (V, E) para *zrand* em termos de custo.

• *Steer*: A funcão *Steer*(*znearest*, *zrand*) resolve para uma entrada de controle u[0,T] que leva o sistema de x(0) = *znearest* para *x*(T) = *zrand* pela trajetória x: [0,T] → X.

• *Collision Check*: A função ObstacleFree(*x*) verifica se o caminho x: [0,T] → X encontra-se dentro de X*free*.

• *Insert Node*: A função *InsertNode*(*zparent*, *znew*, τ) adiciona um nó *znew* a V na árvore τ = (V, E) e conecta ele a um *zparent* já existente como seu pai, adicionando por fim esta primitiva para E. Desta forma um custo é atribuído para *znew*, o qual é a soma do custo do pai e o custo da Euclidiana gerado pela distância entre *znew* e *zparent*.

• *Near-by Vertices*: Dado um estado $z \in X$, a árvore τ = (V, E) e um número *n*, a função *Znearby* = *Near*(τ, *z*, *n*) retorna o vértice em V que são próximos de *z*. Mais precisamente, defina $\text{Reach}(z, l) = z' \in X | \text{Dist}(z, z') \leq l \vee \text{Dist}(z', z) \leq l$, e escolha *l*(*n*) tal que Alcance(z, l(n)) contem uma bola de volume $\gamma((\log n)/n)$, onde γ é um número fixo. [Karaman and Frazzoli 2010]

• *ChooseParent*: Após a RRT considerar todos os nós presentes na vizinhança de *znew*, o ChooseParent avalia o custo total como a combinação dos custos associados para alcançar o nó pai mais o custo da trajetória para *znew*, o nó que gera o menor custo torna-se o novo nó pai e é adicionada a árvore. [Karaman et al. 2011]

• *ReWire*: Verifica cada nó *znear* nas imediações do *znew* para ver se alcançar *znear* via *znew* acarretará menores custos do que seguir através do pai atual. Se esta conexão reduz o custo total associado ao *znear*, então o algoritmo modifica a árvore para fazer *znew* o pai de *znear*. [Karaman et al. 2011]

IV. RESULTADOS

A parte prática deste trabalho foi particularmente voltada para a análise da eficiência de uma trajetória designada pelo RRT* em uma determinada área com algumas obstruções ao seu decorrer, para isso a utilização de passos graduais foi necessária, como primeiro passo se concluiu a conexão do Microsoft Kinect com o software MATLAB e neste ultimo criar portas de entrada de vídeo para a câmara RGB e para o sensor de profundidade, logo após foram feitas duas fotografias da área analisada, uma colorida (Figura 2) e outra de profundidade (Figura 3).

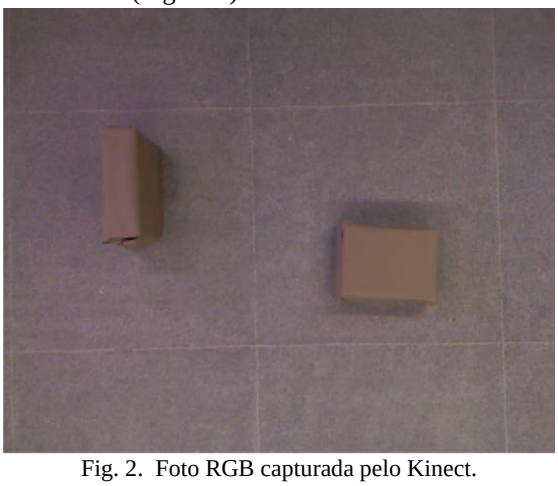

Fig. 2. Foto RGB capturada pelo Kinect.

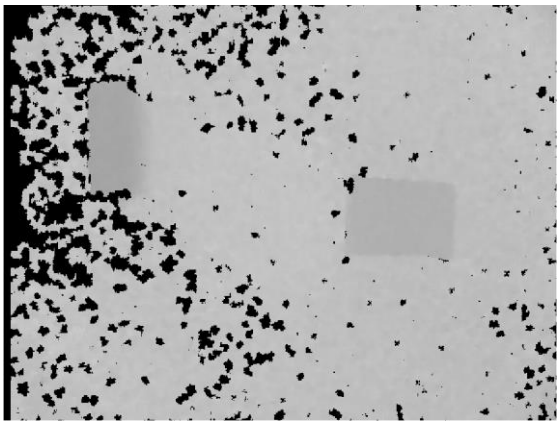

Fig. 3. Imagem do sensor de profundidade capturada pelo Kinect

Pela análise das fotografias é possível perceber que na imagem obtida pelo sensor de profundidade é possível identificar objetos através de sua distância do sensor, porém da mesma forma nessa mesma imagem se contém uma gama de bits decorridos da iluminação do local, tendo estes à possibilidade de atrapalhar na detecção de obstáculos, para um aperfeiçoamento da varredura são utilizados dois filtros básicos no ambiente MATLAB.

Um dado pela conversão da imagem de profundidade em binário e logo após efetuar a eliminação de áreas menores que 10 pixels, e o outro foca na conversão dos dados contidos na imagem de profundidade em uma matriz de distância com a função pccloud. Em seguida criar uma Matriz T do tamanho da matriz distância e definir cada elemento relevante de T de acordo a um limite de distância observado na matriz distancia do Kinect e sendo dada como a distância do sensor ao plano de fundo, neste caso foi-se utilizado3, 0 metros como distância limite devido a uma análise visual da área.

O algoritmo de filtragem de distância para os dados anteriores e o resultado podem ser vistos a seguir respectivamente no Algoritmo 2 e na Figura 4.

Algoritimo 2 Filtros

```
1: [pccloud,distance] = depthToCloud(depth);
2:    x=distance
3:    T = zeros(size(x));
4:    for ii = 1:numel(x)
5:          if  x(ii)<3.0
6:            T(ii) = x(ii);
7:          else
8:            T(ii) = NaN;
9:          end
10:  end
 11:    img=T;
```

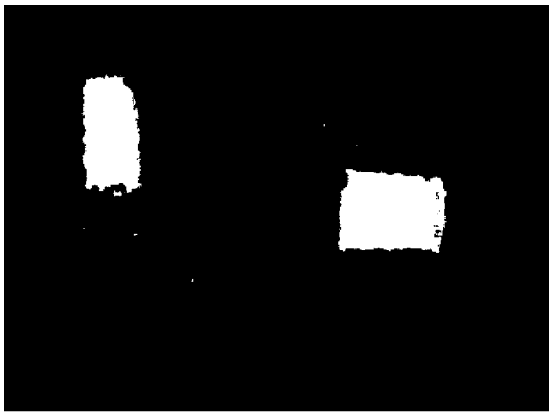

Fig. 4. Filtro baseado na distância do sensor de profundidade e na área de pixels.

Esta aplicação tornou visível o contraste entre os possíveis obstáculos e o plano de fundo e seus possíveis ruídos neste intervalo de consideração. Então neste ponto surge o último passo que simplesmente é uma aplicação de dados no algoritmo RRT*, o qual através destes dados traça um caminho, com pontos de inicio e fim pré-estabelecidos, de modo a se evitar colisões, logo ele cria um raio de segurança que circunscreve o objeto de modo a evitar que o caminho passe por essa área.

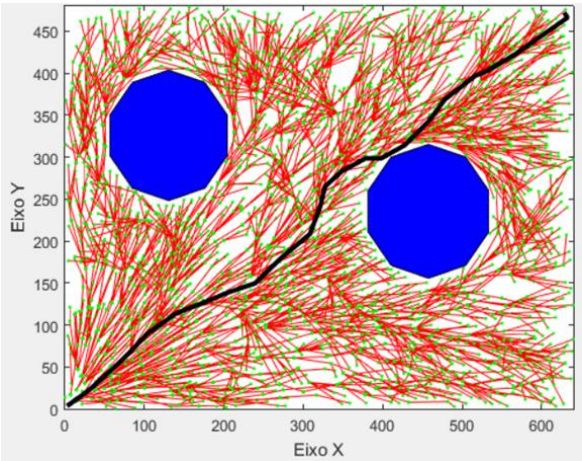

Fig. 5. Resultado do RRT* para a área analisada

V. CONCLUSÕES

Neste trabalho foi apresentada uma aplicação do reconhecimento de área ao planejamento de trajetórias de um robô móvel por via de um filtro baseado na distância de um sensor para toda área analisada.

Deste modo, utilizou-se o Microsoft Kinect devido a sua demonstração de acessibilidade em relação à sua conexão com o Software MATLAB que oferece diversos recursos voltados para o processamento de imagens, além disso, o Kinect tem como seu principal diferencial o seu sensor de profundidade que disponibilizou informações suficientes para conseguir detectar objetos em um intervalo de distância pré-determinado do sensor em toda a área analisada, de modo a eliminar ruídos do plano de fundo.

Este avanço ao ser aplicado ao algoritmo RRT* foi eficiente na criação da árvore de caminhos resultada do algoritmo para de obter um percurso viável para ser utilizada como rota para um robô móvel na área em questão.

Deste modo percebeu-se que o plano de navegação resultado do RRT* foi extremamente prático e simples no plano de navegação de um robô ao ser auxiliado por um reconhecimento de área de um Microsoft Kinect e seu sensor de profundidade, os quais permitiram um bom desempenho em sua função de auxiliar o processo de planejamento de trajetórias ao estudar a área em questão.

Como trabalho futuro, pretende-se implementar essa técnica de modo a auxiliar a navegação autônoma do robô que encontra-se em fase de construção, além da aplicação do reconhecimento de área do ponto de vista do robô para uma maior eficiência no programa de controle necessária para o deslocamento seguro do robô devido a necessidade da técnica utilizada neste artigo depender de uma análise visual da área antes de ser aplicada para definir uma distância base entre o chão e o sensor.

REFERÊNCIAS

[1]. F.S. Fikke & B.K. Gardiner (2013) "3D Image Face Recognition: Image Acquisition." 93f. Electrical Engineering. Delft University of Technology, 2013.

[2]. Karaman, S. and Frazzoli, E. (2010). Optimal kino-dynamic motion planning using incremental sampling-based methods. In 49th IEEE Conference on Decision and Control.

[3]. Karaman, S., Walter, M. R., Perez, A., Frazzoli, E., and Teller, S. (2011). Anytime motion planning using the rrt. In IEEE International Conference on Robotics and Automation (ICRA).*

[4]. Khoshelham, K.; Elberink S.O. (2012) "Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications." Faculty of Geo-Information Science and Earth Observation, University of Twente.

[5]. *MICROSOFT DOWNLOAD CENTER. Kinect for Windows SDK 1.8. [S.l.], (2015). Disponível em:<http://www.microsoft.com/enus/download/details.aspx?id=40278>. Acesso em: 27 fev. 2016.*

[6]. Nasir, J., Islam, F., Malik, U., Ayaz, Y., Hasan, O., Khan, M., and Muhammad, M. S. (2013). Rrt-smart: A rapid convergence implementation of rrt*. In International Journal of Advanced Robotic Systems, pages 299–311*

[7]. Ottoni, G. L.(2000) "Planejamento de trajetórias para robôs móveis." 82f. Curso de Engenharia de Computação. Universidade Federal do Rio Grande, 2000.

[8]. Pegas, L. G. (2014) "Rastreamento Visual para Robôs usando Microsoft Kinect." 93f. Departamento de Engenharia Elétrica. Universidade Federal de São Carlos, 2014.

[9]. Souza, S. C. B. (2008) "Planejamento de trajetória para um robô móvel com duas rodas utilizando um algoritmo A-Estrela modificado". Dissertação de mestrado, Programa de Engenharia Elétrica, UFRJ/COPPE, Rio de Janeiro, RJ.