Segmentação de Teto para a Localização de Robôs

Maurício Armani Lopes
Programa de Pós-Graduação em Ciência da Computação
Pontifícia Universidade Católica do Rio Grande do Sul
Porto Alegre, Rio Grande do Sul
Email: mauricio.armani@acad.pucrs.br

Ezequiel Malvestido Simeoni
Programa de Pós-Graduação em Ciência da Computação
Pontifícia Universidade Católica do Rio Grande do Sul
Porto Alegre, Rio Grande do Sul
Email: ezequiel.simeoni@acad.pucrs.br

***Resumo*—Este trabalho propõe uma nova abordagem para a localização de robôs móveis em um ambiente conhecido. A utilização de uma Rede Totalmente Convolucional avalia a área de teto de cada imagem extraída do robô durante sua navegação no ambiente. Com o uso da Localização de Markov (Filtro de Partículas) é possível fundir estes dados com a odometria, compará-los com a informação esperada da planta-baixa do local e, assim, inferir a posição exata do robô. Os resultados da segmentação se mostraram bastante promissores se aproximando muito da segmentação feita manualmente por seres humanos.**

***Keywords*—Segmentação Semântica, Redes Convolucionais, Robótica, Localização de Robôs.**

I. Introdução

Atualmente a indústria conta com o auxílio de robôs para diversas etapas dos seus processos de fabricação: soldagem, pintura, empacotamento e etc. Porém, em todos estes casos tratamos de robôs apenas com movimento relativo a sua própria base estática. Sempre esteve nos planos da indústria o desenvolvimento de robótica móvel. Robôs capazes de se movimentar e atuar em tarefas que outrora apenas pessoas eram capazes de executar, permitiria uma escalabilidade maior da indústria para atendimento de suas demandas. Uma vez estabelecida, a robótica móvel nos aproximará do antigo sonho de não precisamos mais exercer funções manuais. Reduzindo assim os riscos à saúde mental e física do trabalhador.

Em paralelo, o desenvolvimento da Inteligência Artificial vem cada vez mais fornecendo ferramentas para que passos largos sejam dados nessa direção. Robôs agora são capazes de extrair informação útil a partir de sensores e identificar padrões para simular o entendimento humano dos dados. Seja um humanoide capaz de carregar objetos frágeis, seja um carro capaz de desviar de obstáculos indesejáveis na pista apenas com uma câmera.

As técnicas de Aprendizado de Máquina têm se mostrado capazes de atuar nas mais diversas áreas. Em especial, as Redes Convolucionais - aqui utilizadas - são capazes de extrair semântica a partir de imagens sem prévio processamento dos dados. Estabelecendo-se como o novo estado-da-arte em diversas aplicações, tais redes podem facilmente ser aplicadas à robótica em seus inúmeros obstáculos. Dentre estes está a inerente dificuldade dos robôs móveis de se localizar. Neste trabalho abordaremos as dificuldades da localização de um robô mesmo em um ambiente conhecido e controlado.

Dentre as diversas técnicas já desenvolvidas para a localização de robôs, estão as técnicas que consistem na utilização de partículas virtuais. Estas contém informação sobre cada local do ambiente para posterior comparação com o *ground-truth*. Neste caso, uma a planta-baixa de um andar predial. Na medida que o robô se movimenta, a informação extraída da imagem é processada por uma rede neural pré-treinada com imagens do local e, com o uso de um Filtro de Monte Carlo, é fundida com a informação odométrica. Este processo permite que partículas virtuais - que possuem informação divergente da informação sendo extraída do robô em movimento - sejam eliminadas. Com o progressivo aumento da informação, o método converge para um único *cluster* de partículas determinando, assim, a localização do robô.

No presente trabalho, abordamos a utilização da visão vertical do teto para auxiliar na localização do robô. Tal abordagem consiste em extrair do teto a área total de teto sendo alcançada pelo ângulo da câmera e comparar com a informação esperada pela planta-baixa. Não sendo um problema de solução trivial, decidimos buscar no estado-da-arte da segmentação semântica uma alternativa para este problema. Com isto, foi possível validar nossa metodologia e propor futuras pesquisas.

Projetos futuros consistirão em simplificar a arquitetura da rede a fim de dar velocidade e eficiência à tarefa específica de localização do robô.

II. Trabalhos Relacionados

Aqui descrevemos brevemente trabalhos relacionados com a utilização do teto para a localização de robôs. Ambos projetos possuem similaridades com a nossa abordagem, mas diferem na forma de utilizar os dados extraídos da imagem.

Seung-Hun Kim et al. [3] propõe uma módulo de localização baseado em padrões extraídos do teto. Luzes no teto são usadas como referência para comparar com as posições previamente conhecidas. Dados provenientes das imagens do teto e de uma Unidade de Medição Inercial (IMU) são combinados e utilizados na Localização de Markov para estimar a real localização do robô.

Francisco Dias et al. [2] propõe o uso da segmentação de teto para identificar linhas dos cantos entre o teto e as paredes. A partir de destas linhas é possível fazer a localização e mapeamento simultâneo do ambiente (SLAM).

III. Metodologia

Neste capítulo abordaremos as etapas de geração do dataset e treinamento do algoritmo para a tarefa de segmentar o teto. Tal abordagem possui algumas vantagens ao que seria mais usual (vista frontal). Primeiro, o sistema não depende de escala

já que toda informação está contida no teto a uma distância fixa com relação ao robô. Segundo, entre o robô e o teto não é usual que tenham objetos impondo restrição à visão. Além disso, objetos móveis - como pessoas - exercem pouquíssima influência na visibilidade do teto.

A. Recursos Utilizados

Os recursos utilizados para a geração do dataset foram: uma câmera Raspberry Pi-V2 e uma lente Lensoul LS-WD2. Ambas acopladas a um robô Neato XV-12 e direcionadas verticalmente para cima como ilustrado na Figura 1.

Figura 1: Robô utilizado para geração do dataset.

B. Dataset

O dataset foi gerado a partir da movimentação do robô em uma parte do ambiente para extrair diferentes ângulos de visualização dos corredores. Com uma média de uma imagem por segundo, foram extraídas cerca de 2000 imagens. Algumas delas ilustradas na Figura 2.

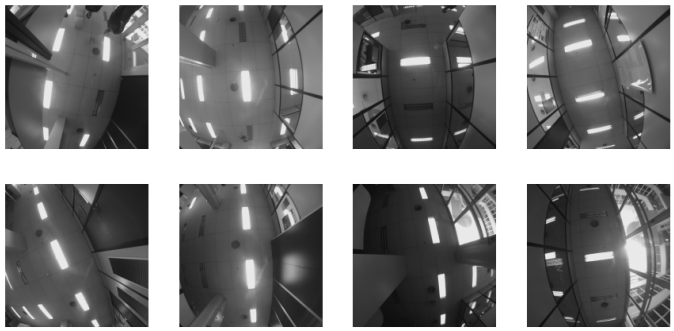

Figura 2: Imagens do dataset.

Tendo em vista que a textura do teto de todo o ambiente é a mesma e que, portanto, muitas das imagens extraídas no trajeto são similares entre si, apenas 5% das imagens foram utilizadas para o treinamento da rede. Julgamos que apenas 5% das imagens já continham toda informação necessária para o algoritmo generalizar corretamente todas as possíveis variações de formas do teto para este ambiente.

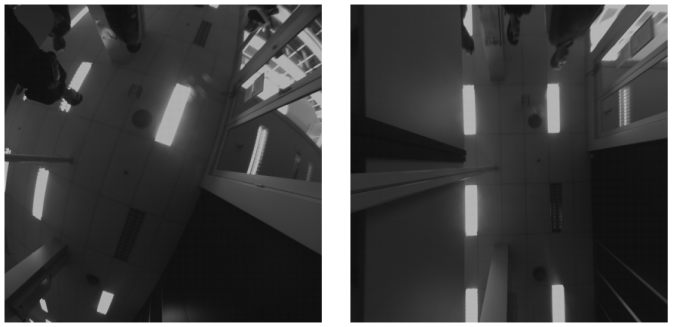

Figura 3: Imagem ajustada para anotação.

Embora, redes convolucionais sejam capazes de atuar mesmo em imagens de forma complexa, optamos por corrigir a angulação da câmera conforme Figura 3 para facilitar o processo de anotação.

C. FCN-VGG19

Para a tarefa de segmentar o teto optamos por utilizar a rede FCN-VGG19 proposta em [1]. É uma adaptação da rede VGG19 para permitir a classificação pixel-a-pixel de imagens.

A tradicional VGG-19 obteve as primeiras colocações nos desafios de localização e classificação do ImageNet ILSVRC-2014. Portanto, seus parâmetros já são capazes de distinguir entre 1000 classes distintas. Por isso, acreditamos que posteriormente será possível simplificar a arquitetura consideravelmente para o caso de classificação de teto.

A FCN-VGG19 consiste na utilização de convoluções de *stride* fracionado para gerar uma saída de dimensões espaciais maior que as de sua entrada. O objetivo final da arquitetura é gerar uma saída, não mais de uma única dimensão, mas de três dimensões. Sendo elas: largura, altura e o número de classes. Podendo assim ser interpretada como a probabilidade de cada pixel pertencer a uma das classes. Dado que nosso objetivo é apenas distinguir entre o que é teto e o que não é, o formato de saída esperado é de $256 \times 256 \times 2$.

D. Filtro de Partículas

O filtro de partículas é conhecido como uma família de algoritmos. Este, é recursivo por natureza e opera em três fases: predição, atualização, re-amostragem. Depois de cada ação, cada partícula é modificada de acordo com o modelo existente "etapa de predição", incluindo a adição de ruído aleatório a fim de simular o efeito do ruído sobre a variável de interesse. As variáveis de interesse em especifico neste caso são a pose do robô ($odometria$) e a leitura do laser ($scan$), as quais são representadas por um conjunto de N amostras (as $partculas$). Cada partícula está constituída de uma cópia da variável de interesse ($odometria$) e um peso que define a contribuição desta partícula para a estimação global da variável.

Neste experimento foram utilizadas a odometria do robô para dar os valores X, Y e $Theta$, e os valores retornados do laser para dar o peso W às partículas. Cada partícula precisa

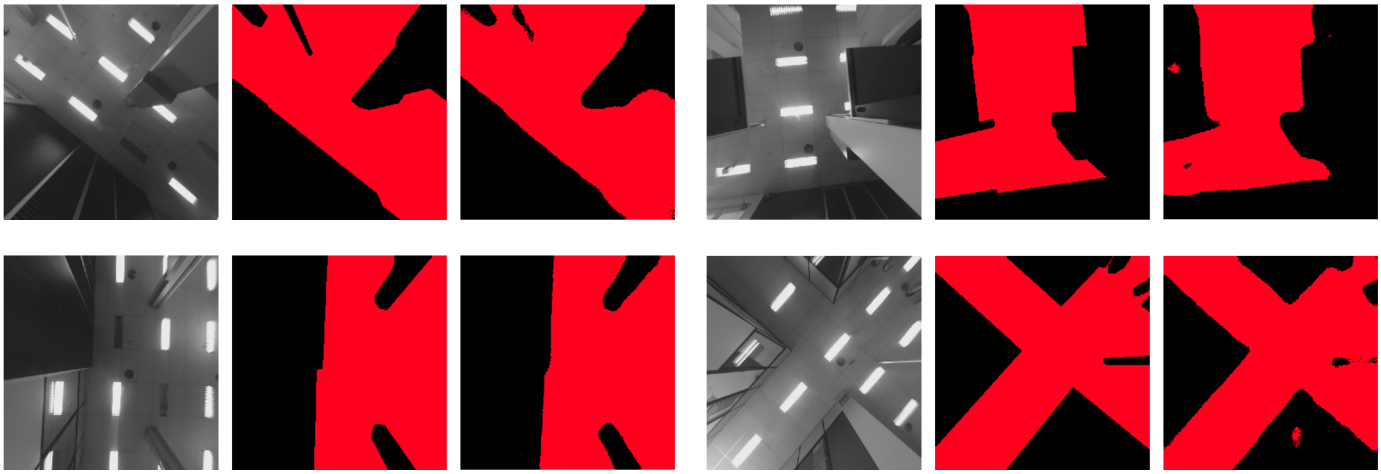

Figura 4: Imagem, *ground-truth* e predição

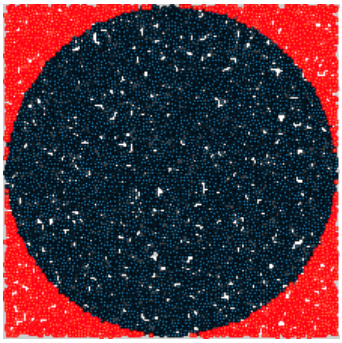

Figura 5: *Filtro de partículas simples*

de um peso ideal à probabilidade de representar a verdadeira posição do robô. Essa probabilidade raramente é computável, portanto, necessita-se apenas que seja proporcional a essa probabilidade, que é computável. Na inicialização, não há motivos para favorecer uma partícula sobre outra, então será atribuído um peso de $1/N$, por N partículas. Usamos $1/N$ para que a soma de todas as probabilidades seja igual a uma. A combinação de partículas e pesos forma a distribuição de probabilidade para o problema de localização.

Após dar o peso inicial as partículas se faz necessária a predição das mesmas, conforme a movimentação do robô ($odometria$). Nesse passo também será adicionado ruído as variáveis X e Y, já que a movimentação do robô não é perfeita. Neste experimento utilizamos a amostra abaixo:

$$dist = (u[1] * dt) + (randn(N) * std[1])$$
$$particles[:, 0] += np.cos(particles[:, 2]) * dist$$
$$particles[:, 1] += np.sin(particles[:, 2]) * dist$$

O próximo passo é fazer a atualização das partículas. Foi atribuída uma probabilidade a cada posição que chamamos de prioridade. Quando uma nova medida entrar, multiplicamos a probabilidade atual dessa posição (prioridade) pela probabilidade de que a medida correspondesse a essa localização. Para isso foi-se baseado no teorema de Bayes.

$$P(A \mid B) = \frac{P(B \mid A)\, P(A)}{P(B)} \tag{1}$$

Por ultimo, será feita a re-amostragem das partículas. A re-amostragem é o passo com maior relevância entre todos, pois é nele que será definido quais partículas devem ser mantidas, e quais devem ser excluídas. Partículas com pesos muito pequenos não descrevem significativamente a distribuição de probabilidade do robô. O algoritmo de re-amostragem descarta

Figura 6: *Amostragem Sistemática*

partículas com probabilidade muito baixa e as substitui por novas partículas com maior probabilidade. Isso ocorre através da duplicidade das partículas com probabilidade relativamente alta. As partículas duplicadas são ligeiramente dispersas pelo ruído adicionado na etapa de predição. Isso resulta em um conjunto de pontos em que uma grande maioria das partículas representam com precisão, a distribuição de probabilidade. Existem vários algoritmos de re-amostragem porém neste experimento optamos por utilizar o $systematic\ resampling$ ou amostragem sistemática. Foi-se optado por está pois a mesma faz um excelente trabalho para assegurar que seja feita a amostra de todas as partes do espaço de partículas, ao mesmo tempo em que garante que partículas com pesos W maiores são proporcionalmente re-amostradas com maior frequência.

E. Comparação

Tivemos a oportunidade de fazer uma comparação entre o AMCL (Filtro padrão do ROS) e o filtro deste experimento. Com isso obtivemos algumas métricas nos indicando que o filtro deste experimento se comporta muito bem em comparação ao AMCL. A diferença entre o agrupamento de partículas de ambos foi de aproximadamente 14%. Para extrair está métrica foi coletada uma amostra (5 partículas) e foi medida a distância

```
cumulative_sum = np.cumsum(weights)
i, j = 0, 0
while i < N:
    if positions[i] < cumulative_sum[j]:
        newParticles.append(deepcopy(self.particle_cloud[j]))
        i += 1
    else:
        j += 1
```

Figura 7: *Trecho de código utilizado no experimento*

que cada uma ficava do robô a cada movimento. Foi concluído que o AMCL tem uma acurácia de 93% após o décimo movimento do robot enquanto o filtro utilizado neste experimento tem uma acurácia de 79%. Os experimentos e comparação podem ser observados nos links https://youtu.be/sSDb32Uu2Pc e https://youtu.be/Emuxr_PubYA.

IV. Experimento

A. Treinamento e Validação

Durante o processo de treinamento foram utilizadas 143 imagens, sendo destas 10 para validação e 16 para teste. O algoritmo rodou durante 70 épocas e obteve os melhores resultados na época 51, onde atingiu-se o menor *loss* de validação ainda menor que o *loss* de treino.

O treinamento é feito calculando-se o *softmax* por pixel entre as diferentes classes e minimizando o *loss*. A validação é calculada pela média das interseções dos pixels sobre a união, com a média retirada de ambas as classes.

B. Resultados

Utilizamos como métrica de avaliação a Precisão, a Revocação e o total de acertos *pixel* a *pixel* das 16 imagens utilizadas no teste. A Precisão indica quanto dos *pixels* que foram preditos como teto realmente são teto. A Revocação mostra quanto dos *pixels* anotados como teto foram preditos como teto pelo algoritmo. O total de acertos indica a porcentagem de *pixels* corretamente preditos, sejam eles teto ou não.

Nossos resultados estão todos expostos na Figura 4 e da Tabela I. Em média nosso método foi capaz de acertar 97.8% dos *pixels* das imagens. Sendo que em média 95.8% são corretamente classificados como teto abrangendo 98% de tudo que é teto. Na Tabela II estão expostos os resultados do modelo proposto em [4] para comparação.

Note pelas imagens que mesmo objetos pequenos atravessando a região do teto são facilmente detectados pelo algoritmo. Tal resultado lido *frame* a *frame* dá uma excelente estimativa da área de teto contida na visão do robô.

V. Conclusão e Projetos Futuros

Ainda que a arquitetura utilizada tenha obtido bons resultados na tarefa de segmentar o teto, acreditamos que uma arquitetura mais simples possa obter resultados satisfatórios, diminuindo o tempo de predição e a exigência do hardware.

Também será necessária a integração entre o filtro de partículas deste experimento com a saída dada pela segmentação de teto para que seja possível localizar o robô pela segmentação de teto.

Tabela I: Resultados do modelo FCN-VGG19.

Imagem	Precisão (%)	Revocação (%)	Total de acertos (%)
1	0.986	0.986	0.987
2	0.968	0.973	0.967
3	0.983	0.988	0.983
4	0.982	0.993	0.987
5	0.991	0.995	0.993
6	0.989	0.994	0.991
7	0.962	0.973	0.962
8	0.988	0.987	0.992
9	0.954	0.971	0.984
10	0.820	0.983	0.952
11	0.983	0.994	0.993
12	0.965	0.986	0.970
13	0.992	0.978	0.988
14	0.823	0.902	0.930
15	0.971	0.993	0.990
16	0.971	0.989	0.989
Média	0.958	0.980	0.978
Min	0.820	0.902	0.930
Máx	0.992	0.995	0.993

Tabela II: Resultados do modelo H. Choi.

Imagem	Precisão (%)	Revocação (%)	Total de acertos (%)
1	0.846	0.972	0.908
2	0.799	0.991	0.859
3	0.956	0.977	0.960
4	0.928	0.936	0.927
5	0.934	0.945	0.934
6	0.907	0.966	0.928
7	0.853	0.938	0.872
8	0.949	0.993	0.980
9	0.592	1.000	0.855
10	0.626	1.000	0.876
11	0.974	0.981	0.987
12	0.943	0.981	0.952
13	0.984	0.968	0.982
14	0.750	0.957	0.913
15	0.680	0.996	0.872
16	0.694	0.968	0.880
Média	0.839	0.973	0.918
Min	0.592	0.936	0.855
Máx	0.839	0.973	0.987

Referências

[1] E. Shelhamer, J. Long and T. Darrell, *Fully Convolutional Networks for Semantic Segmentation*, 2016.

[2] F. Dias, H. Schafer, L. Natal, C. Cardeira, *Mobile robot localisation for indoor environments based on ceiling pattern recognition*, 2015.

[3] Seung-Hun Kim, Changwoo Park, *Ceiling-view and Front-view Localization Module with Single Camera for Mobile Robot*, 2012.

[4] H. choi, D. Kim, J. Hwang, C. Park, E. Kim, *Efficient Simultaneous Localization and Mapping Based on Ceiling-View: Ceiling Boundary Feature Map Approach*, 2012.

[5] Ioannis M. Rekleitis. *A particle filter tutorial for mobile robot localization. Technical Report TR-CIM-04-02, Center for Intelligent Machines*, McGill University, Montreal, Quebec, Canada 2004.

[6] Labbe R., *Kalman and Bayesian Filters in Python*, 2015.