CONFIGURAÇÕES DO ROS PARA UTILIZAR O PACOTE WHYCON

Nota: Esse tutorial foi construído seguindo o tutorial dos autores do pacote Whycon, no link https://github.com/LCAS/whycon

Breve introdução

O pacote whycon é uma ferramenta útil para o feedback de posição no controle de robôs móveis. Em essência, o pacote contém a aplicação de uma técnica de *position-based visual servoing*, i.e., obtenção de um feedback de posição através de sensores de visão (servovisão por câmera).

Os pré-requisitos para utilizar o pacote é ter o ROS Kinect instalado, o pacote Whycon instalado, e uma câmera calibrada (ver como calibrar clicando aqui).

O princípio de funcionamento do pacote é servir de um detector de padrões circulares planares em preto-e-branco (ver Figuras 1 e 2).

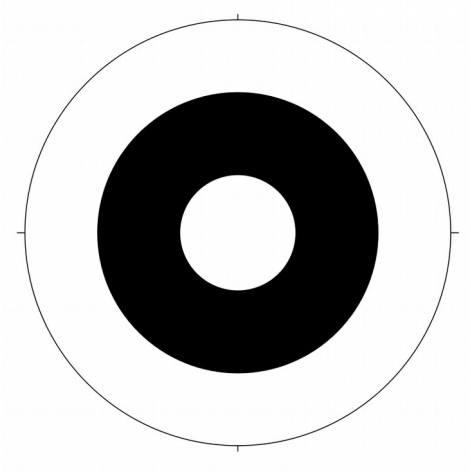

Figura 1 – Padrão utilizado para detecção por pacote Whycon.

Figura 2 – Exemplo de aplicação para robôs móveis.

Para calcular a área do círculo formado pelos pixels pretos, o pacote utiliza a técnica de processamento de imagem chamada flood-fill. Para estimar a distância entre o eixo coordenado da câmera e o círculo, uma transformação utilizando a matriz de projeção da câmera converte a imagem 2D em coordenadas 3D. Essa matriz de projeção é obtida pela calibração da câmera, por isso é necessário ter uma câmera calibrada para utilizar o pacote.

Mais informações podem ser encontrados nos tutoriais da página https://github.com/LCAS/whycon.

Tutorial de instalação

1. Instale o pacote Whycon no ROS conforme o passo a passo a seguir:

Have ROS Kinetic and appropriate camera driver installed. Also have a calibrated camera with distortion model "plumb bob".

0. Install the required libraries

```
> sudo apt-get install libsdl1.2-dev libsdl-ttf2.0-dev libncurses5-dev graphicsmagick-libmagick-dev-compat
```

2. Download the code from GitHub into a catkin workspace.
3. Compile the code - just type `catkin_make` in workspace directory.
4. Source setup script in package directory into shell enviroment e.g. `source devel/setup.bash`

Tutorial de uso

1. Certifique-se de possuir o AuRoRa 2018 atualizado, e copie o conteúdo da pasta “/Bebop e Whycon/bebop_scripts” para dentro do workspace bebop_ws na sua instalação do ROS “/home/bebop_ws/src/bebop_scripts”.

2. Faça o launch do robô Bebop utilizando o arquivo “B_whycon.launch” (ou “B_opti_whycon.launch” caso queira utilizar o ROS como servidor para o OptiTrack).

```
> roslaunch bebop_scripts B_whycon.launch
```

3. Após o launch, uma janela como a da Figura 3 irá aparecer. Mude os parâmetros para que fiquem semelhantes ao apresentados na Figura 3. Em específico, garanta o campo ‘`identify`’ desmarcado e ‘`numMarkers` = 1’, pois valores diferente destes farão o sistema funcionar, mas consumindo maior tempo de processamento.

IMPORTANTE: No campo ‘`circleDiameter`’, deve-se pôr o diâmetro em centímetros do padrão que está utilizando. Essa configuração é **imprescindível** para o funcionamento correto do pacote.

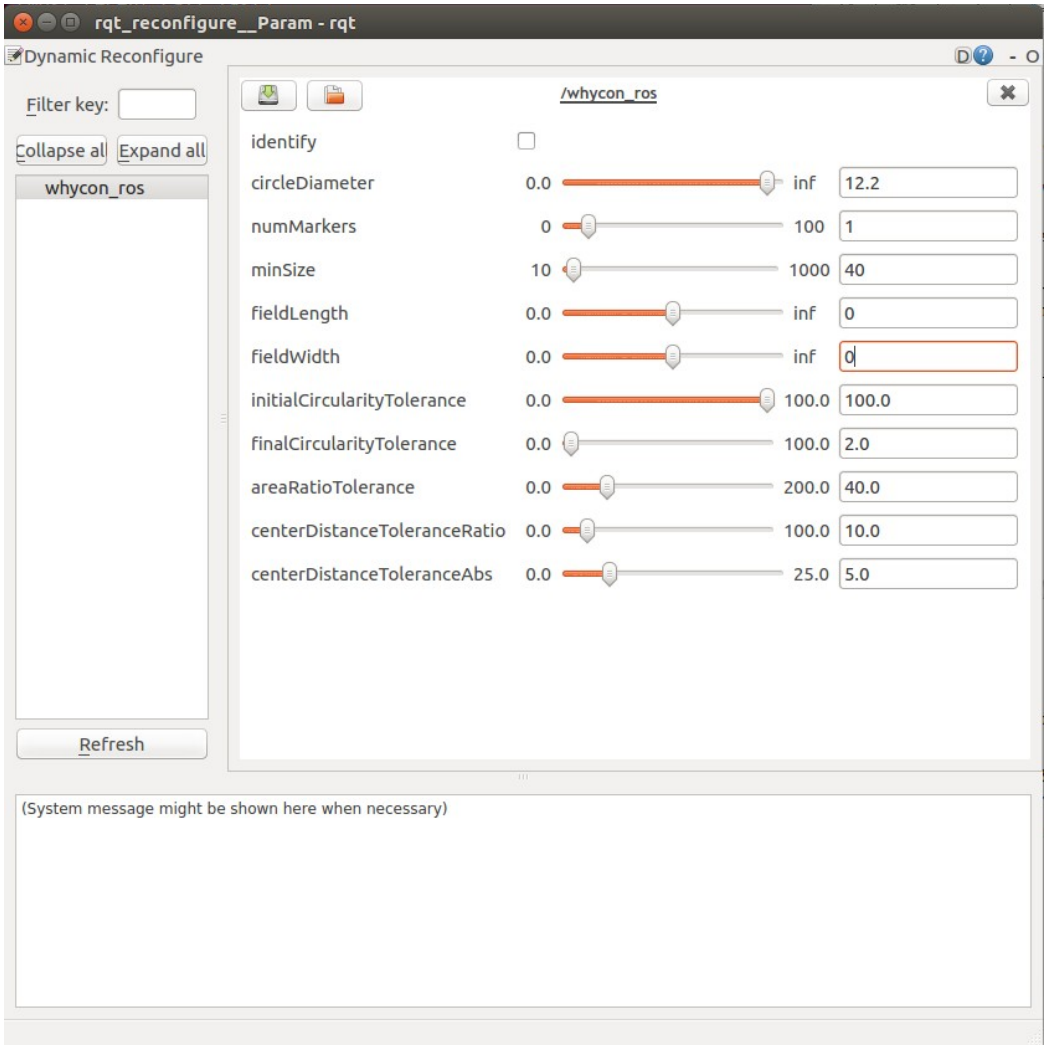

Figura 3 – Janela de configuração do pacote Whycon.

Pronto! Está feita a instalação e a inicialização do pacote. O último passo é imprimir o padrão circular preto-e-branco (está na pasta desse tutorial com nome **padrao 12,2 cm.pdf**). Conforme o próprio nome do arquivo nos conta, esse padrão possui 12.2 cm de diâmetro. Para realizar as leituras, essas podem ser feitas:

- Via ROS, com comando: `rostopic echo /B1/whycon_lab`, i.e., sendo subscriber do tópico whycon_lab e realizando sua leitura.

- Via Matlab, utilizando o script rGetMarker.m, que está dentro da classe @Bebop no AuRoRa 2018. Esse script irá atribuir à pose do Bebop (**B.pPos.X(1:6)**), a distância entre o Bebop e o padrão Whycon. Um script de teste executando a tarefa de Líder-Seguidor entre o Bebop e o padrão Whycon está disponibilizado na pasta "/AuRoRA 2018/ROS/Scripts/Test Scripts/Exp_Daniel_Bebop_Seguidor.m". Favor executá-lo para verificar o funcionamento do algoritmo.

OBS: Recomendo primeiramente testar o script sem o comando de *Takeoff* no Bebop, e ir manipulando o padrão Whycon na frente da câmera do Bebop, e printando no *Command Window* o valor do sinal de

controle, vendo se este sinal de controle está de acordo com a posição desejada na tarefa Líder-Seguidor proposta pelo exemplo.

Por fim e não menos importante, segue abaixo algumas informações acerca do uso:

- Somente a porção preta do padrão da Figura 1 é utilizado para a detecção. Portanto, é o diâmetro desse círculo que deve ser medido e inserido na Janela de configuração apresentada na Figura 3. No entanto, é importante que o padrão seja impresso da forma como apresentada na Figura 1, pois as bordas em branco até o fino arco em preto dão maior estabilidade à detecção do algoritmo.
- Ao executar o comando `roslaunch`, um feedback do que a câmera está recebendo e das posturas medidas podem ser vistas em uma janela que é aberta junto com a da configuração (Figura 3). O layout dessa janela pode ser visto na Figura 4.

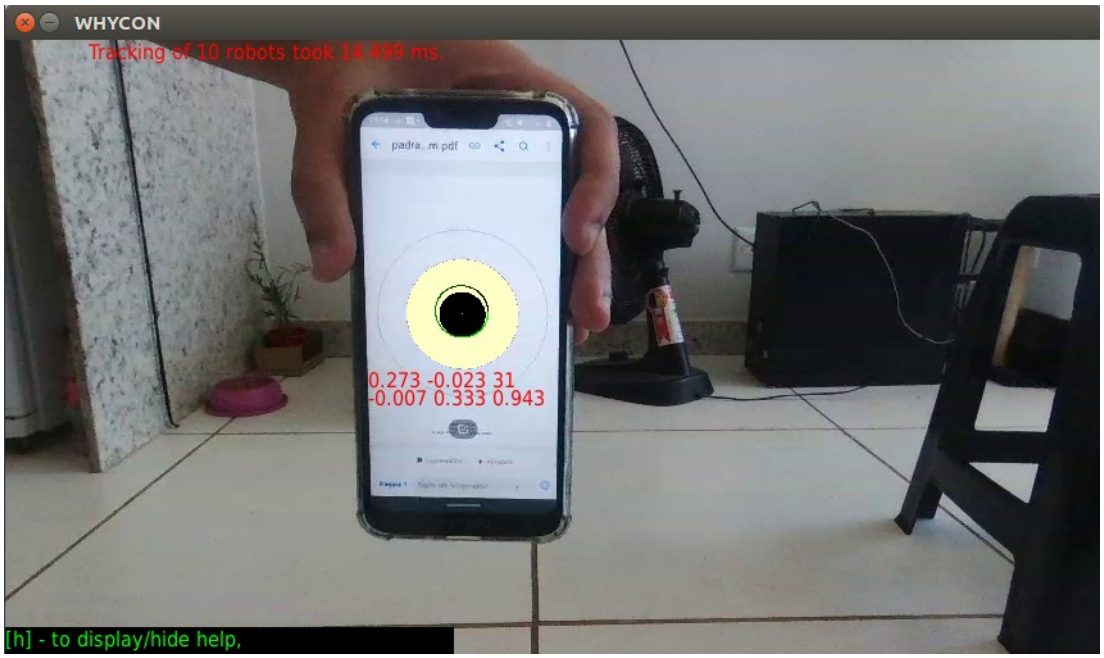

Figura 4 – Janela de output do Whycon em funcionamento. Os números em vermelhos se referem a pose do objeto rastreado.

- No uso padrão do pacote, as informações de postura estão presentes no tópico '/whycon_ros/visual'. No entanto, as mensagens desse tópico são do tipo *visualization_msgs*, e estão em um padrão diferente do utilizado pelo Matlab, impossibilitando sua leitura. Em função disso, Pacheco, o mago do ROS, criou um script chamado 'whycon_publisher.py' para transportar as informações obtidas no whycon para o tópico '/whycon_lab', utilizando mensagens do tipo *geometry_msgs*, que são normalmente lidas pelo Matlab. Paguem uma cerveja para ele.

Considerações finais

Sugestões, dúvidas, ou erros encontrados podem ser enviados para Daniel Villa (danielkdv@gmail.com).

Colaboradores

Vinicius Pacheco Bacheti

Mauro Sérgio Mafra Moreira