

Robótica Computacional - Projeto 1 em grupo

Última modificação: {{ file.mtime }}

Resumo

Seu grupo deve programar um robô em ROS para que reaja de forma diferente a dois objetos, seguindo um deles e usando IMU e LiDAR para se autopreservar.

Veja os [projetos do ano passado](#) para ter uma idéia.

Sempre que possível leia este enunciado e as rubricas online. Estes documentos são atualizados constantemente.

Objetivos

Objetivos do projeto: * Programar identificação de objetos em OpenCV * Fazer o sistema de controle para um robô (a.k.a arquitetura cognitiva) * Praticar programação por eventos (listeners) * Aprender sobre ROS e Linux * Praticar gerenciamento de projetos

Gestão de projetos

Se todos deixarem para trabalhar só durante as aulas e não houver responsáveis claros pelas tarefas o projeto não vai sair. Este projeto tem o desafio de aproveitar bem todas as pessoas dos grupos. Para tanto, por favor sigam as recomendações:

- Planejem ter tudo pronto bem antes do prazo final. Problemas práticos vão surgir.
- Lembrem-se de que só vale o que for documentado em vídeo
- O grupo deve se organizar usando Trello, Asana ou algum outro suporte de sua preferência.
- Seu Github deve servir de documentação - use o README.MD para explicar o que está acontecendo.
- É importante que os professores Fábio de Miranda e Andrew Kurauchi (fmiranda@gmail.com e andrew.kurauchi@gmail.com) sejam incluídos para acompanhar o grupo (se você usar o Asana os e-mails são FabioMiranda@insper.edu.br e AndrewTNK@insper.edu.br).
- É importante que o board do grupo deixe claro **quem** é responsável por cada tarefa em andamento

- Sempre que uma tarefa for concluída, deixar claro quem a concluiu e colocar o link para o commit do Github que a resolveu.
- Todos os commits do Github precisam ter comentários relevantes
- Fiquem à vontade para eleger um *Scrum Master* e para alternar esta função.

Os membros dos grupos devem avaliar uns aos outros segundo [esta rubrica de trabalho em equipe](#). Esta avaliação por pares **não entra na nota**.

Lembrem-se de que o projeto tem entregáveis de código final e entregáveis de aprendizado/testes. Saibam aproveitar a quantidade de pessoas disponíveis no grupo e distribuir bem ambos os tipos de atividades.

Datas

March 2018

Su	Mo	Tu	We	Th	Fr	Sa
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

April 2018

Su	Mo	Tu	We	Th	Fr	Sa
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

Momentos importantes

- 15/3 - Início
- 19/3 - todos os grupos já devem ter planejado um primeiro sprint e estarem aptos a desenvolver o projeto (alguma solução para ROS)
- 19/3 - Indicar quem é seu grupo
- 26/3 - Fim do primeiro sprint e início do segundo
- 30/3 - Questionário intermediário de avaliação - todos os membros do grupo recebem as respostas anonimizadas

- 10/4 - Dia da prova. Os grupos devem vir num horário agendado entre 12:00 e 17:00 e demonstrar o estado atual do projeto.
- 16/4 - Entrega final do vídeo (no Youtube, pode ser oculto) e do código e questionário final de avaliação do trabalho em grupo

Critérios

Veja [a rubrica do projeto](#)

Integridade intelectual

Você pode se basear em qualquer código desde que o entenda não seja código de outro grupo. Cada grupo deve desenvolver seu projeto de forma independente, mesmo que dois grupos se baseiem na mesma solução. Em caso de dúvida se algo é ou não plágio consulte o professor.

Lembre-se que agora [dois plágios significam expulsão do curso](#) (requer *login*).

Links úteis

*ROS** Exemplos básicos de ROS no Github da Disciplina https://github.com/Insper/robot18/tree/master/ros/exemplos_python/scripts

ROS Robot Programming - Capítulo 7 https://www.dropbox.com/s/3qk4qarl0vxvqyd/ROS_Robot_Programming_EN.pdf?dl=0

Exemplo de ROS com OpenCV https://github.com/Insper/robot18/blob/master/ros/exemplos_python/scripts/cor.py

Exemplos de Publishers e Subscribers da ROS Wiki <http://wiki.ros.org/rospy/Overview/Publishers%20and%20Subscribers>

Máquinas de estados Smach - Máquinas de Estados <http://wiki.ros.org/smach/Tutorials>

Visual States - outra alternativa para fazer máquina de estados <http://jderobot.org/VisualStates>

OpenCV Cam shift com OpenCV <http://www.pirobot.org/blog/0016/>

Fluxo óptico https://docs.opencv.org/3.3.1/d7/d8b/tutorial_py_lucas_kanade.html

SLAM Turtlebot3 Slam <https://www.youtube.com/watch?v=hX6pFcf29c>

SLAM https://www.youtube.com/watch?v=7mEKrT_cKWI

SLAM com marcadores Aruco [<http://wiki.ros.org/fiducials>]

Alvar

Marcadores de realidade aumentada (Alvar) http://wiki.ros.org/ar_track_alvar

São outros marcadores fiduciais e podem servir para você implementar seu próprio sistema de localização, vejam o exemplo de Alvar feito por Rachel Moraes (pioneira computação): https://github.com/mirwox/robot17/blob/master/exemplos_projeto1/scripts/marcador.py

Documentação do Alvar / ROS http://wiki.ros.org/ar_track_alvar

IMU - Inertial Measurement Unit Subscriber simples para IMU (em C++) http://wiki.ros.org/evarobot_minimu9/Tutorials/indigo/Writing%20a%20Simple%20Subscriber%20for%20IMU

YOLO Demo do YOLO (Vimos em aula) <https://www.youtube.com/watch?v=VOC3huqHrss> Módulo do YOLO para ROS https://github.com/pgigoli/darknet_ros