

# ROS Project

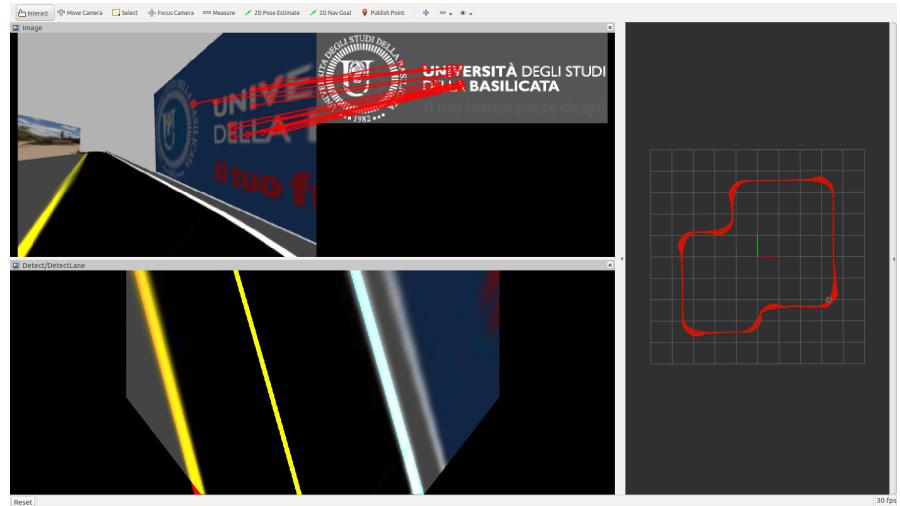
**Corso di Sistemi Informativi A.A. 2018/19**

***Docente***

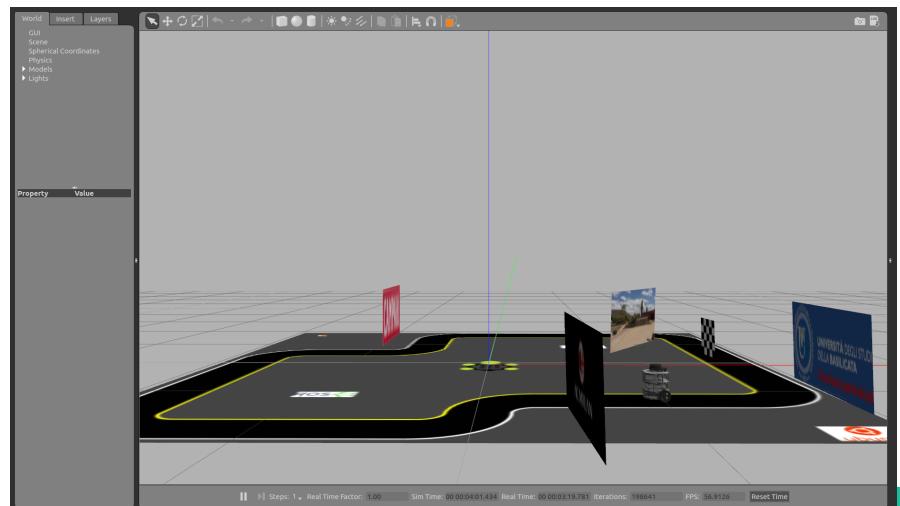
***Domenico Daniele Bloisi***

# Il Progetto

- **Turtelbot con giuda autonoma e riconoscimento immagini**



- **Simulato in ambiente Gazebo e visualizzazione in Rviz**



# Il Progetto

- Questo progetto utilizza ROS per dimostrare che Turtlebot segue un percorso in un ambiente Gazebo simulato.
- E' un'applicazione di progettazione per un robot che richiede una videocamera ed una corsia delimitata da bande.



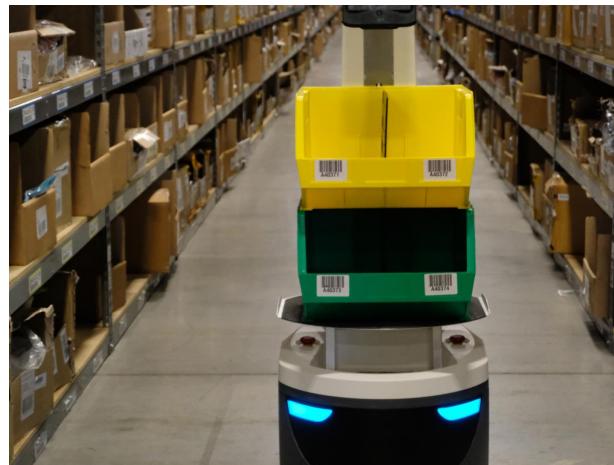
# Il Progetto

- Può essere utilizzato per creare modelli complessi di alto livello utilizzando più robot in più percorsi e può essere implementato nelle industrie e nelle officine per la movimentazione e la distribuzione dei materiali.



# Il Progetto

- **Questi robot possono essere realizzati per seguire autonomamente percorsi in intervalli periodici e quindi automatizzare la funzione dell'intero sistema di movimentazione dei materiali in modo economicamente efficiente.**



# Procedura

**Inizialmente è stato creato un mondo Gazebo contenente alcune immagini e un percorso da seguire per il turtlebot.**

- **Questo progetto ha due nodi:**
- **1 nodo per la trasformazione dell'immagine con la matrice di omografia per il rilevamento della linea e delle immagini**
- **1 nodo visualizzatore**



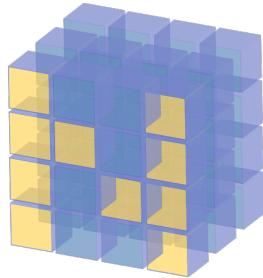
# Requisiti

- **Ubuntu 16.04**
- **ROS con Kinetic**



# Librerie

- **import rospy**
- **import numpy as np**
- **import math**
- **import os**
- **import cv2**
- **from enum import Enum**
- **from std\_msgs.msg import UInt8**
- **from sensor\_msgs.msg import Image, CompressedImage**
- **from cv\_bridge import CvBridge, CvBridgeError**



NumPy



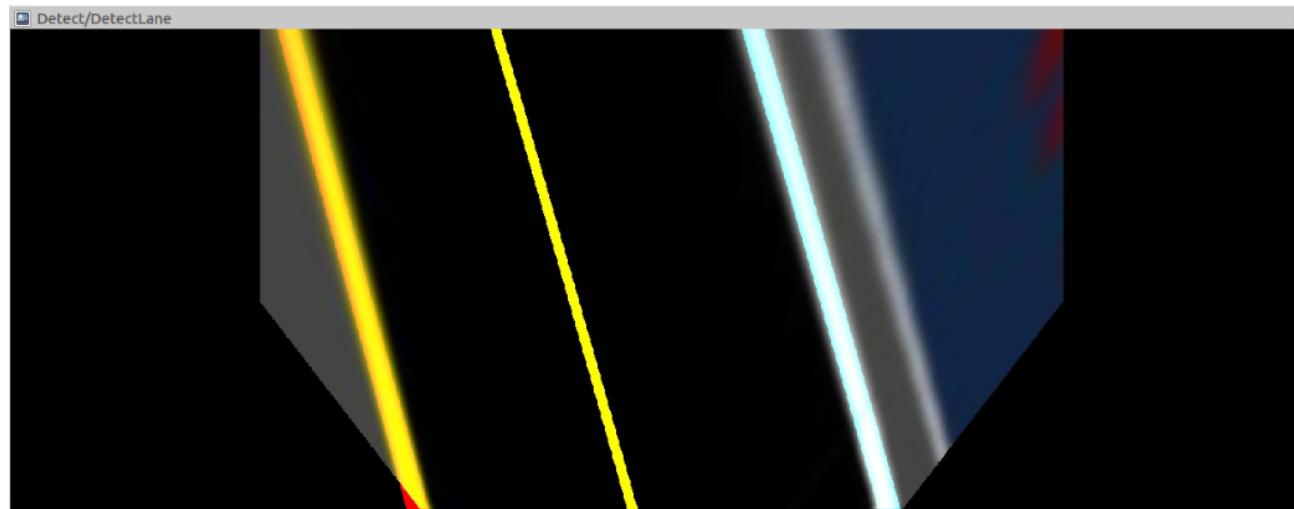
ROS.org

# Rilevamento linee

**Per la guida autonoma sono utilizzati nodi ROS**

- **detect\_lane:** rileva le linee una gialla e una bianca nell'immagine proiettata e compensata.
- **control\_lane:** riceve un ingresso dalla corsia di rilevamento e elabora un cmd\_vel per spostare il robot nella giusta direzione.
- **image\_compensation:** compensa l'immagine proiettata. Viene usato ma il suo effetto è nullo perché non è necessaria la compensazione nell'ambiente di simulazione.
- **image\_projection:** proietta l'ingresso della telecamera.

# Screenshot



# OpenCV

- **OpenCV** (Open Source Computer Vision Library: <http://opencv.org> ) è una libreria open source BSD con licenza che include diverse centinaia di algoritmi di visione del computer.
- è molto popolare e ben documentata. È un ottimo strumento per qualsiasi visione artificiale o applicazione di visione artificiale.



# Algoritmo SIFT

- Nel 2004, D.Lowe , University of British Columbia, ha elaborato un nuovo algoritmo, Scale Invariant Feature Transform (SIFT) nel suo articolo, Distinctive Image Features from Scale-Invariant Keypoints , che estrae i punti chiave e calcola i descrittori.
- Ci sono principalmente quattro passaggi coinvolti nell'algoritmo SIFT:
  1. Scale-space Extrema Detection
  2. Keypoint Localization
  3. Orientation Assignment
  4. Keypoint Descriptor



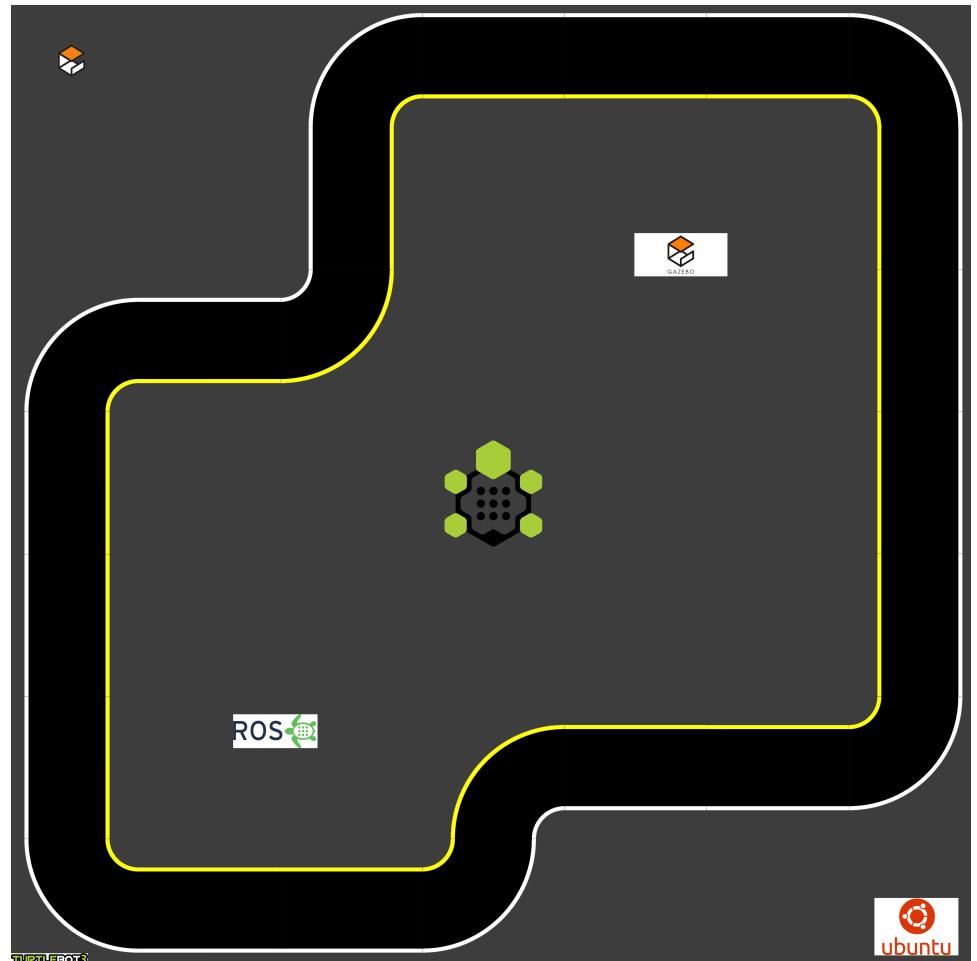
# Model

- La **texture** è stata realizzata con un software grafico e poi utilizzata per creare il “model” in gazebo così come le immagini al lato del percorso.



# Texture

- Il percorso è caratterizzato da una corsia delimitata da una banda gialla ed una bianca che sono ottimi punti di riferimento quando si vuole realizzare una guida autonoma



# Installazione

- **Link del progetto:**

[https://github.com/Luca23Dev/unibas\\_ros\\_line\\_follower.git](https://github.com/Luca23Dev/unibas_ros_line_follower.git)

