Exam prep tasks

SLAM:
Simultaneous localization and mapping

Writing a simple implementation for rangefinder scanner sensors

Exam final results

- We will get simple SLAM implementation usable to some extent in real world applications
- This algorithm will give us not only robot's trajectory but also map of surrounding areas
- Examples of this algorithm work can be seen in the **slam*.mp4** videos

Reading and animating data

- Review **animation.py** script
- Run it. You should get an animation of rangefinder data in polar coordinates
- You can see recorder version of this animation in **polar_animation.mp4** video
- Learn from this script how to open dataset files and how to make animations using matplotlib

Running doctest

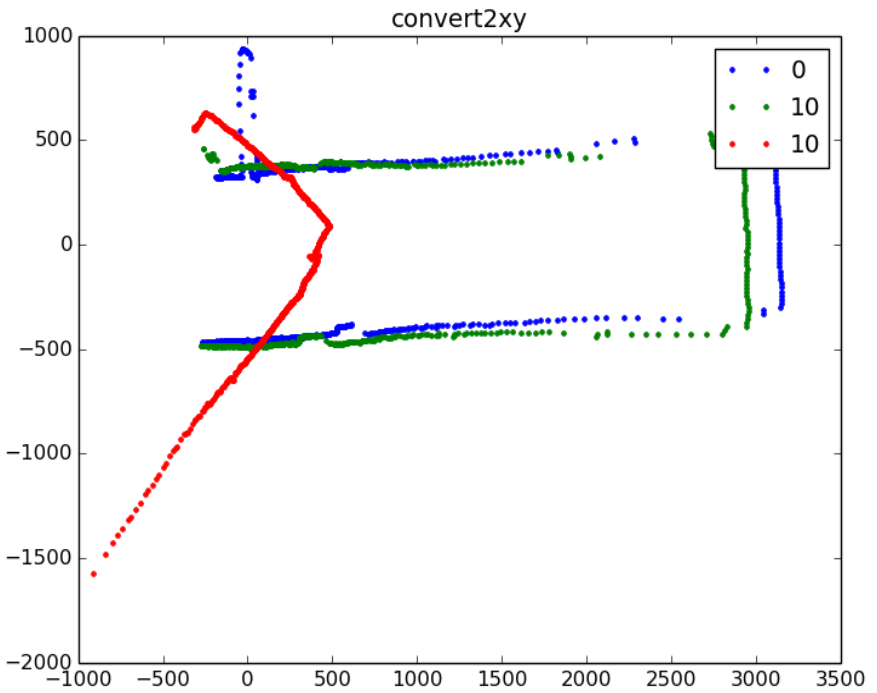
- Review **doctest_exmple.py** script
- Run doctests either by simply running script (i.e. **python3 doctest_exmple.py**) or using other methods (e.g. **python3 -m doctest -v doctest_exmple.py** or other environment dependent methods)
- Try change functions and run doctests again

Writing helper function for SLAM

- Review **slam_exam.py** script and function's docstrings inside it
- You should see empty functions with documentation describing what they should do and some doctests
- Further work will mainly consist of writing this functions so they pass doctests
- Note: due to possible (but unlikely) problems with computations precision your implementation can be correct but do not pass this strict doctests, in this case notify examiners

Converting scan data to Cartesian coordinates

- Implement `convert2xy` function
- In case of successful implementation you should get from doctest plot like this:

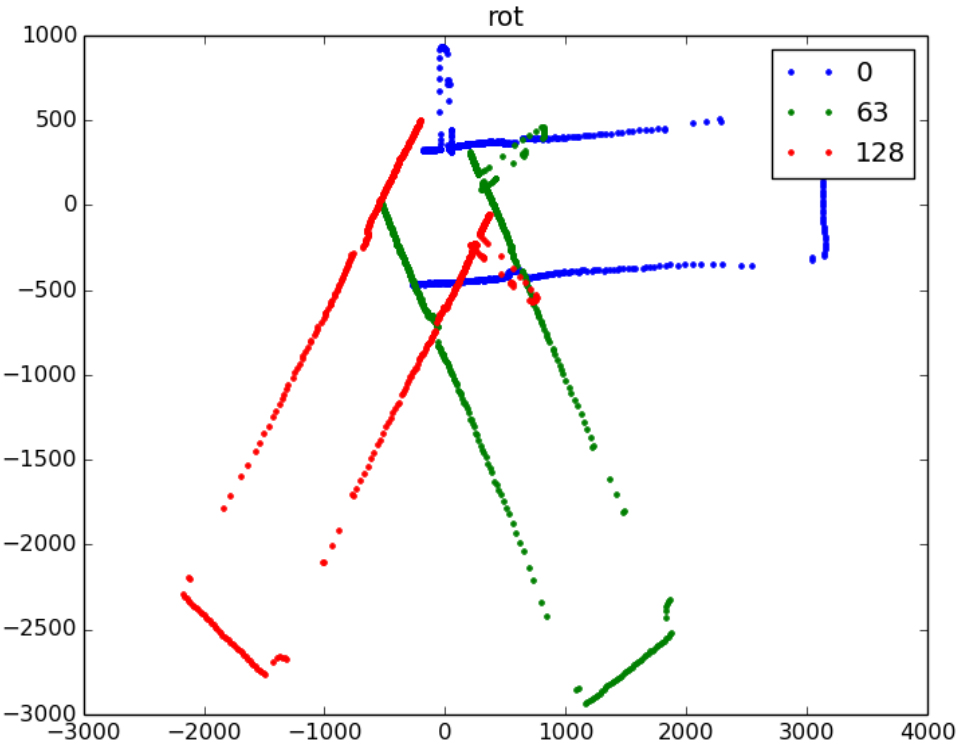


Animating XY points

- Animate scan measurements in Cartesian reference frame by modifying **animation.py** script and using `convert2xy` function

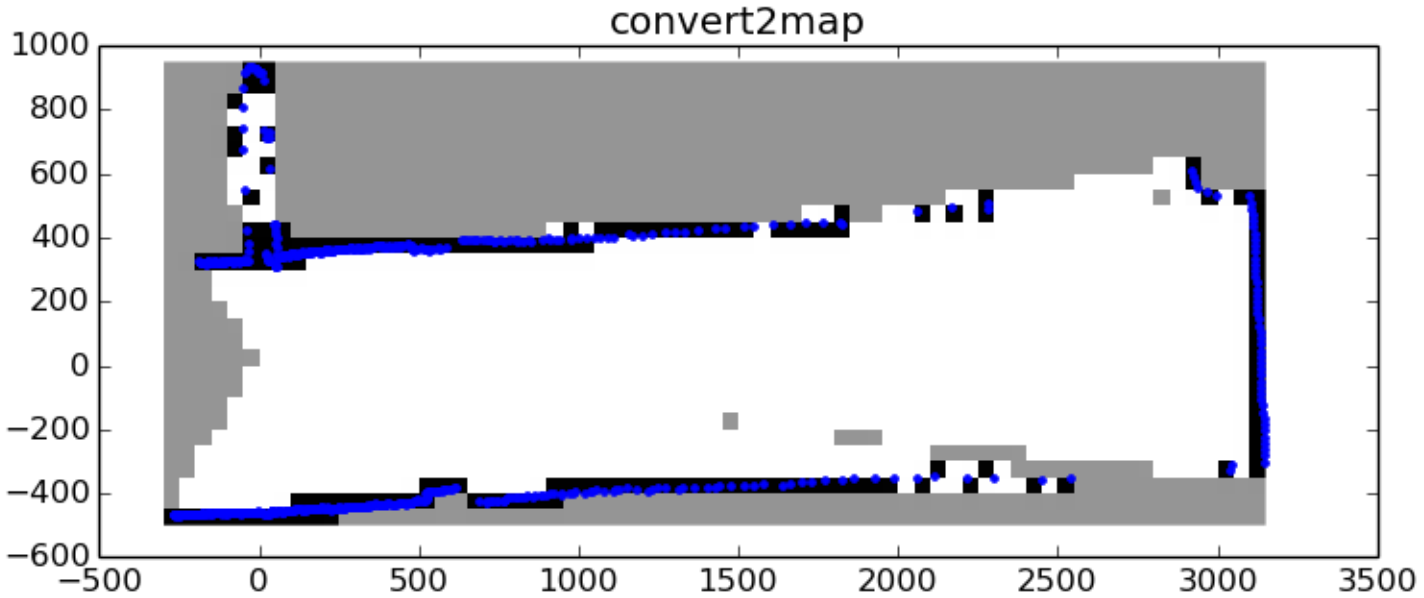
Rotating XY points

- Implement `rot` function
- In case of successful implementation you should get from doctest plot like this:



Converting points to map

- Implement `convert2map` function
- In case of successful implementation you should get from doctest plot like this:



convert2map algorithm description

- Convert list of measured point coordinates to list of pixel coordinates (e.g. if pixel size is 50 mm then point (33, 170) will be converted to (0, 3))
- Create image with necessary size using PIL's `Image.new('L', img_size)` (for determining iamge size you can use numpy's `min` and `max` on pixel coordinates list)
- Create draw object from image using PIL's `ImageDraw.Draw`
- Using draw object's `line` method draw lines from sensor position to measured points
- Convert image to numpy array using Image's `fromstring` and numpy's `tostring`

Animating map and points

- Animate map and points using for reference **animate.py** and `convert2map` docstrings
- Note: you will need change extent for map using `set_extent` method in addition to `set_data`.
- You can see example of recorded animation in **map_animation.mp4** video