1- Using Machine Learning and Simulation to predict the traversability of a given terrain by a simple robot

Supervisors: Alessandro Giusti, Jerome Guzzi, Luca Gambardella

The main goals of the project are:

- learn to simulate a simple robot (such as a three-wheeled robot, or a small car) moving on a complex 3D terrain
- from many simulations, build a dataset which associates a terrain patch to a label (i.e. whether the robot was able to traverse the patch or not)
- using this dataset, build a classifier (such as a neural network) to predict the traversability of a given terrain patch.

Detailed tasks:

- Familiarize yourself with ROS and a 3D robot simulation package such as Gazebo or V-REP
- Build a realistic terrain environment by importing an heightmap
- Experiment running a simple robot model on the map. Make it so that the map has some areas which are easily traversed and others where the robot gets stuck
- Experiment with writing a controller for the simulated robot in such a way that the robot attempts to follow a predeterminate path (e.g. straight or circle or spiral).
- Learn how to control the simulation programmatically and write code to automatically analyze what happened during a simulation (e.g. find out which areas the robot traversed correctly and where the robot got stuck or capsized).
- Run hundreds of simulations automatically and create a dataset with all the terrain areas where the robot passed, got stuck, or capsized.
- Analyze the dataset with supervisors and investigate options to apply machine learning to automatically estimate whether a given area is traversable or not.

We suggest building the system using Python. Matlab or another high-level language the student is familiar with is also accepted.

2- Automatic generation of realistic terrain height-maps from GIS information

Supervisor: Jerome Guzzi, Luca Gambardella

Given high-level GIS information about an area (e.g. 100m x 100m), generate a realistic terrain heightmap. GIS information reports high-level data such as land use, where roads are, etc. The generated heightmap should be plausible and realistic, and could be used for running robotics simulations.

Tasks:

- Familiarize yourself with spatial data formats like GML that provide, among others, informations on elevation and type of terrain.
- Develop an algorithm that given a portion of a GIS map, outputs an height map (stored as a grayscale image where intensity corresponds to elevation or as a matrix) of the terrain by just considering the elevation information. Try different methods to interpolate between elevation curves.
- Improve the algorithm to make use of terrain information (e.g. a grass field is generally smooth, while a stony ground is more irregular).
- Add parameters to tune the mapping between terrain and height map (granularity, roughness, …) and select realistic settings.
- Add randomness to improve the realisticity of the generated heigh maps.

It has to be possible to import the resulting height maps in robotics simulators like V-REP and Gazebo but no additional work is expected to be done within the simulators.

3- Ball tracking in videos using machine learning

Supervisor: Alessandro Giusti, Luca Gambardella

The task of the project is to implement part of a pipeline for tracking a ball in an image using machine learning.
Preferred language choice is Python or Matlab.

The specific machine learning problem to be solved can be described as follows:
Given a 100x100 pixel image patch, predict:

- whether there is a ball in the image
- if there is a ball: where is its center in the patch (pixel coordinates)
- if there is a ball: how big it is (pixel radius)

We can solve this problem by training a deep neural network using e.g. Brainstorm https://github.com/IDSIA/brainstorm, Caffe http://caffe.berkeleyvision.org/ or any other deep learning frameworks.
Once we have a deep net solving this problem, we can apply it to track a ball in a video by applying the net to a 100x100 image patch centered on the last known position of the ball.

The project can be decomposed in the following tasks:
<u>Task 1</u>
Render a dataset of ball images: using e.g. Blender (https://www.blender.org/) and its Python scripting interface (or another software), generate 1k images of a ball with uniform color with different (random) lighting conditions. Output: 1k 210x210 pixel RGBA PNG images, each of which represents a ball with uniform texture and different lighting, with radius 100px and centered in the center of the image. The background is transparent.
<u>Task 2</u>
render a training dataset for the neural network. We want to generate 100k 100x100 grayscale image patches. For each patch, the following needs to be known:

- whether there is a ball in the image
- if there is a ball: where is its center in the patch (pixel coordinates)
- if there is a ball: how big it is (pixel radius)

This can be obtained as follows: first, download a dataset of images to be used as background, such as: http://dags.stanford.edu/projects/scenedataset.html. Then, for each of the 100k images to be generated:

- take a random 100x100 patch from a random background image
- with probability 0.5:
    - take a random ball image
    - rescale it to a random size so that the radius of the ball is uniformly distributed in the interval [5 50] px.
    - paste it in a random location of the patch so that at least part of the ball overlaps the image.

Save the resulting dataset in a reasonable format (directory of images + CSV, or better, HDF5 https://www.hdfgroup.org/HDF5/). You need to save each image (grayscale, 1 byte per pixel), whether the ball is present (boolean), and the corresponding (x,y,r) information.

Task 3

Using a suitable library (Caffe, or others), train and test a deep neural network for solving the classification problem: given an image patch, is there a ball in the image? Output: yes/no.
For images with a ball in it, also consider the associated regression problem: given an image with a ball in it, find (x,y,r). IDSIA can provide suitable network architectures or take care of this task if needed. There is no requirement of previous knowledge of deep learning from the student(s).

Task 4

Apply the system to a few simple real-world images shot on purpose.

Task 5

Test whether the classifier can track a ball in a video where the ball moves slowly. Given an initial position of the ball, initialize the system on the initial position. For each new frame, we consider a 100x100 patch extracted from the last known position of the ball, apply the net, and find the new (x,y) position of the ball's center. Iterate for several frames.

Ideas for possible extensions

- Extend the approach to also consider the radius of the ball (always crop a patch 4 times the found radius, then rescale it to 100x100 so that the expected radius of the ball is 25px)
- Find the initial position of the ball by applying the net to random patches until the ball is found.
- Consider textured balls
- Consider common Image degradations such as motion blur in the training set
- Integrate the approach in a simple kalman filter which tracks the ball's position, radius and velocity

4- Automated Trail Following: Comparing Deep Nets on Clean vs Original Datasets

Supervisors: Alessandro Giusti, Jerome Guzzi, Luca Gambardella

IDSIA has recently developed a system for automated trail following using a quadrotor: http://bit.ly/perceivingtrails
The system is based on machine learning and several hours of video have been acquired on many trails in the area. Read in the linked paper about the dataset acquisition mechanism. Note that as a consequence of such mechanism, a non-trivial percentage of samples (about 5-15%) are incorrectly labeled.

The main goals of the project are:

- write a system to quickly manually verify and relabel the data. In particular, the user is shown an image and should be able to quickly enter the correct label. As we have a real person labeling the data, we can also extend the set of labels. For example, by adding a special class for "no trail visible", or by adding an additional classification output related to the lateral drift with respect to the trail.
- train a classifier on the manually-labeled data
- compare the new classifier with the old one (which was trained on a "dirty" dataset): did the performance improve?

Detailed tasks:

- download and familiarize yourself with the dataset from the link above
- write code that randomly selects a sample from the dataset and shows it to the user
- devise a very quick way for the user to enter its classification (e.g. a single click on the arrow keys: fwd (trail straight ahead), left (trail heading clearly left), right (trail heading clearly right), back (trail not visible), space (very poor quality image).
- experiment whether there is a simple and effective way to capture "soft" data, e.g. by clicking on the screen in the direction you would have to walk to stay on the trail.
- experiment the best way to capture information about the lateral shift from the trail centerline
- find online other video sources representing trails (e.g. from mountain biking videos)
- recruit people to classify a good amount of images (expected: about 10k classified images, corresponding to about 8h total work assuming 3 seconds per image, i.e. 0.5h per person if you recruit 16 friends).