BADGR: An Autonomous Self-Supervised Learning-Based Navigation System

Matthias Gubler, Tim Krenz, Daniele Colombo

Zurich University
of Applied Sciences

zh
aw School of
Engineering

Table of Contents

- Introduction
- The robot platform
- **4 Steps to the solution:**
  - Data collection
  - Data labelling
  - Training a neuronal network
  - Use network for predictions.
- Results

Introduction

About the Paper

- Published 2020 by Gregory Kahn, Pieter Abbeel and Sergey Levine
- BADGR = Berkeley Autonomous Driving Ground Robot

Introduction – The Problem

> «Navigation is a geometric problem, in which the robot's objective is to perceive the geometry of the environment to plan collision-free paths towards a desired goal»

purely geometric reasoning isn't enough!

- No difference between paved path and bumby grass.
- Tall grass likely to be identified as an obstacle

Introduction – The solution

Simple Question:

"Can we enable robots to reason directly from images”?

Solution is learning by doing!

- Robot learns through own experiences in the real world.
- Without simulation
- Without human supervision

Introduction – The solution

BADGR works by:

1. Autonomously collecting data (timesteps)
2. Automatically labelling the data with self-supervision
3. Training an image-based neural network predictive model
4. Using the model to plan into the future.

The Robot Platform

They used:

- Clearpath Jackal (508mm × 430mm × 250mm, 17kg)
- **Sensors suite measures**:
  - linear acceleration
  - angular velocity
  - global position (GPS)
  - images with two forward-facing cameras.
- NVIDIA Jetson TX2 computer (ideal for deep learning)
- External SSD to store 1.3GB per minute from Sensors.

Data collection

- Robot explores enviroment with time-correlated random walk.
- Collects data f.E image, linear and angular velocity, position etc.
- Collision should be answered with an automatic reset.

Self supervised Data Labelling

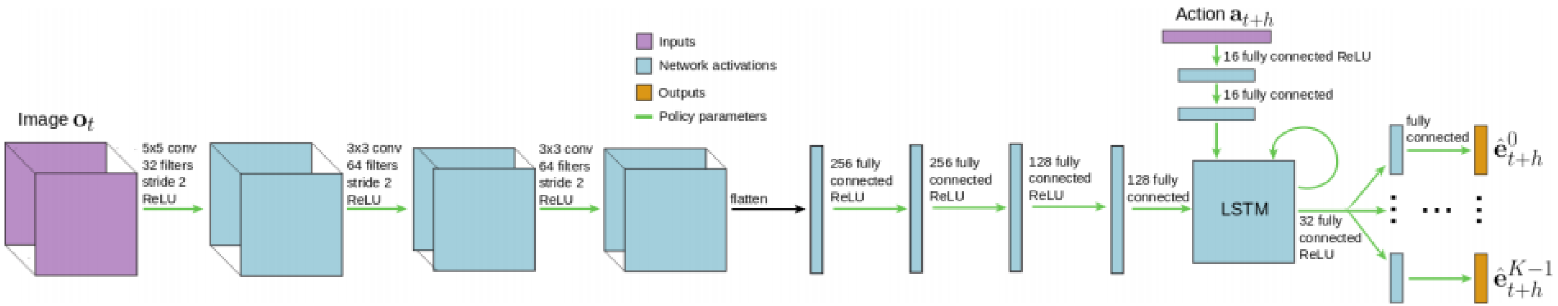
- Robot has to label the data for every time-step.
- Write small code snippets to map raw sensor data to a label.

```
double angularVelocity = readAngularVelocity();
if (isLarge(angularVelocity)) {
    return Label.BUMBY_ROAD;
}
```

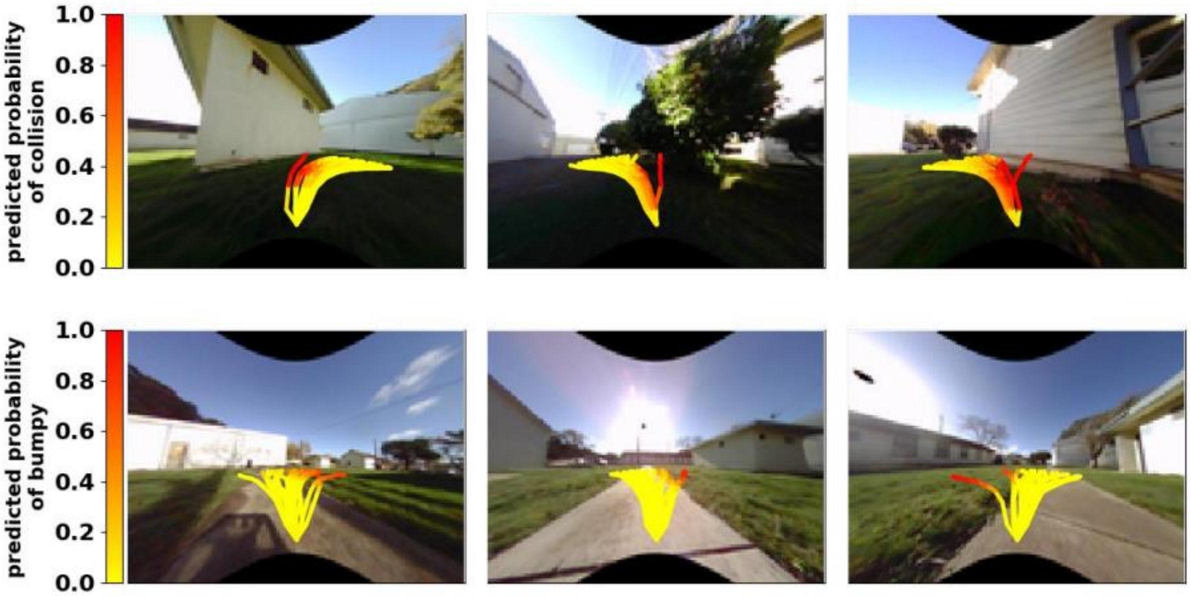
- Yes we need labels for a Neuronal-Network! (next Step)

Training a neural network

From images and labels to a deep neuronal network:



Prediction



predicted probability of collision
1.0
0.8
0.6
0.4
0.2
0.0
predicted probability of bumpy
1.0
0.8
0.6
0.4
0.2
0.0

The prediction together with a reward-function will allow the robot to create an optimal plan to reach his goal.

Results

https://bair.berkeley.edu/blog/2020/03/12/badgr/