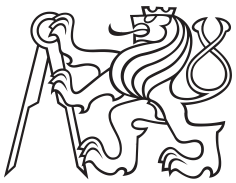
Bachelor Project



Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Drone detection using neural networks from combined RGB camera and LiDAR data

Adam Škuta

Supervisor: Matouš Vrba
Supervisor–specialist: Martin Saska
Field of study: Mathematical Engineering
Subfield: Mathematical Modelling
February 2017

Acknowledgements

Děkuji ČVUT, že mi je tak dobrou *alma mater*.

Declaration

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškerou použitou literaturu.

V Praze, 10. February 2017

Abstract

Keywords: word, key

Supervisor: Matouš Vrba
Ústav X,
Uliční 5,
Praha 99

Abstrakt

Klíčová slova: slovo, klíč

Překlad názvu: Moje bakalářka se strašně, ale hrozně dlouhým předlouhým názvem — Cesta do tajů kdovíčeho

Contents

Figures

Tables

Chapter 1

Introduction

The goal of this thesis is to prove whether a usage of LiDAR data coupled with images from RGB is useful for the localization of UAVs in contrast to the usage of image data alone. The LiDAR and RGB camera will be mounted on top of the scanner UAV. All the measurements will be taken inside a virtual environment, with a realistic UAV and sensor simulation. The dataset will then be preprocessed and used as the input to a Convolutional neural network for the object detection. The preprocessing will be as following:

- Coordinate transformation for the non-matching coordinate systems
- Projection of 3d points into a 2d image
- Using a Convolutional neural network to generate more points in a sparse LiDAR pointcloud

Infographic of the process

1.1 State of the art

- **Expandable YOLO**: A convolutional neural network model bases on YOLOv3 Darknet53 backbone and detection heads expanded to the 3rd dimension. The model can detect 3D bounding boxes from RGB images and stereo camera depth maps. In contrast to other more complex and robust 3D detection neural networks, this model aims to be usable in real-time enviroment by using a simpler structure

- **YOLO3D**
- **YOLOv5**
- **Complex YOLO**

Chapter 2

Convolutional neural networks

A convolutional neural network will be used in two problems in the thesis. First one is used on a sparse LiDAR pointcloud generating more points and therefore making it more dense. Second one is used for object, int this case drone, detection, using RGB and RGBD data as the input.

2.1 Sparse to dense

Sparse to dense is a Convolutional neural network written in PyTorch. It predicts the depth measurements from sparse depth dataset. The size of the network is modifiable and can be chosen as training parameters.

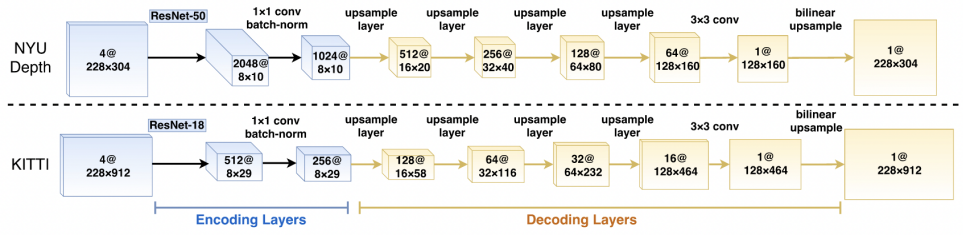


Figure 2.1: Example of different network architectures available

For the encoding and therefore input layers a ResNet-50 or ResNet-18 can be chosen, depending on the size of the input image for memory constraints. The decoding layers consist of 4 upsampling layers and a deconvolutional layer with either stride 2 or 3 or uprojection layer or upconvolutional layer as

a choice for training. The default loss function is least absolute deviations also known as L_1 error:

$$L_1 = \sum_{i=1}^{n} |y_{true} - y_{predicted}| \tag{2.1}$$

where:

- n is batch size
- y_{true} are real depth values
- $y_{predicted}$ are predicted depth values

The input to the network are RGBD images and the output is a depth map with the same dimensions as input. The depth input D is sampled from the ground truth depth map D^* with the following formula:

$$D(i,j) = \begin{cases} D^*(i,j), & \text{with probability } p \\ 0, & \text{otherwise} \end{cases} \tag{2.2}$$

where:

- i, j are coordinates of the input image
- $p = \frac{m}{n}$, where m number of depth samples to be chosen at the start of training and n is the total amount of available depth samples

During training several input data augmentations take place. These augmentations include:

- Scaling the input image by a random number $s \in [1, 1.5]$
- Rotating the input image by a random degree $r \in [-5, 5]$
- Scaling the brightness, contrast and saturation of the RGB component of the image by a random number $k \in [0.6, 1.4]$
- Normalizing the RGB component of the image
- Flipping the image horizontally with a 50% chance

The output of the network is a dense depth image with the dimensions of the input. Every pixel contains predicted depth measurement in meters. The output of the Sparse to Dense network will be used for further training later.

2.2 YOLOv3

citations

You only look once (YOLO) is a convolutional neural network model used mainly for object detection and recognition. The main advantage is its simplicity in comparison to similar convolutional neural networks, resulting in faster detection speeds. It belongs to the state of the art convolutional neural networks for object detection and recognition. The version used in this work is the third version YOLOv3. The backbone called Darknet53 consists of 53 convolutional layers. The original detector consists of 3 detection heads each responsible for detecting objects of various sizes. YOLOv3 takes n-channel

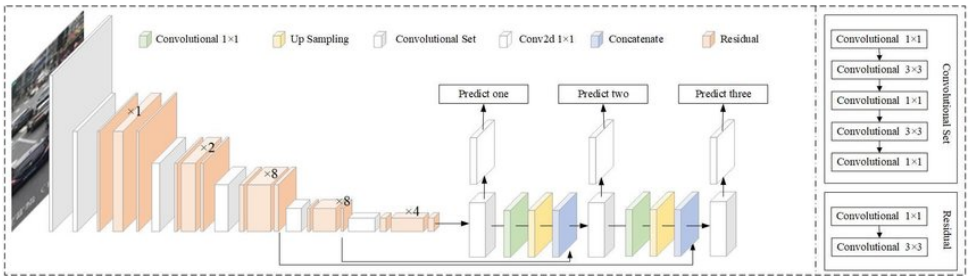


Figure 2.2: YOLOv3 network architecture

images as the input and outputs are as following:

- Offsets of the bounding box centers relative to the size of input image
- Scales of the bounding boxes relative to their respective anchors
- Objectness score, which dictates the confidence of object detection in the particular cell
- Class score, which dictates the confidence of object recognition in the particular cell for each of the classes used during training

The loss function used during the training was sum squared error loss or L_2 error described as follows:

$$L_2 = \sum_{i=1}^{n} (y_{true} - y_{predicted})^2 \tag{2.3}$$

where:

- n is batch size
- y_{true} are ground truth bounding box values
- $y_{predicted}$ are predicted bounding box values

Chapter 3

Sensors

3.1 Coordinate systems

In order to correctly label the data for training, a position of the second drone in relation to the camera mounted on the first one is required. The API call in AirSim returns a position in relation to its starting point. Therefore a transformation from the starting point of the second drone to the camera mounted on the first one is required. We can write this transformation as follows:

insert picture showing the positions of the drones

$$\mathbf{T} = \mathbf{T}_{d1}^{c}\mathbf{T}_{s1}^{d1}\mathbf{T}_{s2}^{s1} \tag{3.1}$$

where:

- $\mathbf{T}_{s2}^{s1}$ is transformation from the starting point of the second drone to the starting point of the first drone
- $\mathbf{T}_{s1}^{d1}$ is transformation from the starting point of the first drone to the body of the first drone
- $\mathbf{T}_{d1}^{c}$ is transformation from the body of the first drone to the cameras coordinate system

Transformation matrix $\mathbf{T}$ can generally be described as follows:

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{p} \\ \mathbf{0}^{T} & 1 \end{bmatrix} \tag{3.2}$$

where:

- $\mathbf{R}$ is a 3x3 rotation matrix
- $\mathbf{p}$ is a 3x1 translation column vector
- $\mathbf{0}^T$ is a 1x3 row vector of zeros

3.2 Camera Model

Picture of pinhole camera

For the creation of the bounding boxes used for training a transformation from coordinate system of the camera to the pixel values of the image needs to be defined. For this task a pinhole camera model is used. The transformation is then defined as follows:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} f\frac{x}{z} + c_x \\ f\frac{y}{z} + c_y \end{bmatrix} \tag{3.3}$$

where:

- x' and y' are pixel coordinate values on the image
- x,y,z are coordinate values of a point to be transformed
- f is focal length of the camera
- c_x,c_y are offsets on the image plane

Chapter 4

Dataset

The dataset for this work can be generated in two ways. The first is real-life drone shots mixed with point clouds from LiDAR mounted on top of a drone. The second is generating a dataset using a realistic virtual environment where a drone, camera and LiDAR are being emulated very close to their real-life counterparts. An advantage to this approach is that a great variety of environments can be chosen a lot of them often inaccessible otherwise (power plant, airport, snowy mountains out of season etc.). Therefore this approach will be chosen for the task.

4.1 Unreal Engine

Unreal Engine is a software tool used for creating realistic 3d environments, most often used as a video game engine. It is written in C++ and open-source supporting a variety of pre-built environments and assets. For this work three diferent environments will be used for the creation of the dataset:

citation https://www.unrealengine.com/en-US/features

- exact name. Snow
- exact name. Park

exact name. City centre

Pictures of the environments

Together

exact number of pictures

pictures and labels were generated using two drones. One drone was equipped with RGB camera and LiDAR sensor and was responsible for taking the pictures and pointclouds from LiDAR. The second one was used as a model for drone detection.

4.2 AirSim

airsim zdroj

Open-source plugin for Unreal Engine called AirSim was used for the generation of the dataset. It simulates realistic flight motions of drones as well as seven types of sensors , including RGB camera and LiDAR used for this task. AirSim supports both a C++ API as well as Python API, latter which was used for controlling the motion and capturing the dataset. Location of the second drone was generated through API call, which produces a location of the drone in global coordinate system of the map, which is later transformed to the local coordinates of the first drone carrying the LiDAR and RGB sensors.. The capturing drone traveled on each map on a 3d cube grid:

Transformacna matica?

Grafika kocky po ktorej lietal dron

Chapter 5

Training

For the training purposes 5320 samples were taken using AirSim simulator. Each sample consists of:

- 640x640 3-channel RGB image from the camera
- 640x640 1-channel sparse depth image from the LiDAR sensor
- Label file containing ground truth bounding boxes values

This dataset was split 90% training samples to 10% validation samples.

5.1 Sparse to Dense

First the data were trained using Sparse to Dense neural network to receive dense depth image. The training was done for 15 epochs using batch size of 8. The backbone was Resnet18 and decoder was set to Deconv3. The network was pretrained on Kitti dataset.

put images of the sparse2dense output

5.2 YOLOv3

After the first network trained multiple different options of input to the YOLOv3 networks were applied. These include:

- Concatenating the direct output of Sparse to dense neural network with the RGB images creating a RGBD image
- Manually altering the dense depth image values that were inaccurate (f.e. sky) a changing them to a high value
- Filtering the parts of the dense depth image that were very sparse in the original sparse depth image
- Using OpenCV library to directly impaint the sparse depth image
- Using only the RGB image skipping the depth entirely

YOLOv3 PyTorch implementation was used for training. Further modifications were required to be made. Theo original implementation of YOLOv3 supports 3-channel RGB images as inputs. For the sake of this work a 4-channel RGB input option using H5 file system was implemented. The dataset consisted of drones of various sizes ranging from very small (few pixels) to very large closeups. Therefore 5 detection heads were implemented instead of the original 3. This ensured much higher detection confidence of smaller far away drones as well as bigger more close ones. The following parameters were used for training all inputs:

- Number of epochs was set to 120
- Batch size was set to 128
- Size of the input images was 416x416xn, where $n \in \{3, 4\}$
- Learning rate was set to 0.001

add images

The sample outputs were as following:

Chapter 6

Results and Conclusion