AUTONOMOUS NANO DRONE PROJECT REPORT

May to August, 2020

Prepared By:
Vishal Sharma
Senior-year, B. Tech (Electrical Engineering)
Delhi Technological University, India

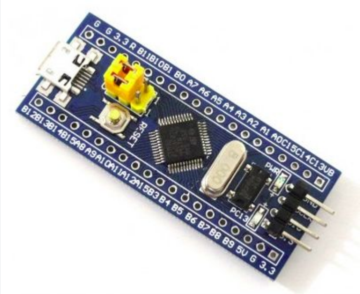
Under the Guidance of:
Prof. Marian Verhelst
Nimish Shah
Steven Colleman

INTRODUCTION

Key Features:

- Weighs < 28g
- Total Power Budget of < 10 W
- Performance: 6 fps @ 64 mW, 18 fps @ 272 mW
- **Complete Autonomy** i.e. no human operator, ad-hoc external signals, or remote base-station!

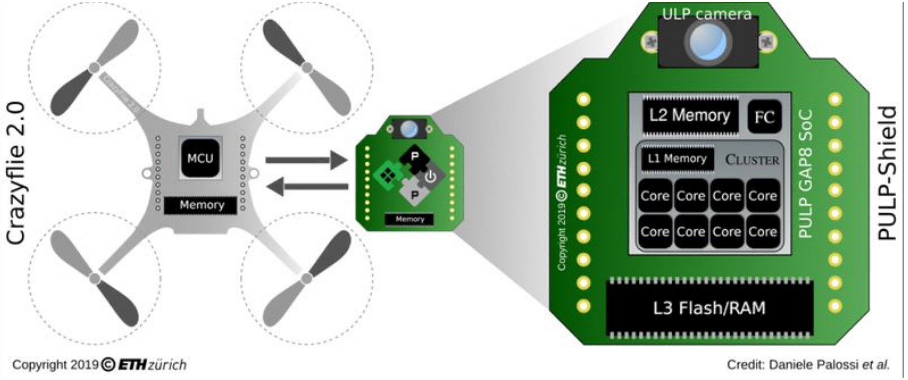
HARDWARE STACK



STM32 ‘Blue Pill’ acts as the Main MCO
Function: simple but real-time & critical tasks i.e.
- estimating current kinematic state
- actuation control
- motion prediction

GAP8 SoC is used to run the Visual Navigation Engine

Pulp Shield

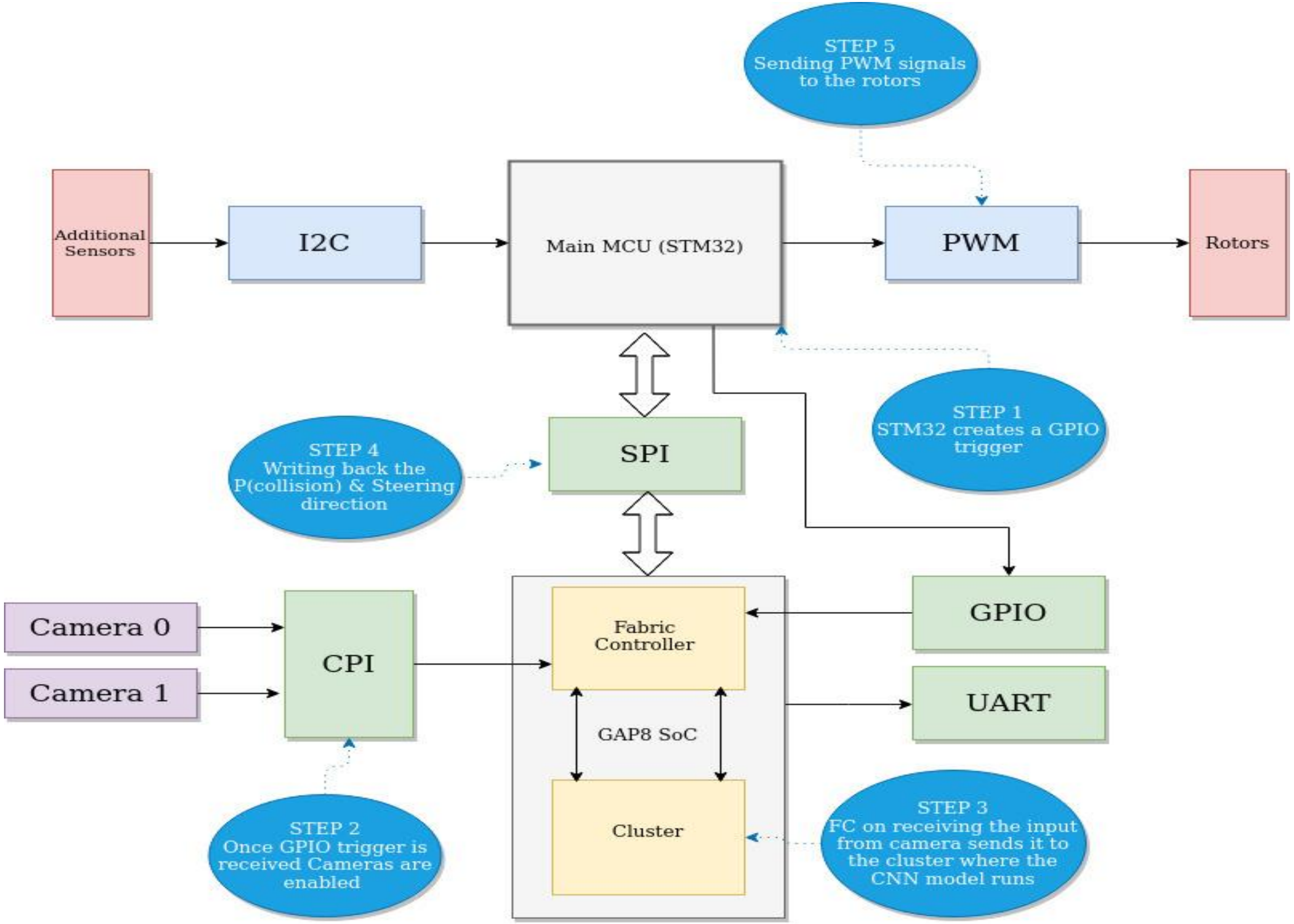


Peak Theoretical energy efficiency = 211 GOPS/W

Peripherals integrated:
- 2 SPI (1 master, 1 slave)
- GPIOs
- Bootup ROM
- JTAG interface for debugging and testing

The SoC features nine general purpose RISC-V-based cores organised in an on-chip microcontroller (1 core, called Fabric Ctrl) and a cluster accelerator of 8 cores.

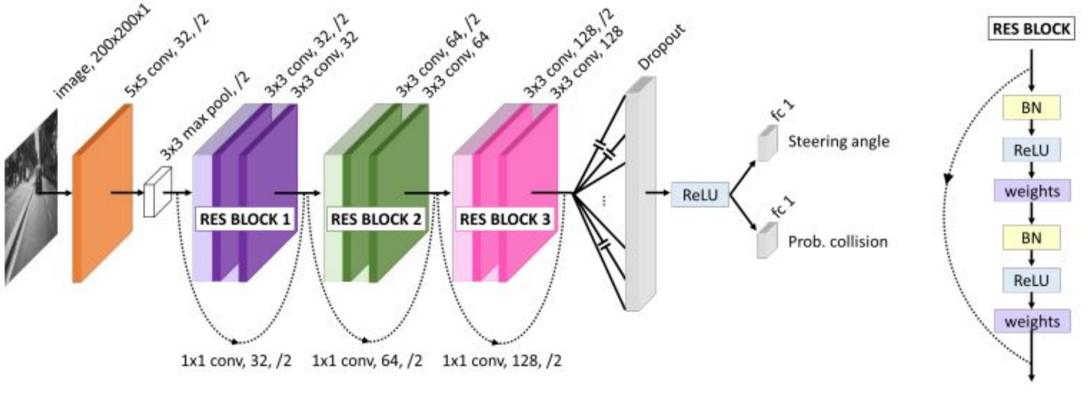
Working Flow



SOFTWARE STACK

The algorithmic heart of the autonomous navigation engine is based on DroNet, a Convolutional Neural Network (CNN) that was originally developed at the Robotic and Perception Group (RPG) of the University of Zürich.

DroNet



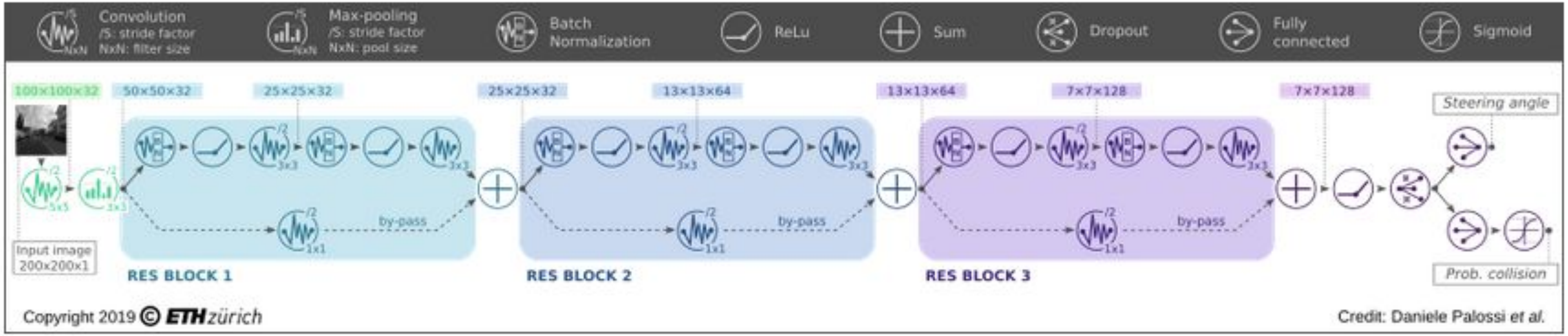
(a) (b)

- Topology based on ResNet
- 8 layers of residual network
- Converts an unprocessed image into:
  1. Estimation of probability of collision (used to determine target velocity)
  2. Desired steering direction using visual clues such as obstacles, white line on floor, etc.

- **Forward velocity:** vx;target[t] = α · vmax · (1 − Pcoll[t]) + (1 − α) · vx;target[t − 1]
- **Target Yaw Rate:** wyaw;target[t] = β · θsteer[t] (2) + (1 − β) · wyaw;target[t − 1]

1. Steering Prediction is a **regression problem,** i.e. the output is real and continuous.
Mean Squared Error (MCE) is used to train the steering predictions.

2. Collision Prediction is a **binary classification problem,** i.e. the output belong to a discrete set of values.
Binary Cross Entropy (BCE) is used to train collision predictions.



Optimization of DroNet for Embedded Deployment

Given the constraints imposed by limited resources, the navigation algorithm must execute at a frame rate required for satisfactory closed-loop control.

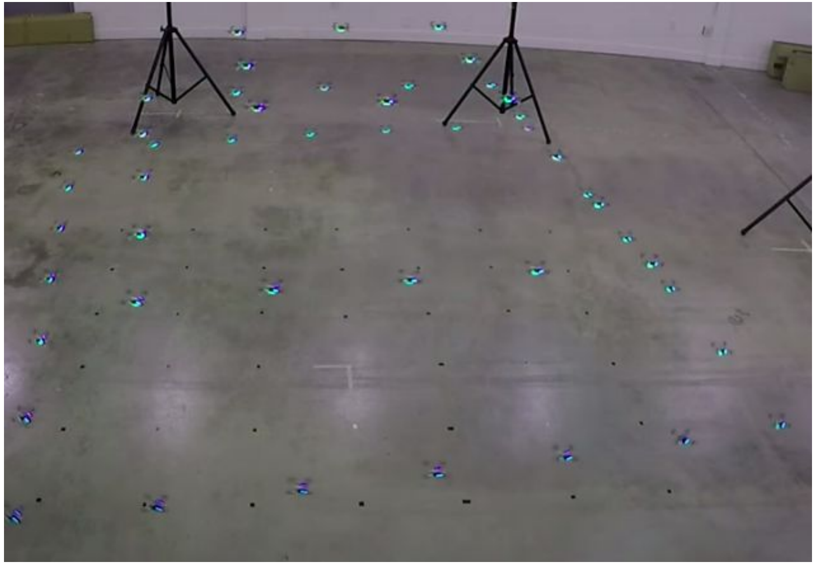
1. **Data fine-tuning & n/w quantization:** With lower precision in data representation & lower resolution image, results should remain similar to original algorithm.
- Grayscale QVGA-resolution
- CNN pooling layers reduced from 3*3 to 2*2

2. **Batch-Norm Folding:** We apply inverse folding on the bypass convolutional layer to counteract the folding.

Some Applications of Nano-Drones

1. SCAMP: The Flying, Perching, Climbing Robot

2. Crazyswarm: USC Nano-Drone Swarm Project

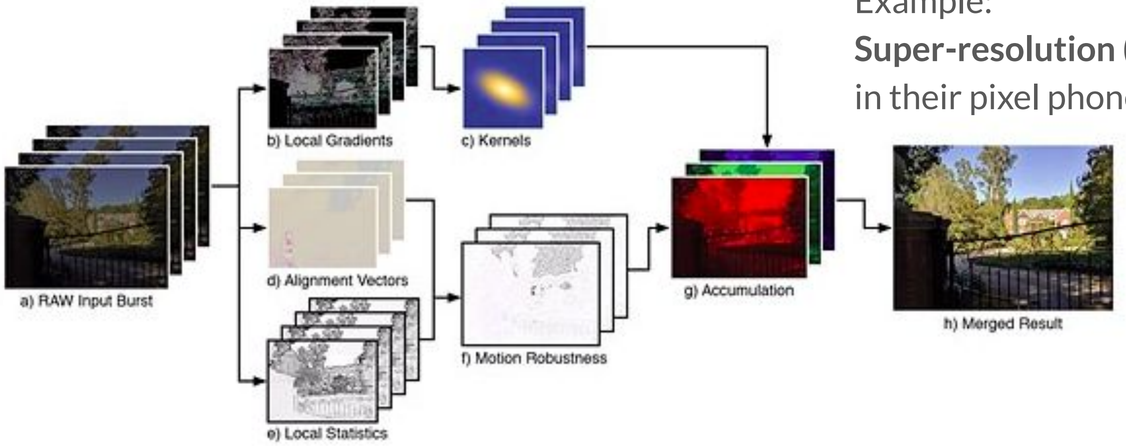


HOW CAN A MULTI-CAMERA SYSTEM IMPROVE THE ACCURACY?

1: RESOLUTION ENHANCEMENT

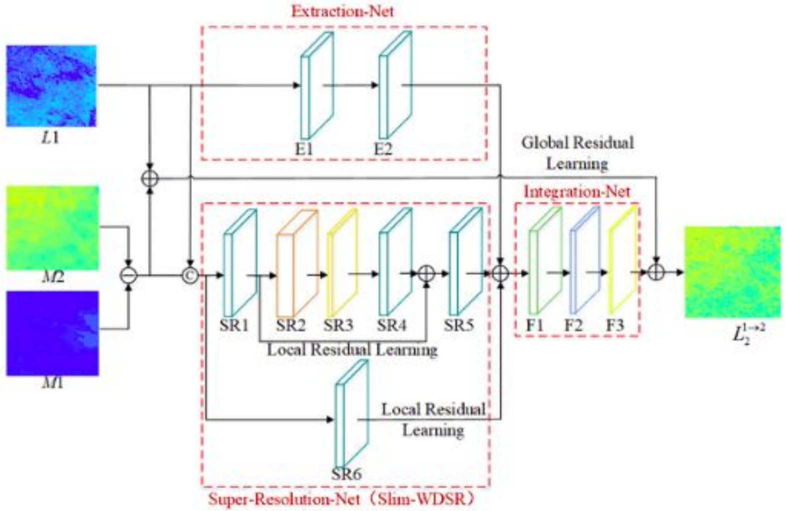
SR Reconstruction

Super-resolution (SR) consists of processing an image or a set of images in order to enhance the resolution of a single frame.



Example:
Super-resolution (SR) algorithm used by Google in their pixel phones.

Fusion SR using Multi-Camera Array



In fusion, super-resolution and high-resolution images are constructed from several observed low-resolution images, thereby increasing the high-frequency components and removing the degradations caused by the recording process of low-resolution imaging acquisition devices.

2: ASYNCHRONOUS FEED TO THE CNN MODEL

(near) Real-Time Predictions

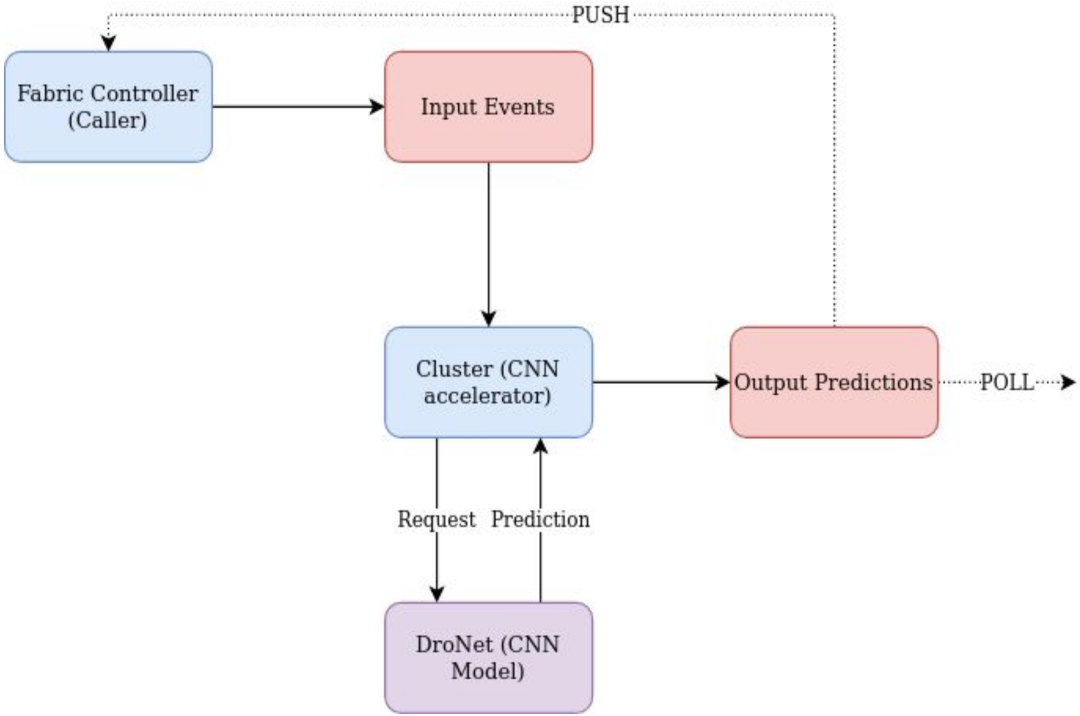
In real-time prediction, the model usually receives a single data point (i.e. image) from the caller (here, Fabric Controller), and is expected to provide a prediction for this data point in (near) real time.

Real-time predictions can be achieved using 2 ways:
1. **Synchronously:**
- request for prediction and the response (the prediction) are performed in sequence between the caller and the CNN model
- caller waits until it receives the prediction from the CNN service before performing the subsequent steps.
2. **Asynchronously:**
- predictions based on events streaming data
- predictions delivered to the caller independently of the request for prediction.

ASYNCHRONOUS SERVICE MODEL

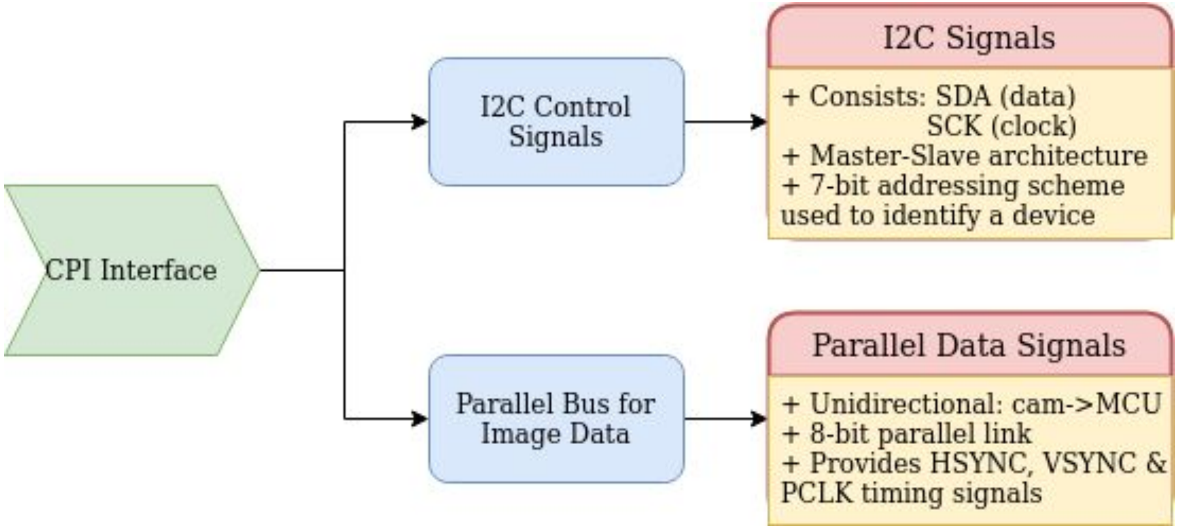
Asynchronous service predictions include:

1. **Push:** The model generates predictions and pushes them to the caller as a notification. For example triggering a response in case an obstacle is identified.
2. **Poll:** The model generates predictions and stores them in a low read-latency database. The caller periodically polls the database for available predictions. For example: determining velocity, altitude.



INTERFACING MULTI-CAMERA SYSTEM

Camera Parallel Interface (CPI)



$$\text{Theoretical Frame Rate} = \frac{\text{Maximum supported PCLK}}{\text{Horizontal size} * \text{Vertical size} * \frac{\text{Bytes}}{\text{pixel}}}$$

PRELIMINARY RESULTS

Based on CNN applications run on GVSOC

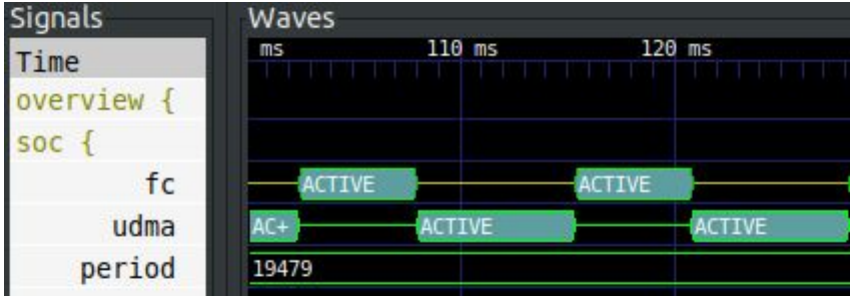
Canny Edge Detection

SYNCHRONOUS

Runtime:
(2 images, 1 cam each)
Total Time = 3.132s
Actual Run Time = 0.240s

ASYNCHRONOUS

Runtime:
(2 images, 1 cam each)
Total Time = 3.169s
Actual Run Time = 0.182s



Multi-Scale Pedestrian Detection

Runtime:
(1 image, 1 cam)
  Total Time = 10.223s
  Actual Run Time = 0.776s

For more information, please visit the github repository:
https://github.com/VishalSharma0309/nano_drone

THANK YOU FOR THE OPPORTUNITY!