# Project Portfolio*

Qian Qian
xeonqq@gmail.com

July 2, 2017

## Contents



*The projects are listed in most recent first order

# 1 Automatic takeoff and landing of the Songbird VTOL UAV

**Programming Language**: C++, Javascript
**Hardware**: Arduino Mega2560, mpu6050, pixhawk
**Techniques**: Attitude and Heading Reference System (AHRS), Sensor fusion, PID control, Real-time scheduler, Gazebo simulation
**Duration**: 1 year 8 month
**Company and position**: germandrones gmbh, senior system engineer
**Refer to**: Demo video

Between 2015.11 and 2017.06, I was working at germandrones gmbh, a startup building vertical take off and landing UAVs. The UAV is designed as a fixed-wing plane but with four tilting rotors, so it can takeoff/land vertically and still fly like a plane.

Figure 1: Songbird1400 from germandrones

My responsibilty was to develop an autopilot for the UAV which will allows it to takeoff, transition to mission and land automatically without human intervention. Worth to mention is that I was the only software engineer in this company, so I have taken great responsibilties in problem analysis, software design, simulation and flight test. For the autopilot software, I have designed the software archtecture, real-time scheduler, control algorithms, attitude estimation, etc. Through almost two years' effect, I have accomplished this task and converted the UAV from fully manual controlled to fully autonomous. Major tasks includes the design of:

- Quaternion based attitude estimation.
- IMU calibration algorithm.
- Real time scheduler.
- PID controller for attitude, altitude and positon control in copter mode.
- Softare achitecture design.
- Auto takeoff, transition and landing logic.
- Gazebo simulation interface.
- GCS communication interface.

## 2 Self-balance robot (Hobby Project)

**Programming Language**: C++
**Hardware**: Arduino, IMU, bluetooth
**Techniques**: PID control, Kalman filter for IMU, PWM, openSCAD 3D modeling
**Duration**: 1 month
**Refer to**: Demo video

I always wanted to build a Segway-like self-balance robot, so I started doing this hobby project after my thesis.

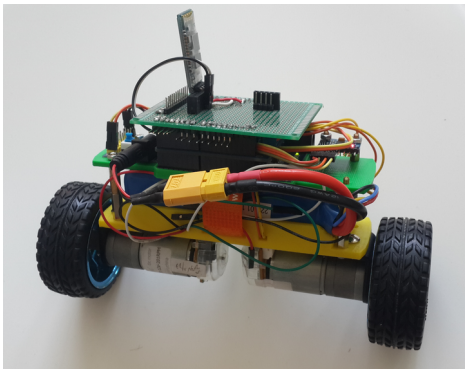

Figure 2: Self-made balance robot

I designed the structure of the balance robot using openSCAD and printed it using a 3D-printer. The stucture is to hold the motors as well as the Arduino and the IMU. I used a Kalman filter to calculate the angle of the robot according to the gyro and accelerometer input. The angle is then used as the input of a PID controller to control the speed of the motors. The bluetooth module enables me to tune the PID parameters easily from my phone. For detailed code and list of hardwares, please visit my Github.

## 3 Localization of Humanoid robot NAO in RoboCup SPL (Master Thesis)

**Programming Language**: C++, python, cython
**Hardware**: NAO robot V4 (Intel Atom@1.6GHz, 1GB RAM)
**Techniques**: Kalman filter, Particle filter, feature detection and matching, robotics programming, python optimization
**Duration**: 1 year
**Refer to**: my master thesis

DAI-Labor
Distributed Artificial Intelligence Laboratory

This is the *master thesis* project I have done under DAInamite team in TU Berlin. In this project I have investigated and implemented a multi-hypotheses Kalman filter based localization for NAO robot in RoboCup.

The Kalman filter based algorithm takes motion and vision result as input and performs odometry update followed by a measurement update. The update is performed for each Extended Kalman Filter (EKF) sample.

In the end, based on how good the feature landmarks are matched with the correspondence, the best sample is chosen and its state's value is used as the robot pose.

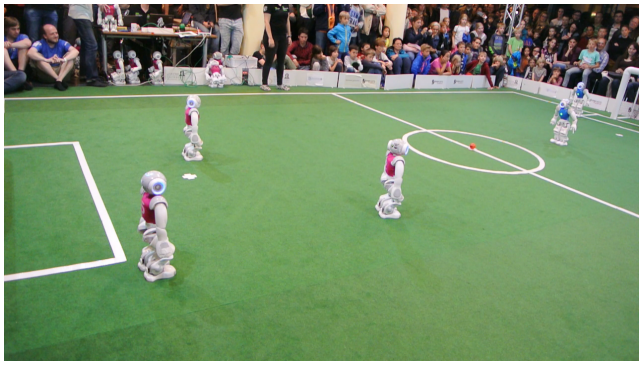

Figure 3: Algorithm tested in real game at Frankfurt

The main challenges lies in this project are that the robot has only partial view of the field and the landmarks are ambiguous, so hard to find the correspondence. To tackle these problems, method like nearest neighbor feature matching and L, T, X junction detection has been deployed. Particle filter technique like resampling by landmark is also used to help recovery kidnapped robot.

# 4 Hand Detection using OpenCV and Robot Arm Control

**Programming Language**: C++
**Hardware**: Freescale 8-core embedded platform
**Techniques**: OpenCV image processing, background substraction, convex hull, mean-shift, UDP
**Duration**: 4 months
**Refer to**: Demo video

Fraunhofer FOKUS

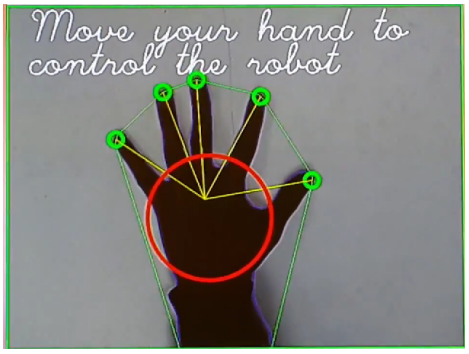


Figure 4: OpenCV detection result

During my *internship* at Fraunhofer Fokus, I developed an algorithm which can detect and track human hands using a down-facing camera. It is shown as part of Fraunhofer's demonstrations in Internationale Luft- und Raumfahrtaustellung (ILA), Berlin.

The algorithm works by first using background substraction to extract the contour of the hand. Then I find the convex hull and convex defects of the

hand contour. The number of convex defects are to determine whether the hand is a palm or a fist. Meanshift method is used to track the hand motion.

In the end, the hand position in image plane is then sent out by UDP to control the movement of a robot arm. The robot arm can also grab items when the detected hand turn from palm to fist.

# 5 Virtual drum player using wireless sensor nodes equipped with accelerometer

**Programming Language**: nesC
**Hardware**: Telosb
**Techniques**: TinyOS, digital signal processing and filtering, broadcast, UDP communication, IoT
**Duration**: 1 month
**Refer to**: Demo video

Mini project for the Wireless Sensor Network course. I designed a virtual drum player, in which user can play drum by holding a sensor node, and the end PC will play the sound of the drum according to the movement of the user in real time. There is another node in the system which can record and reply the drum song.


Figure 5: Telosb hardware

To detect specific motion of the user, the accelerometer data is first filtered through an average filter. Then the gesture of drumming is detected by analysing the accelerometer data signal pattern.The detected result along with the timestamp are broadcasted to the PC as well as the intermediate node. So the drum pattern could be played in real-time as well as be able to re-play later.

# 6 Building, stabilization and vision control of a Quadcopter

In this project we managed to build and assemble a flying quadcopter from scatch. In the setup, an STM32F3Discovery board is used for lower-level control and stablization of the quadcopter; a Raspberry Pi is used for higher level control using camera vision.

**Programming Language**: C, C++
**Hardware**: STM32F3Discovery, Raspberry Pi and RPi camera
**Techniques**: Quadcopter assemble, ArUco marker detection, OpenCV face detection, PID tuning, FreeRTOS
**Duration**: 5 months

Figure 6: Self-made quadcopter

To achieve real-time control and rich functionality, FreeRTOS based OpenPilot firmware is deployed on the STM32 to fuse the sensor data and control the motors. In order to fly stably, the PID parameters of the control loop are carefully tuned for our quadcopter.

The vision processing part is implemented on the more computationally powerful Raspberry Pi. Two algorithms are ported on it, namely OpenCV face tracking algorithm and ArUco marker detection algorithm. The end results are sent to openPilot to direcly control the motion of the quadcopter.

# 7 JPEG decoding using three-core multiprocessor

**Programming Language**: C
**Hardware**: three core micro-blazer microprocessor
**Techniques**: JPEG decoding pipeline, multi-core programming, synchronizations between cores
**Duration**: 5 months

Ported a JPEG decoder to a three core micro-blazer microprocessor, use data parallelism and function parallelism to speedup the program. After the parallelization, the code can runs up-to 3 times faster.

# 8 GPU optimization for Viola-Jones face detection

**Programming Language**: C, Cuda
**Hardware**: Nvidia Tesla GPU
**Techniques**: GPU programming
**Duration**: 2 months

Technische Universiteit Eindhoven University of Technology

Ported a x86 C-code of Viola-Jones face detection to the GPU, use the parallel computing capability of the GPU to make it run as fast as possible. After porting to GPU, the optimized running time of the face detection code *reducd from 2.2s to 130ms*.

# 9 Android Phone Based Robot with Internet Remote Control (Bachelor Thesis)

**Programming Language**: C, Java
**Hardware**: STM32 micro-controller, Samsung Galaxy II, PC
**Techniques**: Android development, digital audio signal processing (FFT), DMA, servo and motor control, mechanical building of a robot car
**Duration**: 6 months
**Refer to**: Demo video

This is the *bachelor thesis* I have done in Nanjing University. I realized an innovative way of controlling a robot car using the combination of an Android phone and a PC. Through the camera of the phone, the manipulator can have the first person view of the robot *i.e.*monitor the objects in front of the robot car through the PC screen and wirelessly control the car to move using the PC keyboard.

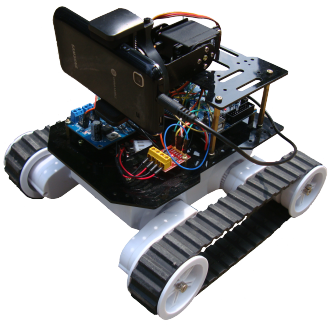

Figure 7: The self-made robot with Android phone equipped

The communication between the Android phone and the micro-controller is using DTMF audio signal. After decoding using FFT, the control signals will be used to control the speed of the robot motors and the angle of the servo which

holds the phone. Moreover, the image data of the android phone is sent through the network to the PC terminal and the PC terminal can send commands to the phone as well.