BioMAV: the present-inspecting robot

Robin Wellner and Roland Meertens

Februari 1, 2012

1 Introduction

What is BioMAV

The robot used in the BioMAV is the Parrot AR.Drone 1.0. It is able to communicate via USB and Wi-Fi. The drone has two cameras, one ground-facing and one forward-facing. However, the driver sofware we used was only able to access the image data from the forward-facing camera.

2 Background

IMAV

Previous BioMAV

This year's biological inspiration

ROS (for Robot Operating System) is an open-source framework for robot controller software, especially intended for re-use of libraries (called "nodes") between different projects and for different types of robots. This allowed us to make use of existing solutions for blob detection and inter-node communication.

3 Motivation

Motivation on working on BioMAV

Motivation on the current implementation?

4 Goal

4.1 Initial goals

The initial goals of this project were:

1. To establish a platform, based on the old BioMAV project, on which future BioMAV projects can build.
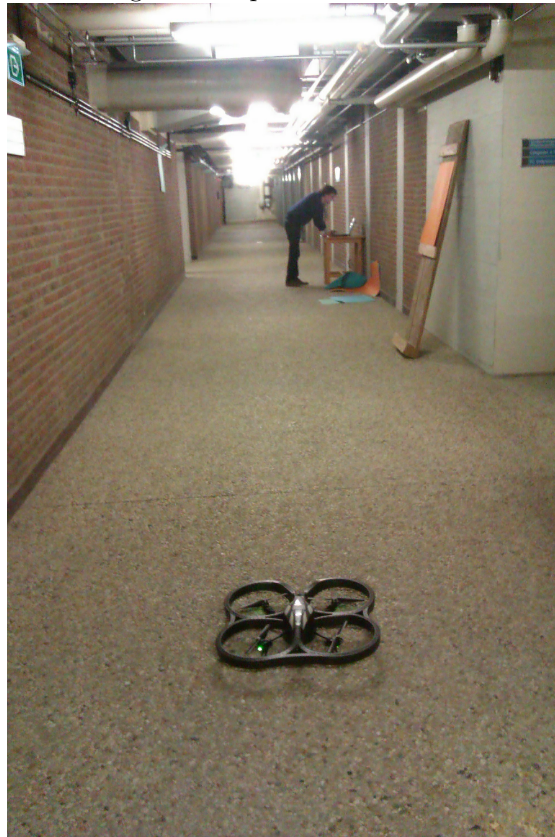
2. To let the drone perform a task autonomously. Which task that would be was to be determined.

3. To demonstrate the drone performing that task.

4.2   Final goal

Our final goal was to have the drone perform the packet-inspection task and light-following task.

5   Pseudocode

Figure 1: A picture of a drone.

In this section several of the implemented algorithms are discussed. Every subsection starts with the code in pseudocode and ends with an explanation of the algorithm.

5.1 Detection of objects

```
targets = getAllBlobsThatMatchOurTarget
coolDownHeatMap
activateHeatMap(targets)
if dot in heatmap is high enough
if sum of heatmap is high enough
inspect present
else
go toward average of activation
```

Our program first determines that targets we have by aquiring all blobs that match our target. After this is done our heatmap is cooled down using a xxxxxxx function. The heatmap then receives activation again by raising the value of all the locations of the detected blobs. When a dot in our heatmap is activated enough it is assumed that this is because our target is at this position. When the total sum of the heatmap is activated enough we "inspect" the present (see section xxxxxx). When there is not enough activation the drone flies towards the average of the activation (see section xxxxxxx)

5.2 Corridor following

```
targets = getAllBlobsThatMatchOurTarget
myTarget = getTargetLowestOnCamera
if isVeryMuchToTheLeft(myTarget)
flyVeryHardToTheleft
else if isVeryMuchToTheRight(myTarget)
flyVeryHardToTheRight
else if isSlightlyToTheLeft(myTarget)
flySlowlyToTheLeft
else if isSlightlyToTheRight(myTarget)
flySlowlyToTheRight
else
flyForward
```

We found out that by always flying towards the target that was the "lowest" on the camera (in the y-space) our drone could smoothly follow the lights in the hallway. By implementing this function we were able to get a corridor-following behaviour. Our program first gets all blobs that are matching our target colour (the color of the lights). The blob with the lowest y value is chosen as our primary target. When this blob is very much to the left or right the drone will turn with a great speed in this direction. When this blob is only slightly to the left or right the drone will turn slowly in this direction. Otherwise the drone will fly forwards this blob.

5.3 Flying towards objects

```
direction = XvalueOfAverageOfActivation
turnTowards(direction)
```

```
flyForward
```

Flying towards a target was first solved by only flying towards a specific blob. As soon as we started working with objects that were not detected as a "whole" blob and we had an activation value we thought of another function. In the new fly-towards function the X value of the average of the activation is taken. This value is devided by the length of the camera image and mapped to our drone in such a way that we aquire a directional value. We let the drone "spin" in this direction and fly forwards.

5.4   Present inspection

```
if sum(activation) > threshold
if getNewTarget(currentTarget)
currentTarget = getNewTarget(currentTarget)
else
startLanding
```

For the "inspection" of the presents there is not a really special behaviour. As soon as the total activation in the heatmap for the presents exceeds the set threshold the next target is chosen. When there is no new target specified in our goals-array the drone will land.

6   Project history and division of labour

The first BioMAV project started in xxxxxx with an enthousiastic group of people. After the initial project was a large succes and obtained the third price in the imav competition on xxxxxinsert datexxxx a second project was started. For this second project one had to write a sollicitation letter and send this to the teachers. Initially the group consisted of xxxxxx people (Youetta, Rick, Jurriaan, Roland, Robin and xxxxx). These people were devided into two teams, the "image processing" team and the "drone control" group. The task of the "image processing" team was defined as getting useful features from the image data, which would be given to the "drone control" group. The "drone control" group, consisting of Roland and Youetta, was supposed to use these features to control the drone.

Initially both groups were both trying to decode the largely undocumented code made by the previous team. One of the wishes of the group was a code implementation using ROS. This was another software part that both teams had to figure out how to use. Around june the first implementation of a control program using ROS was finished by Roland. This program consisted of a server extension of the original software written by the first BioMAV team. The ROS component connected to a socket that listened to calls coming from the ROS component of the software. Using a simple syntax to parse fly-commands it was possible to fly with the ARDrone using ROS.

Unfortunately, other team members turned out to be not that enthousiastic about the project anymore. Some stopped pretty early and told the whole group about it, others stopped very late without telling anybody. This all led to the fact that the project slowed down and was not really coordinated anymore.

During a meeting in juli it was agreed that this setup was not usefull enough and that another implementation had to be found that only used ROS. Unfortunately this meant that Robin and

Roland had to start over again, throwing away all old code (including all code used by BioMAV 1). After we threw away all old code we started with searching this "easy usable" library that should have been available on the internet. Unfortunately, this did cost us more time than we expected. Every library we found was eiter not usable, not downloadable (findable) anymore or not uploaded yet (but sounded very promising). It was only after a lot of hours searching that one of the librarys suddenly was pullable again.

After we were able to control the drone using this library we started looking at how we would be able to recognise objects. We started with looking at the recognition of colours, which thanks to a remark by Bas Bootsma turned out to be relatively simple. His idea was to recognize blobs using the xxxxxx library. Starting this library turned out to be pretty easy, tweaking it turned out to cost more time than expected.

Although recognizing blobs worked for recognizing objects it also yielded a lot of "false-positives", objects that did not have to be recognized. As an alternative we started to look at alternatives, including using AR markers and image-recognition. Unfortunately it proved to be too much of a challenge to use the algorithms available on ROS with the image data that we aquired.

Unfortunately this did not solve our problem about recognizing other objects as target objects. The first solution we came up with was searching for the most dull location, this turned out to be the corridor under the Spinoza building. Although this hallway did not contain a lot of colours and distractions from our real target there were still a lot of small blobs recognised by our blob-detector.

Our first solution was to simply ignore small blobs and try to use colours that were more different than those colours found in the hallway. This did lead to a more stable blob recognition, but still was not perfect.

The small blobs scattered a lot while the blob of our real object was pretty stable. Eventually we came up with the idea to use a "heat-map", where all detected blobs would yield an activation. Over the course of several frames this activation would decay leading to the fact that only locations where for longer times blobs were detected would be activated.

This led to a very smooth detection of the target object, as is visible in figure xxxxx.

Figure 2: A picture of a drone.
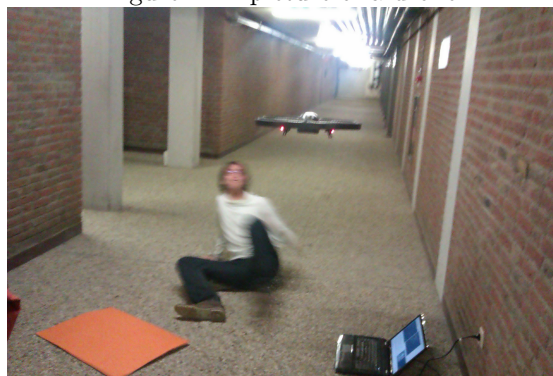
7   Reflection

Although the bioMAV project has been a rough project this year the final result is something that looks very cool. To build a program that lets a small UAV autonomously follow bread-containers, corridors and inspect packages is something to be proud of.

Of course, one starts to think about how much more awesome the initial plans of the complete team were. Unfortunately, since more than half of the team decided to stop early, these plans were never finished and kept for the next team.

Another thing that was very unfortunate was the fact that the whole old BioMAV project was thrown away in the end. Although a lot of time was spend in learning how to interpret the old code and extending the project to work with ROS all these work was thrown away. Another thing that was not used after a lot of time went into researching this were the AR markers, unfortunately these proved to be impossible for us to implement.

Of course we are all very contend with the final result, and are very proud about what we eventually reached in such a short time.

Figure 3: A picture of a drone.

References