Introduction to Vision and Robotics
Assessed Practical 1: Robot Tracking

Ran Guan Thea Koutsoukis

4pm March 7, 2013

Contents

1 Introduction

Overview of main ideas used in our approach

2 Methods

Which methods were used?

2.1 Method 1

Give a functional outline of how this method was implemented and the structure of the code. Explain how each part of it is meant to work. Where suitable, justify your decisions, e.g. why you used one method rather than another, what you tried that didn't work as expected

2.2 Method 2

etc.

3 Results

You should provide some actual data, from repeated trials (with the camera or robots in different positions) on how well your algorithm performs, as described previously. Show an example of your results for each stage of the detection. Well documented failure will get more marks than unsupported claims of success (well-documented success would be even better!).

4 Discussion

Assess the success of your program with regard to the reported results, and explain any limitations, problems or improvements you would make.

5 Code

euclidianDist2D

```
% Finds the euclidean distance between points (x1,y1) and (x2,y2).

function distance = euclideanDist2D(x1,y1,x2,y2)

distance = sqrt((x1—x2)^2 + (y1—y2)^2);

end
```

euclidianDist3D

```
% Finds the normalised distance between coloured points (r1,g1,b1) and (r2,g2,b2).
% Euclidean distance is not suitable, as each colour has a different
% behavior. The values taken by the red pixel are
% every feature may not have similar behaviors. For example if the values taken by the first feature
% are very concentrated around 0, and $ \theta_2$ takes uniform values on an interval, then a big di
% and $ \theta_1^{(q)}$ is much more significant than the same big difference between $ \theta_2^{(i
% Euclidean distance does not take into account this possible asymmetry.

function distance = euclideanDist3D(r1,g1,b1,r2,g2,b2)

a = 2.2;
b = 1.00;
c = 0.75;

dist=sqrt(((x1—x2)/a)^2 + ((y1—y2)/b)^2 + ((z1—z2)/c)^2);

end
```