UG3 Introduction to Vision and Robotics
Vision Assignment

Clemens Wolff, Toms Bergmanis

March 7th, 2013

1 Introduction

The goal of this report is to propose an algorithm for object tracking. Object tracking, in general, is a challenging problem. Difculties in tracking objects can arise due to abrupt object motion or changing appearance patterns of both the object and its surroundings and other factors. Tracking is usually performed in the context of higher-level applications that require the location and/or shape of the object in every frame. Typically, assumptions are made to constrain the tracking problem in the context of a particular application[1].

2 Methods

Several simplifying assumptions were made to constrain the tracking problem. It was assumed that:

1. objects to be tracked will be puck-like robots and that they will each appear in different shades of red or blue, or green.
2. camera will be set up in an angle not less than 45% with respect to the plane it is observing.
3. on the top of the robot will be a triangle indicating it's direction and that it will be in darker colour than the colour indicating the robot.
4. background of the image will be in colour other than any of the colours of the robots.

2.1 Detection of Robots

Input

I, a three channel image of dimensions $m \times n$ in the RGB colorspace.

Output

M, a $m \times n \times 3$ binary matrix where for each pixel Pij of I, it holds that:
$M(i, j, 1) = 1 \leftrightarrow P_{ij}$ belongs to the red robot,
$M(i, j, 2) = 1 \leftrightarrow P_{ij}$ belongs to the green robot,
$M(i, j, 3) = 1 \leftrightarrow P_{ij}$ belongs to the blue robot.

Algorithm

1. Apply approximate RGB-normalisation to I, giving I_n:
   - For each pixel in I_n, calculate the sum S_{rgb} of the red, green, and blue values of that pixel.
   - If $S_{rgb} \neq 0$ (the pixel is not absolute black), set each of the pixel's red, green, and blue values to that value divided by S_{rgb}.
2. Calculate μ_r, μ_g, μ_b and $\sigma_r, \sigma_g, \sigma_b$, the means and standard deviations of the values in the three channels of I_n.
3. Assign each pixel P_{ij} in I to one of the robots or to the background:
   - Normalise P's red, green, and blue values, giving P_n.
   - Calculate the probabilities p_r, p_g, p_b that P_n was generated by the gaussian distributions $\mathcal{N}_r = (\mu_r, \sigma_r), \mathcal{N}_g = (\mu_g, \sigma_g), \mathcal{N}_b = (\mu_b, \sigma_b)$.
   - Calculate P's hue value h.
   - If h is whithin a certain range defined as red and p_r is sufficiently small, set $M(i, j, 1) = 1$ (similarly for ranges defined as green/blue and p_g/p_b. If none of these conditions are met, set $M(i, j, 1) = M(i, j, 2) = M(i, j, 3) = 0$.
4. Remove noise from each channel in M:
   - Set pixels to zero if they have fewer neighbours with value one than they have adjacent pixels with value zero.
   - Set zero-valued pixels to one if they have two one-valued horizontal or vertical neighbours.
5. Remove components that are distant from the main concentration of mass in each channel in M:
   - Compute the center of mass C of the channel.
   - Compute $c_1, c_2, \ldots$, the centers of mass of each connected component in the channel.
   - Compute the mean distance δ of the c_k to C.
   - Set $M(i, j) = 0$ for all the pixels (i, j) in the components k that satisfy $c_k > \tau\delta$ for some fixed threshold τ.
6. Exploit the fact that all robots have similar sizes by setting every channel in M to all-zeros if the number of pixels set in that channel is smaller than the number of pixels set in the most populated channel by some margin.

Figure 1 shows a visualisation of the output matrix M.

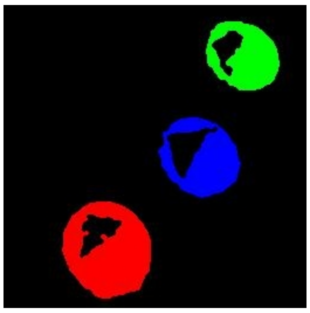


Figure 1: Result of Robot detection

2.2 Finding Robot Directions

Input

I, a three channel image of dimensions $m \times n$ in the RGB colorspace.

Output

$\Lambda = \{(c_r^m, c_r^t), (c_g^m, c_g^t), (c_b^m, c_b^t)\}$, a set where c_r^m is the center of mass of the red robot and c_r^t is the point towards which the robot is facing (similarly for the green and blue robots).

Algorithm

1. Get a matrix of robot masks M using the algorithm in Section 2.1. Let M_i be the ith channel of M i.e. the set of points $\{M(a, b, i) | 1 \leq a \leq m, 1 \leq b \leq n\}$. Apply the remainder of the algorithm to each channel ξ in M.

2. Calculate the convex hull H of the points in the channel and create the set of pixels of I that are inside H: $P = \{p_{ij} | p_{ij} \in \xi \wedge M(i, j, \xi) = 1\}$

3. Calculate μ, the average rgb-value over P. Generate $\Pi = \{p | p \in P \wedge rgbvalue(p) < \mu\}$, the set of pixels in P that have a below-average rgb value.

4. The black triangles on the robots are the pixels in Π. Get rid of them by setting the relevant indices in M to zero.
   Recompute the convex hull of M.

5. Repeat the previous step and remember the pixels in Π.
   This reduces noise in M by giving a tighter estimate on the robot's pixels when the triangles were under-detected by the algorithm in Section 2.1.
   Figure 2 shows the result of this step - a notable improvement in clarity of the triangles compared to Figure 1.

6. Update Λ: c_ξ^m is the center of mass of M_ξ, c_ξ^t is the center of mass of Π.
   A line from c_ξ^m to c_ξ^t indicates the direction of the robot.

Figure 3 shows a visualisation of the output set Λ.

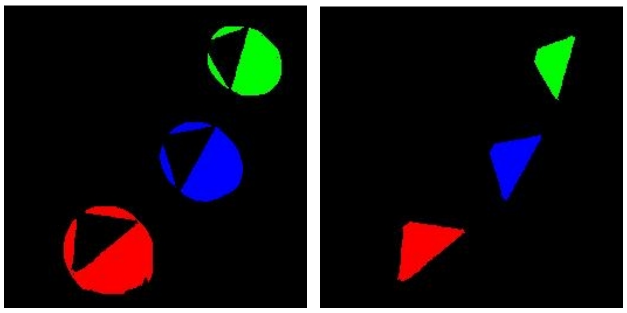

(a) After Removing Triangles (b) Removed Triangles

Figure 2: Triangle Detection via Local Thresholding

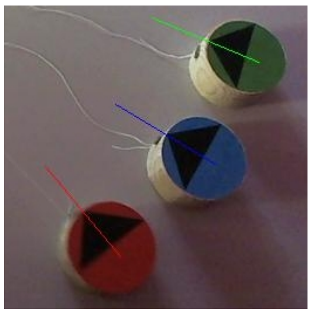


Figure 3: Detected Directions

2.3 Tracking Robots Over a Sequences of Frames

Input

$\Upsilon = \{I_1, I_2, \cdots\}$, a sequence where each of the I_i is a three channel image of dimensions $m \times n$ in the RGB colorspace.

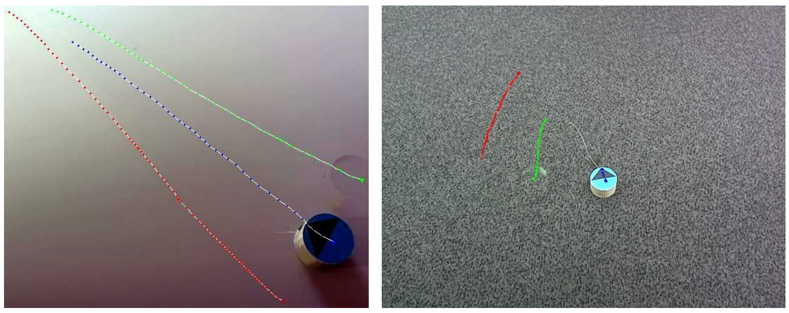
Output

Ω, a visualisation of the robot positions over Υ.

Algorithm

1. Use a median-filter to generate a background Ω from Υ.
   For each $1 \leq i \leq m, 1 \leq j \leq n$:
   - Create $\omega_{ij} = \{I_k(i,j) | I_k \in \Upsilon\}$, the set of the colors of the pixels at location (i,j) of all the images in Υ.
   - Set $\Omega(i,j) = median(\omega_{ij})$.
2. For each $I_i \in \Upsilon$:
   - Use the algorithm in Section 2.2 to get the set Λ. Let $\lambda = \{c | (c, \_) \in \Lambda\}$.
   - Overlay Ω with a line from each element in λ_{i-1} to the corresponding element in λ_i, thus linking the centroids from image I_{i-1} to the centroids in image I_i.

Figure 4 shows a visualisation of the resulting track.



(a) Dataset 1 (b) Dataset 2

Figure 4: Output of Tracing Algorithm

3 Results

This section evaluates and visualises the performance of the three algorithms presented in Sections 2.1, 2.2, and 2.3. Table 1 describes the properties of the datasets used for this evaluation.

#	Background	Robot Size	Robot Color	Illumination
1	uniform, gray	large	saturated, dark	uniform, red hue
2	noisy, gray	small	faded, blue robot is cyan	histograms are bell-shaped
3	patterned, brown	large	saturated	daylight only
4	patterned, brown	large	saturated	daylight and artificial light

Table 1: Properties of evaluation datasets

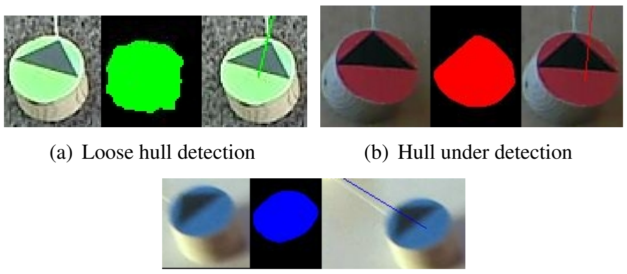
3.1 Detection of Robots

The algorithm described in Section 2.1, worked perfectly on datasets 1 and 2.
Evaluation on the third dataset led to the worst performance over all datasets, with ∼60% of the occurences of the blue robot being undetected and ∼40% of the occurences of the green robot being under-detected (leading to bad direction detection).
The performance on the fourth dataset was interesting: the red robot was under- detected in ∼45% of the cases (with the blue and green robots being found just fine) - while in the other datasets the red robot was usually detected with the highest confidence. Over all four datasets, about 10% of the robot instances were badly detected.

3.2 Detection of Directions

Performance of detection of the directions was heavily dependent on the performance of the detection of the robots. In case of precisely detected robot detected direction perfectly matched the actual direction. In case of loose detection - detection where some addition non-robot region is misleadingly detected as a robot - detected direction perfectly matched the actual direction due to algorithms ability to filter noisy detections.

In case of under-detection - detection where some parts of the image representing the robot where omitted - detected direction was skewed on the side opposite (from the axis matching the actual robot's direction) to the misdirected fragment of the robot. Error was proportional to the error of under-detected area of the robot.


(a) Loose hull detection (b) Hull under detection

(c) Good hull detection

Figure 5: Direction detections for convex hulls of different qualities

3.3 Tracking of the robots

Section 2.3's algorithm to track robots over consecutive frames is trivial - a mere visualisation of half of the results of the robot-direction- detection algorithm presented in Section 2.2. The tracking algorithm's performance is therefore directly related to the performance of the robot-direction-detection algorithm and the same observations as in Section 3.2 apply: generally speaking, the algorithm performed well.
Figure 4 begets one additional observation related to the evaluation of the robot-tracking algorithm: both datasets considered in this report exhibit the property that one of the objects of interest does not move much for most of the frames. This entails that generating a background from data employing a simple frame-difference base approach (such as the median- filter used in Section 2.3) to perform background subtraction is bound to fail as one of the objects of interest will be considered a part of the background due to being mostly stationary. This is unfortunate since pre-processing the datasets with background subtraction would increase the accuracy of Section 2.1's algorithm by reducing noise and increasing resolution in the image.

4 Discussion

Overall performance of the algorithm can be evaluated as good. It operates under little or no simplifying assumptions. Robot detection could be improved [WRITE PLEASE HOW] Another way how to improve the detection of the robots could be utilisation of second order spatial statistics of the robot images. Using such approach might give good results combined with our current approach. Direction detection could be improved by making it less dependent on detection of the robots. It could be achieved by performing local search near the regions detected as robots. Such approach would improve accuracy of the directions detected, however it would make algorithm much slower. Another possible direction of development could be shape analysis - currently algorithms utilise only image's colour statistics due to limited information about the scale of the images and the possible placements of the camera.

5 Code

the new Matlab code that you developed for this assignment. Do not include code that you downloaded from the course web pages. Any other code that you downloaded should be recorded in the report, but does not need to be included in the appendix.

References

[1] Yilmaz, A., Javed, O., and Shah, M., “Object Tracking: A Survey”, ACM Comput. Surv. 38, 4, Article 13, 2006.