Advanced Vision Assignment # 1

Student Numbers s1107496 & s1119520

February 12, 2015

1 Introduction

In this assignment, we sought to track the motion of various balls over the course of a given video. To facilitate this, we were supplied with a representative background frame of the environment, as well as ground truth positions for each of the balls which could be used to evaluate our program.

2 Methods

2.1 Frame processing pipeline

To detect the balls within a frame, the frame went through a number of processing steps, which ultimately resulted in the detection of a number of connected components. These steps are described below.

2.1.1 Background subtraction

The function `background_sub` was used to subtract the supplied background from the current frame, in order to isolate changed regions which potentially might be classified as balls. To achieve this, both the R channels of the RGB versions of the background and current frame and the S channels of the HSV versions of the background and current frame were isolated. For each of these channels, absolute differences between the established background and current frame values were calculated per-pixel, then thresholded using hand-optimized parameters. If a pixel significantly differed in either the R or S channels as per these thresholds, it was considered to be a non-background pixel. Finally, the resultant binary mask had the `bwmorph` "`erode`" and "`close`" functions applied to it, as well as `medfilt2`, in order to clean up stray pixels in the mask.

2.1.2 Separation of balls sharing a connected component

In some instances, after background subtraction multiple balls may be located within the same connected component within our generated binary mask. This might result from a number of circumstances, including our background subtraction method being too sensitive, the motion blur of multiple balls overlapping, or one ball occluding another. In any case, we wish in these instances to separate the balls into multiple distinct connected components, for later use within our tracking system.

To perform this step, we use the function `separate_balls`, passing our background mask and the current frame. For each connected component within our background mask,

we check to see whether its eccentricity is below a certain threshold. If the connected component has low eccentricity according to this threshold, we consider the connected component sufficiently ball-like such that we do not need to perform any separation. Following this, we apply the background mask to our current (RGB) frame, keying out the background and leaving only (potential) foreground objects. We also key out the previously found ball-like connected components. The resultant keyed frame is then passed to the `separate_connected_component` function.

In `separate_connected_component`, we convert the given masked image from RGB into HSV, then isolate the H and S channels. Following this, we apply the Scharr variant of the well-known Sobel edge-detection kernel in both the `x` and `y` directions to the respective channels, providing us derivatives with respect to hue and saturation both vertically and horizontally. In doing so, we are able to distinguish not only differently-colored balls from each other (by virtue of their differing hues and saturations) but also (under ideal circumstances) the background between balls from the balls themselves. We then threshold these derivatives to create further binary masks. Additionally, we isolate the background mask from the given image and apply the `bwmorph` "`clean`" function. We then merge these five masks (including background mask) into a single mask by using the Matlab vector "or" (`|`) operator. We finally apply various functions to improve the quality of the final mask, namely, `medfilt2` and `bwmporh` "`erode`". We subsequently return this mask.

2.1.3 Final connected component extraction

(Description of `extractForegroundObjects` goes here).

2.2 Cross-frame object tracking

Having isolated connected components, we then seek to match these to connected components ("objects") from previous frames within the function `update_ball_tracking`. Each object in the current frame is uniquely mapped to at most one object from previous frames. Matching between objects is performed by first calculating a number of respective properties given the object's connected component, as follows:

- the `x` and `y` coordinates of the connected components's centroid (obj_x^{cur} and obj_y^{cur}, respectively),
- the area of the connected component obj_a^{cur},
- the average hue of the pixels comprising the connected component obj_h^{cur},
- the average saturation of the pixels comprising the connected component obj_s^{cur}, and
- the time at which the connected component was detected obj_t^{cur} (i.e. the current time).

Further to this, the following properties are computed for each current object obj^{cur} with respect to each potential match from a previous frame, obj^{past}, were obj^{cur} to be matched with obj^{past}:

- the x-velocity obj_{vx}^{cur} $((obj_x^{cur} - obj_x^{past})/(obj_t^{cur} - obj_t^{past}))$ and y-velocity obj_{vy}^{cur} $((obj_y^{cur} - obj_y^{past})/(obj_t^{cur} - obj_t^{past}))$,

- the overall object velocity obj_{vm}^{cur} ($\sqrt{(obj_{vx}^{cur})^2 + (obj_{vy}^{cur})^2}$), and
- the predicted x-position obj_{px}^{past} ($obj_{x}^{past} + obj_{vx}^{past}$) and y-position obj_{py}^{past} ($obj_{y}^{past} + obj_{vy}^{past}$) of the past object.

For each potential mapping between current and past objects, the mapping is immediately rejected if any of the following is true:

- the past object has already been matched to an object in the current frame,
- the past object existed too long in the past, as determined by a time thresholding parameter, or
- the calculated velocity of the current object if matched to the past object would exceed a set velocity thresholding parameter.

Otherwise, a cost is calculated for the mapping between objects by calculating a Euclidean distance between these properties in n-dimensional space, with each component approximately normalized to a value of 1 (e.g. obj_x is divided by the width of the frame). Should this cost exceed a predetermined threshold, the mapping is discarded.

While minimizing the costs across all possible mappings constitutes an optimization problem, for the sake of simplicity we instead utilize a greedy "first come, first served" approach whereby the first found connected component/object is paired with the best past object not violating any of the above constraints, the second paired with any past object not including that matched to the first object, *et cetera*. Finally, these mappings are used to determine `prev_x` and `prev_y` properties for the current objects, which are used when plotting object paths.

3 Results

4 Discussion