Lab 2: Brute force tracking

For this lab we will try to do object tracking for the first time. Tracking means establishing the position of a target in subsequent video frames. The most primitive form of tracking is 'brute force' tracking. Brute force tracking uses no tricks whatsoever to speed up the tracking. It is the slowest form of tracking, but it is a good start to develop other tracking algorithms.

Comments on last week's assignment

Some of you tried to use the built-in functions that Matlab provides to create a histogram of an image. Since the function `hist` is not suited for the task at hand (it takes 1-dimensional input), this is not the way to go. You will have to write your own function to compute a histogram from an image.

A histogram is a method of quantizing the colors in the image. A color in the (R,G,B)-space consists of three values R, G and B, and in the histogram, the correlation between these values have to be retained. Therefore, it is not good enough to generate three histograms of the separate color channels (one for each color channel), but every (R,G,B)-triple will have to be "binned" as a whole. If the histogram would be as accurate as possible, for an (R,G,B)-image with pixel values ranging between 0 and 255, the resulting histogram contain 256x256x256 bins. However, this version is quite sensitive to noise, and huge in memory, so what is generally done is map the colors to a lower-dimensional histogram, e.g. a 16x16x16 histogram. You will have to write a function that determines the appropriate bin for a given color.

Tracking

As stated above, tracking means establishing the position of a target in subsequent video frames. So you start with a target at a known position, you create a target representation and then you will try to find the position that best matches the target in the next frame. The only trick you may use in brute force tracking is exploiting the knowledge that the new position of the object will be close to the previous one. So, if the target at the previous frame was at location `(x,y)`, then your search space is restricted from `(x-width, y-height)` to `(x+width, y+height)`, where `(width, height)` determines your search window.

This is basically all there is to it (it is after all *brute force*). Some interesting frames can be found in the **data** folder. The files MoreFrames_part_1.zip, MoreFrames_part_2.zip, MoreFrames_part_3.zip, MoreFrames_part_4.zip, contain 200+ frames and are split into four (quite large) files.

Histogram distances

You need histogram distances to compare the histograms of the target model and the target candidates. So you create a histogram for the candidates in exactly the same fashion as you did for the target model. Now you need to compare the candidate and target histograms, in some meaningful way. The reader discusses several methods. I will repeat two here:

Euclidean distance

The Euclidean distance between histograms calculates the following distance between all histogram bins: `(b1 - b2)^2`, then sums this over all bins and takes the square root.

Bhattacharyya distance

The Bhattacharyya distance is a measure to compare statistical distributions, something that our histograms represent. The Bhattacharyya distance is calculated as follows.
First the Bhattacharyya coefficient is determined:

$$\rho[p,q] = \int \sqrt{p(u) \cdot q(u)}\, du.$$

then the Bhattacharyya distance is determined from:

$$d = \sqrt{1 - \rho[p,q]}$$