Hybrid Tracking in OpenCV

Pushkar Kolhe

August 26, 2011

Abstract

Notes on the hybrid tracker module for OpenCV. Initial version of the code is at `https://github.com/pushkar3/hytrack`.

1 Introduction

The mean shift algorithm was designed so as to search for a local probability density function (PDF) that approximates the empirical PDF. Mean shift has shown promising performance in many circumstances against image occlusions and clutters. In spite of its success in specific applications, however, mean shift has presented less efficiency in the presence of dramatic intensity or color changes in a pre-defined neighborhood. In these situations, additional features may be used as a complement that can improve the capability of the trackers. In our case we use a spare feature based tracker, such as SIFT or SURF. They can be integrated with the Mean Shift tracker for robust tracking.

2 Object description

2.1 For MeanShift

An object is described as a PDF. Object tracking involves searching for the most similar PDF across two neighboring frames. In MeanShift the similarity measure is based on color features. We convert each frame into the HSV color space and use these channels for creating a PDF. For this hybrid tracking our features are in joint feature-spatial space.

$$p_x(x,u) = \frac{1}{N}\sum_{i=1}^{N} w\left(\left|\frac{x-x_i}{\sigma}\right|^2\right) k\left(\left|\frac{u-u_i}{h}\right|^2\right) . \tag{1}$$

Here x is the position of a feature, u is the color feature. σ and h are used to normalize and w and k are kernels, in this case L2 Norms.

The MeanShift algorithm used for tracking works in the following way:

1. Create a histogram based on the chosen region of interest.
2. Backproject the histogram on the image.
3. Use the backprojection for Meanshift alongwith a distance measure.

2.2 Convergence of MeanShift

Meanshift is an instance of gradient ascent with adaptive step size. For asymptotic distributions close to the normal and Gaussian distributions, convergence in meanshift is guaranteed. However color based PDF might not always have these asymptotic properties. MeanShift can get stuck at a local minima along the PDF. Integrating SIFT correspondence helps to make the object PDF asymptotic to the normal distribution, thereby reliably performing meanshift convergence at every frame.

2.3 For SIFT feature tracking

SIFT feature tracking works when the region of interest being tracked is textured or has many features. Contrary to MeanShift, which can track blobs of colors very well, SIFT feature tracking can track textured surfaces. Integrating them makes our tracker more robust.

$$p_x(x,u) = \frac{1}{N}\sum_{i=1}^{N} w\left(\left|\frac{x-x_i}{\sigma}\right|^2\right) f_s(x,u)\ , \tag{2}$$

$$f_s(x,u) = \frac{1}{2\pi\sigma_s^2} e^{-(d(x_{ms},x_{ft}))^2/2\sigma_s^2}\ . \tag{3}$$

Here $f_s(x,u)$ is the Gaussian kernel where the variance is the distance between the detected region centers using mean shifts and feature tracking.

The SIFT tracking algorithm works in the following way:

1. Find SIFT features on the region of interest.
2. Find SIFT features in the next image around the region of interest.
3. Find the best correspondence using SSD/RANSAC. Find displacement and scale of the new region of interest.
4. A Gaussian PDF is created with the new tracked region's center as mean and the difference between MeanShift's predicted region and SIFT's trackers region as variance.
5. Use this Gaussian PDF alongwith distance measure as the object PDF.

A Lucas Kanade Optical Flow based sparse feature tracker is also included in this tracker. This works better in most circumstances.

2.4 Similarity search for the object PDF

A similarity search has to be performed on the object PDF in consecutive frames. This is achieved by using an EM algorithm that minimizes the difference between the mean shifts from the previous frame and the current frame. The object PDF consists of the two PDF's from MeanShift and SIFT matching. They are added in a linear fashion with weights given to each tracker. If the region of interest is textures, SIFT matching gets more weight. If the region of interest can be segmented as a color blob, MeanShift gets more weight. The weight are estimated by using linear regression.

$$p_x(x,u) = \frac{1}{N}\sum_{i=1}^{N} w_1 \left(\left| \frac{x - x_i}{\sigma} \right|^2 \right) k \left(\left| \frac{u - u_i}{h} \right|^2 \right) + w_2 \left(\left| \frac{x - x_i}{\sigma} \right|^2 \right) f_s(x,u) \ . \tag{4}$$

Here w_1 and w_2 are the two weights which are found using linear regression. They have the property that

$$\frac{1}{N}\sum_{i=1}^{N} w_1 \left(\left| \frac{x - x_i}{\sigma} \right|^2 \right) + w_2 \left(\left| \frac{x - x_i}{\sigma} \right|^2 \right) = 1 \ . \tag{5}$$

The overall algorithm for this hybrid tracker is as follows:

1. Define a region of interest in the first frame.
2. Compute the color histogram and backproject it on the image.
3. Generate a PDF based on color histogram and distance measure.
4. Find SIFT features and find best correspondence around the region of interest.
5. Generate a PDF based on the newly detected region of interest and the difference between the detected region using mean shift and SIFT matching.
6. Launch the EM algorithm and iterate until the difference between two meanshift is less than some threshold.
7. Launch linear regression to find weights of the two individual trackers.

3 Experiments

The object PDF is generated by combining PDF's from individual trackers and adding the distance measure. The figure 1 shows the space in which each tracker visualizes the object. It is easy to notice that the MeanShift tracker relies heavily on color blobs, while the SIFT based tracker. Since, the object being tracked does not have good features to track, the SIFT based feature tracker drifts. The EM algorithm still tracks the object by increasing the weight given to the MeanShift based tracker.

A Low pass based filter was added to combine the individual trackers together. Also, an optical flow based Lucas Kanade tracker was added as a sparse feature tracker. Two experiments were conducted to validate the tracker:

1. Weights were recorded while tracking.
2. The tracker was benchmarked against the Bonn benchmark dataset [1].

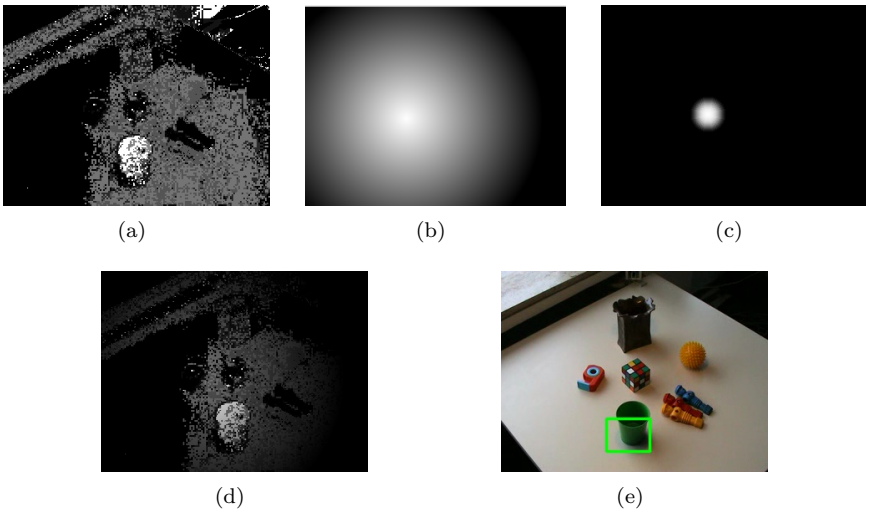


Figure 1: Object PDF. (a)PDF due to color histogram for MeanShift. (b) PDF due to the distance measure. (c) PDF due to SIFT correspondence. (d) Combined PDF for the object. (e) Object being tracked.

3.1 Change in weights

It can be seen that when the tracker is tracking an object with textures, the sparse feature tracker gets more weight. In other cases, when the object does not have much texture the mean shift tracker gets weighted more as shown in Figure 2. This happens due to regression on the weights and the simple Gaussian motion model assumption. The EM algorithm assumes that the object pdf is generated from a single Gaussian mixture. A low pass filter also works in a very similar way, but it can also filter out noise.

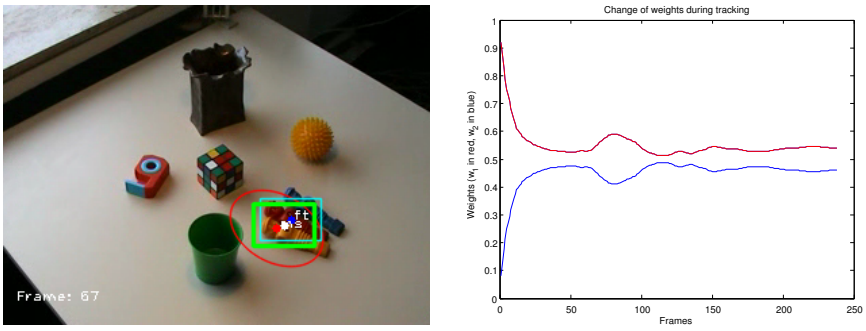
3.2 Benchmarking

The tracker diverges when it makes fast movements. This happens when the sparse feature tracker cannot track efficiently. However, it tends to converge again to the correct region when the movement is slowed down. This can be easily seen in Figure 3. The tracker is still does not work very well again scale changes. This is the reason of the high error in the end. More about the tracker is discussed in the next Section.

4 Issues with the above approach

In the vision literature, hybrid tracking has been used frequently to combine two or more trackers in a certain way. Hence, there is no one way of integrating a hybrid tracker. This makes designing an API hard. In this case seperate structures to configure parameters are used. More parameters can be added when more functionality needs to added to the tracker.

For the current GSOC project a hybrid tracker by [2] had to be implemented. Although their approach looks promising, there are a few details missing. Most



(a) (b)

Figure 2: The weights of individual trackers adjust during tracking. The blue line corresponds to the MeanShift tracker while the red line corresponds to the feature tracker.

of their experiments are conducted on test data where the object of interest does not move, but they track in the presence of occlusion and lighting changes.

Details on the hidden variables and parameter vector for the EM algorithm are not specified. The EM algorithm forms the basis of this approach, and it seems that the controlling parameters are found through the EM algorithm, but this is not clear. It also seems from their psuedo-algorithm that the EM algorithm assumes that the pdf is generated from a single gaussian mixture, but again is not clearly mentioned.

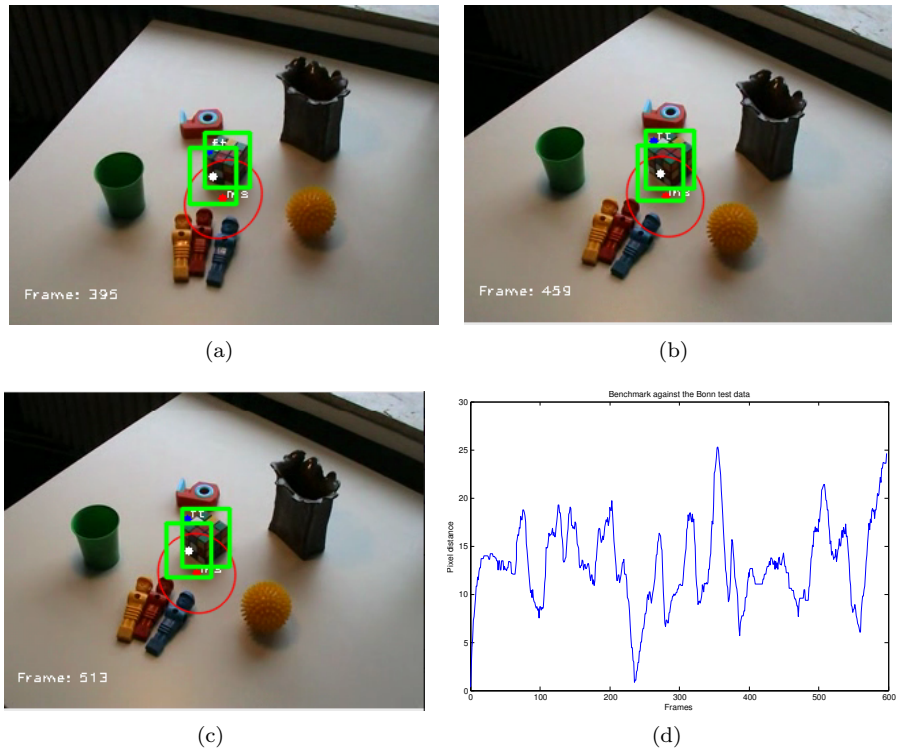
To provide other approaches instead of the EM algorithm, a low pass filter based approach is also added. It takes in the previously tracked object and the new tracked object's position to find the new position based on a gain parameter. In the literature, some have used a boolean switch to switch between trackers when certain condition are met. Such functionalities can be added.

In the current implementation, the EM algorithm frequently returns *nan* as the mean. The input to the algorithm is an array of two dimensional image coordinates based on the object pdf. This bug is yet to be solved.

5 Results

The following were delivered as a part of this project:

1. A hybrid tracker based on Zhou et al. [2].
2. An API/framework for tracking using MeanShift and Feature correspondence in OpenCV.
3. Sample code to use the trackers.
4. Sample code to benchmark the trackers.
5. Documentation for the framework and samples in reST docs.



(a) (b)

(c) (d)

Figure 3: The tracker was benchmarked against hand marked test data. Here, the rubiks cube is being tracked. The red and blue dots show the center or regions detected by MeanShift and feature tracker respectively. (Code included in sample.))

6 Future Work

1. Add Ensemble tracking in this framework.
2. Use existing motion tracking algorithms based on Optical flow with OpenCV in this framework.
3. Work on the spare feature tracker, make configuring parameters easy.
4. Work on the EM algorithm that combines the two trackers.

7 API

The MeanShift and Feature based trackers are parameterized by their parameter classes. For MeanShift, this is:

```
struct CvMeanShiftTrackerParams
{
    enum {  H = 0, HS = 1, HSV = 2  };
    CvMeanShiftTrackerParams(int tracking_type = CvMeanShiftTrackerParams::HS,
```

```
            CvTermCriteria term_crit = CvTermCriteria())
    {
    }

    int tracking_type;
    float h_range[];
    float s_range[];
    float v_range[];
    CvTermCriteria term_crit;
};
```

You can choose if the MeanShift tracker should use the H, HS or HSV space for tracking. For the feature based tracker this is

```
// Feature tracking parameters
struct CvFeatureTrackerParams
{
    enum {  SIFT = 0, SURF = 1, OPTICAL_FLOW = 2 };
    CvFeatureTrackerParams(int feature_type = 0, int window_size = 0)
    {
        feature_type = 0;
        window_size = 0;
    }

    int feature_type; // Feature type to use
    int window_size; // Window size in pixels around which to search for new window
};
```

The feature type to be used for tracking can be specified alongwith the window size for searching.The Hybrid Tracker Parameter class uses these two classes alongwith the Motion Model class which specifies how to integrate the various trackers.

```
 struct CvMotionModel
{
    enum {LOW_PASS_FILTER = 0, KALMAN_FILTER = 1, EM = 2};

    CvMotionModel()
    {
    }

    float low_pass_gain;    // low pass gain
    CvEMParams em_params;   // EM parameters
};

struct CvHybridTrackerParams
{
    CvHybridTrackerParams(float ft_tracker_weight = 0.5, float ms_tracker_weight = 0.5,
            CvFeatureTrackerParams ft_params = CvFeatureTrackerParams(),
            CvMeanShiftTrackerParams ms_params = CvMeanShiftTrackerParams(),
            CvMotionModel model = CvMotionModel())
```

```
    {
    }

    float ft_tracker_weight;
    float ms_tracker_weight;
    CvFeatureTrackerParams ft_params;
    CvMeanShiftTrackerParams ms_params;
    CvEMParams em_params;
    int motion_model;
    float low_pass_gain;
};
```

This API makes it very easy to use the tracker class. The interface excluding the parameter configuration is independent of the trackers used. From the sample code, the tracker is used as:

```
HybridTrackerParams params;
HybridTracker tracker(params);
tracker.newTracker(image, selection);
tracker.updateTracker(image);
```

This API needs to be discussed further considering different types of trackers, rate of update, integration, configuring parameters etc.. and revised. This can act as an initial proposal to such an interface.

References

[1] "Bobot - bonn benchmark on tracking," 2010. [Online]. Available: http://www.iai.uni-bonn.de/~kleind/tracking/index.htm

[2] H. Zhou, Y. Yuan, and C. Shi, "Object tracking using sift features and mean shift," *Computer Vision and Image Understanding*, vol. 113, no. 3, pp. 345–352, 2009.