# Functions for Object Tracking in OpenCV

We have added two new state-of-the-art object tracking algorithms to OpenCV which can robustly track images patches across a video sequence and learn online (i.e., *on-the-fly*) as the appearance of the object changes from frame-to-frame. The input to these trackers is simply an image and an initial bounding box, defining the first location and dimensions of the image patch to be tracked. On each subsequent frame, an updated bounding box is returned with the new location of the object. For these algorithms, the size of the box never changes—whatever is given for width and height in the first frame is used throughout the sequence. Therefore if the object gets much larger or smaller, although it still may keep track, it may not encompass the object as tightly as the first frame.

The images that are used are 1-channel, 8-bit-per-pixel gray-scale images. If a color image is input then it is immediately converted to gray-scale internally. However, any image input should be of type `unsigned char` (`float`, `int`, `double` will just fail). The features used are Haar features, based on the popular offline classifier approach of Viola and Jones which has been so successful in face detection.

To use the trackers, in addition to the usual OpenCV includes, you must include `<object_tracker.h>`. The main object tracking object is `cv::ObjectTracker`. There are several important parameters which are used by the object tracker, the most important of which is the algorithm to be used as the main tracking algorithm (currently, the choices are `CV_ONLINEBOOSTING`, `CV_SEMIONLINEBOOSTING`, and `CV_ONLINEMIL`). These trigger the correct chain to take in the inheritance hierarchy to instantiate the appropriate tracking algorithm. Because these are all online boosting-based algorithms, many of the tracking parameters are shared amongst the different algorithms. The important ones are `num_classifiers_`, which says how many weak classifiers to use in the online tracker, and `num_features_`, which says how many features are in the feature pools for each of the trackers. (The parameters `overlap_` and `search_factor_` are specific to Online Boosting and describe how a new image is search and classified at each pixel). For more details on the algorithms, please see the following papers:

Grabner, H., Grabner, M., and Bischof, H. "Real-time Tracking via On-line Boosting". *Proceedings British Machine Vision Conference*, pp 47—56, 2006.

Grabner, H., Leistner, C., and Bischof, H. "Semi-supervised On-line Boosting for Robust Tracking". *Proceedings European Conference on Computer Vision*, 2008.

Babenko, B., Yang, M. H., and Belongie, S. "Visual Tracking with Online Multiple Instance Learning". *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

Begin with setting up a `cv::ObjectTrackerParams` structure with the algorithm type and associated parameters. Suppose we want to use the *MILTrack* algorithm:

```
cv::ObjectTrackerParams params;
params.algorithm_ = cv::ObjectTrackerParams::CV_ONLINEMIL;
params.num_classifiers_ = 50;
params.num_features_  = 250;
```

Next we instantiate the object tracker with the parameters in the constructor:

```
cv::ObjectTracker tracker(params);
```

There is also a function to set/change the parameters later. Keep in mind a deep copy is performed of the parameters, so if you set them into the object tracker's constructor and then change them (outside) later on, it doesn't get propagated via pointers or anything. Therefore, at any time you can set/reset the parameters:

```
tracker.set_params(params);
```

Then we initialize the tracker with an image and a bounding box of the first location of the image patch to be tracked, as a CvRect. The image can be gray-scale or color 8-bit pixel types. Ultimately color images will be converted to gray-scale anyways. Initialize the tracker as follows:

```
IplImage* I = my_video_source.get_image();
CvRect initial_bounding_box = cvRect(122, 58, 75, 97);
if ( !tracker.initialize(I, initial_bounding_box) )
{
   std::cerr << "Could not initialize tracker!\n" << std::endl;
}
```

If initialization couldn't occur for some reason, false will be returned from the initialize() function. Finally, to track subsequent frames, call the update() function:

```
CvRect theTrack;
while (keep_grabbing_frames)
{
   IplImage* I = my_video_source.get_image();
   if ( !tracker.update(I, theTrack) )
   {
      std::cerr << "Could not update tracker!\n" << std::endl;
   }
}
```

As a final note, the semi-supervised online boosting algorithm, although implemented, sometimes shows issues on Linux machines, but works as designed on windows machines. The symptom on Linux is that it runs but doesn't track successfully. This will be investigated further.