**Question 5**:

The algorithm was executed on frames 1-30 of PETS2014-0101 video, and scaled down by 50% for making our computer to process it.

1. Short description of the algorithm:

The algorithm perform tracing on objects by looking for their histograms in every frame. The objects-to-track (= **models**) initial tracking details are inserted manually (their location, and in what frames they appear), and also their **histogram** for the first frame is calculated. In every consecutive frame, we look for the model in some **region** around it by doing exhaustive search for its histogram in that region. The exhaustive search is quickly performed because we use the region's integral histogram for extracting **best-model-candidates** histograms. In every exhaustive-search-round, the algorithm tests candidates-models with 75%, 100% and 125% size of the original model. This is available due to normalization of the histograms. When the best is chosen, the algorithm updates the state for this model and moves to the next frame. If after consecutive frames the object is not found on the screen it's no longer tracked.

1. Tracking results:

We tested our video on the PETS2014-0101 video from the given DATA folder. The algorithm managed to track objects with only the initial information from the first frame was given (motion is from left to right):



However when arriving to the conus in the last frame above, the algorithm found that the conus should be part of the pedestrians we track, but was able to recover few frames later. However, as can be observed, the conus distraction had impact on the histogram that we will look for in the next frames (the algorithm tracks the body of the pedestrian, without the head):



There's another aspect that affects the algorithm performance: resolution.

There are 2 more pedestrians that the algorithm was told to track on (gray & yellow pedestrians):



Let's look at them a little closer:

 

The histogram of the left pedestrian is very similar to other frames around it, such as just a random square near him:



And is very hard to detect. The other however is very different from his environment and therefore the algorithm manages to track him:



However, as expected, the algorithm loses the gray guy, which looks like his environment.

To summarize up the results: the algorithm can effectively track objects and is sensitive to the model's histogram changes. However the resolution is a major player especially here, since as the resolution is lower it is hard to detect the tracked objects.

1. Continue with Ben

**Question 6:**

1. For tracking objects, we will use the algorithm in question 5, using integral histograms. The method for determining when people are getting close to each other is by looking at their **search region** around them. A search region is the area around the tracked object, that we will exhaustively-search it in the next frame. When 2 search regions are overlapping each other 🡪 the objects may be close to each other in the next frames 🡪 We draw a red line between them.

The method will absolutely fail for objects that are close on X & Y axes, but far on the Z axis (when one object is more "deeper" than the other) because the algorithm do not support the Z axis. The method will also fail when at the 1st frame the two objects are near each other, but in the next frames are getting far from each other. So until their regions won't overlap anymore, even though they are getting far from each other – the algorithm will mark them as "getting close".

1. Implemented in integralHistogramTracking.m. Depend on parameter 'check\_proximity'
2. Implemented in q6\_c.m.