**Overview**

In this project we implemented Kalman Filter based object tracker. Given a short video that has a constant background it assigns an id to each object in the video and attempts to track this object while it is in the video.

The started the program prompts for a video name in the console window. The videos are taken from the folder.

IMAGE

A graphical window is opened showing the video. In the background the program begins to compute the tracking information. GUI controls:

* **Q**: Exit the program
* **Space key**: Stop/play video.
* **Left/right arrows**: Step through the video.
* **Up/down arrows**: Change presentation mode.
* **Home**: Jump to the beginning of the video.
* **End**: Jump the end of the video.
* **R**: Show tracking path.
* **N**: Show ID display.
* **C**: Show blob covariance.
* **Shift-C**: Show tracker covariance and velocity.
* **M**: Show markers (blob centers).
* **D**: Show blob indexes.
* **T**: Show possible transition edges.

**Blob Detection**

We begin by building an image of the background, by taking the median value of each pixel over all frames. Once we have the background we can compute the foreground of each frame as the abs diff with the background. We apply a threshold to the foreground, apply morphological operations to smooth the result and then find the connected components in the image (ignoring very small components).

We represent each connected component as a Gaussian blob: mean point and covariance matrix. The blobs are then passed to the tracking algorithm.

**Tracking Algorithm**

The algorithm uses Kalman Filters for tracking individual object. When multiple objects combine into one blob, it tries to consider all the possible ways the objects could go, and choose the most probable one.

The Kalman filter keeps track of position and velocity. It is defined using the following matrices:

and are set each frame according to the position and covariance of the tracked blob.

Multi-object tracking tries to assign stable ids to each moving object. This isn’t simple because sometimes multiple objects can merge into a single blob, and we have no good way to know how many objects a blob contains. For this reason, we always assume each blob contains a single object, but we also store the probabilities for other objects in this blob. If in later frames we find that one of the probabilities was in fact true, we go back, and change this object from a probability to a tracked object.

The tracking goes as follows: Take the current trackers and probabilities and use Kalman Filtering to update them for the new frame. Turn the highest probabilities into sure trackers, allowing only one tracked object per blob. For each probability chosen in this step we retroactively mark it as a sure marker in previous frames. Assign new ids to blobs with zero trackers.

**Code Architecture**

To let the user use the program before all frames are processed, the program uses multiple threads:

* Video loading thread: loads the video, notifying a condition variable after each frame is loaded.
* Tracking thread: runs image processing on each frame and does the tracking.
* GUI thread: Periodically syncs its state with the tracking thread, and displays its thread in a graphical window.

The entire tracking process state is represented using ProcessState:

For each frame a FrameState is computed: