CSE 252A: Tracking & Recognition

Lecturer: David Kriegman

Scribed by Owen Jow on November 20, 2018

1 Tracking

Sometimes, we want to track meaningful objects over time. Usually this means we only have to focus on a smaller region of the motion field, but it is also made difficult by 3D pose variation, occlusion, illumination changes, camera jitter, and expression variation (among other things).

Formally, in tracking we want to estimate some *state* (location, velocity, etc.) at each frame. We might also need to consider the *state transition dynamics*, the *representation(s)* we're using, *prediction* methods, *data associations* (correspondences between measurements and objects), *correction* methods (based on new measurements), and *initialization* strategies (first state?).

We can define a general probabilistic model for tracking –

- with moving objects having underlying states $\ldots, X_{t-1}, X_t, X_{t+1}, \ldots$, and
- measurements $\ldots, Y_{t-1}, Y_t, Y_{t+1}, \ldots$ being in some cases functions of the state.

Then we can view this as an HMM, treating state as random variables X_t with probabilities $P(X_t)$ or $P(X_t \mid X_{t-1})$, and observations as random variables Y_t with probabilities $P(Y_t \mid X_t)$.

We define three main components of tracking:

1. **Prediction.** Compute a representation of $P(X_t \mid Y_0 = y_0, \ldots, Y_{t-1} = y_{t-1})$.
2. **Data association.** Identify measurements y_t relevant to the state we care about.
3. **Correction.** Compute a representation of $P(X_t \mid Y_0 = y_0, \ldots, Y_t = y_t)$.

Most commonly, we will use a linear dynamic model for this. We will treat the state and observation vectors as multivariate Gaussians (with potentially correlated entries).

$$x_t = \mathcal{N}(D_{t-1}x_{t-1}; \Sigma_{d_t})$$
$$y_t = \mathcal{N}(M_t x_t; \Sigma_{m_t})$$

for covariance matrices Σ_{d_t} and Σ_{m_t}. D_{t-1} and M_t are some transformation matrices.

Together, these equations define our linear dynamic model. Usually, we will then use a multivariate Kalman filter to solve the tracking (prediction and correction) problem.

2 Recognition

In a standard recognition problem, we want to identify the presence of known objects in an image. There are many task-dependent levels of identification ("human," "girl", "Christine") that we might

want to produce. We might want to do *category* or *instance* recognition ("girl" versus "Christine"). Also, we might want to detect the 2D location and/or 3D pose of the object(s): *where* in addition to *what*. **Bottom line: we can ask many questions related to recognition.**

This is a challenging problem, due in large part to variability in pose, lighting, clutter, etc.

It has also seen a lot of development.

- In the early days, simple (multi-) template matching was a common approach to recognition. However, this requires one to know, e.g., the scale of the objects he/she is looking for.
- Next, people started extracting features from images & matching features instead of raw pixels.
- At some point, people tried to match 3D shapes and other abstract representations, and also features generated from abstract representations of *collections* of objects.
- Later, classifiers were introduced to distinguish between feature vectors.
- Finally, it turned into deep networks doing all the work.

In essence, a typical pre-"deep learning craze" recognition architecture might be (1) *extract (hand-crafted?) features from the image*, (2) *perform classification on the features*.