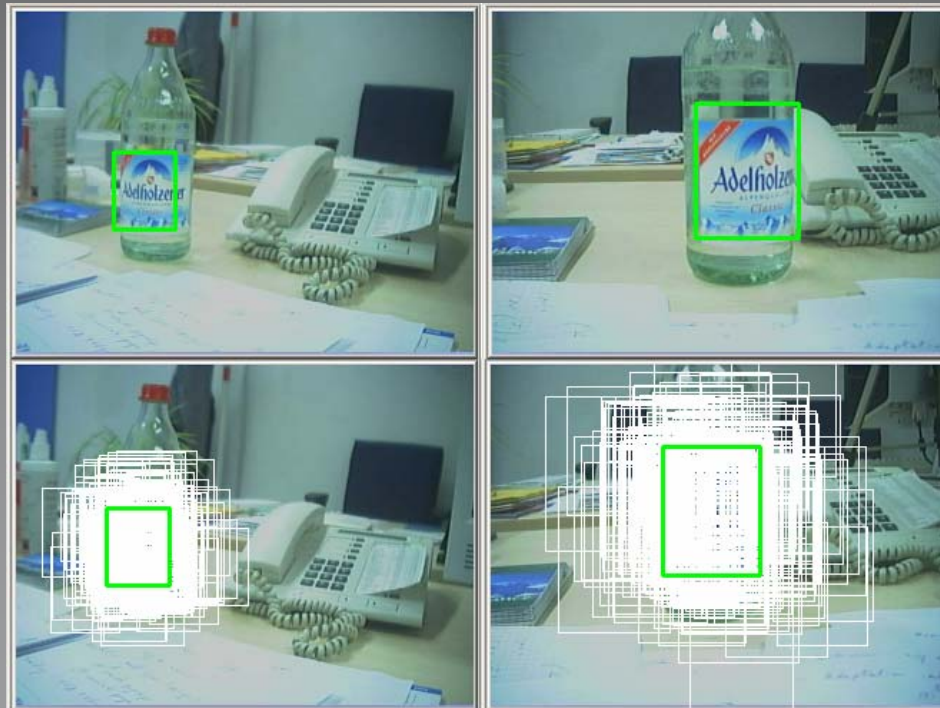
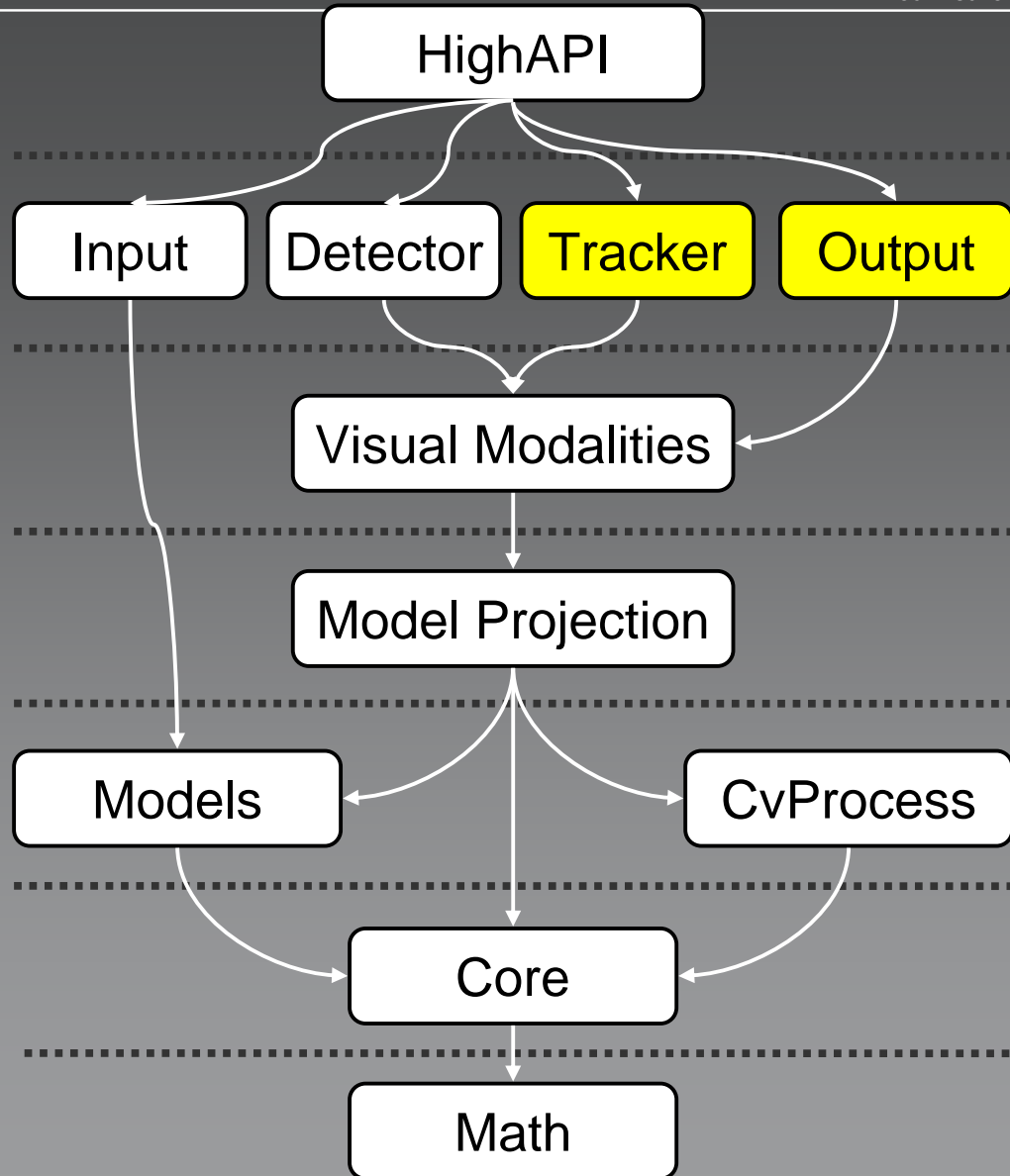


OpenTL – Tutorial 5

- GOAL: Instantiate a SIR particle filter, and track the object





Tracker

- `opentl::tracker` namespace

The Tracker abstraction is common to all Bayesian filters, for single- or multi-target problems

```
opentl::tracker::Tracker
```

```
{
```

- `Tracker(Likelihood, Warp)`
Construct a tracker, and link to a Likelihood instance
- `init(initStates)`
Initialize the state density estimate, $P(s_0)$
- `predict()`
From the last posterior $P(s_{t-1})$ and the dynamical model, predict the state prior $P(s_t^-)$
- `correct()`
Perform a measurement (using the Likelihood instance) and update the state posterior $P(s_t^-) \rightarrow P(s_t)$
- `output()`
Evaluate the output state from $P(s_t)$ (e.g. weighted average)

```
}
```

SIR particle filter

opentl::tracker::SIRParticle

- $P(s)$ is internally represented by a set of weighted particles: $P(s) \sim \{s_t^{(i)}, w_t^{(i)}\}$

- The output state is: $\bar{s}_t = \sum_{i=1}^N w_t^{(i)} s_t^{(i)}$

- In order to get the output:

```
std::vector<States> * SIRParticle::getTargets();           // Only one State in this vector!
```

- In order to get a single particle:

```
State * SIRParticle::getParticles(int i)
```

- In order to get a particle weight:

```
double SIRParticle::getParticleWeight(int i)
```

Output namespace

- `opentl::output::Output`

This namespace provides output visualization and post-processing

Output class

- `output::Output`

The `Output` class contains static functions for visualization, not based on OpenGL

- `output::Output::drawPose(`
`&outImage, // Destination image`
`shape, // pointer to ShapeAppearance model`
`warp, // pointer to Warp class`
`pose, // pointer to Pose`
`1, // Edge thickness`
`CV_RGB(255,255,255), // Color`
`false, // Draw also edge normals`
`true, // Draw edges`
`camIdx); // Camera ID number`