Tomáš Hodaň (xhodan04@stud.fit.vutbr.cz), 17. 4. 2012

Object Detection Framework

Main Idea

The main purpose of this framework is **to unify and simplify wrapping of object detectors into ROS system** while preserving availability of all its functionality.

The framework consists of three main functional elements – an object detector, a detection tracker and a matcher of detections and predictions. One of the main features of the design is a separation of the detection tracker (used to predict the next state of a detection), which performs a functionality universal for all different object detectors, from a particular detector-dependent functionality, which is performed by the detector itself and the matcher of predictions and detections. The reason why we consider matching as detector-dependent functionality is that for each detector, there can be different features suitable for this purpose.

Implementation

From the ROS viewpoint, there are two main components in this framework, a detector node and a tracker node, both implemented as ROS nodes. The detector node employs (inherits relevant C++ classes) implementation of a detector and a suitable matcher of detections and predictions. The tracker node offers a service through which it can provide a prediction of a detection state (position and size of its bounding box) at any time in the future. This tracking is currently done using Kalman filter.

Communication with another ROS nodes is realized by the prediction service, which can be used to periodically obtain the current prediction of detections state. This is to satisfy a possible need of an external node to be the current state updated with some specified rate. If there is no need for a periodical update, one can also subscribe to a topic (provided by the detector node) on which detections are published as soon as they are obtained.

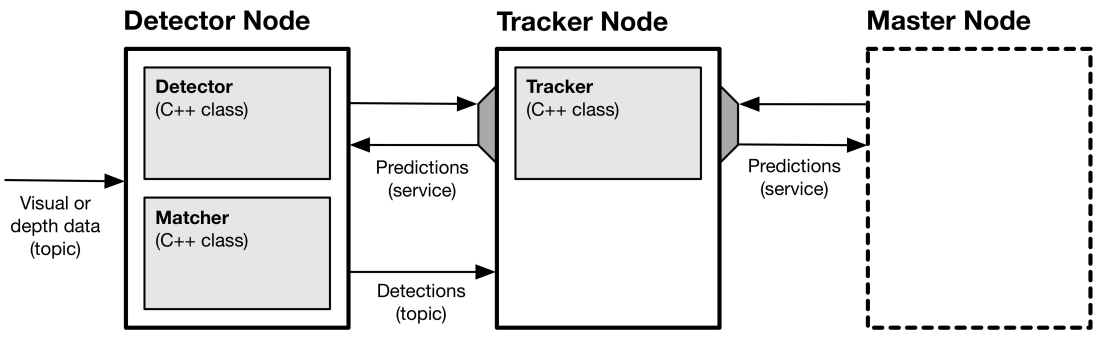

Figure 1: *Scheme of ObjDet Framework*

The roles of the main components can be better understood from the working

pipeline mentioned below (see also Fig. 1).

1. The detector node receives visual or depth data (depending on its type).
2. The detector node asks the tracker node for predictions of the currently considered detections through a ROS service (a detection is not considered any more when there is no detection of the same object for a specified number of detection runs).
3. Object detection using the data received in the first step (predictions can be used as a prior knowledge, if the detector supports it).
4. Predictions and the new detections are matched. This is important to know which detection corresponds to some already detected object and which is a detection of a new object, which has been unseen so far.
5. The new detections (with possibly modified corresponding object ID) are published and thus provided to the tracker node.
6. The tracker node subscribes the new detections, which are used to update its state, i.e. the state of Kalman filters (there is one Kalman filter for each detected object).
7. A master node periodically asks for the current prediction of detections state through the prediction service. This node is not a part of the framework and represents just some ROS node using object detections for some other task.

The described implementation is realized within **but_object_detection** stack. We believe that the design of our framework is general and extendable enough, so we will make this stack public and thus usable by ROS community.