Vision

Core Functionality

- The vision system on the sub is performed by multiple vision nodes. Each currently contains a number of different processors, explained later, which perform a task on the latest images captured from the cameras. These images are synchronized with each other in time.
- There is one vision node spawned for each task required to be performed using the vision.launch file. For more information on using this launch file, see Running the Vision System in ROS below.
- The processors used by the vision node currently are as follows and described below.
    - ColorProcessor
    - StereoProcessor
    - FeatureProcessor

Processors

ColorProcessor

Purpose

Performs color masking according to the parameter file for the node.

Structure

The ColorProcessor contains a FilterSet. This FilterSet will sequentially apply each Filter it contains which are each a single OpenCV operation on an image.

Parameters

The parameters for the ColorProcessor part of a vision node should be formatted as follows:

```
filters:
    - [filter name]: {[filter parameters]}
    - [filter name]: {[filter parameters]}
```

Where the [filter name] is known by the ColorProcessor allowing it to create the proper filter and the [filter parameters] are parsed by the filter. An unlimited number of filters is allowed. Each filter in the list will operate on the result of the previous filter's output.

StereoProcessor

Purpose

Utilizes the two forward facing cameras to determine the distance an object is away from the cameras. This requires the cameras to be calibrated.

Structure

This processor is currently under construction and is not finalized. This section will be changed in the future.

Parameters

The StereoProcessor currently does not use any parameters. This may change in the future, however.

FeatureProcessor

Purpose

More accurately identifies contours in the image as being a particular object.

Structure

The FeatureProcessor contains a single ObstacleDetector which is an abstract class that the detectors for each obstacle in the course inherit from. Each detector must also contain a static init function which takes an ObstacleDetector pointer and initializes it to a new instance of that detector.

Parameters

The parameters used by the FeatureProcessor of each vision node are formatted as follows:

```
features:
    detector: [detector name]
    params: {[detector parameters]}
```

There should be only one of these blocks in each parameter file where [detector name] is the string mapped to the init function for the desired detector and [detector parameters] is the set of parameters required by the detector.

Cameras

See the Cameras page for more information on the details and use of the cameras.

Running the Vision System in ROS

After making sure that the cameras are running if needed, use the following command to start the vision system.

```
roslaunch robosub vision.launch
```

To remap left and right camera topics, append

```
leftImage:=[newTopic]
```

and/or

```
rightImage:=[newTopic]
```

(bottom camera to be implemented) the topics for the simulator are

```
/camera/(left|right|bottom).
```

To use simulator color parameters and listen to simulator topics by default (they can still be overwritten using the above leftImage and rightImage remaps), append

```
simulated:=true
```

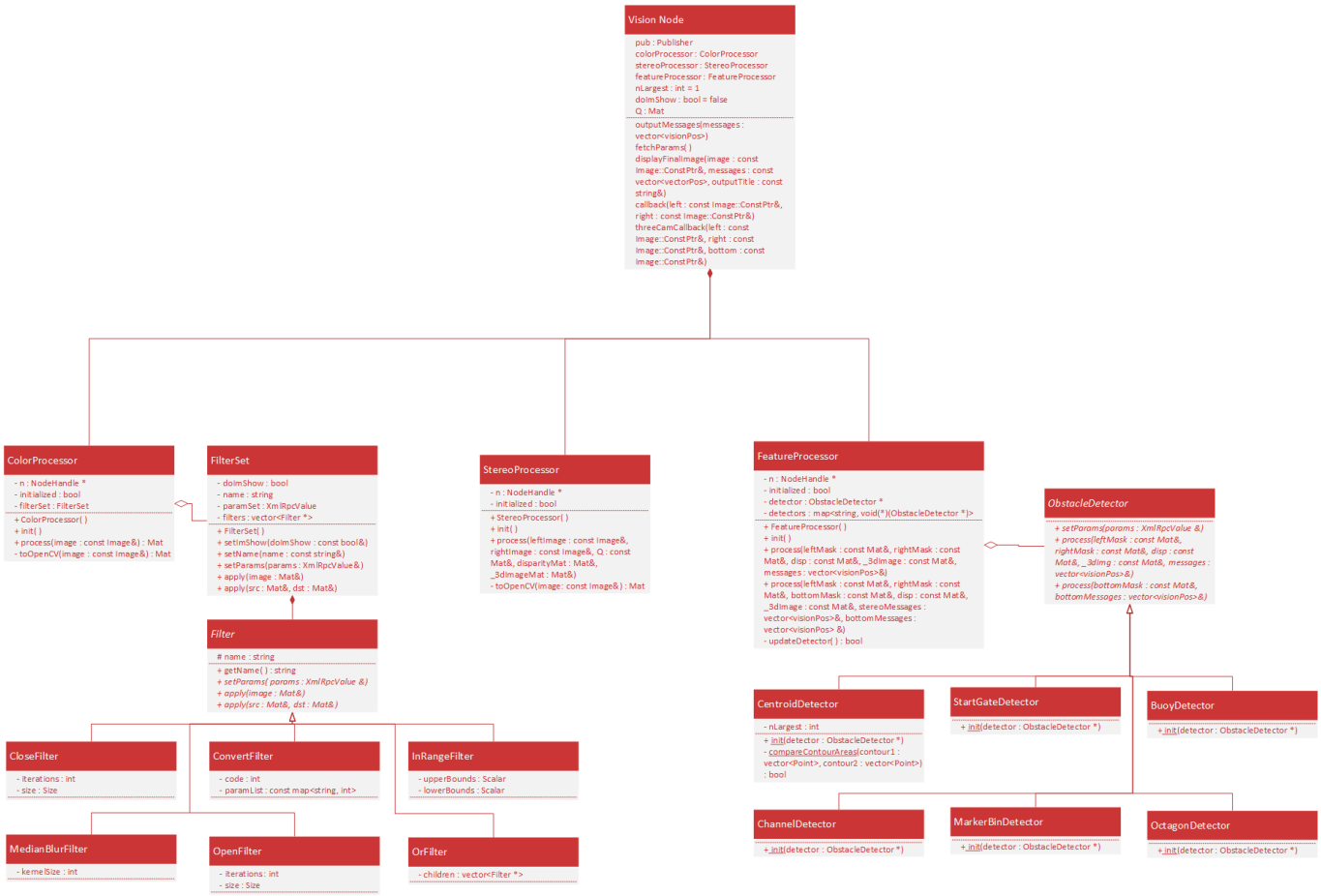
(This feature could change in the near future) After this, you will see the launch file spin up multiple nodes. The vision system is running!

If you would like to see the images that the system is using, you can run the following command:

```
rosparam set /[vision node name]/processing/doImShow true
```

You will then see many windows open each with a unique image.

UML



Vision UML