Overhead localization system

Overhead localization system includes two cameras mounted near the right and left walls of the room (right and left cameras) and computers that process images from these cameras and identify and locate objects in the room. Namely, the system identifies AR tags and color blobs and calculates their position (X,Y coordinates and rotation angle) in the floor plane. Resulting information is published to the corresponding ROS topics, separate for right and left cameras. The schema of the localization system functionality is provided on the figure 1. Below we first describe in which order components of the system should be started and configured and then describe how user can obtain information on objects from the system.

Running overhead localization system

Loading camera drivers

We are using a Play Station 3 camera, which can work in following modes:

Mode no.	Resolution, pixels	FPS
0	640 x 480	15
1	640 x 480	30
2	640 x 480	40
3	640 x 480	50
4	640 x 480	60
10	320 x 240	30
11	320 x 240	40
12	320 x 240	50
13	320 x 240	60
14	320 x 240	75
15	320 x 240	100
16	320 x 240	120

Mode 04 is used for the overhead system. The mode can be set up when camera’s driver is loaded using “modprobe” command:

```
modprobe -r gspca-ov534                 #unload the driver first

modprobe gspca-ov534 videomode=04       #Load the driver and set up mode
```

Obviously, driver loading/unloading should be done by a user with root privileges.

Cameras can also switch between the narrow and wide angles of view. From our experience it seems that the system works better if the camera is switched to the narrow angle of view. However, this requires camera to be situated further from the field and thus not always possible.

IMPORTANT: the driver works only with a specific version of the Linux kernel. DO NOT UPDATE the kernel on the notebooks used to control the cameras!

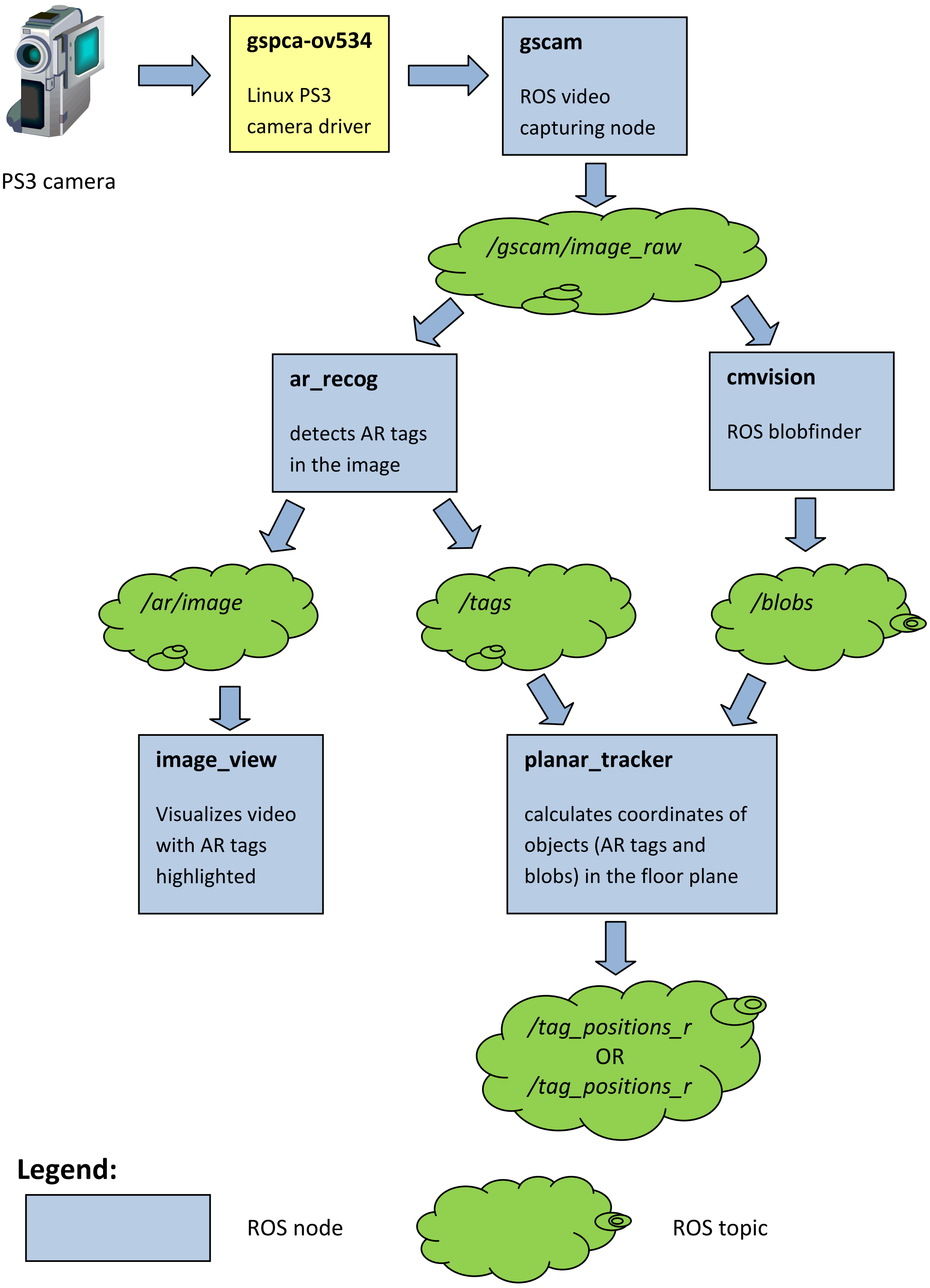

Figure 1: components and data flow in the overhead localization system

Configuring camera

There can be problems recognizing AR tags if the image is too bright. In order for the system to work reliably, camera parameters (exposure, main gain) must be configured. This is done using the `guvcview` program:

```
guvcview -d /dev/video1
```

Please set up following parameters for the left camera:

Exposure:	50
Main gain:	5

Please set up following parameters for the right camera:

Exposure:	70
Main gain:	10

Starting video capture

IMPORTANT: don't forget to launch roscore before launching any ROS nodes. Each of the two notebooks that control right and left cameras must run their own instances of ROS core.

`gscam` is the ROS video capture program. `gscam` relies on certain configuration files, so it must be launched from the `/bin` directory of the ROS package. The following command line launches `gscam` and broadcasts image to the `image:=/gscam/image_raw` ROS topic:

```
roscd ar_recog

cd bin

rosrun ar_recog ar_recog image:=/gscam/image_raw
```

Starting ar_recog

`ar_recog` is the program to recognize AR tags; it is a part of the Brown ROS package. The program analyzes image frames from the `/gscam/image_raw` topic, finds images of AR tags and outputs information on located tags to the `/tags` ROS topic. For each tag, `ar_recog` outputs its ID, X and Y coordinates relative to the camera (in the camera image plane) and Euler angles that describe how the tag is rotated relating to the camera image plane.

Obviously, the program needs to have a description of how every AR tag looks like. These descriptions (patterns) are stored in separate files in the `/bin` folder of the ar_recog ROS package. In the same place there is a configuration file that contains the list of all the AR tags in the ordered fashion; tag ID is defined by its index in that list. Please refer to the description of the `ar_recog` in the `brow-ros-pkg` documentation for a more detailed description.

In order to run `ar_recog`, run following commands:

```
roscd ar_recog

cd bin

rosrun ar_recog ar_recog image:=/gscam/image_raw
```

In addition to tag information, the program also publishes video stream with AR tags highlighted to the /ar/image ROS topic. This stream can be viewed with the image_view ROS program:

```
rosrun image_view image_view image:=/ar/image
```

Starting blobfinder

cmvision is the standard ROS blobfinder; it is a part of the vision ROS stack. cmvision analyzes image frames and then detects objects of the same color range. Color ranges are defined in the colors.txt file. Path to the colors.txt file should be specified by the cmvision/color_file ROS parameter. colors.txt has two sections: [Colors] and [Thresholds]. Each line in the [Thresholds] section defines a color range in the YUV color space for the object to be detected. For every color range there should be a corresponding entry in the [Colors] section. When the cmvision detects a blob that matches a corresponding threshold, then it publishes a "blob" (rectangle) of the corresponding color to the /blobs ROS topic.

To configure the thresholds, the user should use color_gui program:

```
rosrun cmvision color_gui
```

The program will display video from the camera, so the user can click with a mouse on an object he wants to get color thresholds for (in our case – a ball). Thresholds will be displayed in the corresponding text boxes of the color_gui.

We prepared a special ROS launch scripts to launch the cmvision: cmvision_l.launch for the left camera and cmvision_r.launch for the right camera. These scripts are located in the cmvision folder. To launch cmvision, run following commands:

```
roscd cmvision/

roslaunch cmvision_l.launch
```

Starting planar_tracker

In fact, coordinates of the object reported by ar_recog or cmvision are of a little of use for the overhead localization system because these coordinates are relative to the camera. However, the robot needs coordinates in the Cartesian coordinate system associated with the soccer field, where point (X,Y)=(0,0) corresponds to the lower left corner of the field. In particular, the robot needs to know its X,Y coordinates and yaw angle, as well as X,Y coordinates of the obstacles and the goal. Thus coordinates of the objects in the camera image plane must be transformed to the field coordinate system. This is done using a perspective transform performed by the planar_tracker program. planar_tracker is a part of the brown-ros-pkg package.

planar_tracker accepts positions of the tags from the /tags ROS topic and outputs results to the topic, whose name is specified as a command-line parameter. For the left camera, the name of the topic is /tag_positions_l, and for the right camera - /tag_positions_r.

IMPORTANT: Tag and ball positions are published in the same topic, but in separate messages. planar_tracker reports a separate ID for every different tag. Tag IDs are integer numbers starting from 0. ID=-1 denotes the ball.

In order to run `planar_tracker`, run following commands:

```
roscd planar_tracker/

cd bin

python ./planar_tracker.py <topic name>
```

Here topic name should be `tag_positions_r` for the right caner and `tag_positions_l` for the left camera.

To perform the transform, `planar_tracker` must calculate the transformation matrix. All the necessary information for matrix calculation is stored in the `transformation_points.txt` file. The file contains a number of records; every record is a separate line with a following format:

```
<camera_x>\t\t<camera_y>\t\t<object_x>\t\t<object_y>\t\t<object_theta>
```

Here `<camera_x>` and `<camera_y>` are X and Y coordinates of the object (AR tag) in the image plane. `<object_x>` and `<object_y>` are coordinates of the object on the floor, in the Cartesian coordinate space. `<object_theta>` is the angle of the object relative to the camera.

Calibrating planar_tracker

In order to perform a transformation correctly, the `planar_tracker` has to be calibrated every time the position of the camera or field dimensions are changed.

Calibration involves creating the correct `transformation_points.txt` file. To calibrate `planar_tracker` please launch PS3 driver, `gscam` and `ar_recog` nodes first. Then run "`rostopic echo tags`" command to see the output of the `ar_recog` node. Clear contents of the existing `transformation_points.txt` file and fill it with new records.

For every record, place the tag on the position of the field with known X,Y coordinates in such a way that its real yaw angle is 0 (the tag must lay along the X axis). Enter these coordinates as `<object_x>` and `<object_y>` into the `transformation_points.txt` file. Then Look at the records being output to the `/tags` topic by the `ar_recog` program. Find X, Y and Z_rot values and enter them as `<camera_x>`, `<camera_y>` and `<object_theta>` values.

You should enter at least four records, one for each edge of the field. In addition, you might add more records for better accuracy. We are adding total 6 records to the `transformation_points.txt`: 4 points on the edge of the field and two inside the field.

Launching overhead localization system with the script

For convenience, we created scripts for starting/stopping the localization system. To launch the overhead localization, run `overhead_start_l.sh 04` on the PC that controls the left camera and `overhead_start_r.sh 04` on the PC that controls the right camera. Here "04" command line parameter denotes a video mode for the camera. Scripts will ask you for the root password. You will also have to configure camera's parameters: exposure and main gain.

To stop the overhead localization, please run the `overhead_stop.sh` script without any parameters.

Please take into attention that topic names can be different for right and left cameras. Configuration files, such as `colors.txt`, `cmvision` launch file and `transformation_points.txt` are different for the right and left cameras.

Obtaining localization information

Cameras are publishing information on balls and tags to the `/tag_positions_l` and `/tag_position_r` topics. These topics, however, are local to ROS cores running on PCs that control the cameras. In our environment, `/tag_positions_l` topic is created on the PC named "`bold`" that controls the left camera and `/tag_positions_r` – on the PC with name "`italic`" that controls the right camera. It is not possible to **directly** subscribe to that topic from the PC that controls your robot and runs a different ROS core.

Fortunately, there is a way to relay ROS topics from one ROS core to another. This can be done using a `foreign_relay` ROS node. This node can subscribe to the topic running on the different ROS core and create the topic with the same name on the "local" ROS core. That's what you have to do on your client computer. You have to relay to `/tag_positions_l` and `/tag_position_r` topics running on `bold` and `italic` to your computer and then to locally created, "relayed" topics on your ROS core.

Here we describe steps that **must be performed on your computer**. You are not supposed to run anything related to topic forwarding on bold or italic.

First, you need to add records for `bold` and `italic` into the `/etc/hosts` file on your computer. Suppose, the `bold` has an IP address `10.100.0.203` and `italic` – `10.100.0.204`. You can see real IP addresses by running a ping command "`ping <computername>`". Then run your favorite text editor and add following records to the /etc/hosts:

```
10.100.0.203     bold

10.100.0.204     italic
```

Please note you need to be root for that.

Second, install and build `planar_tracker` and `foreign_relay` nodes in your ROS installation. You will need the `planar_tracker node` to for message definitions, while `foreign_relay` will actually do topic relaying.

Third, make sure that overhead localization system is up and running. Make sure that bold and italic are publishing messages to `/tag_positions_l` and `/tag_position_r` topics respectively. Then run two instances of `foreign_relay`:

```
roscd foreign_relay
```

```
cd ./nodes/src
```

```
python ./foreign_relay sub <full name of the topic> <publisher’s ROS_MASTER_URI>
```

Here `<publisher’s ROS_MASTER_URI>` is the URI of the ROS core for the computer that publishes messages to the topic. `<full name of the topic>` will be the name of the topic that must be relayed, **including the leading slash (“/”).**

With the current configuration we have to run following commands (omitting roscd and cd):

```
python ./foreign_relay sub /tag_positions_l http://bold:11311
```

```
python ./foreign_relay sub /tag_positions_r http://italic:11311
```

Contacts

In case of any problems with the localization system, please contact ***Alex Tarvo*** (`alexta at cs dot brown dot edu`).