From LaserScan to a Simulink image

Submitted by sprinkle on Fri, 03/24/2017 - 6:07pm

In this tutorial we will show how to convert sensor_msgs/LaserScan into an image that can be used in Simulink. Once we have an image, we can use the Image Processing and Computer Vision toolboxes on the laser scan. The model for this tutorial can be found from the Files section, or in the github repository catvehicle_simulink (https://github.com/sprinkjm/catvehicle_simulink) A video that demonstrates and walks through this tutorial can be found at https://youtu.be/B8GBl_xoOTY

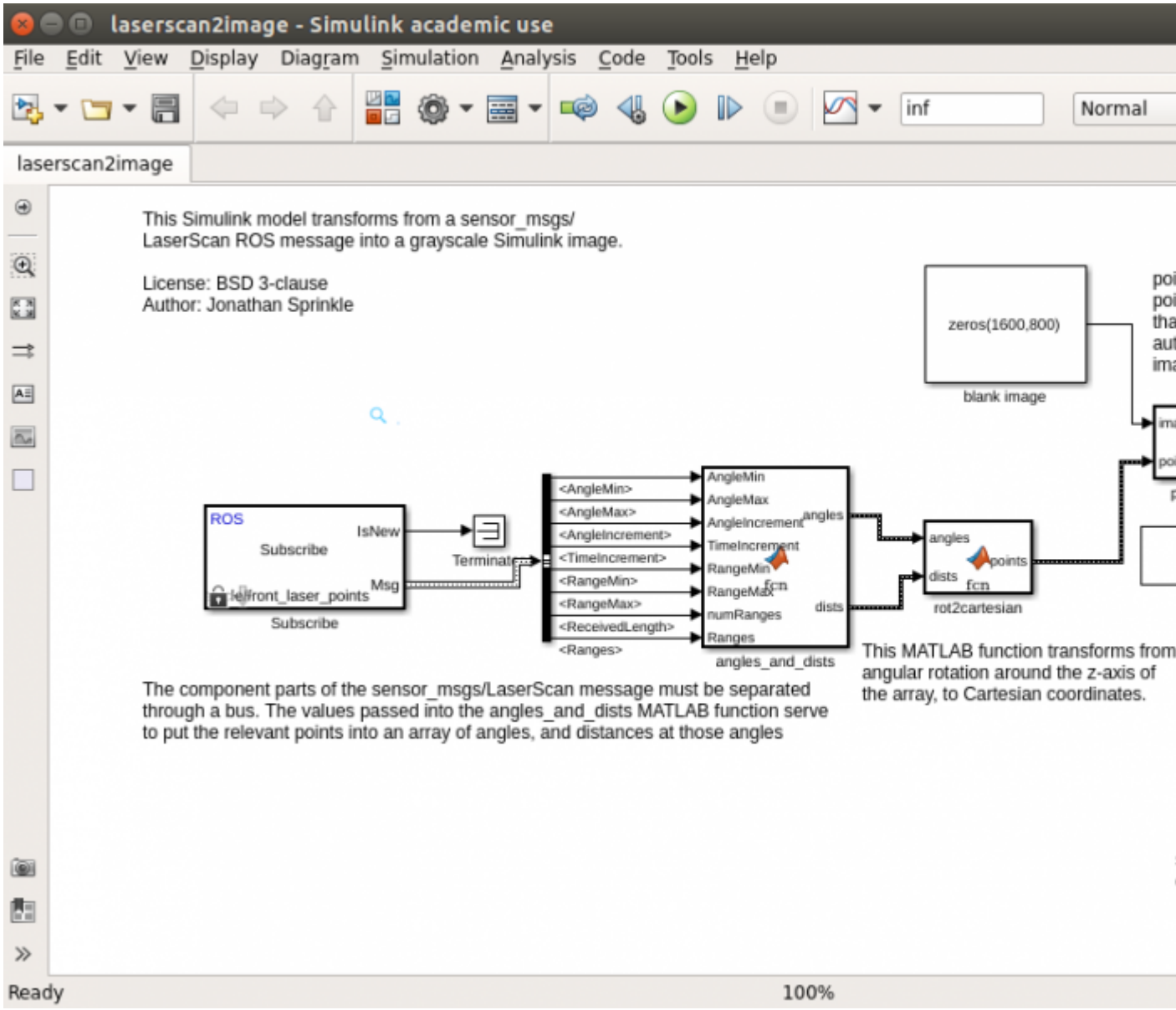
```
git clone https://github.com/sprinkjm/catvehicle_simulink.git
```

Motivation

The sensor_msgs/LaserScan message is an array of distances, where each element in the array increments a fixed angle. It is possible to use rudimentary algorithms to detect nearby points or adjacent points, and infer values from these. However, it is also possible to consider that the returned points constitute an image, and therefore image processing and other kinds of filtering could be applied to the problem in order to identify regions of interest and/or objects in the field, based on their position.

Model

Open up laserscan2image.slx from the github repository, and examine the different components of the model.



The model is decomposed into the following main pieces:

- Subscribe to laser data: this acquires laser information from the catvehicle's front laser scanner
- Unpack the sensors_msgs/LaserScan message: this transforms the ROS message into Simulink data for the ranges at each angle
- Coordinate transform: takes the range/angle data and places it into the Cartesian plane *within the frame of the laser scanner*
- Make an image: with the coordinates of each point, create a grayscale image that represents the scanner information

Below are some interesting pieces for each of these components.

Subscribe

The laser scanner is a 180 degree scanning laser, which provides 180 points with

ranges between 1m and 80m. These messages are defined by ROS, and so we need to do a little effort to get them into a Simulink format.

Unpack

When unpacking the data, we see how Simulink manages information; with the exception of "ReceivedLength" each one of these data values from the Bus are directly from the message. The received distances gives us an array of ranges:

```
dists
80
80
24.225.3
```

The angles_and_dists function, however, takes the single array of Ranges (which for this scanner should be 180 points) and creates a second array, which is the set of angles for each of these points. For a brief example, it would look like:

```
angles          dists
-pi             80
-pi + pi/180    80
-pi + 2pi/180   24.2-pi + 3pi/180   25.3
```

This allows each index of angles to correspond to the distance at that angle.

Coordinate Transform

This function is very simple, it multiplies the distances times the cosine (for x) and sine (for y) of the angle. This produces the location of each point in the Cartesian plane, within the laser sensor's frame of reference.

```
xs = dists.*cos(angles);
ys = dists.*sin(angles);points = [xs,ys];
```

Make an Image

This is, by far, the most complex of the MATLAB function blocks, but it produces a Simulink-compatible image which is a projection of the (x,y) points into a (X,Y) image. By design, we expand each meter into 10 pixels in the image. Thus, our (x,y) of (80,160) possible points in meters, is mapped into a (800,1600) range for (X,Y). We create an image that is, therefore, X by Y pixels large, with the idea that the vehicle is looking from halfway down the image, to the right.

The coordinate system of the laser is for -y to be "up" in the image, and positive x to be "right" in the image, with (0,0) being halfway down the image, all the way to the left. Since a point (0,0) in the (image would be x=0 and y=-80, we can convert from (x,y) into (X,Y) by (X,Y) = (0,80)-(x,y). This is given in the code for the function block as:

```
% add 80 to all the y points
pts(:,2) = 80 - pts(:,2);
```

And since we want to make 10 pixels for each meter:

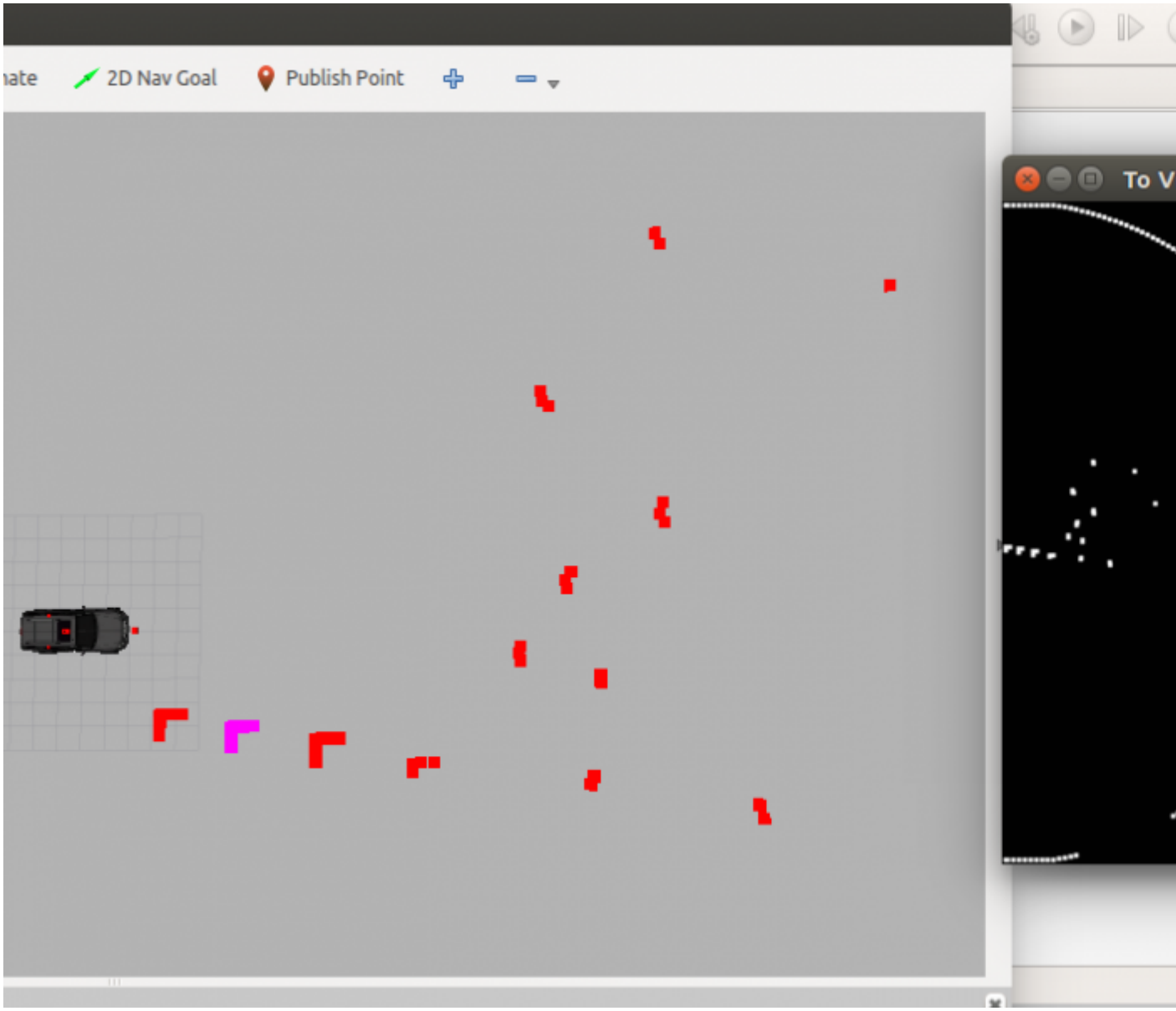
```
% since we are building an 800x1600 image of the 80m (x) by 160m (y), we
% will multiply the data by 10 and take the floor in order to show
% occupancy near that pointpts = 10*pts;
```

And then, the only thing to do is to transform each point into the image, by changing the default color (black) into white.

```
% iterate over these points, and set up a primitive grid map
for ii=1:len
    ii_x = pts(ii,1); % the x point is in the first column
    ii_y = pts(ii,2); % the y point is in the second comlumn
    if(ii_x > 800)
        disp('ru roh');
    end
    index_x = min(max(10,floor(ii_x)),800-10);
    index_y = min(max(10,floor(ii_y)),1600-10);
    % make a square here, of size 10x10 pixels
    % we must swap the (x,y) to (y,x) for MATLAB images
    img(index_y-5:index_y+5,index_x-5:index_x+5) = ...
        img(index_y-5:index_y+5,index_x-5:index_x+5) | 255;end
```

Results

Now, you can see the laser returns as an image which can be used for image processing. Start up your favorite worldfile, and compare what you're seeing in rviz with what the laser returns show you in the image. Here you can also see where the car is sitting in rviz, and match the points on the rviz plane with points in the Simulink "To Video Display" image.



What's next?

Now you're ready to use some image processing on the laser images!

Backlinks:

- Tutorials

catvehicle_v2