Tracking a Ground Vehicle from an Autonomous Helicopter

Mark Woodward

The goal of my project was to build a system that can track and follow a moving ground vehicle from an autonomous helicopter. There were four main components to the system 1) Locating the vehicle in an image 2) Transforming the image location to a world location 3) Filtering the world estimates and 4) Generating trajectories for the helicopter to follow.

The major hardware components for this project are shown in Figure 1. They are the X-Cell Tempest, a 10th scale R/C truck, and a wireless vision system.

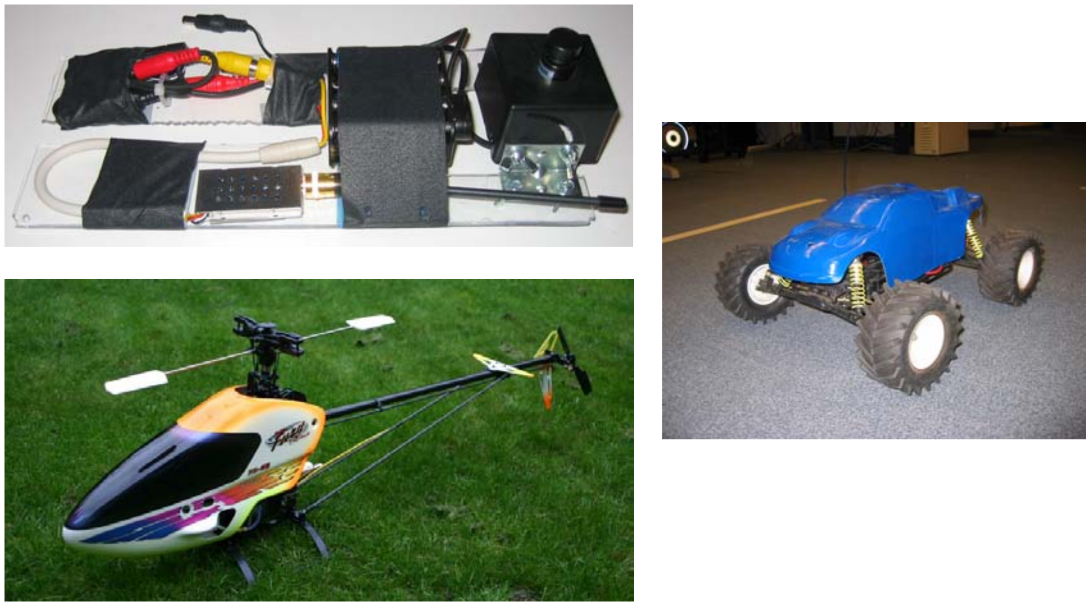
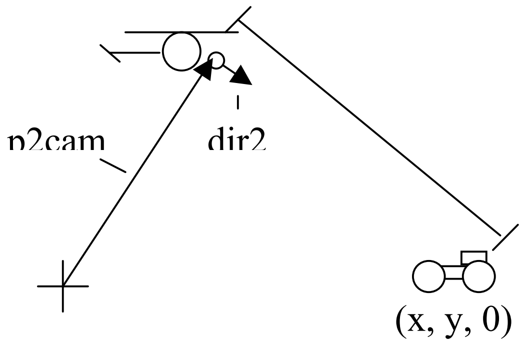


Figure 1.

The vehicle is located using a recursive color-centroid algorithm. Figure 2. contains the pseudo code for the algorithm. At a high level, the centroid of all pixels from a specific window that match a filter is computed, then a new smaller window is generated with it's center at the previous centroid. This recurses until a minimum window is reached. This recursive technique was developed because it is very tolerant to outliers.

Once the vehicle is found in image coords, it projected onto the ground plane to find the location of the vehicle in world coordinates. The general idea is depicted below in figure 3. This project involves a number of different transformations: image to camera, camera to helicopter, and helicopter to world.

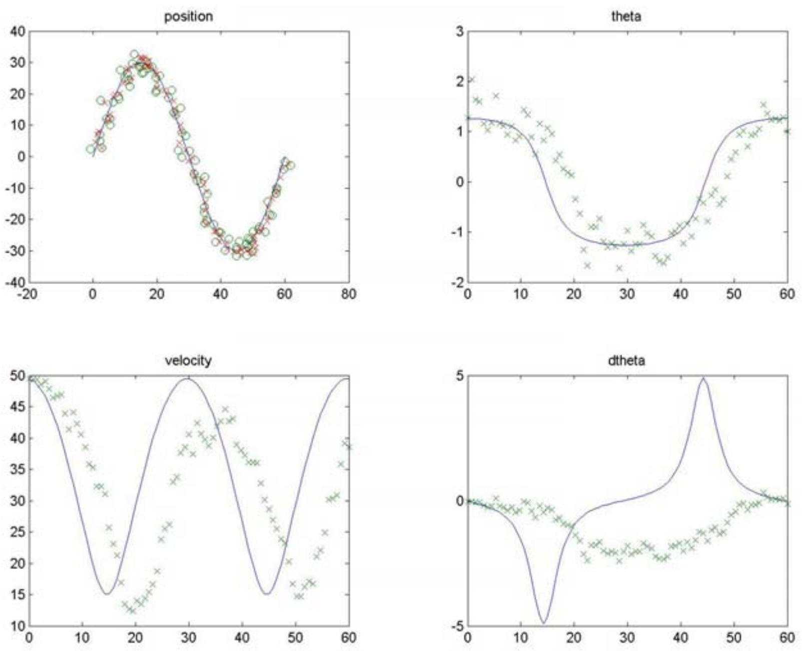
```
convert image to HSV
loop {
  for all pixels in window {
    if pixel satisfies parameters {
      x_total += x;
      y_total += y;
      matches++;
    }
  }
  car_x = x_total/matches
  car_y = y_total/matches

  if (window < min_window)
    break

   window_center = (car_x, car_y)
   window = window/2
}
```



x = p2camera.x + t*dir2car.x
y = p2camera.y + t*dir2car.y
0 = p2camera.z + t*dir2car.z

Figure 2. **Figure 3.**

I then use a Kalman filter to estimates the state of the car. The Kalman filter is used for two main reasons, 1) because the single estimates of the car's location can be noisy, and 2) because the derivative information of velocity, orientation, and angular velocity are useful when generating intercept trajectories. Figure 4 shows the Kalman filter run in simulation, along with the motion model used for the car.



```
     x = x + v*cos(theta)*del_t
     y = y + v*sin(theta)*del_t
     v = v
 theta = theta + dtheta*del_t
dtheta = dtheta
```

Figure 4.

Presently, a PD hover controller is used to command the helicopter to locations near the tracked target. As we move to more aggressive controllers, the helicopter will pitch, yaw, and roll as it approaches the target. Due to these motions, the camera can loose view of the target. I will next implement an A* search for trajectories that, when followed by a given controller, will trade off maximize the time that the vehicle is in view and minimize the time to reach the target. The heuristic for this search will be learned in simulation. Figure 5 outlines the major points of this approach.

- **A***
    - State:
      S = (S_heli, S_car, S_heli – S_car)
    - Cost of each node:
      C(n) = a*Time + b*(Percent time not in view)
    - Heuristic for each node:
      H(S) = Theta*S
- **Learning the heuristic**
    - Run in Simulation
    - After each step, update theta to minimize error
      error = (H(s_t+1) – H(s_t)) – (C(s_t+1)-C(s_t))

Figure 5.

In summary the system successfully tracks and follows a moving ground vehicle. Future work will allow the system to more aggressively approach and follow the target.

I would like to thank my advisor Andrew Ng, and a number of other people who have made this project possible through advising and contributing code: Adam Coates, Peter Abbeel, Sudeep Banerjee, Patrick Mihelich, and Alok Deshpande.