Visual Servoing Control Design Report

ENN586 - Decision and Control

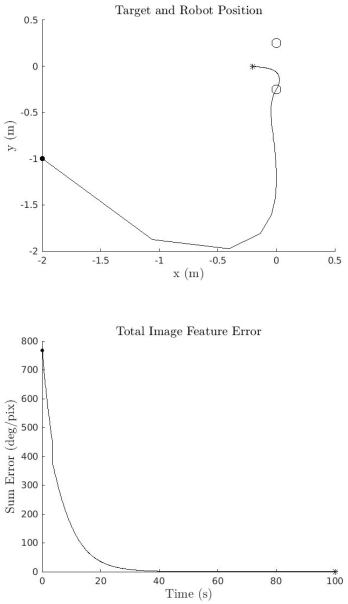
Cameron Stroud

Contents

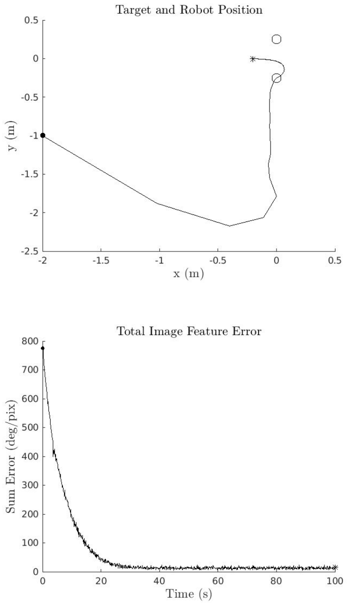
The Problem

- Test Cases
  - 1: No error, no measurement noise case
  - 2: Measure error is ON. No other error sources.
  - 3: Range error is ON. No other error sources.
  - 4: Feature error is ON. No other error sources.
  - 5: Wind error is ON. No other error sources.
  - 6: Measurement, range, feature, Wind errors are all ON.
- Assumptions
  - Target Features are hovering drones
    - Eg. swarm scenario

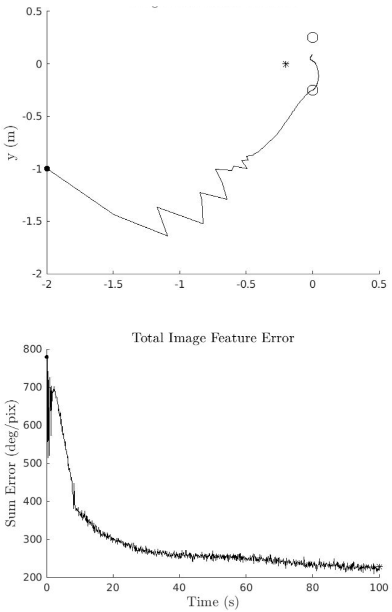
Existing Visual Servoing Controller



1-1



2-1



6-1

Proposed Design

PID Controller

+ Add robustness to control design, stability in wider range of scenarios

Kalman Filter

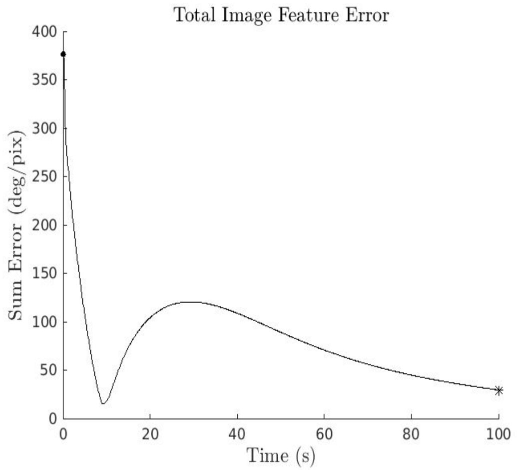
+ Mitigate error introduced by noisy measurement

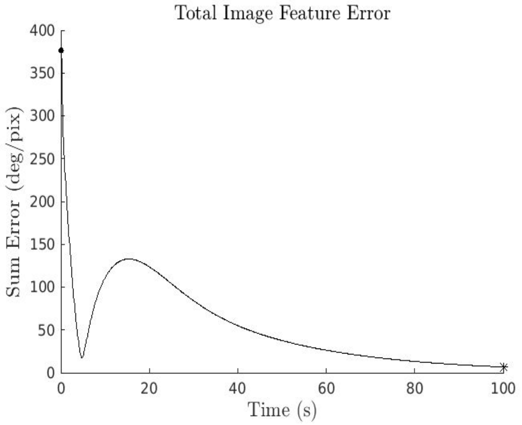
PID Control

Based on provided PID Controller

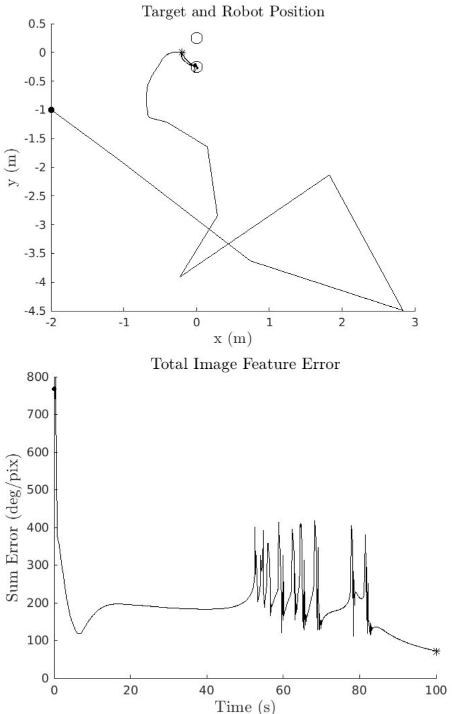
$$K_p = 0.3$$
$$K_i = -0.0009$$
$$K_d = -0.3$$

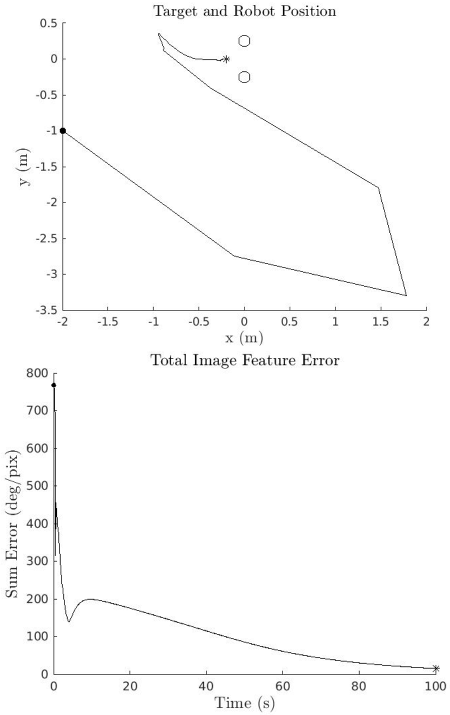
Start case 4:



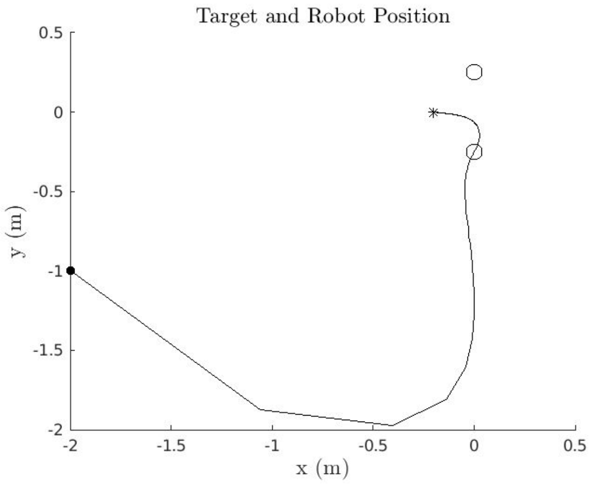


PID Control: Base vs New (Test Case 1-1)

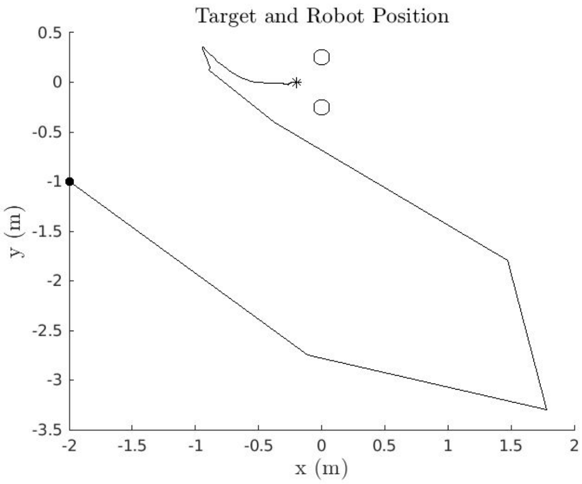




PID Performance: Test Case 1, Start Case 1

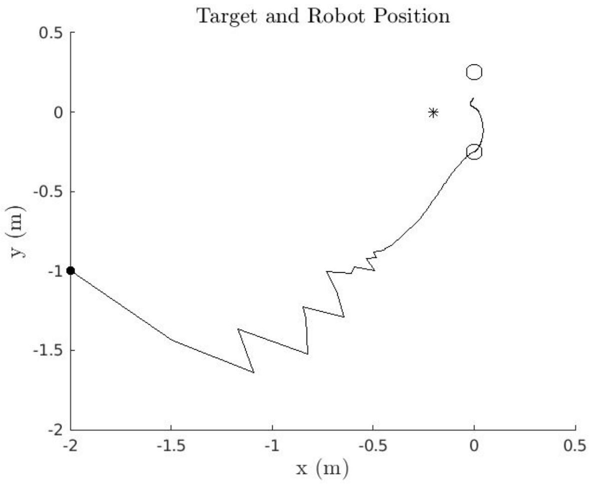


Existing Controller

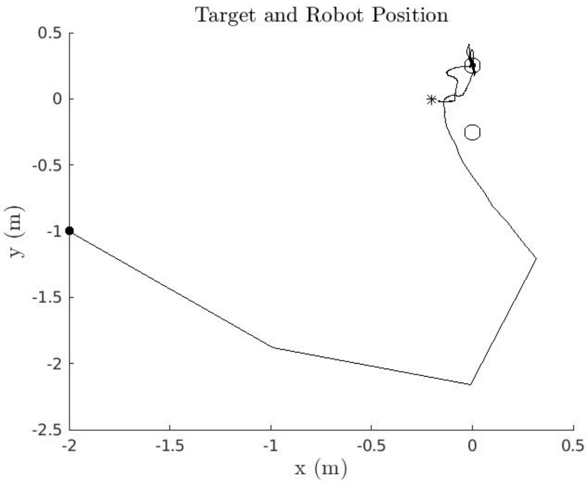


PID Controller

PID Performance: Test Case 6, Start Case 1

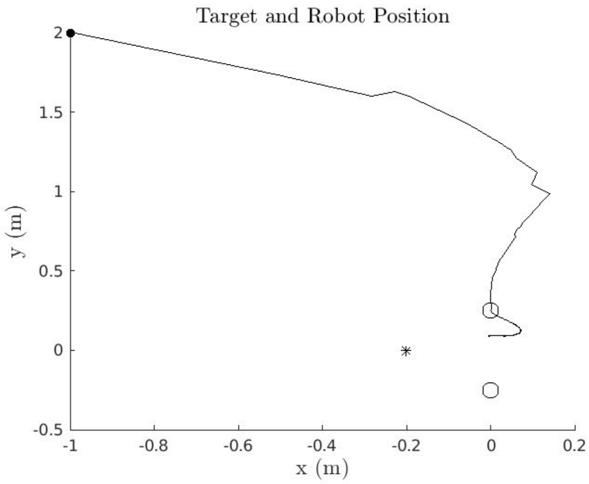


Existing Controller

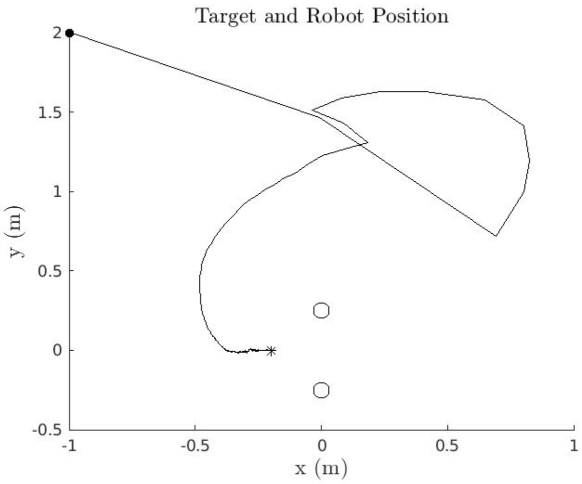


PID Controller

PID Performance: Test Case 6, Start Case 2

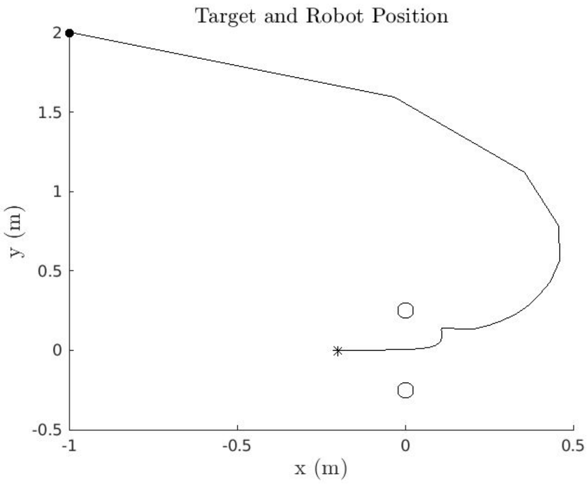


Existing Controller

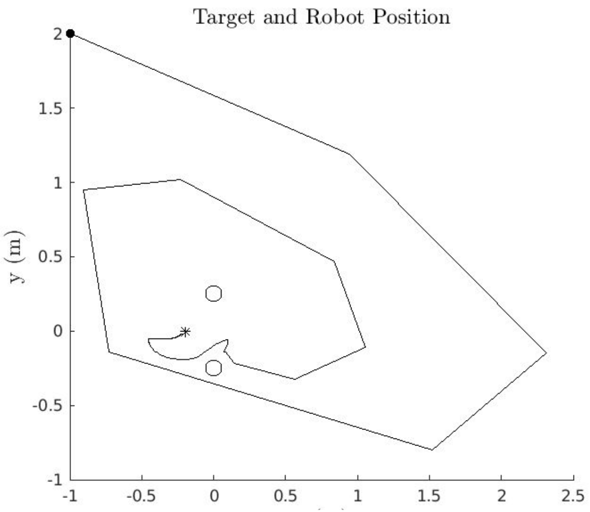


PID Controller

PID Performance: Test Case 1, Start Case 2



Existing Controller



PID Controller

The Kalman Filter

```
% Kalman Filter
YourVariables.kf.A = eye(8);
YourVariables.kf.R = 25;
YourVariables.kf.Q = diag(0.01*ones([8 1]));
YourVariables.kf.H = eye(8);
YourVariables.kf.P_prev = 1000 * eye(8);
YourVariables.kf.B_prev = 0.01*rand([8 6]);
YourVariables.kf.u_prev = zeros([6 1]);
YourVariables.kf.predict = @kalman_predict;
YourVariables.kf.measure = @kalman_measure;

x_0 = 5*(rand([8 1]) - rand([8 1]));
YourVariables.kf.e_est = x_0;
```

$$A = I_{8\times 8}$$
$$R = 25$$
$$Q = 0.01 \times I_{8\times 8}$$
$$H = I_{8\times 8}$$

Kalman Filter Performance - No Error

	Total Control Energy/Effort (actual) (J)	Final Robot Position Error (m)	Total Image Feature Error (deg/pix)
Start Case 1	174.1013	0.00011726	0.18196
Start Case 2	95.6084	1.5437e-06	0.0023358
Start Case 3	53.8423	4.4557e-06	0.0068299
Start Case 4	19.2171	2.3142e-06	0.0035573

Table 4: Test Case 1 Kalman Filter-only Performance

	Total Control Energy/Effort (actual) (J)	Final Robot Position Error (m)	Total Image Feature Error (deg/pix)
Start Case 1	170.2261	1.3837e-07	0.00019657
Start Case 2	96.2538	1.628e-07	0.0002306
Start Case 3	54.1759	7.0003e-08	8.7361e-05
Start Case 4	19.528	9.4745e-08	0.00011181

Table 1: Test Case 1 Provided System Performance

Kalman Filter Performance - Measurement Error

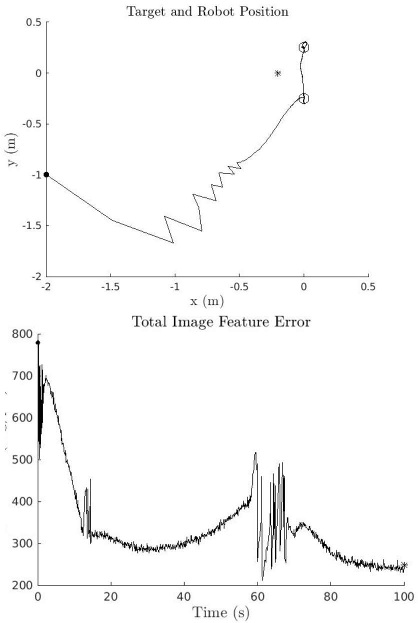
	Total Control Energy/Effort (actual) (J)	Final Robot Position Error (m)	Total Image Feature Error (deg/pix)
Start Case 1	183.8961	0.0056205	16.057
Start Case 2	94.3981	0.0054807	16.1138
Start Case 3	91.9945	0.005498	16.1069
Start Case 4	23.8046	0.005494	16.1085

Table 8: Test Case 2 Kalman Filter-only Performance

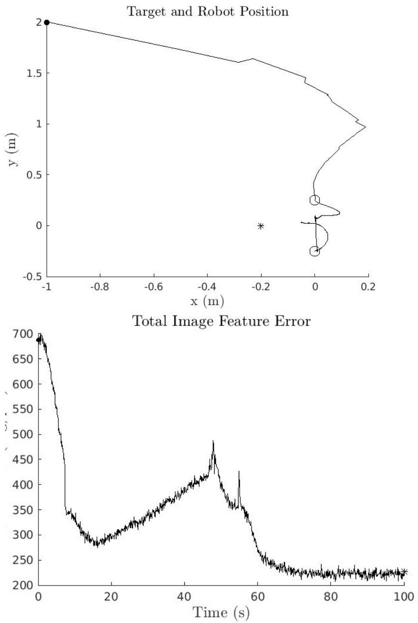
	Total Control Energy/Effort (actual) (J)	Final Robot Position Error (m)	Total Image Feature Error (deg/pix)
Start Case 1	180.063	0.0050248	16.3357
Start Case 2	95.1837	0.005025	16.3356
Start Case 3	92.4306	0.0050249	16.3358
Start Case 4	23.9756	0.0050249	16.3358

Table 5: Test Case 2 Provided System Performance

Kalman Filter Performance: 6-1, 6-2

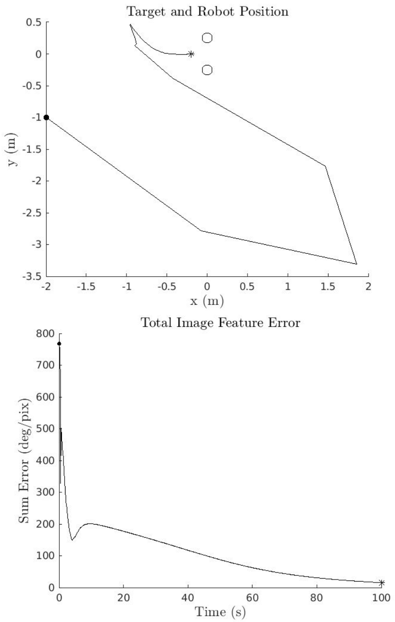


6-1

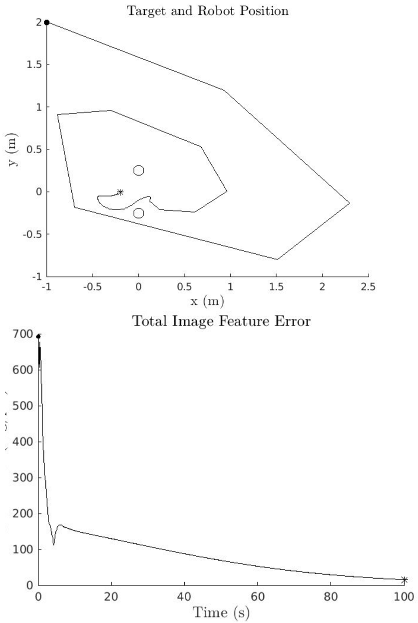


6-2

Full System Performance: 1-1, 1-2

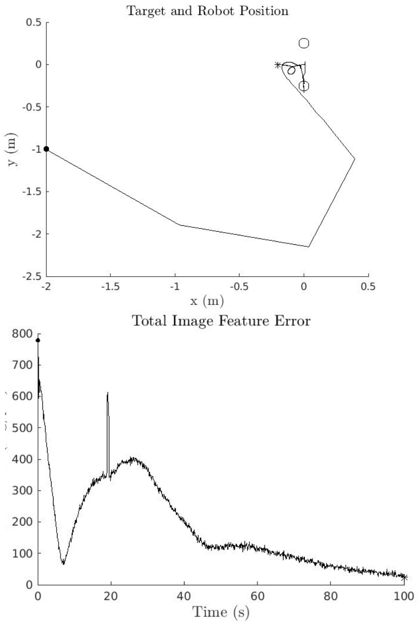


1-1

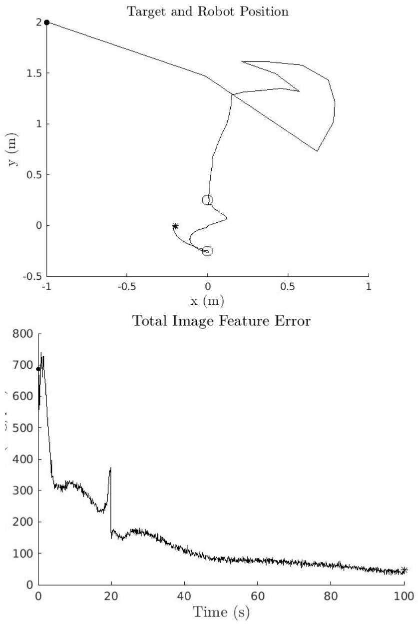


1-2

Full System Performance: 6-1, 6-2



6-1



6-2

Conclusion

- PID offers higher robustness at cost of efficiency
- Kalman Filter has major model mismatch - extreme rework or alternative filtering
- Need further consideration of range error