Image-Based Visual Servoing of Unmanned Aerial Vehicle

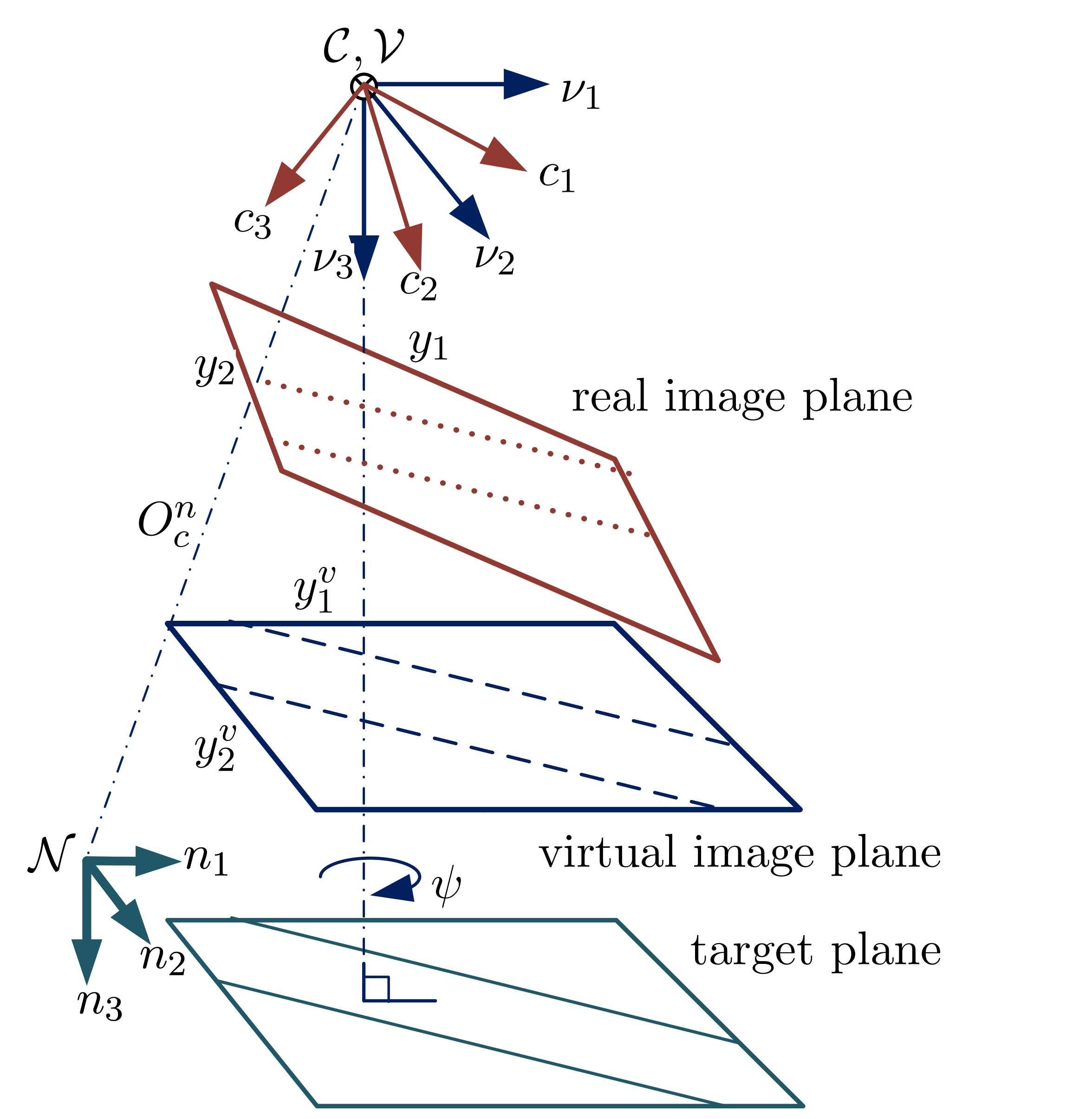
Zhijun Xue, Awais Muhammad, Alan Lynch

zxue2@ualberta.ca, mrafique@ualberta.ca, alan.lynch@ualberta.ca

Contribution: Line Following *using* Image-Based Visual Servoing(IBVS)

Implement of control algorithm for IBVS, which include modifying px4 firmware, creating ROS packages, image processing on nVIDIA Jetson platform.

Frames



Navigation frame $\mathcal{N} = \{n_1, n_2, n_3\}$, Real camera frame $\mathcal{C} = \{c_1, c_2, c_3\}$, Virtual camera frame $\mathcal{V} = \{\nu_1, \nu_2, \nu_3\}$, real image frame $\{y_1, y_2\}$ and virtual image frame $\{y_1^v, y_2^v\}$

The frames are related to each other by unitary rotation matrices $R \in SO(3)$.The rotation matrices for converting any point or vector from real camera frame to virtual camera frame is given by

$$R_c^v = R_\theta R_\phi. \tag{1}$$

Quadrotor UAV Dynamics

The quadrotor velocity kinematics in virtual camera frame derived by multiplying nominal body frame kinematics presented in [1] with R_c^v, and the angular dynamics in real camera frame are given as follows

$$\dot{v}^v = -\left[\dot{\psi} e_3\right]_\times v^v + g e_3 + \frac{F^v}{m} + \delta \tag{2a}$$

$$\dot{\eta} = W(\eta)\omega^c \tag{2b}$$

$$\dot{\omega}^c = -J^{-1}[\omega^c]_\times J\omega^c + J^{-1}\tau^c \tag{2c}$$

simplify F^v (2) as:

$$F^v = R_c^v F^c = -K_T f_T \begin{bmatrix} \sin\theta\cos\phi \\ -\sin\phi \\ \cos\theta\cos\phi \end{bmatrix}. \tag{3}$$

Line Modeling and Dynamics

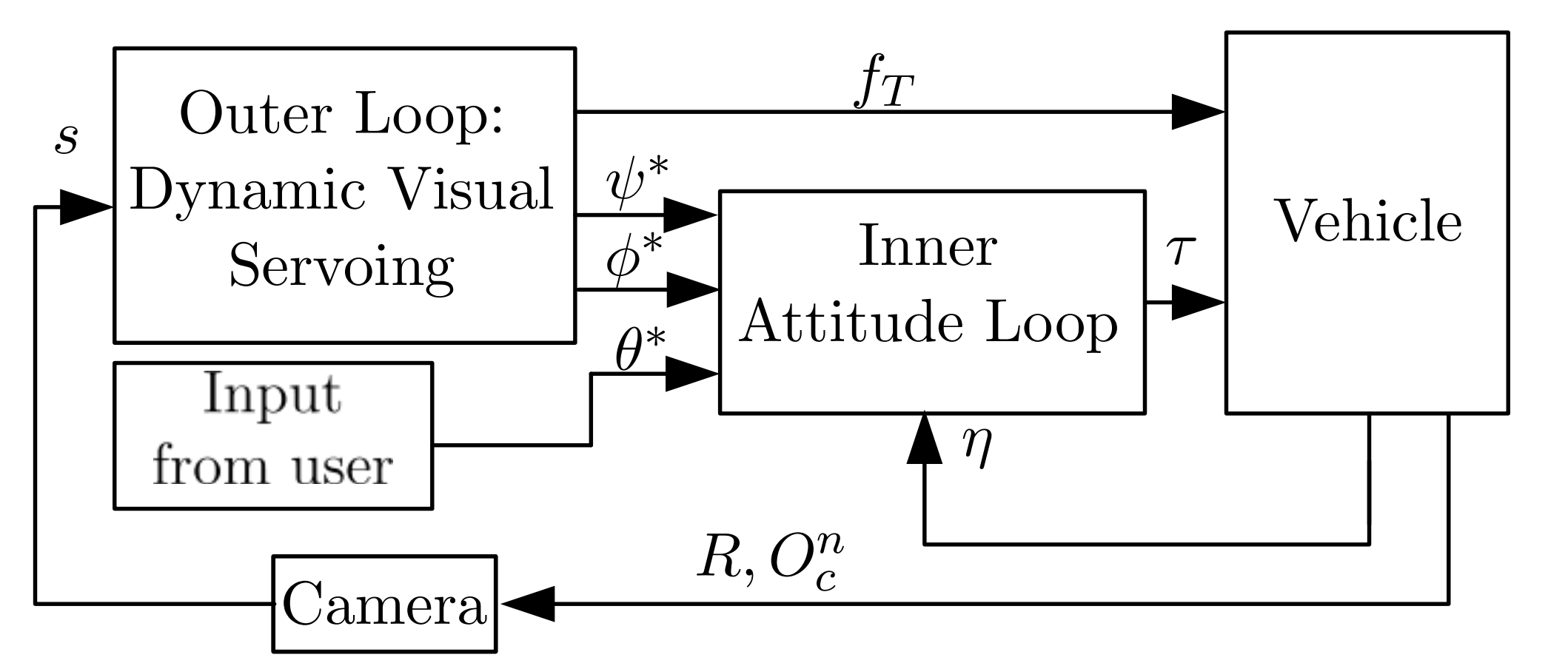
Consider a static 3D point in the camera frame represented by a vector $P = (X_1, X_2, X_3)^T$, its projection $P^c = (y_1, y_2)^T$ onto the 2D image plane is given by

$$p = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \lambda \begin{bmatrix} \frac{X_1}{X_3} \\ \frac{X_2}{X_3} \end{bmatrix} \tag{4}$$

The line feature kinematics for kth line:

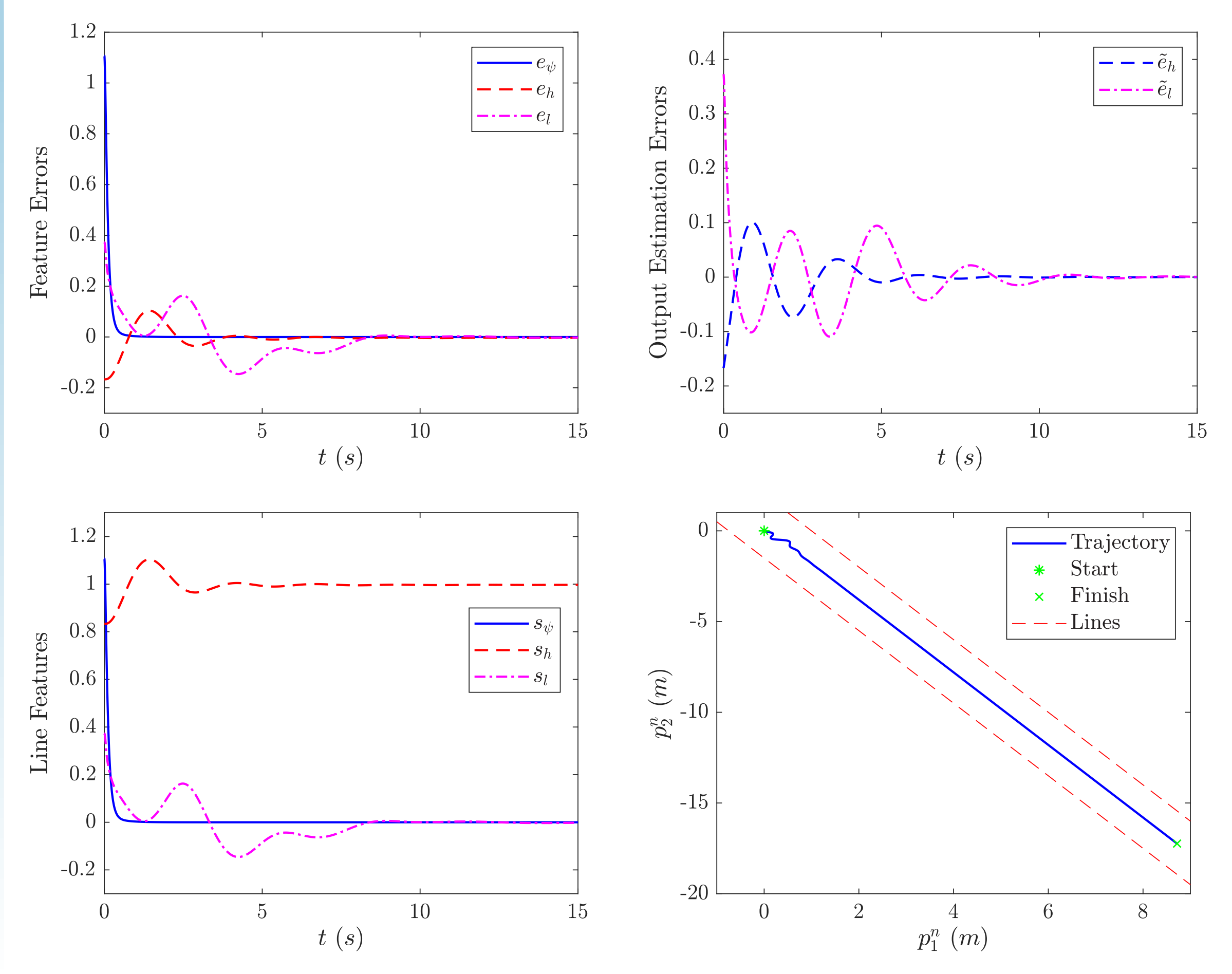
$$\dot{\alpha}_k^v = \dot{\psi}$$
$$\dot{\rho}_k^v = \frac{-1}{X_3^v}\begin{bmatrix} \lambda\sin\alpha_k^v & \lambda\cos\alpha_k^v & -\rho_k^v \end{bmatrix}\begin{bmatrix} v_1^v \\ v_2^v \\ v_3^v \end{bmatrix} \tag{5}$$

Control Structure

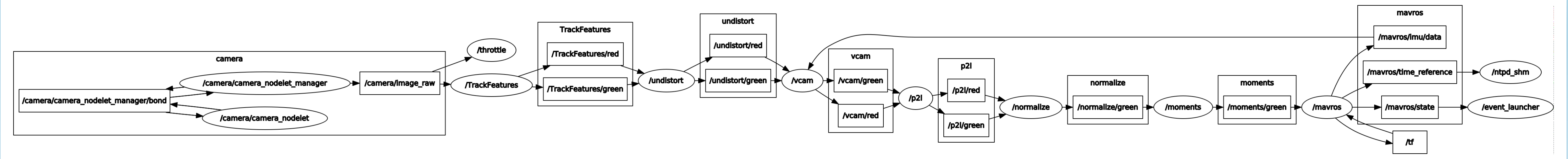


Inner-outer loop control structure for line

Simulation Results



Create ROS Package to process images and communicate with Quadrotor UVA



The work flow of our ROS package is showed in the picture, the rectangles represent ROS topic and the ovals represent ROS topics. In the project Feature Detector, which is our ROS package to process images, calculate movements and passing the movements feature to the drone. Here are few things that are meaningful to address:

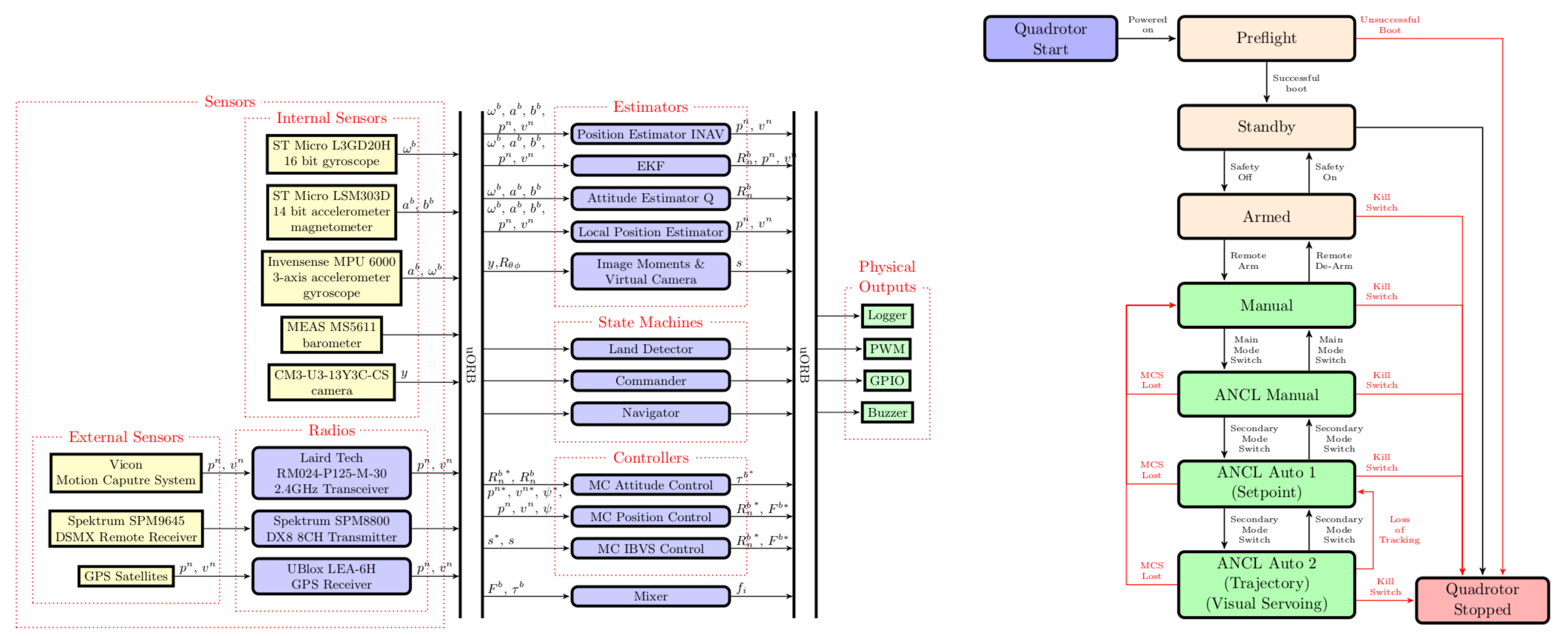
1. All publications and the subscriptions in the project are all handled by the function:

`void getSubscribersAndPublishers()`

Source code: `http://gitlab/gfink/feature_detector/blob/master/feature_detector/include/feature_detector/Msgs.hpp`

2.The .yawl files in the launch fold, setting the parameters of the nodes, for exampl vcam-point.yawl, setting the node vcam to subscript to topic /undistort/green and /mavros/imu/data, publish to /vcam/green.And the message type are object and imu, whose definitions can be find on Object.msg.

Controller Implement on PX4



Setting up develop environment on Emacs

Emacs is a vary powerfull tool for me to understand and develop with these open source project. Here are the emacs packages I used for C++ development.

Package	Description
which-key	brings up contextual key binding help
auto-complete	just as the name says
flycheck	in line, live syntax checking
yasnippet	snippets and expansion
ggtags	source code navigation

More details:`http://cestlaz.github.io/posts/using-emacs-32-cpp/#.W3u5gHZKhWM`

References

[1] Awais Muhammad, Alan Lynch Output-feedback Image-based Visual Servoing

[2] Hui Xie Dynamic Visual Servoing of Rotary Wing Unmanned Aerial Vehicles PhD thesis, 2016