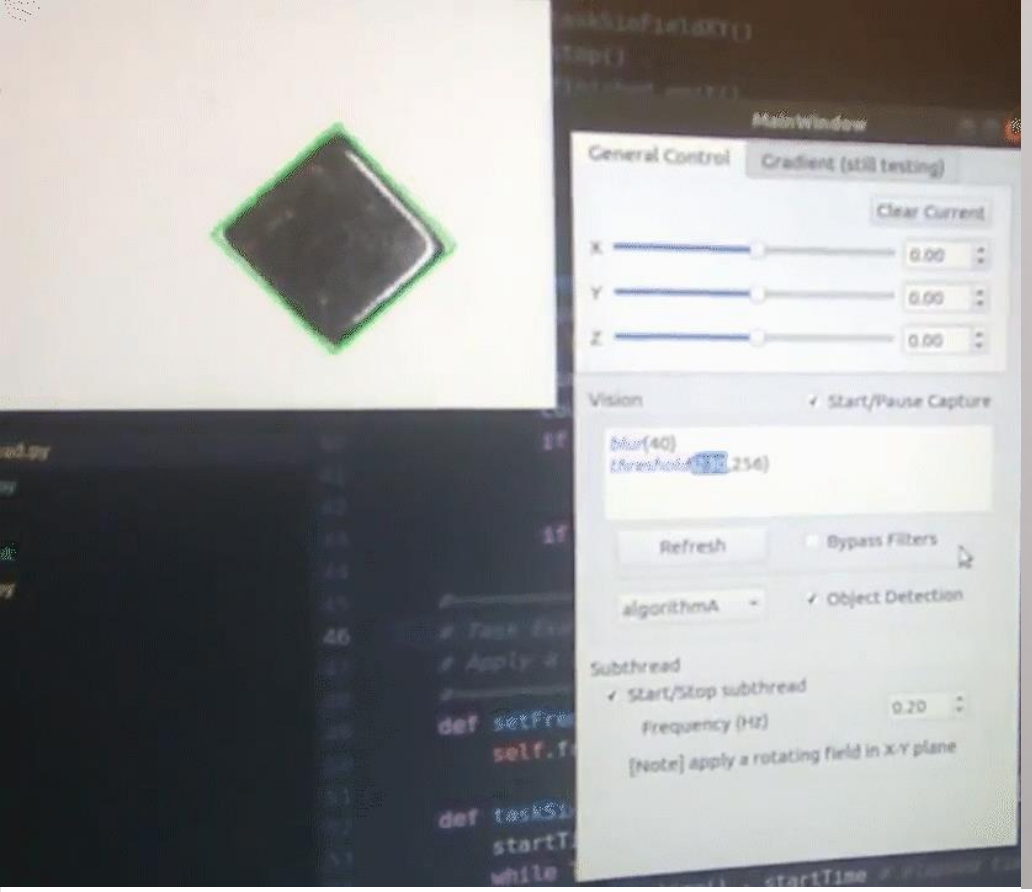


Guide to the Python-based Coil System

https://github.com/atelier-ritz/CoilSystemPython

Table of contents

Dependencies

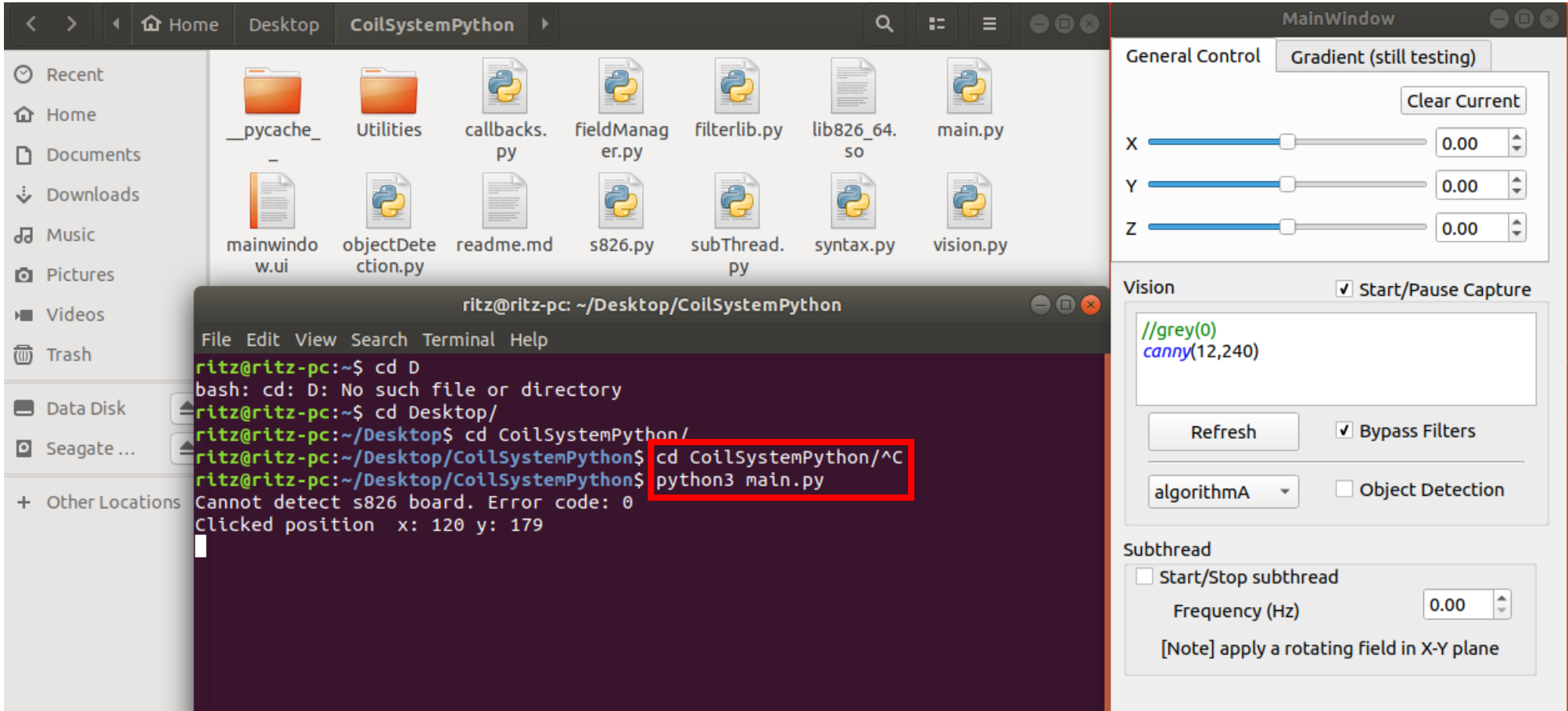
====== Tested on Ubuntu 17.10 ======

- Python 3.6 pre-installed in Ubuntu 17.10
- PyQt5 pip3 install pyqt5
  - *What is PyQt https://riverbankcomputing.com/software/pyqt/intro*
- Opencv pip3 install opencv-python, pip3 install opencv-contrib-python
- Pydc1394
  - *Firewire camera module https://github.com/jordens/pydc1394*
- Qt-designer sudo apt-get install qt4-designer
  - *GUI designer*

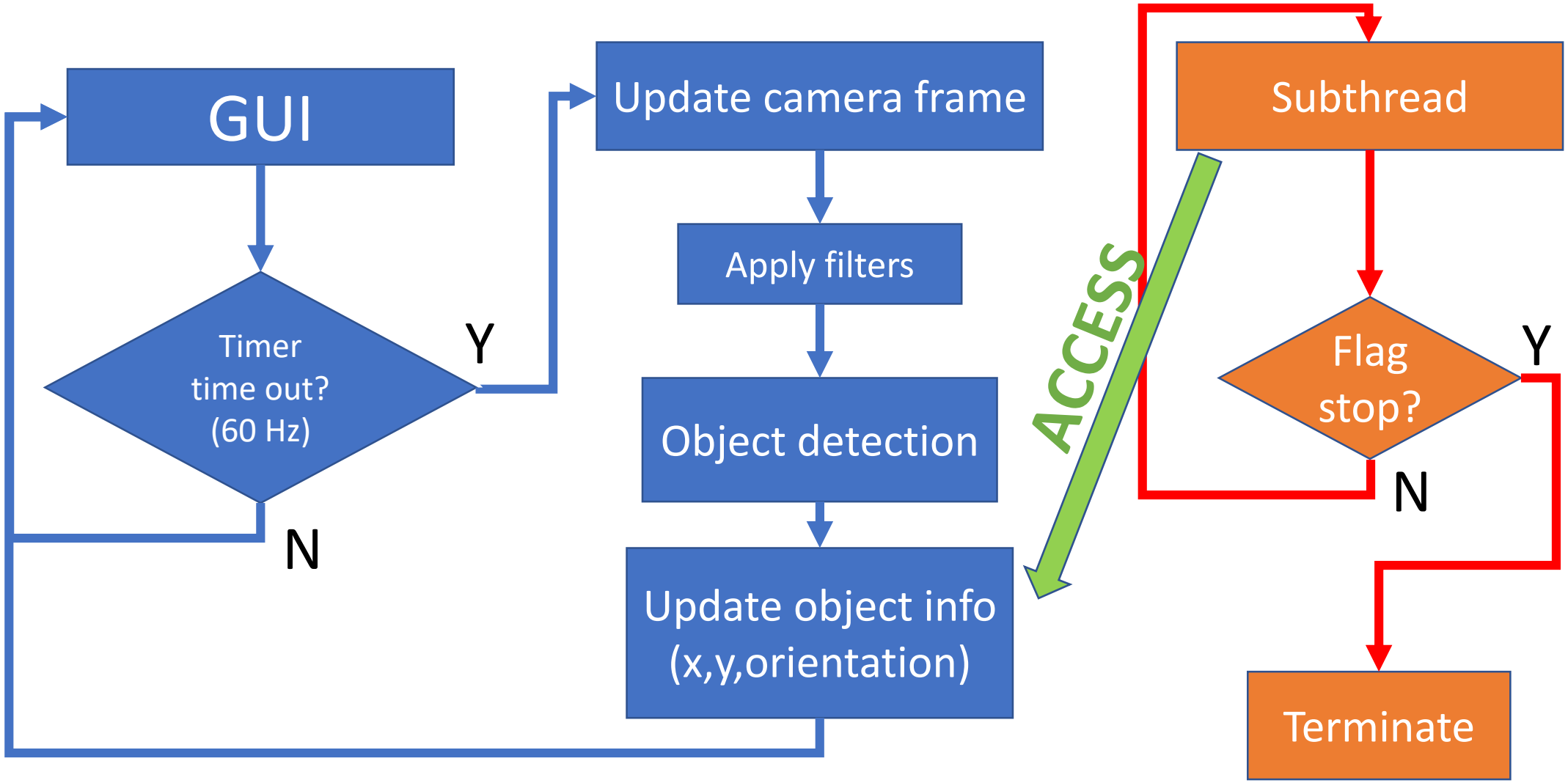
Usage

Go to the working directory.

Run “python3 main.py”



Program structure



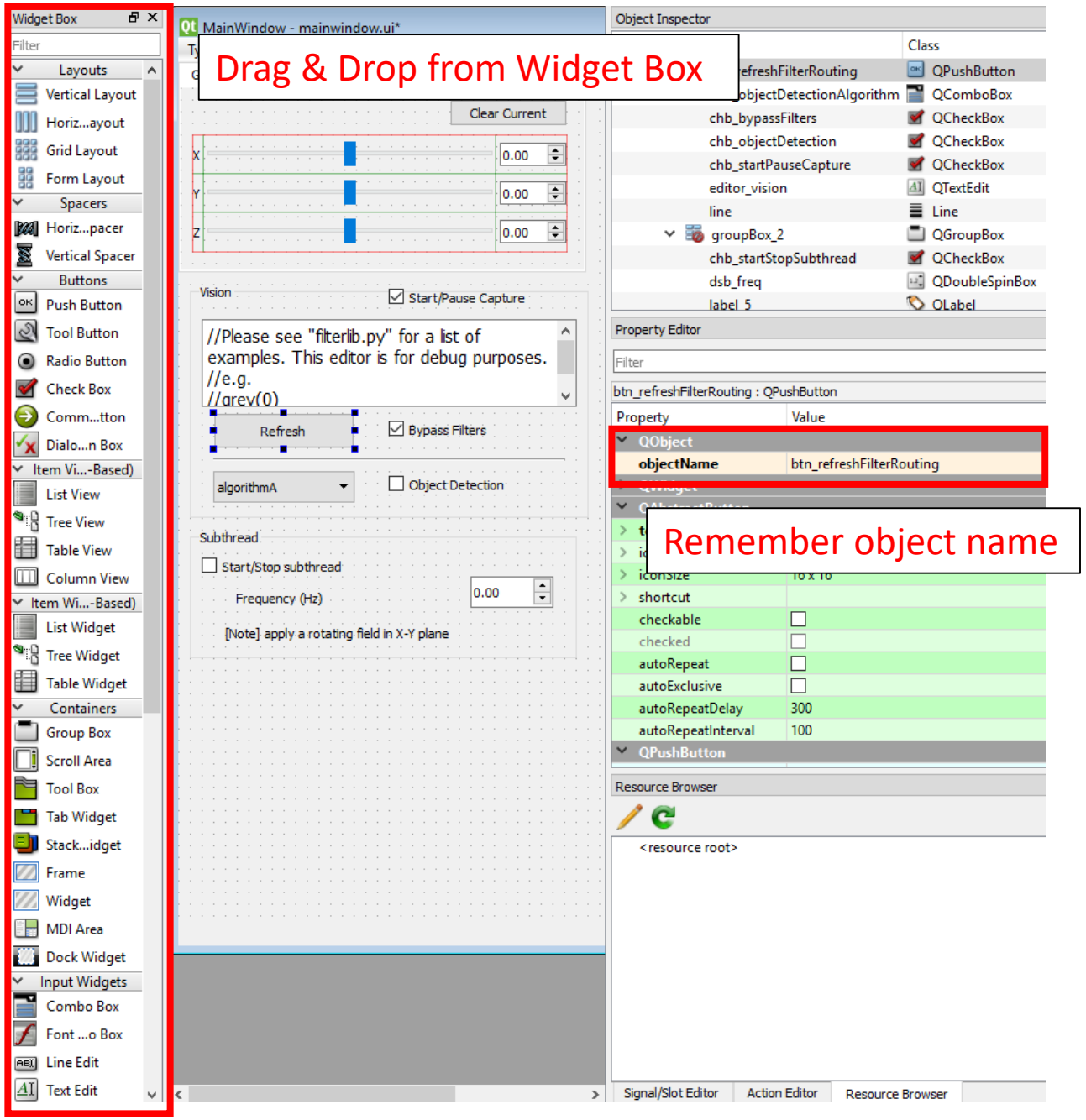
Program structure

```
main.py

callbacks.py  Add your code here
|
|
└──syntax.py [highlight the keywords in GUI editor_vision]
|
└──fieldManager.py [send commands to s826; store XYZ field strength]
|       |   s826.py [control s826 I/O]
|
|
└──visoin.py [capture frames; apply filters; detect objects]
|       |   filterlib.py [define filters] Add your code here
|       |   objectDetection.py [define object detection algorithms] Add your code here
|
|
└──subthread.py [run multithreading tasks] Add your code here
```

Modify GUI

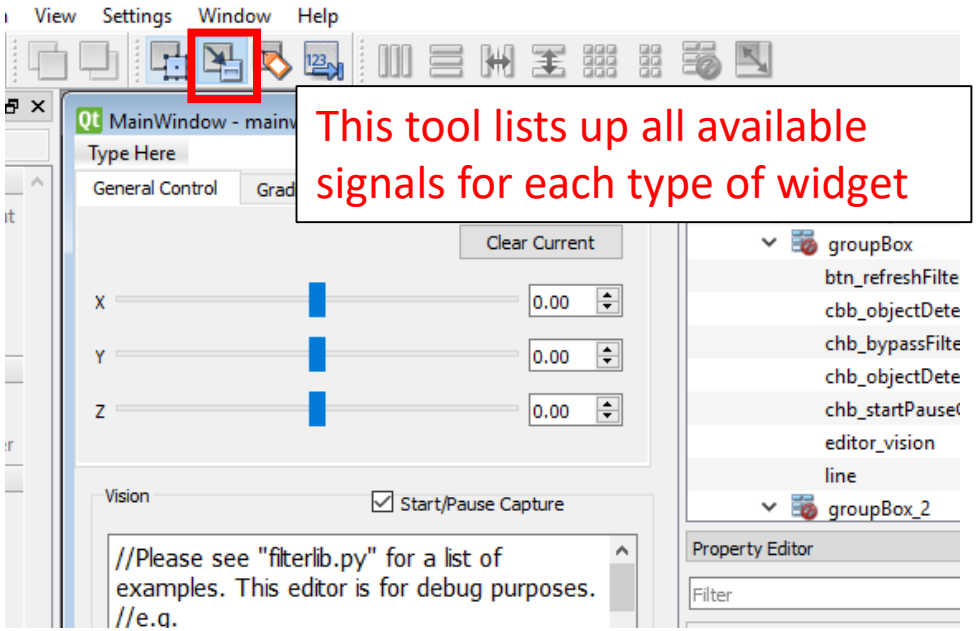
1. Open “Mainwindow.ui” with qt-designer.



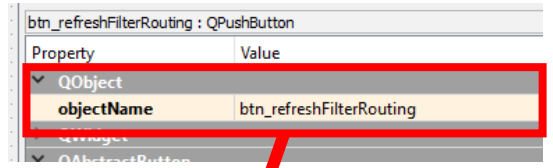
Modify GUI

1. Open “Mainwindow.ui” with qt-designer.
2. Oepn “callbacks.py” and edit connectSignals()

More about <signal> and <slot> http://pyqt.sourceforge.net/Docs/PyQt4/new_style_signals_slots.html



This tool lists up all available signals for each type of widget



<objectName>

```
def connectSignals(self):
    # General Control Tab
    self.dsb_x.valueChanged.connect(self.setFieldXYZ)
    self.dsb_y.valueChanged.connect(self.setFieldXYZ)
    self.dsb_z.valueChanged.connect(self.setFieldXYZ)
    self.btn_clearCurrent.clicked.connect(self.clearField)
    self.dsb_xGradient.valueChanged.connect(self.setFieldXYZGradient)
    self.dsb_yGradient.valueChanged.connect(self.setFieldXYZGradient)
    self.dsb_zGradient.valueChanged.connect(self.setFieldXYZGradient)
    # Vision Tab
    self.highlighter = syntax.Highlighter(self.editor_vision.document())
    self.chb_bypassFilters.toggled.connect(self.on_chb_bypassFilters)
    self.chb_startPauseCapture.toggled.connect(self.on_chb_startPauseCapture)
    self.btn_refreshFilterRouting.clicked.connect(self.on_btn_refreshFilterRouting)
    # object detection
    self.chb_startStopSubthread.toggled.connect(self.on_chb_startStopSubthread)
    self.dsb_freq.valueChanged.connect(self.thrd.setFreq)
```

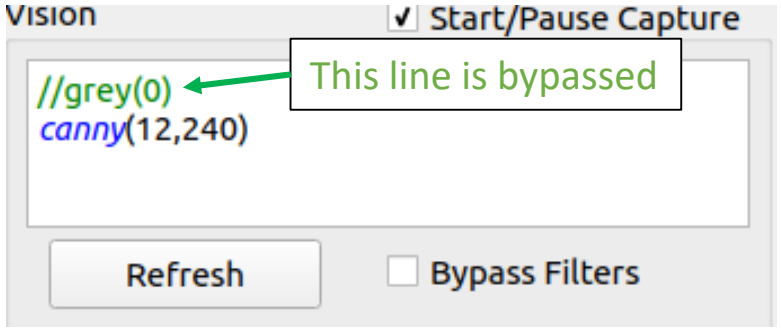
self.<objectName>.<signal>.connect(<slot>)

Image filters

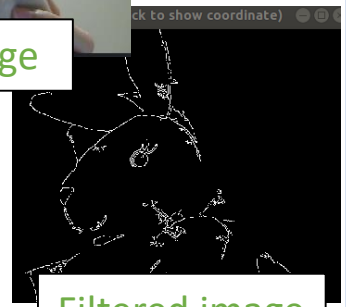
Note:
ONLY include alphabets,numbers, and underbars in your filter name.

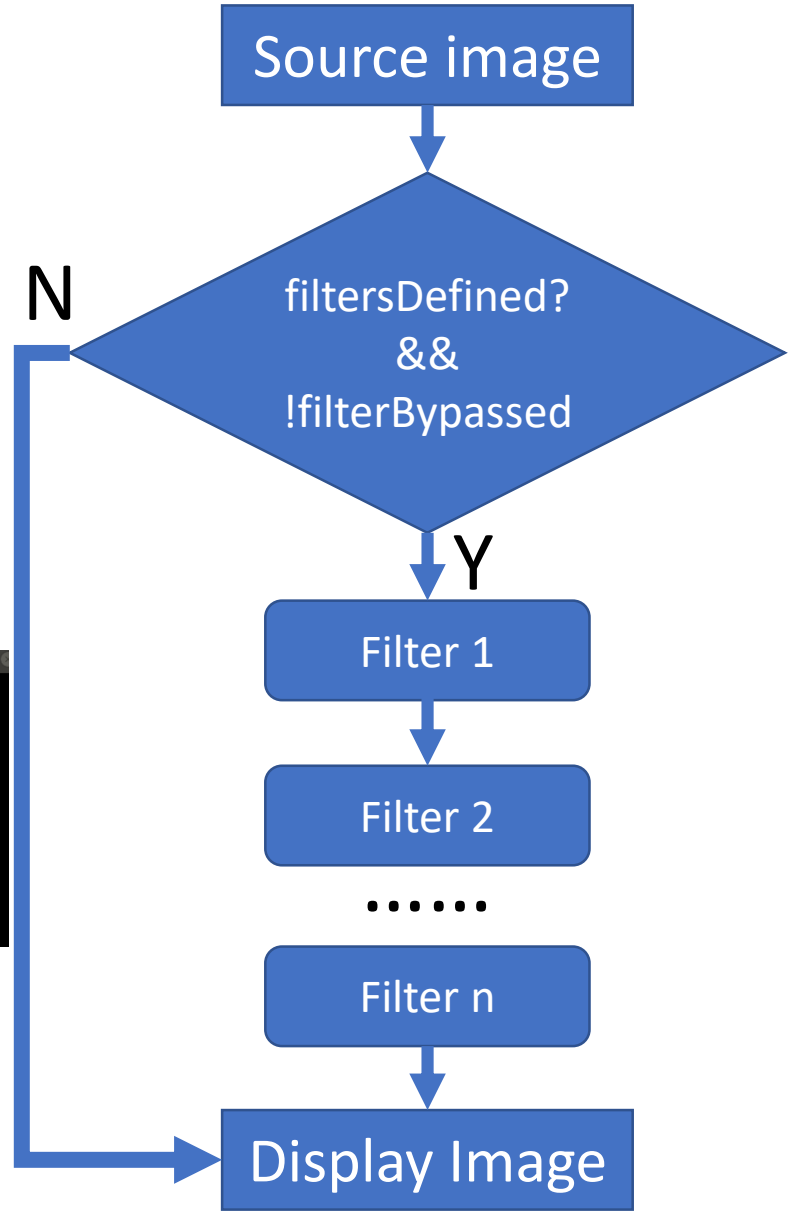
1. Open “filterlib.py” and add your custom filter.
   Attention: need to handle variable conversion by yourself
   E.g. str -> int/float, define upper/lower bounds
2. Use it directly in the GUI.
   Double slash to comment it out

Filters are connected in series and applied in order.




```
#========================================
# canny(minVal,maxVal)
# Input must be a greyscale image
#========================================
def canny(inputImage,args):
    arg = args.split(',')
    return cv2.Canny(inputImage,int(arg[0]),int(arg[1]))
```



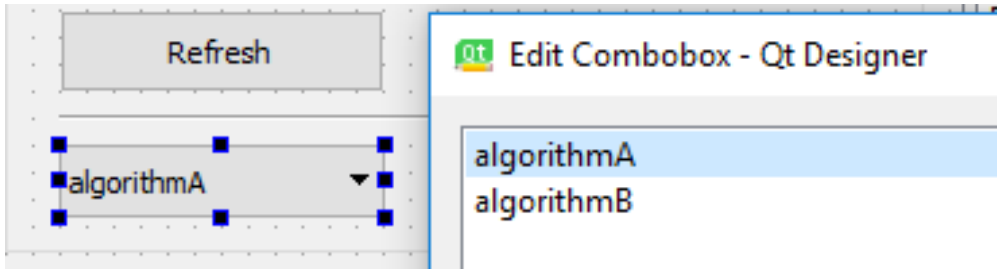


Object detection

Note:
When object detection is enabled, the original image overlayed with the detected object will be shown instead of the filtered image.

1. Add the name of your object detection algorithm to the GUI.
2. Define your algorithm in “objectDetection.py”.

   See sample algorithmA() or google “opencv object detection python”




```python
def algorithmA(imageFiltered,imageOriginal,agent):
    nOfSamples = 2
    im2, contours, hierarchy = cv2.findContours(imageFiltered, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
    cnts = sorted(contours, key = cv2.contourArea, reverse = True)[:nOfSamples]
    if len(cnts) > 1:
        targetCnt = cnts[1] # cnt[0] is the edge of the screen
        rect = cv2.minAreaRect(targetCnt)
        box = np.int0(cv2.boxPoints(rect)) # vertices of the bounding rect
        center = np.int0(np.sum(box, axis=0)/4) # [centerX, centerY] dataType: int
        agent.set(center[0],center[1]) # update the position of the agnet
        imageOriginal = cv2.drawContours(imageOriginal,[box],0,(0,255,0), 3) # draw boundingRect on the
    return imageOriginal
```

```python
class Agent():
    def __init__(self):
        self.x = 0
        self.y = 0
        self.orientation = 0
```

3. Instances of **Agent** class can be accessed via “vision.<agentName>”. Information about the agents are often used in a subthread.

Subthread

Use a subthread when you want to apply a time-varying magnetic field with respect to the position/orientation of the agents.

```
def taskSinFieldXY(self):
    startTime = time.time()
    while True:
        t = time.time() - startTime # elapsed time (sec)
        theta = 2 * pi * self.freq * t
        fieldX = 2 * cos(theta)
        fieldY = 2 * sin(theta)
        self.field.setX(fieldX)
        self.field.setY(fieldY)
        print('X: {}, Y: {}'.format(self.vision.agent1.x,self.vision.agent1.y))
        if self.stopped:
            return
```

Obtain elapsed time

Access detected objects

Stop flag

Some useful features

- Left click on the camera image returns xy coordinate in the terminal.
- ……
- We need you to improve it!