

Guide to the Python-based Coil System

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

Table of contents

- Dependencies
- In a nutshell
- Program Structure
- Modify GUI
- Image filters
- Object detection
- Multithreading

Dependencies

Recommend Ubuntu16

===== 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

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



MainWindow

General Control Gradient (still testing)

Clear Current

X 0.00

Y 0.00

Z 0.00

Vision ☒ Start/Pause Capture

//Please see "filterlib.py" for a list of examples. This editor is for debug purposes.
//e.g.

Refresh ☒ Bypass Filters

algorithmA ☐ Object Detection

Subthread

☒ Start/Stop subthread

param0 0.00

param1 0.00

param2 0.00

param3 0.00

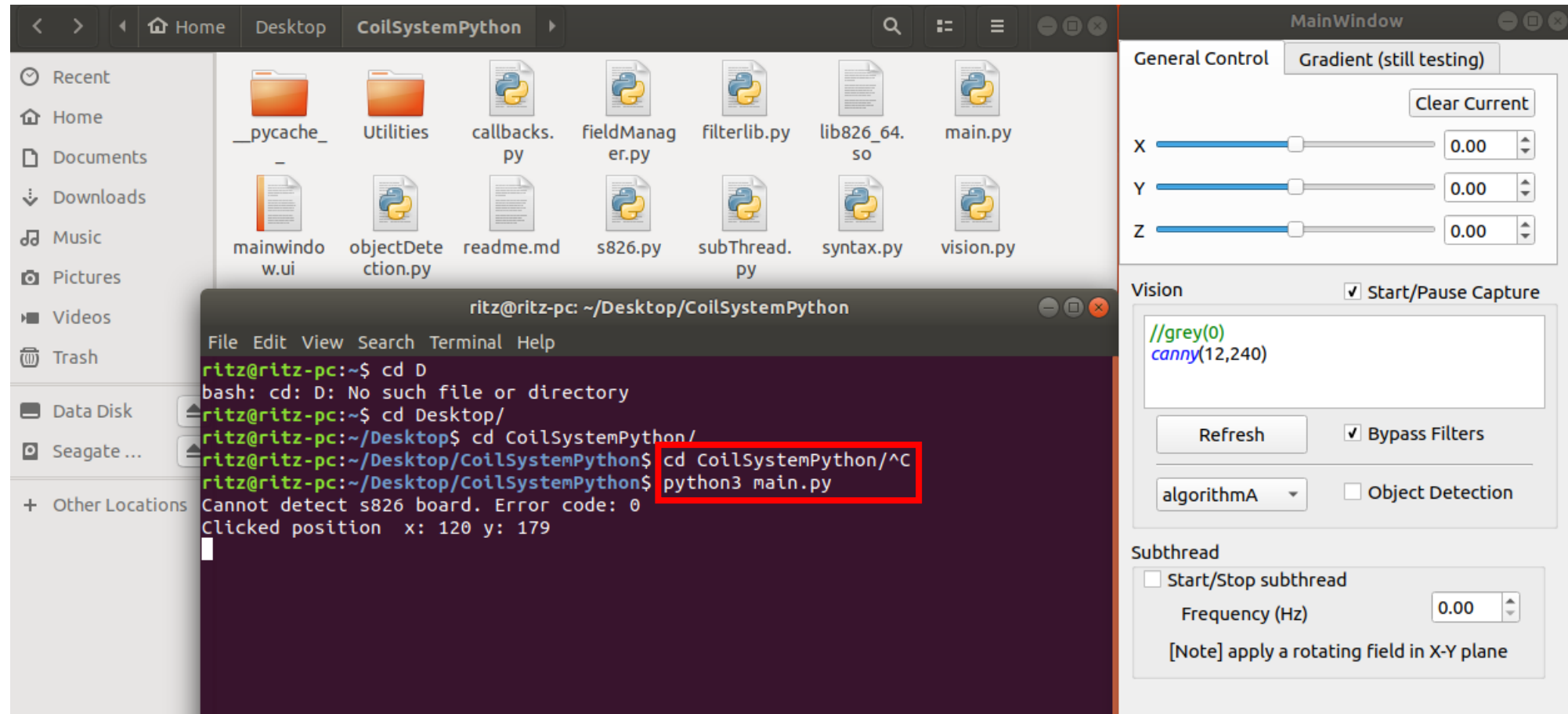
param4 0.00

```
File Edit View Search Terminal Help
ritz@ritz-pc:~$ cd '/home/ritz/Desktop/CoilSystemPython'
ritz@ritz-pc:~/Desktop/CoilSystemPython$ python3 main.py
=====
Vendor: b'NET GmbH'
Model: b'F0124TC'
GUID: 2672909587849792
Mode: 640x480_Y8
Framerate: 60.0
Available modes ['640x480_YUV422', '640x480_Y8', '640x480_Y16', 'FORMAT7_0']
=====
Subthread "" starts.
Subthread not defined.
Subthread is terminated.
□
```

In a nutshell

Go to the working directory.

Run “python3 main.py”



General Control

Gradient (still testing)

Clear Current

X  0.00Y  0.00Z  0.00

Vision

☒ Start/Pause Capture

```
//Please see "filterlib.py" for a list of
examples. This editor is for debug
purposes.
//e.g.
```

Refresh

☒ Bypass Filters

algorithmA

☐ Object Detection

Subthread

☐ Start/Stop subthread

param0 0.00

param1 0.00

param2 0.00

param3 0.00

param4 0.00

General Control

Gradient (still testing)

General Control Section

Clear Current

X 0.00

Y 0.00

Z 0.00

Subthread

☐ Start/Stop subthread

param0 0.00

param1 0.00

param2 0.00

param3 0.00

param4 0.00

Subthread Section

Vision

☒ Start/Pause Capture

```
//Please see "filterlib.py" for a list of
examples. This editor is for debug
purposes.
//e.g.
```

Refresh

☒ Bypass Filters

algorithmA

☐ Object Detection

Vision Section

MainWindow

General Control

Gradient (still testing)

Clear Current

X 0.00

Y 0.00

0.00

General Control Section

fieldManager.py

Subthread

☐ Start/Stop subthread

param0 0.00

param1 0.00

param2 0.00

param3 0.00

param4 0.00

Subthread Section

subThread.py

Vision

☒ Start/Pause Capture

```
//Please see "filterlib.py" for a list of
examples. This editor is for debug
purposes.
//e.g.
```

Refresh

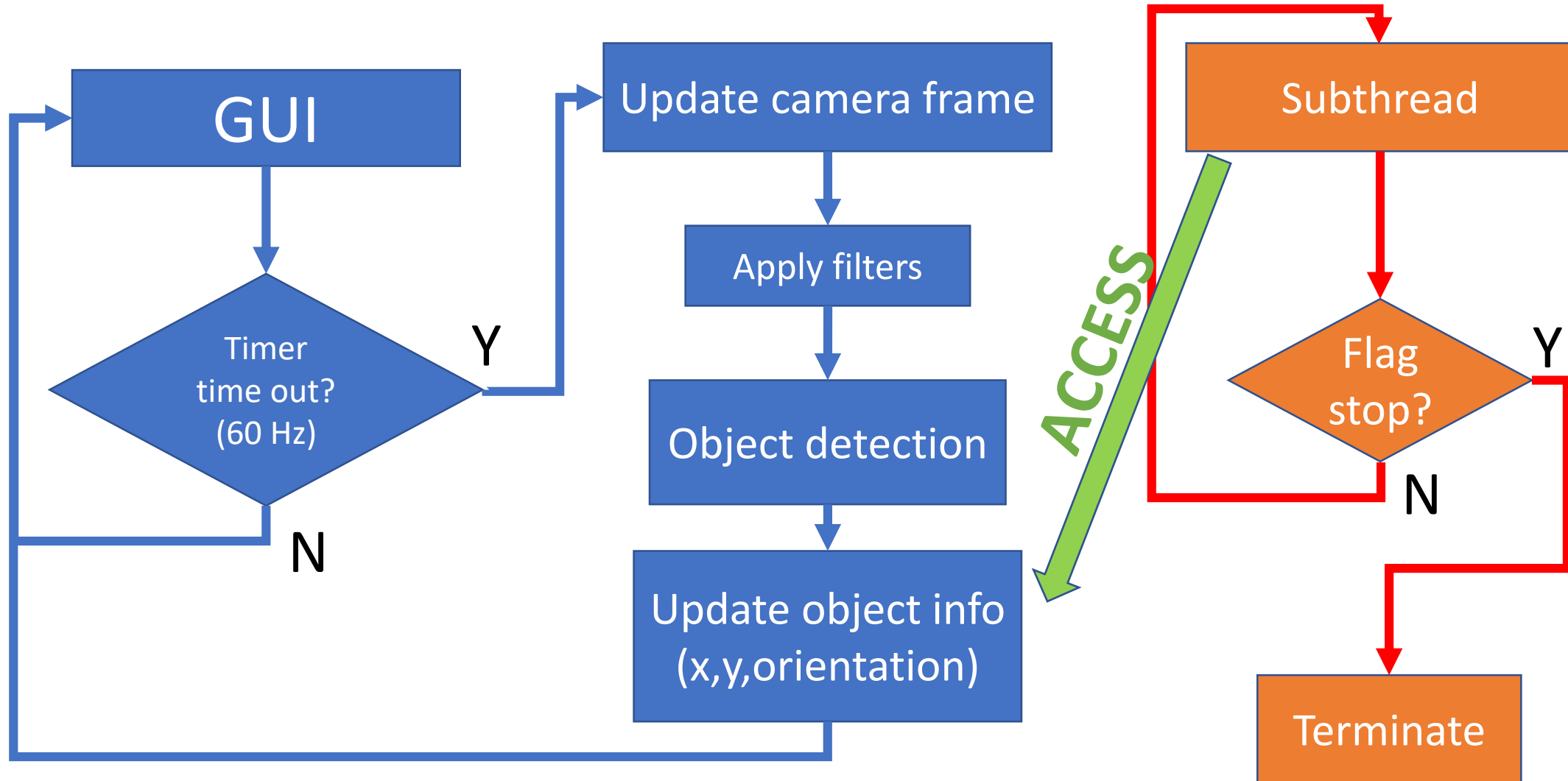
☒ Bypass Filters

algorithmA

Vision.py
filterlib.py
objectDetection.py

Vision Section

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*

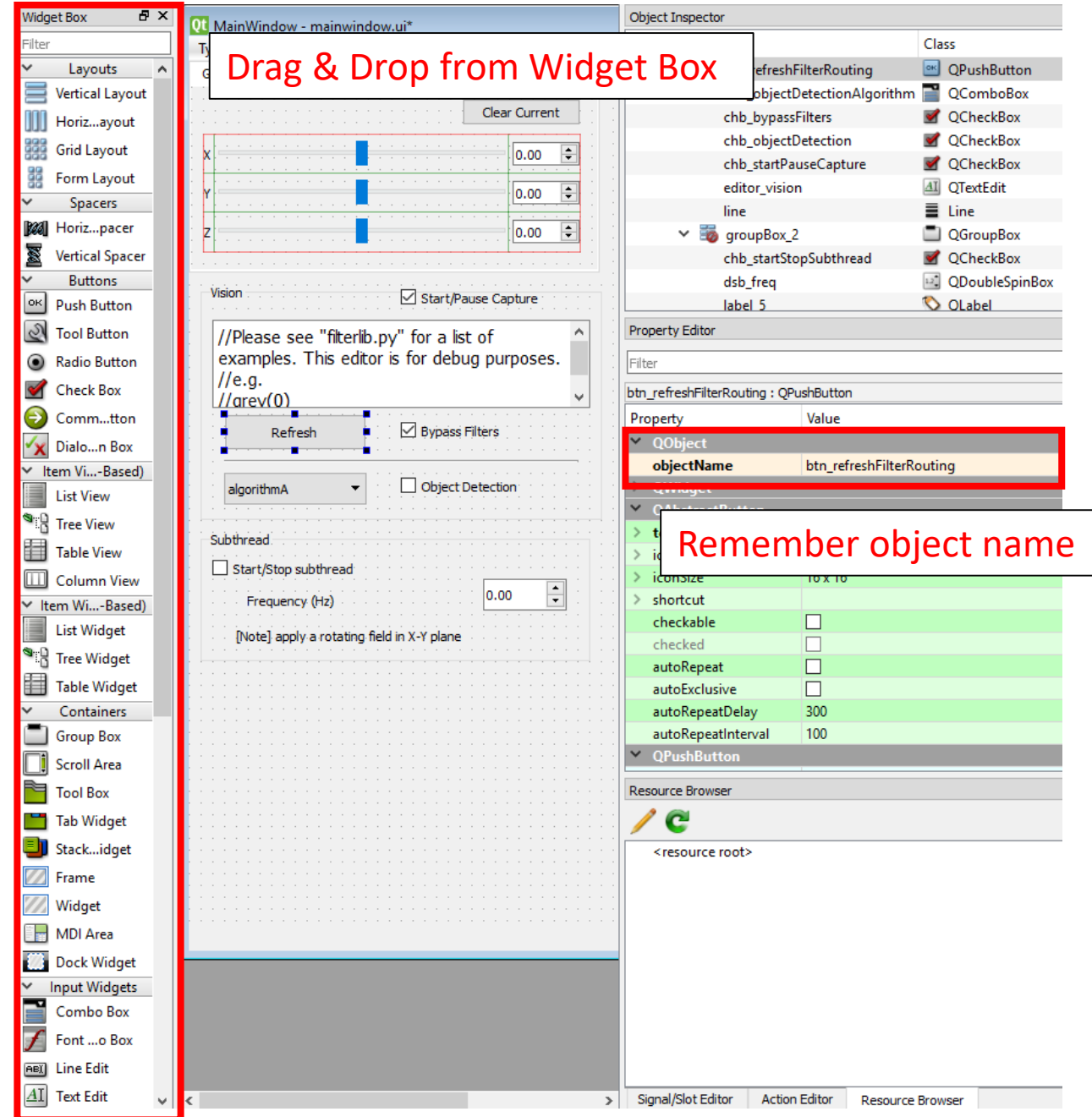
| | objectDetection.py [define object detection algorithms] *Add your code here*

|
|
|

└─subthread.py [run multithreading tasks] *Add your code here*

Modify GUI

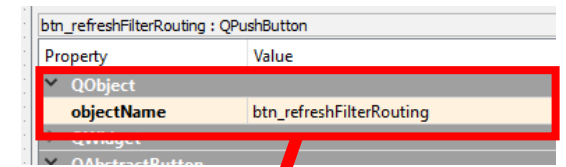
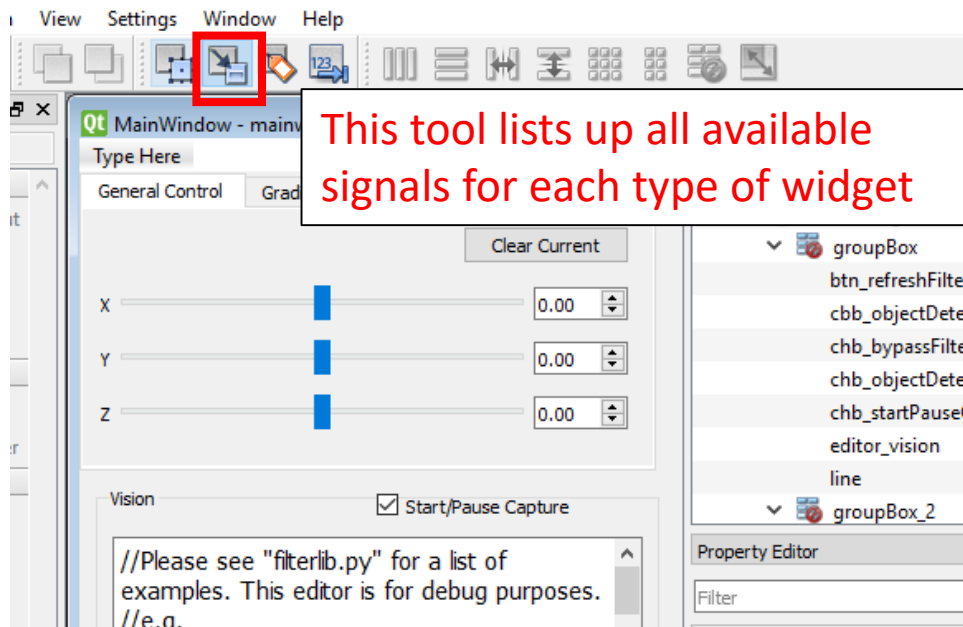
1. Open "Mainwindow.ui" with qt-designer.



Modify GUI

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

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



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

```
self.chb_startStopSubthread.toggled.connect(self.on_chb_startStopSubthread)  
self.dsb_freq.valueChanged.connect(self.thrd.setFreq)
```

Image filters

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

1. Open “filterlib.py” and add your custom filter.

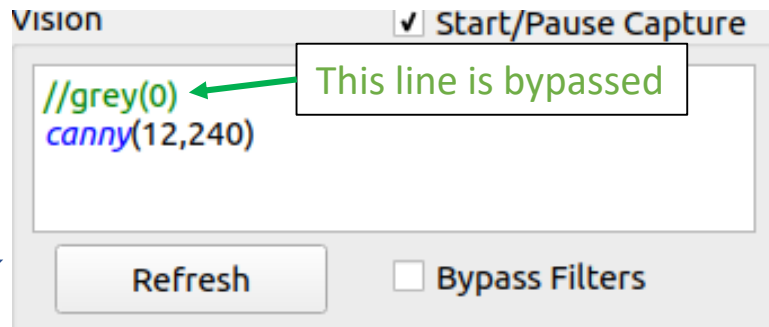
Attention: need to handle variable conversion by yourself

E.g. str -> int/float, define upper/lower bounds

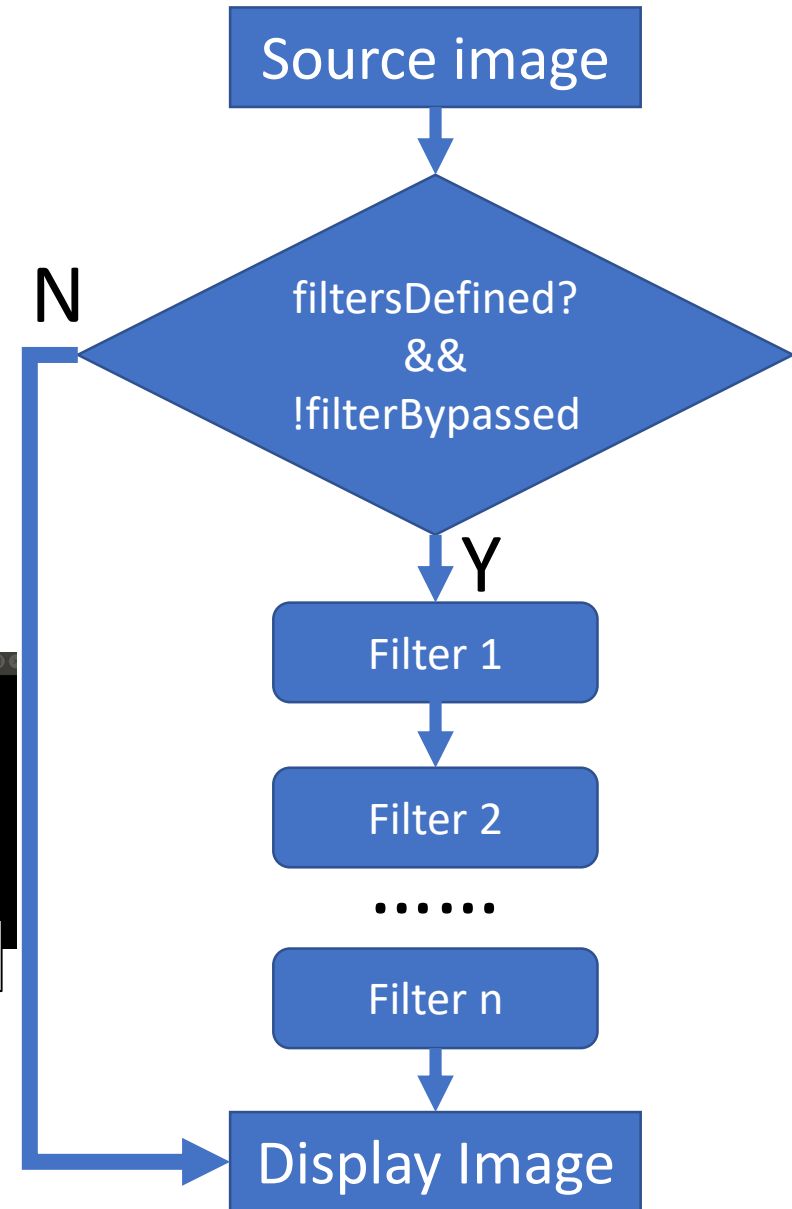
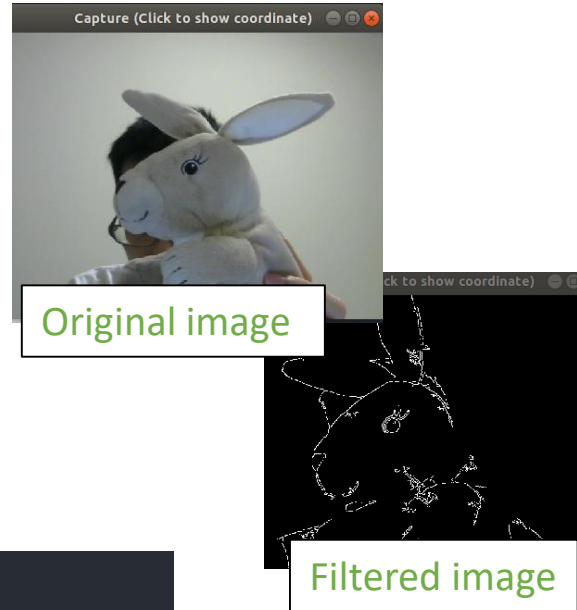
2. Use it directly in the GUI.

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]))
```



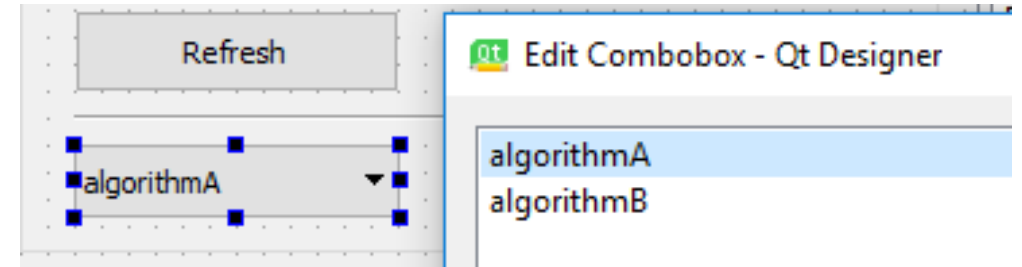
Object detection

Note:

When object detection is enabled, the original image overlaid 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”



```
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 agent
        imageOriginal = cv2.drawContours(imageOriginal, [box], 0, (0, 255, 0), 3) # draw boundingRect on the
    return imageOriginal
```

```
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

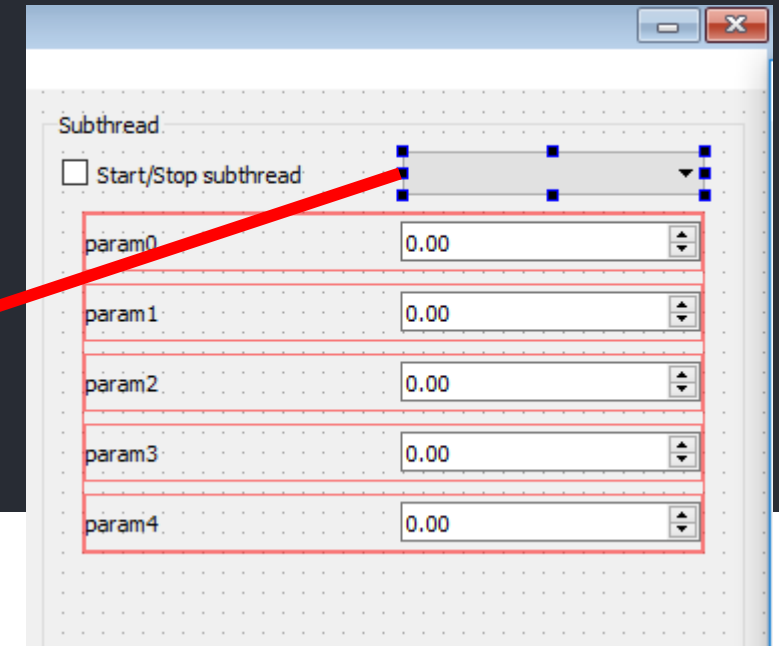
Use a subthread when you want to apply a time-varying magnetic field with respect to the position/orientation of the agents.

```
self.labelOnGui = {'twistField': ['Frequency (Hz)', 'Magniude (mT)', 'AzimuthalAngle (deg)', 'PolarAngle (deg)', 'SpanAngle (deg)'],  
                  'rotateXY': ['Frequency (Hz)', 'Magniude (mT)', 'N/A', 'N/A', 'N/A'],  
                  'rotateYZ': ['Frequency (Hz)', 'Magniude (mT)', 'N/A', 'N/A', 'N/A'],  
                  'rotateXZ': ['Frequency (Hz)', 'Magniude (mT)', 'N/A', 'N/A', 'N/A'],  
                  'default': ['param0', 'param1', 'param2', 'param3', 'param4']}]  
  
self.minOnGui = {'twistField': [-100, 0, -1080, 0, 0],  
                'rotateXY': [-100, 0, 0, 0, 0],  
                'rotateYZ': [-100, 0, 0, 0, 0],  
                'rotateXZ': [-100, 0, 0, 0, 0],  
                'default': [0, 0, 0, 0, 0]}  
  
self.maxOnGui = {'twistField': [100, 14, 1080, 180, 360],  
                'rotateXY': [100, 14, 0, 0, 0],  
                'rotateYZ': [100, 14, 0, 0, 0],  
                'rotateXZ': [100, 14, 0, 0, 0],  
                'default': [0, 0, 0, 0, 0]}
```

subThread.py

Qt Edit Combobox - Qt Designer

- twistField
- rotateXY
- rotateXZ
- rotateYZ



```
def twistField(self):
```

```
#=====d=====
# reference params
# 0 'Frequency (Hz)'
# 1 'Magniude (mT)'
# 2 'AzimuthalAngle (deg)'
# 3 'PolarAngle (deg)'
# 4 'SpanAngle (deg)'
#=====
```

params[0], params[1], ...

```
startTime = time.time()
```

```
record = 'Time(s), FieldX(mT), FiledY(mT), FieldZ(mT), X(pixel), Y(pixel) \n' # output to a txt file
```

```
counter = 0
```

```
while True:
```

```
t = time.time() - startTime # elapsed time (sec)
```

Obtain elapsed time

```
fieldX = self.params[1]* ( cosd(self.params[2])*cosd(self.params[3])*cosd(90-self.params[4]*0.5)*cos(2*pi*self.params[0]*t) - s
```

```
fieldY = self.params[1]* ( sind(self.params[2])*cosd(self.params[3])*cosd(90-self.params[4]*0.5)*cos(2*pi*self.params[0]*t) + c
```

```
fieldZ = self.params[1]* (-sind(self.params[3])*cosd(90-self.params[4]*0.5)*cos(2*pi*self.params[0]*t) + cosd(self.params[3])*c
```

```
self.field.setX(fieldX)
```

setField

```
self.field.setY(fieldY)
```

```
self.field.setZ(fieldZ)
```

```
# save to txt
```

```
counter += 1
```

```
if counter > 300:
```

```
    counter = 0
```

```
    record = record + '{:.5f}, {:.2f}, {:.2f}, {:.2f}, {}, {} \n'.format(t, self.field.x, self.field.y, self.field.z, self.vision.ag
```

Obtain XYZ field strength

Stop flag

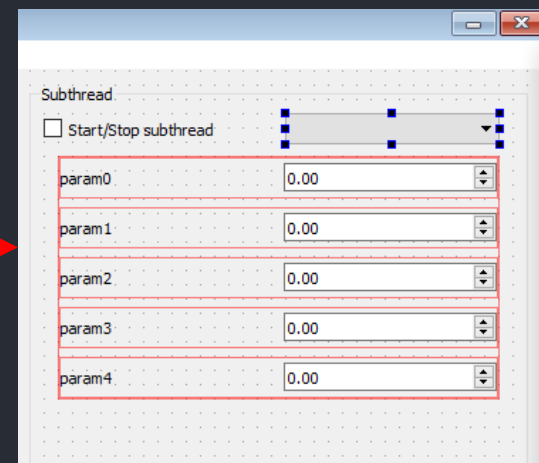
```
    if self.stopped:
```

```
        text_file = open("Output.txt", "w")
```

```
        text_file.write(record)
```

Obtain object position

```
self.vision.agent1.x, self.vision.agent1.y
```



Tips

- Left click on the camera image returns xy coordinate in the terminal.
- If you are using Ubuntu 17, use green-recorder for screen recording.
 - *Ubuntu 17.10 rolls back to GNOME shell (Ubuntu has been using Unity shell since Ubuntu 11), so some screen recording software is not working properly.*
 - <https://github.com/foss-project/green-recorder>
 - *You might have trouble converting WEBM format. One of the reasons I would recommend Ubuntu 16.*
- We need you to improve it!