

# Electric Soup ~ each word is malleable

## OpenCV and OpenGL using Python

29 Saturday Aug 2015

POSTED BY RDMILLIGAN IN TECHNOLOGY

≈ LEAVE A COMMENT

Tags

3D, Cascade Classifier, Computer Vision, Haar Cascades, Lego, Lego Detection, Object Detection, OpenCV, OpenGL, PyOpenGL, PyOpenGLAccelerate, Python, Python Tools for Visual Studio, Webcam

Arkwood was playing with one of those *ball on a string attached to a wooden cup* thingamajigs. ‘It’s tricky,’ he stated.

‘Not as tricky as computer vision and 3D graphics,’ I retorted. As his ball hit the rim of the cup and fell to a dangling loss, he told me to shut the fuck up. Charming!

Let’s cut to the chase. I am going to use OpenCV computer vision to detect a Lego policeman on my webcam. Then I’m going to use OpenGL 3D graphics to render the Lego policeman real-time onto a cube. The integration of OpenCV and OpenGL holds much promise, and this post is but a first step.

## OpenCV computer vision

So how to detect a Lego policeman using OpenCV? An OpenCV haar cascade classifier, that’s how! My post Augmented Reality using OpenCV and Python has the detail.

We use our Webcam class to retrieve a snap:

```

1  import cv2
2  from threading import Thread
3
4  class Webcam:
5
6      def __init__(self):
7          self.video_capture = cv2.VideoCapture(0)
8          self.current_frame = self.video_capture.read()[1]
9
10     # create thread for capturing images
11     def start(self):
12         Thread(target=self._update_frame, args=()).start()
13
14     def _update_frame(self):
15         while(True):
16             self.current_frame = self.video_capture.read()[1]
17
18     # get the current frame
19     def get_current_frame(self):
20         return self.current_frame

```

And the Detection class uses my [Lego Policeman haar cascade](#) to find officers in the webcam snap:


```

1  import cv2
2
3  class Detection(object):
4
5      def get_items_in_image(self, item_cascade_path, image):
6
7          # detect items in image
8          item_cascade = cv2.CascadeClassifier(item_cascade_path)
9          gray_image = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
10         items = item_cascade.detectMultiScale(gray_image, scaleFactor=1.1, minN
11
12         # debug: draw rectangle around detected items
13         for (x,y,w,h) in items:
14             cv2.rectangle(image, (x,y), (x+w,y+h), (255,0,0), 2)
15
16         # debug: show detected items in image
17         cv2.imshow('OpenCV Detection', image)
18         cv2.waitKey(100)
19
20         # return items
21         return items

```

#### Advertisements

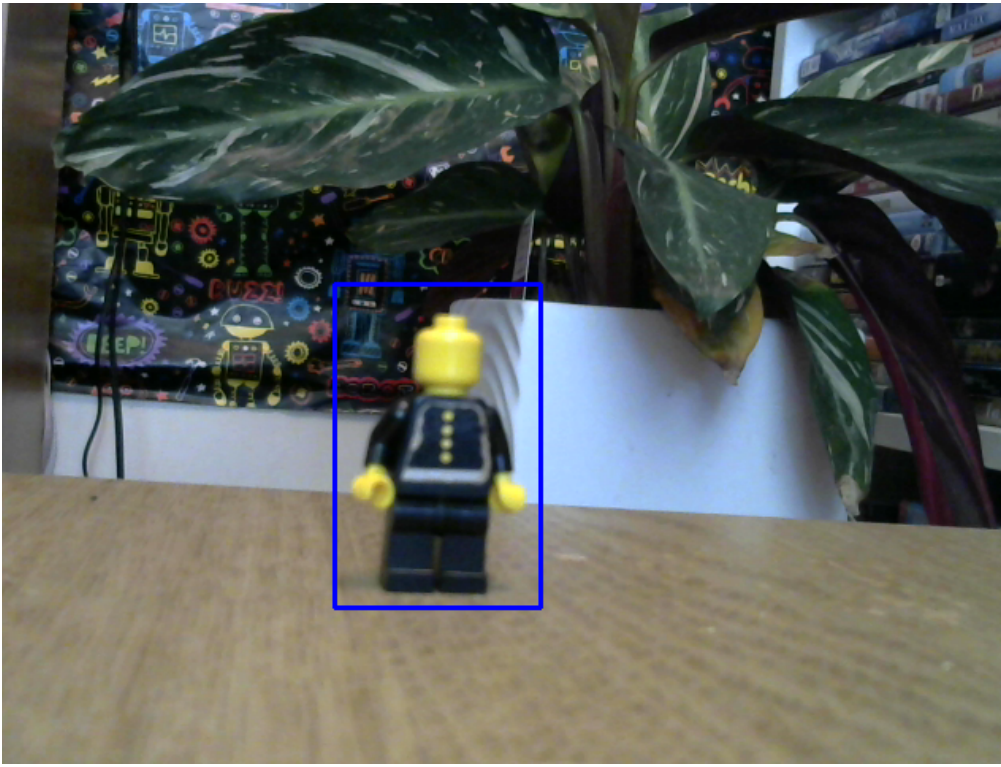
Your Journal for Life.  
The #1 app for journaling.



DAY ONE

[REPORT THIS AD](#)

Here's an example of a Lego policeman detected in a webcam snap:



As we'll see, the policeman will be extracted from the snap and displayed on a 3D cube.

## OpenGL 3D graphics

Okay, I need to get my hands on a Python binding for OpenGL. Enter PyOpenGL. I downloaded it from <http://www.lfd.uci.edu/~gohlke/pythonlibs/>

Specifically, I was after PyOpenGL and PyOpenGL Accelerate for Python version 2.7, to run on my Windows 7 PC 64-bit using the free Python Tools for Visual Studio – so plumped for PyOpenGL-3.1.1a1-cp27-none-win\_amd64.whl and PyOpenGL\_accelerate-3.1.1a1-cp27-none-win\_amd64.whl.

Here's the Lego Tracker class, which makes use of the aforementioned Webcam and Detection classes to render a Lego policeman onto a 3D cube:

```

1  from OpenGL.GL import *
2  from OpenGL.GLUT import *
3  from OpenGL.GLU import *
4  import cv2
5  import Image
6  from webcam import Webcam
7  from detection import Detection
8
9  class LegoTracker:
10
11     def __init__(self):
12         self.webcam = Webcam()
13         self.webcam.start()
14
15         self.detection = Detection()
16
17         self.x_axis = 0.0
18         self.y_axis = 0.0
19         self.z_axis = 0.0
20
21     def _update_image(self):
22         # get image from webcam
23         image = self.webcam.get_current_frame()
24
25         # detect Lego policemen in image
26         items = self.detection.get_items_in_image('haarcascade_lego_policeman.
27
28         if(len(items) > 0):
29
30             # get coordinates of first Lego policeman
31             roi_points = items[0]
32             x = roi_points[0]
33             y = roi_points[1]
34             w = roi_points[2]
35             h = roi_points[3]
36
37             # extract Lego policeman from image
38             roi = image[y:y+h,x:x+w]
39
40             # convert to OpenGL texture format
41             gl_image = Image.fromarray(roi)
42             ix = gl_image.size[0]
43             iy = gl_image.size[1]
44             gl_image = gl_image.tostring("raw", "BGRX", 0, -1)
45
46             # apply texture
47             glTexImage2D(GL_TEXTURE_2D, 0, 3, ix, iy, 0, GL_RGBA, GL_UNSIGNED_
48
49     def _draw_cube(self):
50         # draw cube
51         glBegin(GL_QUADS);
52         glTexCoord2f(0.0, 0.0); glVertex3f(-1.0, -1.0, 1.0)
53         glTexCoord2f(1.0, 0.0); glVertex3f( 1.0, -1.0, 1.0)
54         glTexCoord2f(1.0, 1.0); glVertex3f( 1.0,  1.0, 1.0)
55         glTexCoord2f(0.0, 1.0); glVertex3f(-1.0,  1.0, 1.0)
56         glTexCoord2f(1.0, 0.0); glVertex3f(-1.0, -1.0, -1.0)
57         glTexCoord2f(1.0, 1.0); glVertex3f(-1.0,  1.0, -1.0)
58         glTexCoord2f(0.0, 1.0); glVertex3f( 1.0,  1.0, -1.0)
59         glTexCoord2f(0.0, 0.0); glVertex3f( 1.0, -1.0, -1.0)
60         glTexCoord2f(0.0, 1.0); glVertex3f(-1.0,  1.0, -1.0)
61         glTexCoord2f(0.0, 0.0); glVertex3f(-1.0, -1.0,  1.0)

```

```

62         glTexCoord2f(1.0, 0.0); glVertex3f( 1.0, 1.0, 1.0)
63         glTexCoord2f(1.0, 1.0); glVertex3f( 1.0, 1.0, -1.0)
64         glTexCoord2f(1.0, 1.0); glVertex3f(-1.0, -1.0, -1.0)
65         glTexCoord2f(0.0, 1.0); glVertex3f( 1.0, -1.0, -1.0)
66         glTexCoord2f(0.0, 0.0); glVertex3f( 1.0, -1.0, 1.0)
67         glTexCoord2f(1.0, 0.0); glVertex3f(-1.0, -1.0, 1.0)
68         glTexCoord2f(1.0, 0.0); glVertex3f( 1.0, -1.0, -1.0)
69         glTexCoord2f(1.0, 1.0); glVertex3f( 1.0, 1.0, -1.0)
70         glTexCoord2f(0.0, 1.0); glVertex3f( 1.0, 1.0, 1.0)
71         glTexCoord2f(0.0, 0.0); glVertex3f( 1.0, -1.0, 1.0)
72         glTexCoord2f(0.0, 0.0); glVertex3f(-1.0, -1.0, -1.0)
73         glTexCoord2f(1.0, 0.0); glVertex3f(-1.0, -1.0, 1.0)
74         glTexCoord2f(1.0, 1.0); glVertex3f(-1.0, 1.0, 1.0)
75         glTexCoord2f(0.0, 1.0); glVertex3f(-1.0, 1.0, -1.0)
76         glEnd();
77
78     def _init_gl(self, Width, Height):
79         # initialize incl. texture
80         glClearColor(0.0, 0.0, 0.0, 0.0)
81         glClearDepth(1.0)
82         glDepthFunc(GL_LESS)
83         glEnable(GL_DEPTH_TEST)
84         glShadeModel(GL_SMOOTH)
85         glMatrixMode(GL_PROJECTION)
86         glLoadIdentity()
87         gluPerspective(45.0, float(Width)/float(Height), 0.1, 100.0)
88         glMatrixMode(GL_MODELVIEW)
89
90         glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST)
91         glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST)
92         glEnable(GL_TEXTURE_2D)
93
94     def _draw_scene(self):
95         # update texture image
96         self._update_image()
97
98         glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
99         glLoadIdentity();
100
101         # position and rotate cube
102         glTranslatef(0.0,0.0,-7.0);
103         glRotatef(self.x_axis,1.0,0.0,0.0)
104         glRotatef(self.y_axis,0.0,1.0,0.0)
105         glRotatef(self.z_axis,0.0,0.0,1.0)
106
107         # draw cube
108         self._draw_cube()
109
110         # update rotation values
111         self.x_axis = self.x_axis - 0.30
112         self.z_axis = self.z_axis - 0.30
113
114         glutSwapBuffers()
115
116     def main(self):
117         # setup and run OpenGL
118         glutInit()
119         glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH)
120         glutInitWindowSize(640, 480)
121         glutInitWindowPosition(0, 0)
122         glutCreateWindow("OpenGL Lego Tracker")
123         glutDisplayFunc(self._draw_scene)

```

```
124         glutIdleFunc(self._draw_scene)
125         self._init_gl(640, 480)
126         glutMainLoop()
127
128     # run instance of Lego Tracker
129     legoTracker = LegoTracker()
130     legoTracker.main()
```

The [NeHe tutorials](#) are a great help in understanding OpenGL (though operations may be legacy-mode). The [Mixing OpenGL and OpenCV tutorial](#) at TutorialsPlay provides detail of rendering video as OpenGL textures.

Time for a demo! Here's the Lego Tracker in action:



You'll notice that each time the Lego policeman is successfully detected, the cube is updated.

'So what the fuck is the point of that?' Arkwood snarled, the wooden cup broken in rage.

'It's the start of something beautiful,' I replied, 'and can you please mind your bastard Ps and Qs.'

The end of something beautiful, he muttered.

Ciao!

#### Sponsored Content

**Leonardo DiCaprio Buys  
Neighbor's Home for \$10.5  
Million Mansion Global | Sponso  
red**

**Ivanka Trump and Jared  
Kushner Close massive \$32  
Million Dollar Deal in Miami**

**[Gallery] The Most  
Inappropriate Dress on The Met  
Gala Red Carpet HeraldWeekly |  
Sponsored**

Condo Mansion Global | Sponsored

**Take A Look - Designer's Corner**  
Mansion Global | Sponsored

**For an Architect, the Metaverse Is Nothing New, Says Piero Lissoni**  
Mansion Global | Sponsored

**Improve Poor Sight! Do This immediately!** healthyeyes.live | Sponsored

**Helping industries become**

Electric | Sponsored

**Kolkata - Stair Lifts Could Be a**

air Lift | Search Ads | Sponsored

**[Pics] Royal Family Confirms**

ponsored

[Create a free website or blog at WordPress.com.](#)