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.

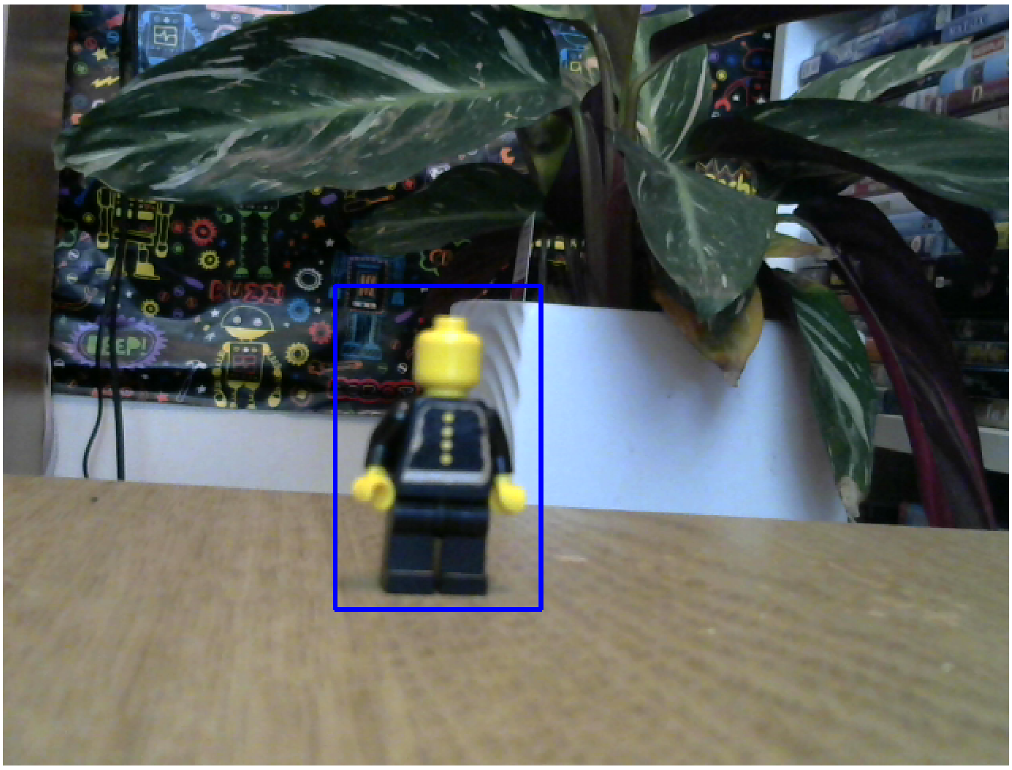
We use our Webcam class to retrieve a snap:

```python
import cv2
from threading import Thread

class Webcam:

    def __init__(self):
        self.video_capture = cv2.VideoCapture(0)
        self.current_frame = self.video_capture.read()[1]

    # create thread for capturing images
    def start(self):
        Thread(target=self._update_frame, args=()).start()

    def _update_frame(self):
        while(True):
            self.current_frame = self.video_capture.read()[1]

    # get the current frame
    def get_current_frame(self):
        return self.current_frame
```

And the Detection class uses my Lego Policeman haar cascade to find officers in the webcam snap:

```python
import cv2

class Detection(object):

    def get_items_in_image(self, item_cascade_path, image):

        # detect items in image
        item_cascade = cv2.CascadeClassifier(item_cascade_path)
        gray_image = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
        items = item_cascade.detectMultiScale(gray_image, scaleFactor=1.1, minN

        # debug: draw rectangle around detected items
        for (x,y,w,h) in items:
            cv2.rectangle(image,(x,y),(x+w,y+h),(255,0,0),2)

        # debug: show detected items in image
        cv2.imshow('OpenCV Detection', image)
        cv2.waitKey(100)

        # return items
        return items
```

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:

```python
from OpenGL.GL import *
from OpenGL.GLUT import *
from OpenGL.GLU import *
import cv2
import Image
from webcam import Webcam
from detection import Detection

class LegoTracker:

    def __init__(self):
        self.webcam = Webcam()
        self.webcam.start()

        self.detection = Detection()

        self.x_axis = 0.0
        self.y_axis = 0.0
        self.z_axis = 0.0

    def _update_image(self):
        # get image from webcam
        image = self.webcam.get_current_frame()

        # detect Lego policemen in image
        items = self.detection.get_items_in_image('haarcascade_lego_policeman.

        if(len(items) > 0):

            # get coordinates of first Lego policeman
            roi_points = items[0]
            x = roi_points[0]
            y = roi_points[1]
            w = roi_points[2]
            h = roi_points[3]

            # extract Lego policeman from image
            roi = image[y:y+h,x:x+w]

            # convert to OpenGL texture format
            gl_image = Image.fromarray(roi)
            ix = gl_image.size[0]
            iy = gl_image.size[1]
            gl_image = gl_image.tostring("raw", "BGRX", 0, -1)

            # apply texture
            glTexImage2D(GL_TEXTURE_2D, 0, 3, ix, iy, 0, GL_RGBA, GL_UNSIGNED_

    def _draw_cube(self):
        # draw cube
        glBegin(GL_QUADS);
        glTexCoord2f(0.0, 0.0); glVertex3f(-1.0, -1.0,  1.0)
        glTexCoord2f(1.0, 0.0); glVertex3f( 1.0, -1.0,  1.0)
        glTexCoord2f(1.0, 1.0); glVertex3f( 1.0,  1.0,  1.0)
        glTexCoord2f(0.0, 1.0); glVertex3f(-1.0,  1.0,  1.0)
        glTexCoord2f(1.0, 0.0); glVertex3f(-1.0, -1.0, -1.0)
        glTexCoord2f(1.0, 1.0); glVertex3f(-1.0,  1.0, -1.0)
        glTexCoord2f(0.0, 1.0); glVertex3f( 1.0,  1.0, -1.0)
        glTexCoord2f(0.0, 0.0); glVertex3f( 1.0, -1.0, -1.0)
        glTexCoord2f(0.0, 1.0); glVertex3f(-1.0,  1.0, -1.0)
        glTexCoord2f(0.0, 0.0); glVertex3f(-1.0,  1.0,  1.0)
        glTexCoord2f(1.0, 0.0); glVertex3f( 1.0,  1.0,  1.0)
        glTexCoord2f(1.0, 1.0); glVertex3f( 1.0,  1.0, -1.0)
        glTexCoord2f(1.0, 1.0); glVertex3f(-1.0, -1.0, -1.0)
        glTexCoord2f(0.0, 1.0); glVertex3f( 1.0, -1.0, -1.0)
        glTexCoord2f(0.0, 0.0); glVertex3f( 1.0, -1.0,  1.0)
        glTexCoord2f(1.0, 0.0); glVertex3f(-1.0, -1.0,  1.0)
        glTexCoord2f(1.0, 0.0); glVertex3f( 1.0, -1.0, -1.0)
        glTexCoord2f(1.0, 1.0); glVertex3f( 1.0,  1.0, -1.0)
        glTexCoord2f(0.0, 1.0); glVertex3f( 1.0,  1.0,  1.0)
        glTexCoord2f(0.0, 0.0); glVertex3f( 1.0, -1.0,  1.0)
        glTexCoord2f(0.0, 0.0); glVertex3f(-1.0, -1.0, -1.0)
        glTexCoord2f(1.0, 0.0); glVertex3f(-1.0, -1.0,  1.0)
        glTexCoord2f(1.0, 1.0); glVertex3f(-1.0,  1.0,  1.0)
        glTexCoord2f(0.0, 1.0); glVertex3f(-1.0,  1.0, -1.0)
        glEnd();

    def _init_gl(self, Width, Height):
        # initialize incl. texture
        glClearColor(0.0, 0.0, 0.0, 0.0)
        glClearDepth(1.0)
        glDepthFunc(GL_LESS)
        glEnable(GL_DEPTH_TEST)
        glShadeModel(GL_SMOOTH)
        glMatrixMode(GL_PROJECTION)
        glLoadIdentity()
        gluPerspective(45.0, float(Width)/float(Height), 0.1, 100.0)
        glMatrixMode(GL_MODELVIEW)

        glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST)
        glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST)
        glEnable(GL_TEXTURE_2D)

    def _draw_scene(self):
        # update texture image
        self._update_image()

        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        glLoadIdentity();

        # position and rotate cube
        glTranslatef(0.0,0.0,-7.0);
        glRotatef(self.x_axis,1.0,0.0,0.0)
        glRotatef(self.y_axis,0.0,1.0,0.0)
        glRotatef(self.z_axis,0.0,0.0,1.0)

        # draw cube
        self._draw_cube()

        # update rotation values
        self.x_axis = self.x_axis - 0.30
        self.z_axis = self.z_axis - 0.30

        glutSwapBuffers()

    def main(self):
        # setup and run OpenGL
        glutInit()
        glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH)
        glutInitWindowSize(640, 480)
        glutInitWindowPosition(0, 0)
        glutCreateWindow("OpenGL Lego Tracker")
        glutDisplayFunc(self._draw_scene)
        glutIdleFunc(self._draw_scene)
        self._init_gl(640, 480)
        glutMainLoop()

# run instance of Lego Tracker
legoTracker = LegoTracker()
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!