COMPUTER VISION

PyYYC

Computer Vision

Outline

- Setup
- OpenCV
  - *drawing*
  - *filtering and thresholding*
  - *Canny edge detection*
  - *template matching*
  - *deep learning*
- Tesseract
- Discussion/Questions

Setup

- Anaconda
  - *conda supports python code editors (Spyder) in with virtual environments*
  - *pip & virtualenv do not seem to support python editors natively (takes work)*
  - ***anaconda and VS Code are recommended for this tutorial***
  - *https://www.anaconda.com/download/*
  - *https://docs.anaconda.com/anaconda/install/*
  - *https://conda.io/docs/_downloads/conda-cheatsheet.pdf*
- VS Code
  - *Recommended IDE but other IDEs will work too*
  - *Option to install in Anaconda setup: select yes*
  - *https://code.visualstudio.com/docs/getstarted/introvideos*

Setup

- python 3.6
  - ***python 3.6 or higher is recommended for this tutorial***
- OpenCV 3.4.1 (recently added to conda repository)
  - *OpenCV 3.4.1 has many new features including deep learning added*
  - *In terminal "conda create --name OpenCV34 python=3.6"*
  - *After "source activate OpenCV34", "conda install –c conda-forge opencv"; "conda install matplotlib"; "conda install pillow"*
- Git
  - *https://git-scm.com/video/what-is-git*
  - *https://services.github.com/on-demand/downloads/github-git-cheat-sheet.pdf*
  - *VS Code: Ctrl+Shift+P; Git clone;*
  - *https://github.com/cyrust/PyYYC-vision.git*

SETUP BREAK

DEMO

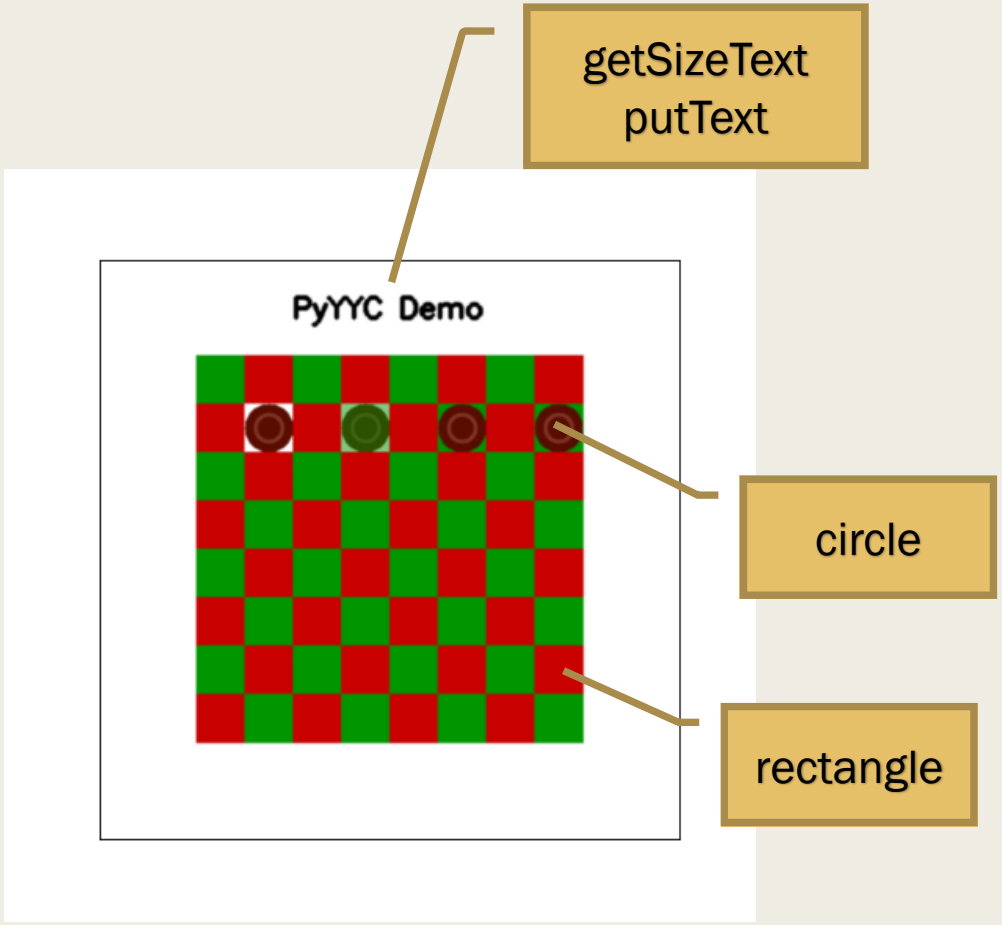
OpenCV Basics

- Tutorial based on a checker board
- OpenCV uses numpy to represent images
  - *“img = np.ones((width, height, channels), dtype=np.uint8) * 255”*
  - *Blank image from running above code*
- Supports different coloring models
  - *BGR (Blue Green Red) by default*
  - *“cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)”*
  - *“cv2.cvtColor(img, cv2.COLOR_BGR2RGB)”*

OpenCV Drawing

- rectangle
  - *# cv2.rectangle(img, pt1, pt2, color[, thickness[, lineType[, shift]]])*
  - *cv2.rectangle(img, (board_x+i*w,board_y+j*w),(board_x+i*w+w,board_y+j*w+w),color,-1)*
- circle
  - *#cv2.circle(img, center, radius, color[, thickness[, lineType[, shift]]]) → None*
  - *cv2.circle(checker, (square_width//2, square_width//2), square_width//2, brown1, -1)*
- text
  - *# cv2.getTextSize(textString, font)-> (textSize, baseline)*
  - *Size, textSize = cv2.getTextSize(text, font, fontScale, fontThickness)*
  - *# cv2.putText(img, text, org, font, color) → None*
  - *cv2.putText(img,text, (img_w//2 - Size[0]//2, (img_h-board_width)//4 + Size[1]//2), font, fontScale, (0,0,0), fontThickness)*

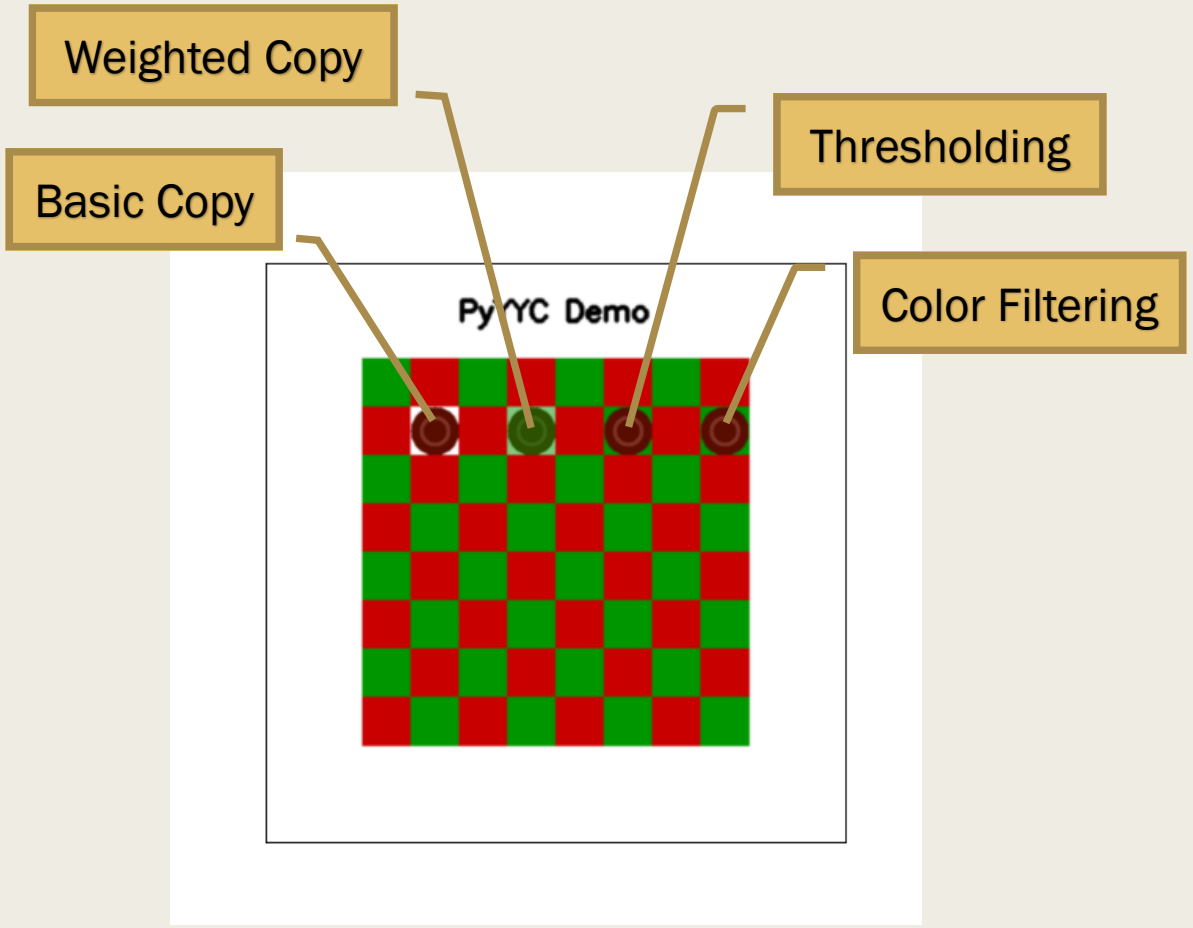
OpenCV Drawing



OpenCV Arrays

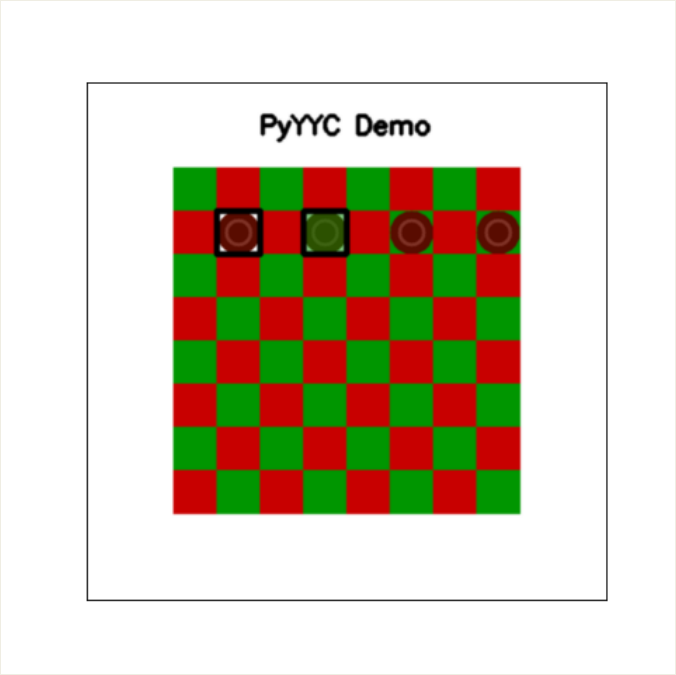
- Copying images over to one another
  - *result = checker*
- Weighted copying
  - *#cv.AddWeighted(src1, alpha, src2, beta, gamma, dst) → None*
  - *result = cv2.addWeighted(square, 0.5, checker, 0.5, 0)*
- Thresholding (mask based on threshold color)
  - *#cv.threshold(src, dst, threshold, maxValue, thresholdType) → None*
  - *cv2.threshold(checkergray, brown2_gry[0,0]+5, 255, cv2.THRESH_BINARY_INV)*
- Color filtering (mask based on color range)
  - *# dst=cv.inRange(src, lowerb, upperb[, dst])*
  - *cv2.inRange(checker_hsv, (0,0,0,), (180,255,253))*

OpenCV Arrays



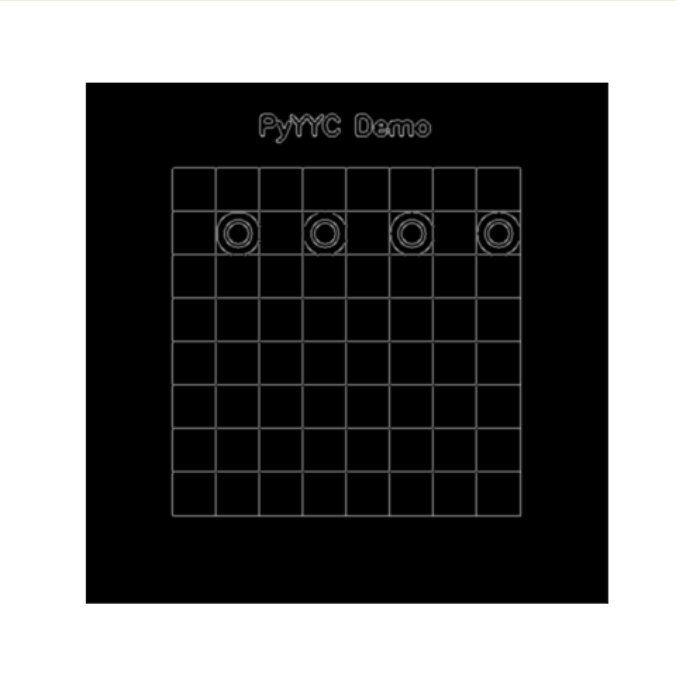
OpenCV Template Matching

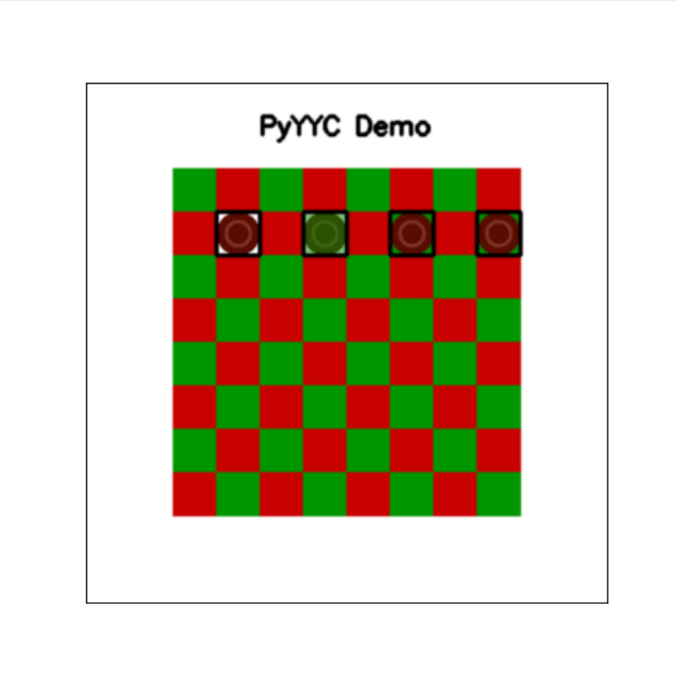
- Finds a rectangular image in another image
- Threshold allows specifying matching error
- See find_image function



OpenCV Template Matching

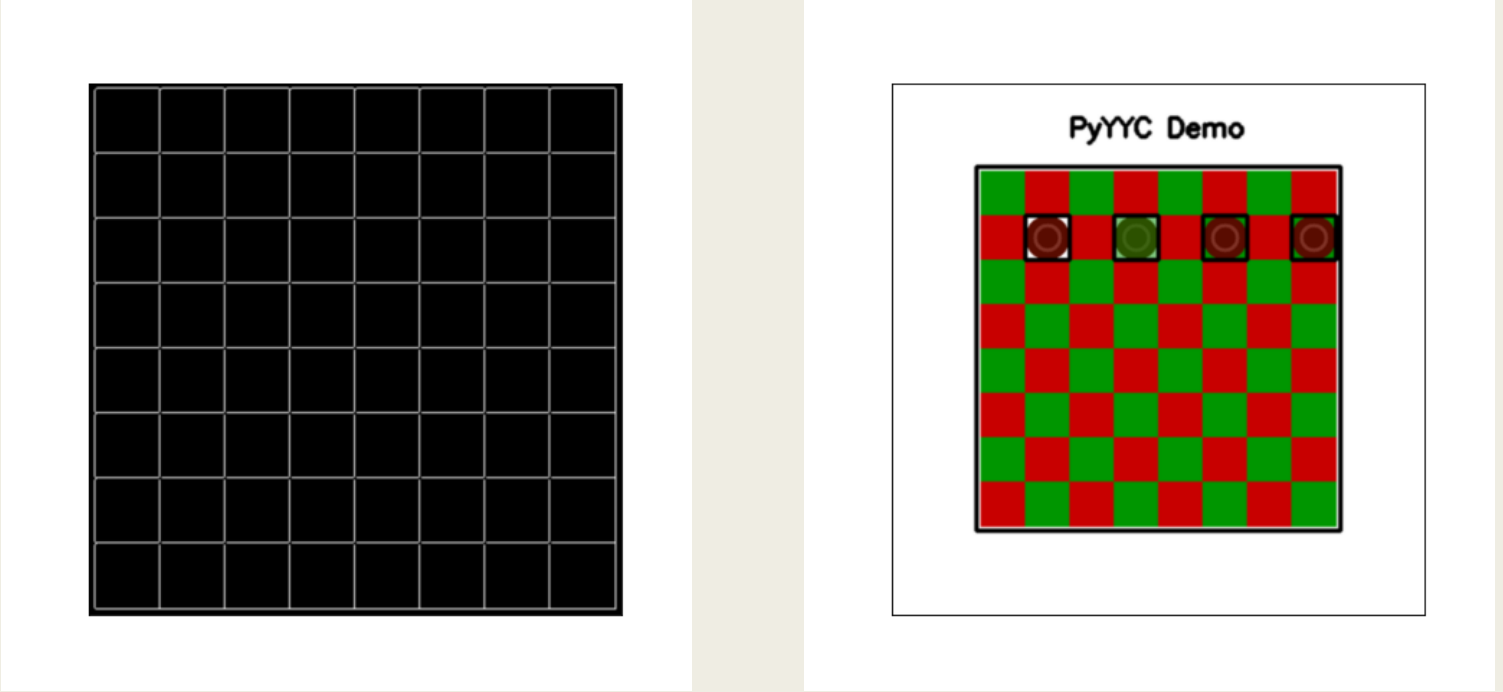
- Template and image can be transformed to features first (ie: edges)
- All checkers now detected





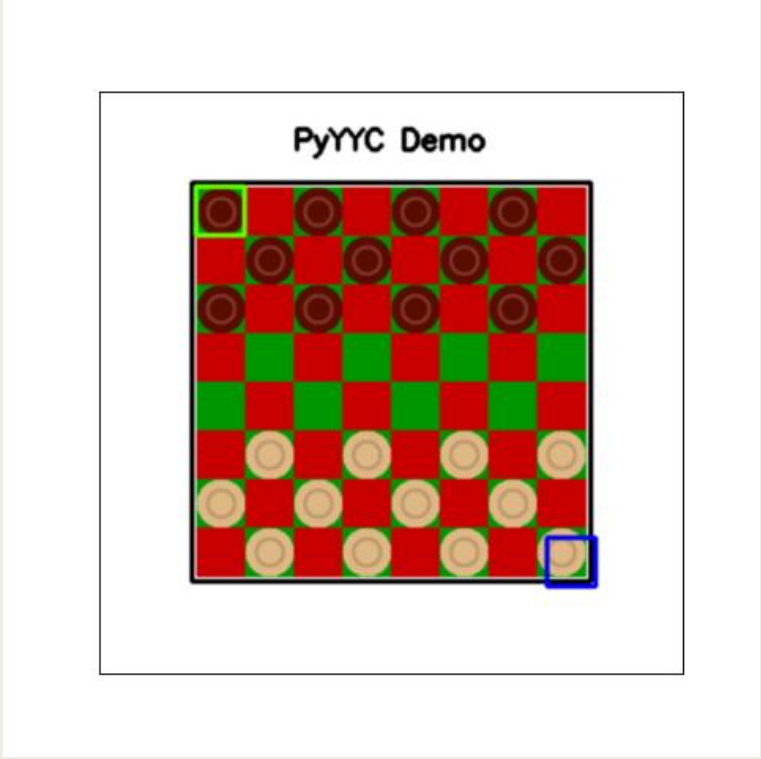
OpenCV Template Matching

- Masks needed for non-rectangular images (requires OpenCV 3 and above)
- Checker board detected through its edge features
- Checkers filtered out using a mask while detecting board

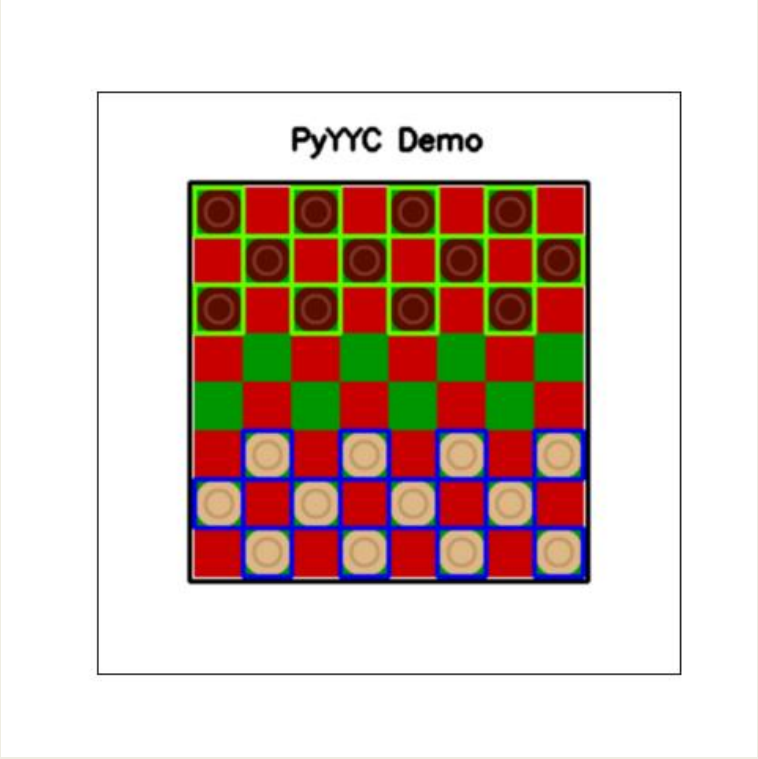


OpenCV Template Matching

Without mask

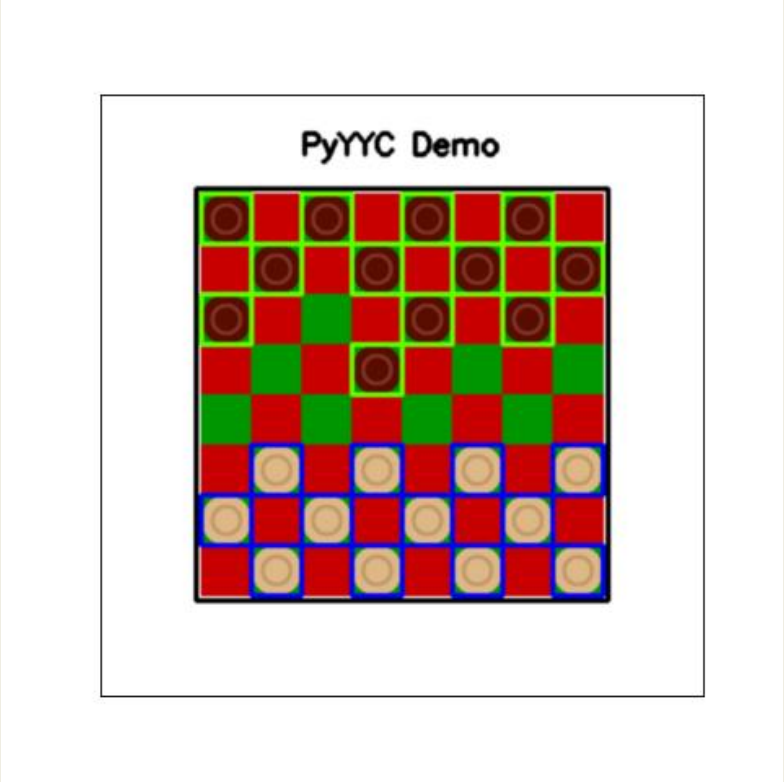


With mask 



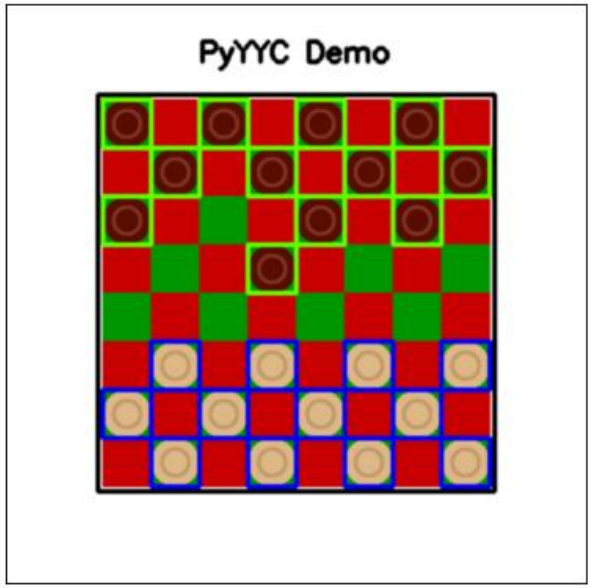
OpenCV Template Matching

- System correctly identifies state on the board after move



Interested in Checkers AI?

- References:
  - *An Introduction to Artificial Intelligence by Philip C. Jackson Jr.*
  - *Open source Checkers Players are available*
  - *Raven 0.4: https://github.com/bcorfman/raven-checkers/releases/tag/0.4*
  - *Not covered in this presentation*



DEEP LEARNING

ImageNet Large Scale Visual Recognition Challenge (ILSVRC)

- Algorithms for object detection and image classification at large scale
- http://image-net.org/challenges/LSVRC/
- OpenCV supports pre-trained models starting from its 3.3 version

ImageNet

Model	Year	Top-5 Error	Link
AlexNet/ SqueezeNet	2012/ 2016	15.3%	https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf
GoogLeNet	2014	6.67%	https://www.cs.unc.edu/~wliu/papers/GoogLeNet.pdf
VGG	2014	7.32%	https://arxiv.org/abs/1409.1556
ResNet	2015	3.57%	https://arxiv.org/abs/1512.03385
SENET	2017	2.25%	https://arxiv.org/abs/1709.01507

Top-5 error rate compares the ground truth against the first 5 predicted classes: an image is deemed correctly classified if the ground truth is among the top-5, regardless of its rank in them.

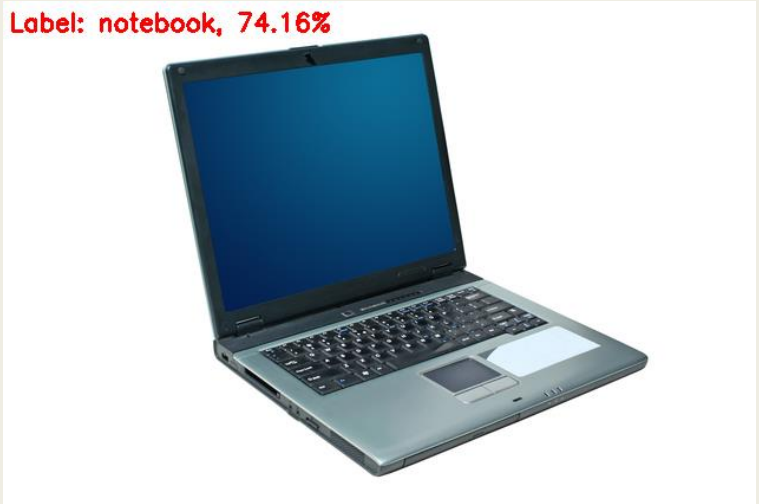
See http://www.deeplearningmodel.net/

Image classification accuracy is improving rapidly!

ImageNet competition is moving to Kaggle.

OpenCV dnn Module

- dnn module in OpenCV allows implementation of pre-trained deep neural networks
- Example: https://www.pyimagesearch.com/2017/08/21/deep-learning-with-opencv/
- “conda install –c conda-forge opencv” worked for my set up
- Let’s test a few examples with GoogLeNet!



FreeImages.com/Alaa Safei

FreeImages.com/Danny de Bruyne

OPTICAL CHARACTER RECOGNITION

OpenCV OCR Sample File

- Optical Character Recognition (OCR) is classification of images into characters
- For python files: https://github.com/opencv/opencv/tree/master/samples/python
- Download digits.py, common.py
- For data files: https://github.com/opencv/opencv/tree/master/samples/data
- Download digits.png
- Change the value of DIGITS_FN to ‘digits.png’ or proper path if you did not save digits.png in the same folder as digits.py
- The sample file performs OCR for digits using kNN and SVM

OpenCV OCR Sample File

kNN Classification

SVM Classification

Red indicates misclassification. SVM more accurate.

Alphabetical OCR

- Similar methods can be applied for alphabetical letters
- It is recommended to use Tesseract in combination with OpenCV for pre-processing for alphabetical character recognition
- See https://www.pyimagesearch.com/2017/07/10/using-tesseract-ocr-python/

tesseract

- Tesseract is an open source OCR engine
- To install “sudo apt install tesseract-ocr” and “sudo apt install libtesseract-dev”
- pytesseract package provides support for tesseract in python
- pytesseract is not available in conda repository
- Simplest way to use pytesseract is to copy pytesseract.py from github into your working folder (do not install with pip if you use conda)
- # try OCR with tesseract
  import pytesseract
  print(pytesseract.image_to_string(img_game2))

QUESTIONS