
What are your ideas?

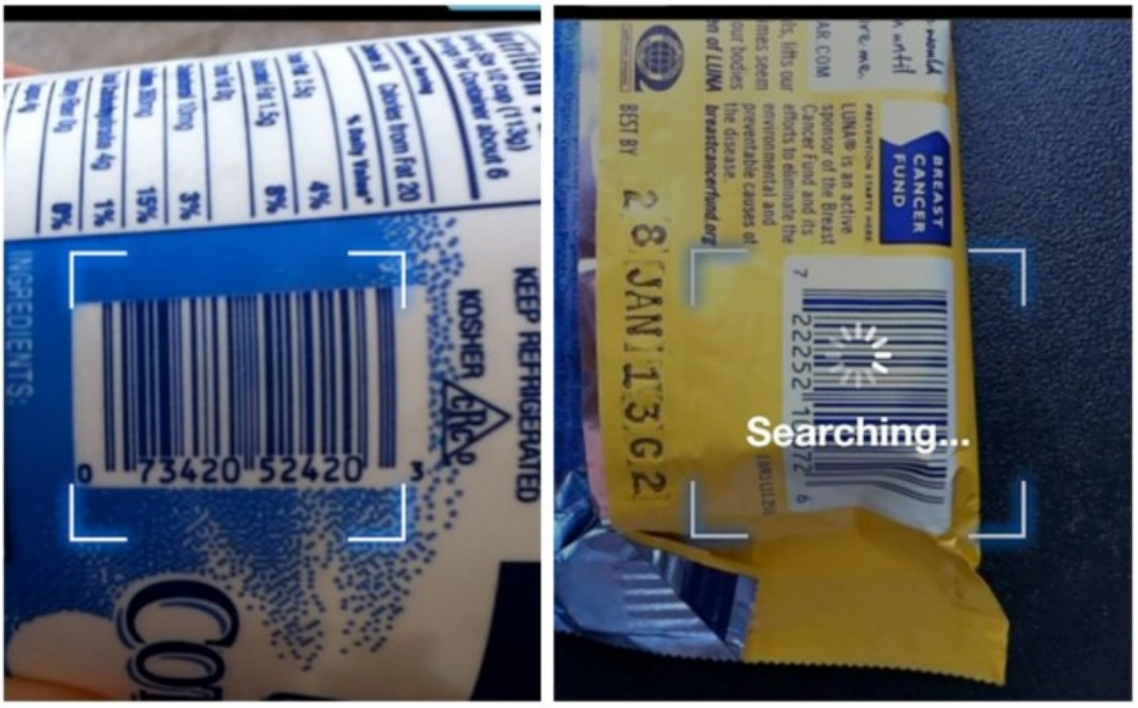
What is Computer Vision?

What is Computer Vision?

Computer vision is extracting information from visual data to:

- Impart knowledge
- Inform action

UPC Symbol Scanning

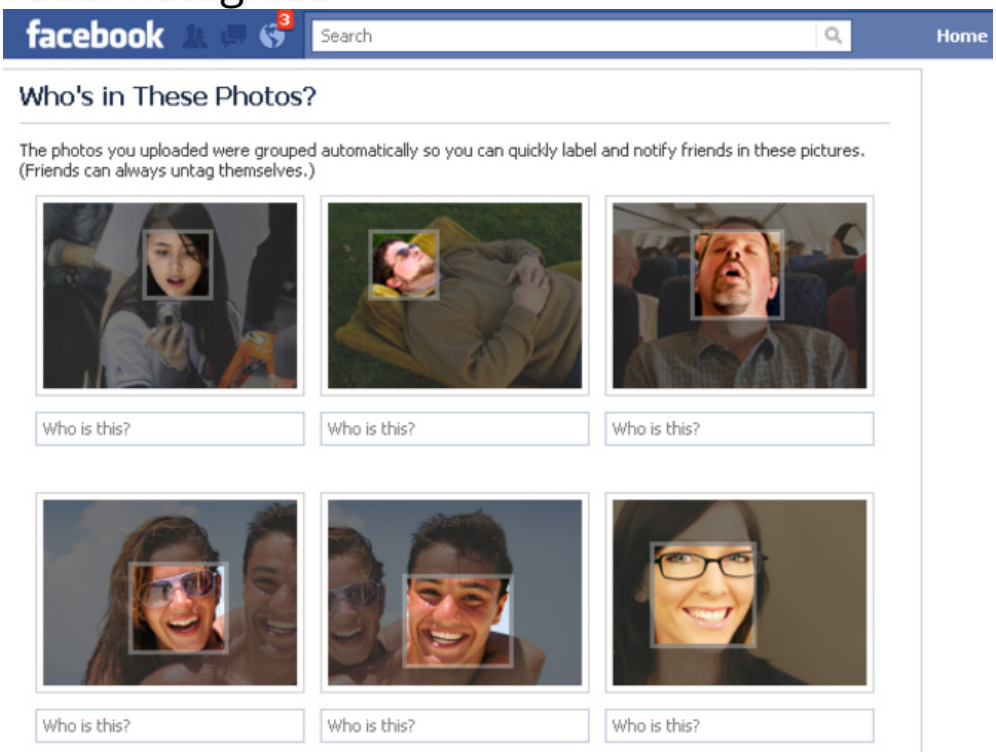


http://bit.ly/cvarp-pytn-1

Google Self Driving Car

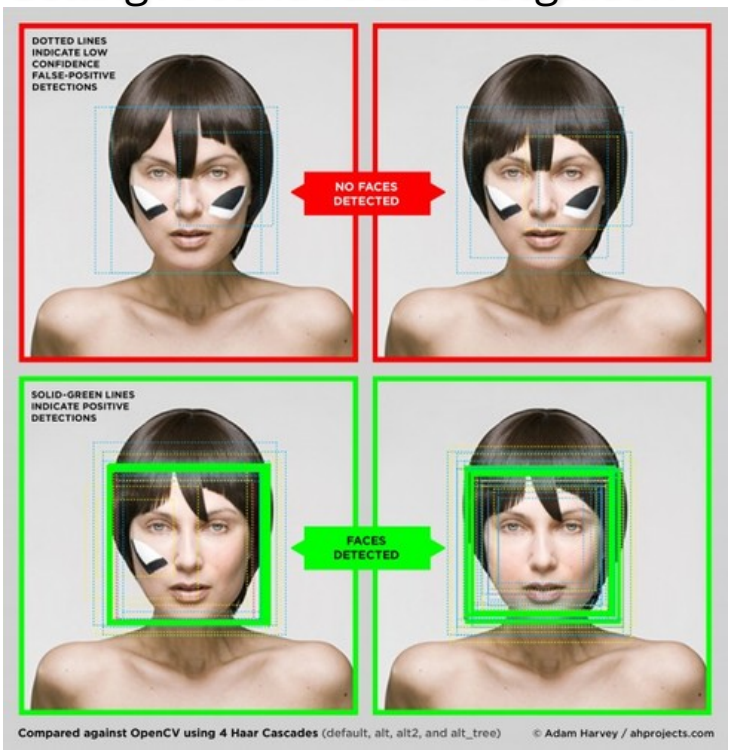
http://bit.ly/cvarp-pytn-2

Facial Recognition



http://bit.ly/cvarp-pytn-3

Getting Around Facial Recognition



http://bit.ly/cvarp-pytn-4

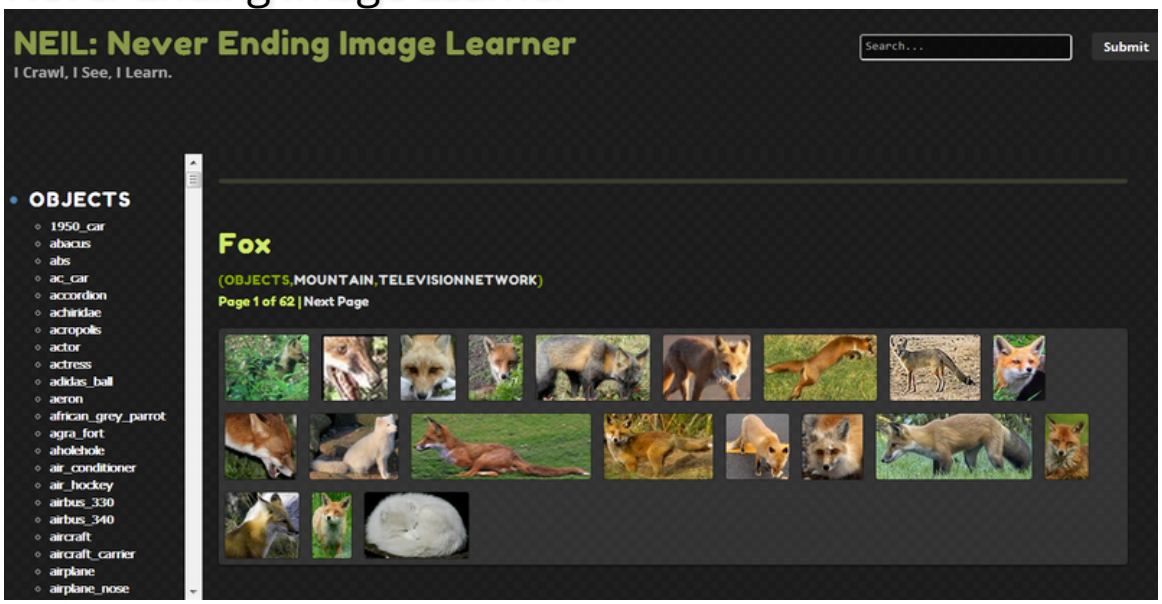
Video Surveillance and Analytics

http://bit.ly/cvarp-pytn-5

Microsoft Kinect

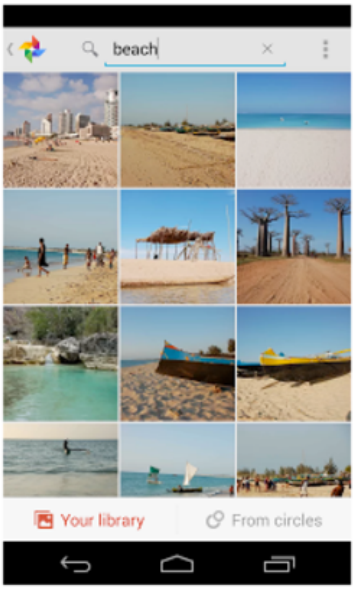
http://bit.ly/cvarp-pytn-6

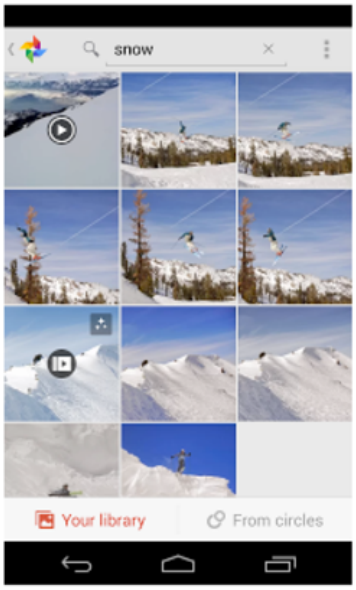
Never Ending Image Learner



http://bit.ly/cvarp-pytn-7

Google Object Recognition





http://bit.ly/cvarp-pytn-8

Google Auto Awesome (Before)

http://bit.ly/cvarp-pytn-9

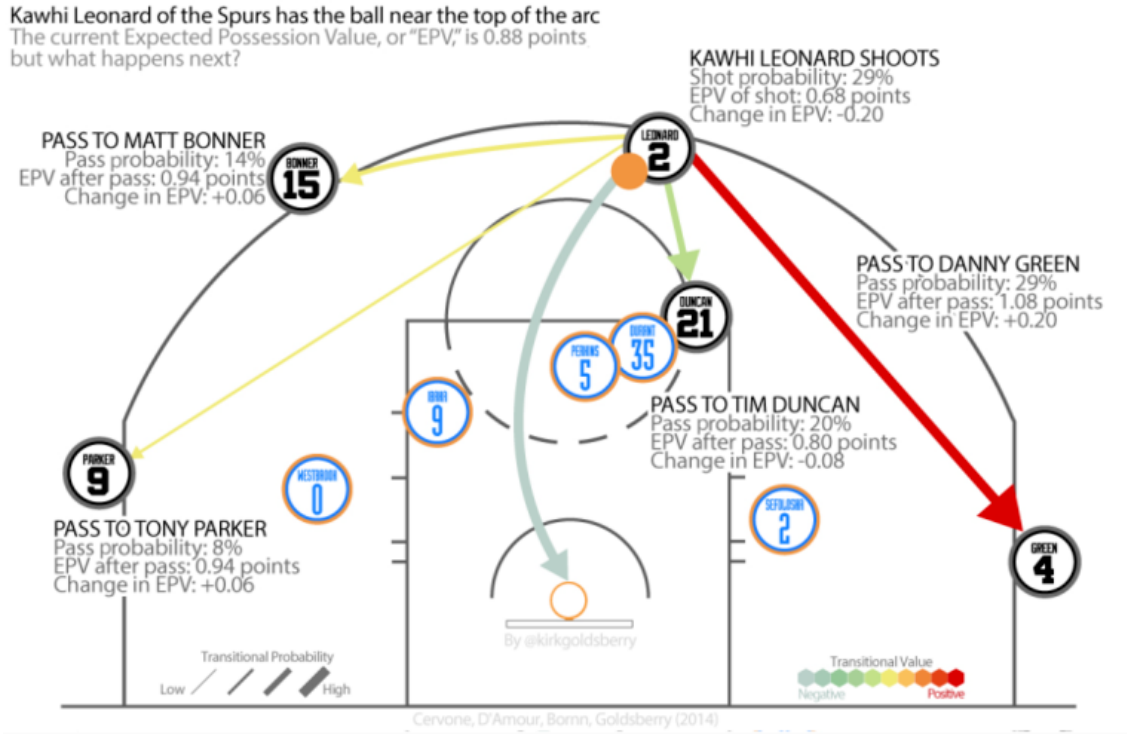
Google Auto Awesome (After)

http://bit.ly/cvarp-pytn-9

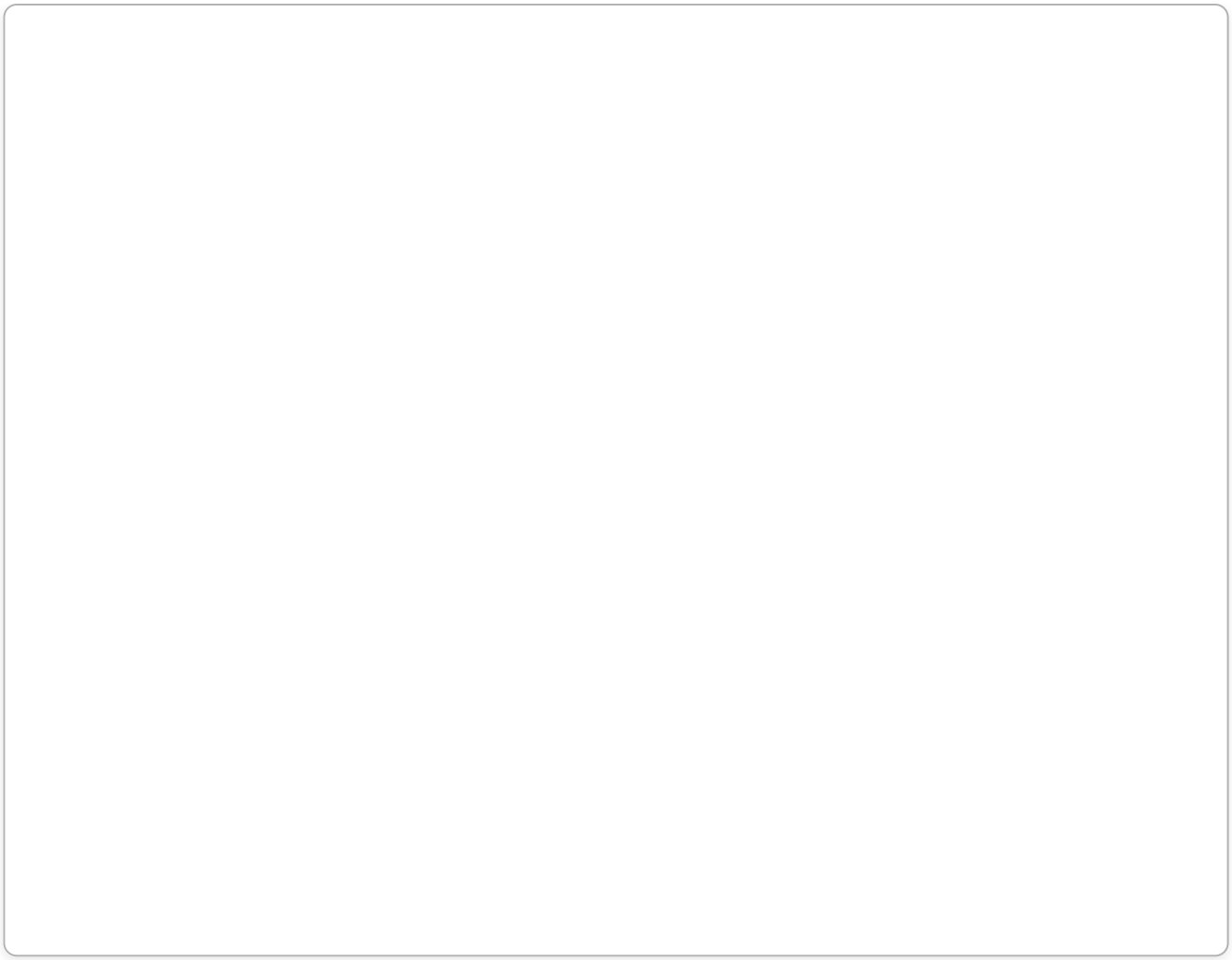
Sports Statistics (Tech)

http://bit.ly/cvarp-pytn-10

Sports Statistics (Results)



http://bit.ly/cvarp-pytn-11



What is Augmented Reality?

Augmented Reality is altering the visual image to:

- Impart knowledge
- Inform action

Augmented reality doesn't have to incorporate computer vision.

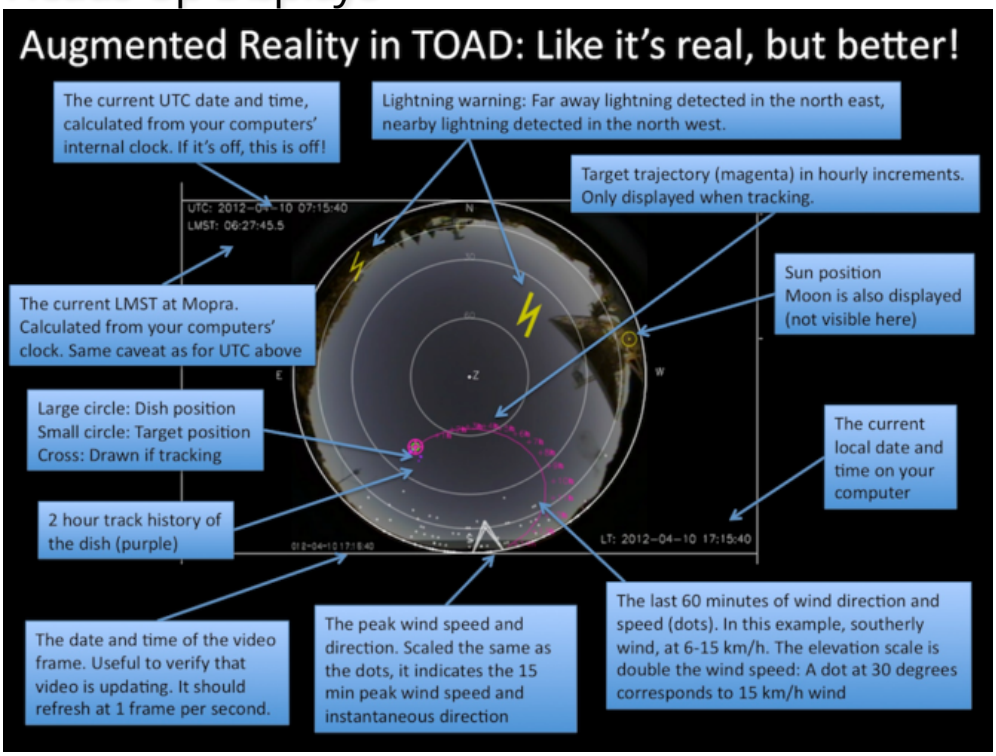
Inputs informing augmented reality display can be non-visual. For example:

- GPS
- Sensor data
- Backend databases

Augmented Reality Without Computer Vision

http://bit.ly/cvarp-pytn-12

Heads-Up Displays

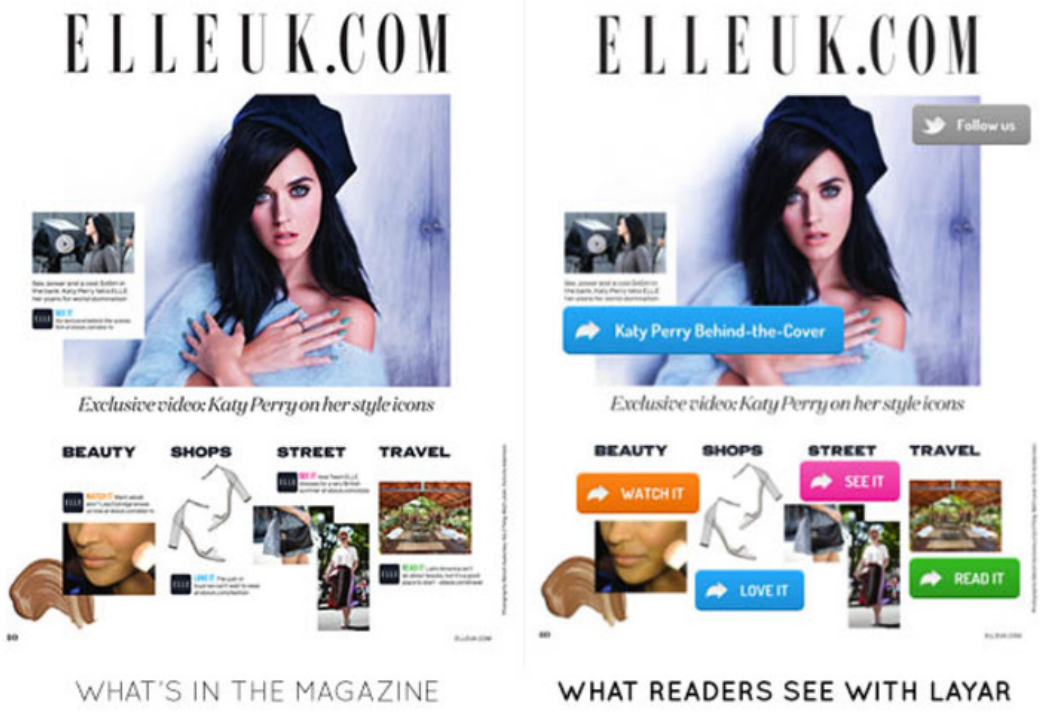


http://bit.ly/cvarp-pytn-13

Football on TV

http://bit.ly/cvarp-pytn-14

Layar App



http://bit.ly/cvarp-pytn-15

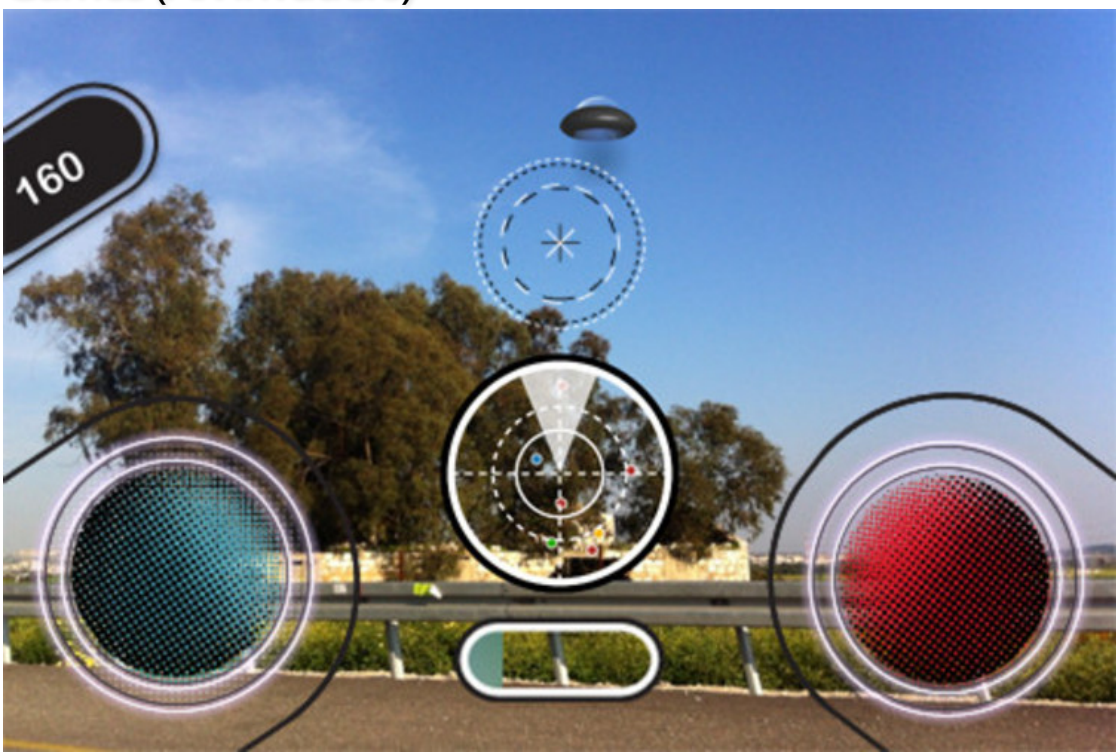
Visualization of Furniture (IKEA)

http://bit.ly/cvarp-pytn-16

Education

http://bit.ly/cvarp-pytn-17

Games (AR Invaders)



http://bit.ly/cvarp-pytn-18

What does it mean to extract information from an image?

- Color and intensity
  - *Is the sky red? Must be sunset.*
- Segmentation
  - *These parts of an image are related to each other*
- Feature detection
  - *This is an edge*
- Object Recognition
  - *This is a hammer*
- Difference
  - *This object has moved from the previous image*

Why now?

Computer Vision and Augmented Reality applications have blossomed with the blossoming of smart phones.

But just as important is the increase in computational power.

You have a *supercomputer* on your lap,

and a *powerful desktop computer* in your pocket.

Why am I interested in computer vision and augmented reality?

I have been taking pictures of the sky for many years.

It is automated.

It takes lots of pictures.

I don't have time to look at them all!

Version 1 of Timelapse Camera

About 1,000,000 images taken once per minute over two years.

About 12GB of 640x480 jpg files.

Version 2 of Timelapse Camera

Taking a picture every 10 seconds, since July of this year.

Currently over 2,000,000 images, and over 1.3 TB of jpg files.

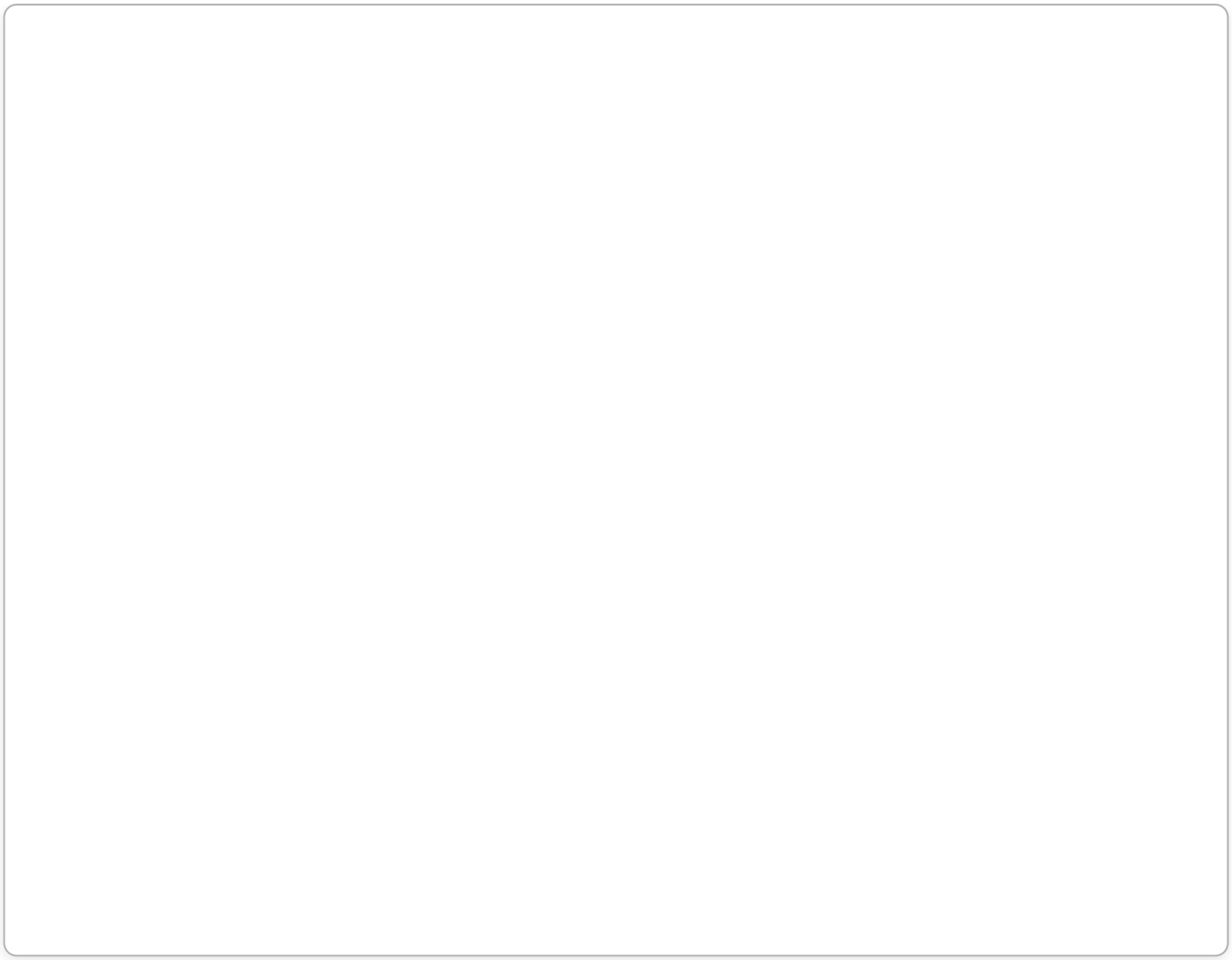
That is A LOT of images to look at!

Why would I want to look at them?

- For pretty sunsets
- For birds
- For cloud movement
- For UFOs

Sunset

Birds



How do we get started in Python?

- Python Imaging Library (pillow)
- OpenCV (pyopencv)
- SimpleCV (simplecv)
- Books

I recommend starting your journey with SimpleCV. PyOpenCV is a wrapper around OpenCV, SimpleCV is a framework.

SimpleCV

- Offers an interactive console based on IPython
- Proves a tutorial
- Provides lots of examples
- Abstracts a lot of hard concepts
- Pulls together complementary technologies (like PIL)

Installing SimpleCV

SimpleCV requires:

- opencv
- numpy
- scipy
- PIL
- ipython
- svgwrite

SimpleCV is in active development (and has a few rough edges).

I think it's best to use the current code from GitHub.

Installing SimpleCV Requirements

Easiest: Install your system's package for opencv, numpy, scipy, and PIL, at least.

Then, make a virtualenv for your SimpleCV work.

```
[dev]$ mkvirtualenv --system-site-packages vision
New python executable in vision/bin/python
Installing Setuptools.................done.
Installing Pip.......................done.
(vision)[dev]$
```

Getting SimpleCV

Next, you'll want to get the current codebase from github.

```
(vision)[dev]$ git clone git://github.com/sightmachine/SimpleCV.git
Cloning into 'SimpleCV'...
remote: Reusing existing pack: 13952, done.
remote: Total 13952 (delta 0), reused 0 (delta 0)
Receiving objects: 100% (13952/13952), 131.98 MiB | 2.67 MiB/s, done.
Resolving deltas: 100% (9123/9123), done.
(vision)[efloehr@agali dev]$ cd SimpleCV/
(vision)[SimpleCV]$
```

Install Any Remaining Requirements

If there are any requirements still needed, and even if not it doesn't hurt, run `pip install`.

```
(vision)[]$ pip install -r requirements.txt
Requirement already satisfied (use --upgrade to upgrade):
  numpy in /usr/lib64/python2.7/site-packages (from -r requirements.t›
Requirement already satisfied (use --upgrade to upgrade):
  scipy in /usr/lib64/python2.7/site-packages (from -r requirements.t›
Downloading/unpacking PIL (from -r requirements.txt (line 3))
  You are installing a potentially insecure and unverifiable file. Fut
  Downloading PIL-1.1.7.tar.gz (506kB): 506kB downloaded
  Running setup.py egg_info for package PIL
...
Successfully installed PIL ipython
Cleaning up...
(vision)[SimpleCV]$
```

Install SimpleCV Itself

The final step is to install SimpleCV.

```
(vision)[SimpleCV]$ python setup.py install
running install
running bdist_egg
running egg_info
creating SimpleCV.egg-info
...
Installed /home/efloehr/.virtualenvs/vision/lib/python2.7/site-package
Processing dependencies for SimpleCV==1.3
Finished processing dependencies for SimpleCV==1.3
(vision)[SimpleCV]$
```

Running SimpleCV

```
(vision)[dev]$ simplecv
+-------------------------------------------------------------+
SimpleCV 1.3.0 [interactive shell] - http://simplecv.org
+-------------------------------------------------------------+

Commands:
        "exit()" or press "Ctrl+ D" to exit the shell
        "clear()" to clear the shell screen
        "tutorial()" to begin the SimpleCV interactive tutorial
        "example()" gives a list of examples you can run
        "forums()" will launch a web browser for the help forums
        "walkthrough()" will launch a web browser with a walkthrough

Documentation:
        help(Image), ?Image, Image?, or Image()? all do the same
        "docs()" will launch webbrowser showing documentation

SimpleCV:1>
```

Running an Example

NOTE: The examples are loaded dynamically from the examples directory. So they *likely* won't be the same numbers for you.

```
SimpleCV:3> example 30
running example: ...SimpleCV/examples/detection/CannyCam.py

This example just takes an image, finds the edges, and draws them
the threshold is used for the edge detection, if you adjust the
max_threshold and threshhold_step values and run the program you will
see it change over time
```

Which Brings me to...Computer Vision as art!

A More Practical Example

```
SimpleCV:12> example 41

This example finds a quarter in the picture and then uses that measure
to determine the rest of the coins in the picture.  Since a quarter is
a certain size we can use it as a reference because it is known.

The sizes of coins are as follows:
penny - 19.05 mm
nickel - 21.21 mm
dime - 17.9 mm
quarter - 24.26 mm
```

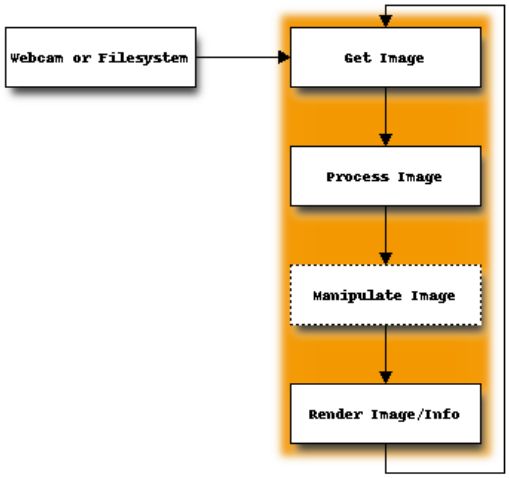
Measures Coins and Identifies Them

Looking at the Examples:

The examples are in SimpleCV/examples in the repo and in the egg. This is a portion of example 30:

```
from SimpleCV import *

cam = Camera() #initialize the camera

while True:
    # get image (or frame) from camera
    image = cam.getImage()
    # flip it so it looks mirrored
    flipped_image = image.flipHorizontal()
    # get the image edges
    edged_image = flipped_image.edges(threshold)

    edged_image.drawText("Current Edge Threshold:" + str(threshold),
                         10,10, fontsize=30, color=Color.GREEN)
    edged_image.show()
```

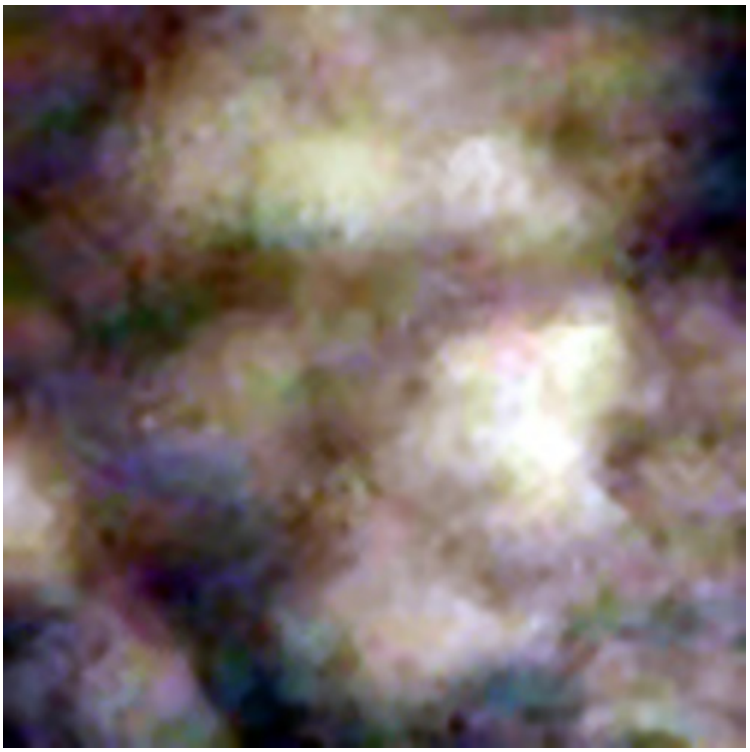
Computer Vision/Augmented Reality Workflow



1. Get Image
2. Process image
3. Manipulate image (optional)
4. Display or save image/info

Finding Faces in the Clouds

CSI Enhance!



Pseudocode

1. Iterate through directory of timelapse pictures
2. Throw away any pictures that are dark (nightime)
3. Look for faces
4. Crop the faces individually and save as a separate file
5. Take original and outline faces for context and save

Get Image

```
import SimpleCV

for imagenum, image in enumerate(images):
    # Don't do for sure dark
    img = SimpleCV.Image(os.path.join(dirpath, day, hour, image))

    # Check if dark
    r, g, b = img[32,32]
    if r<20 and g<20 and b<20:
        darkcount += 1
        continue
```

Process Image

```
# Find a face
scalesize = 1.0
padding = 1.5
scaledimg = img.scale(scalesize)

scalar = 1/scalesize
faces = scaledimg.findHaarFeatures("face.xml",
                                   min_neighbors=17,
                                   min_size=(50,50))

if faces:
    for facenum, face in enumerate(faces):
```

Manipulate Image And Save

```
for facenum, face in enumerate(faces):
    scaled_x = face.x * scalar
    scaled_y = face.y * scalar
    scaled_width = face.width() * scalar * padding
    scaled_height = face.height() * scalar * padding

    faceimg = img.crop(face)

    faceimg.save(os.path.join(resultpath, "{0}-{1}.jpg"\
                 .format(image.split('.')[0], facenum)))

    img.drawRectangle(scaled_x - (scaled_width/2.0),
                      scaled_y - (scaled_height/2.0),
                      scaled_width,
                      scaled_height)

img.save(os.path.join(resultpath, image))
```

The SimpleCV Console

The SimpleCV console is a great tool for exploration. Specificly:

- Tab completion will show available methods
- help() or object? or function()? will display help text
- docs command will bring up a web browser with full documentation

Sudoku

Our goal will be for our computer to get our computer to:

1. Identify a Sudoku puzzle in an image
2. Identify individual squares in a Sudoku puzzle
3. Recognize digits in the Sudoku puzzle
4. Construct a representation of the Sudoku puzzle in the computer
5. Solve the Sudoku puzzle
6. Place the solved digits back into the image
7. Display the image and save the solved puzzle

Kudos

Kudos to Abid Rahman K at http://opencvpython.blogspot.com/ for a great series of posts on solving Sudoku with OpenCV.

Also, there is an implementation of a solver in the Programming Computer Vision with Python book by Jan Erik Solem which also uses OpenCV.

Kudos also to Peter Norvig for his Sudoku Solver at http://norvig.com/sudoku.html which I have used to actually solve the Sudoku puzzles.

Identify a Sudoku puzzle

We generally want to make things as easy on the computer as possible.

There is generally a lot of extraneous information in an image.

Color is one of them.

So our steps will be:

- Remove color and make the image black and white (no grays even)
- Do this in a way that lines and numbers are clear
- Identify the largest rectangle in the image, because that will be the puzzle

Removing Color and Smoothing

```
from SimpleCV import Image
import cv2

# Load the Image
raw_image = Image('sudoku.jpg')

# Remove color
gray_image = raw_image.grayscale()

# Smooth to remove speckle
smooth_image = gray_image.gaussianBlur((5,5),0)
```

Thesholding by Going To OpenCV

```
# Convert to Numpy Array For OpenCV use
cv_image = smooth_image.getGrayNumpyCv2()

# Adaptive threshold does much better than linear
raw_thresh_image = cv2.adaptiveThreshold(cv_image,255,1,1,11,2)

# Convert back to a SimpleCV image
thresh_image = Image(raw_thresh_image)

# For some reason it gets rotated and flipped, reverse
thresh_image = thresh_image.rotate90().flipVertical()
```

Identify Largest Rectangle

```
# Find "blobs" which are interesting items in the image
blobs = thresh_image.findBlobs()

# Assume the largest rectangular blob is our puzzle
puzzle_blob = None
puzzle_area = 0

for blob in blobs:
    if blob.isRectangle() and blob.area() > puzzle_area:
        puzzle_blob = blob
        puzzle_area = blob.area()
```

Identify Individual Squares

```
# Slice into 81 squares (simple approach)
blocks = []
for y in range(0,9):
    for x in range(0,9):
        block = puzzle_image.crop(x*block_width, y*block_height, block
        blocks.append(block)
```

Recognize Digits

We will use the code at http://bit.ly/cvarp-pytn-numrec which:

1. Take a solved Sudoku puzzle picture, hand recognize the digits
2. Take the recognized digits, resize them to the same size, and save as a numpy array
3. Take the saved, recognized digits and load into OpenCV K-Nearest Neighbors *(basically learning by example)*
4. Find only things that could be numbers in each of the 81 blocks of the puzzle
5. Assuming its a number, recognize it

Recognize Digits (Code)

```
# Read the stored digits
samples = np.loadtxt('generalsamples.data',np.float32)
responses = np.loadtxt('generalresponses.data',np.float32)
responses = responses.reshape((responses.size,1))
model = cv2.KNearest()
model.train(samples,responses)

# For each number blob, size to standard size
small = blobblock.blobImage().resize(10,10).getGrayNumpyCv2()
small = blobsmall.reshape((1,100))
small = np.float32(blobsmall)

# And get the nearest neighbor
al, results, neigh_resp, dists = model.find_nearest(blobsmall, k = 1)
t = str(int((results[0][0])))
```

Construct a Representation and Solve

Use a simple string, with '.' an empty block and the number for a recognized number. We can pass this in directly to Norvig's Sudoku's solver.

For example:

.1.2.3....4...5...3.2.6..7.2..1..3...35...68...9..7..2.5..7.2.4.

```
solved_puzzle = solve(puzzle_repr)
print display(solved_puzzle)
```

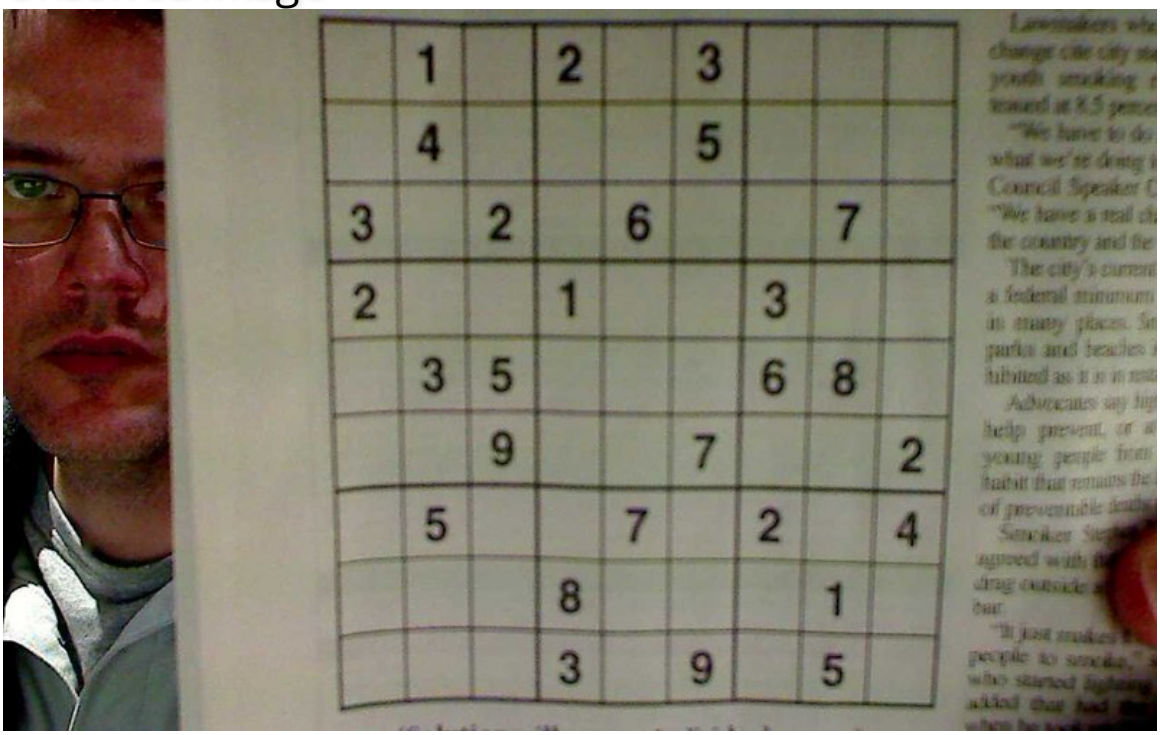
Place Solved Digits Into Image

All we do is take the images of the recognized numbers and use them to place into the original image in the unsolved blocks.

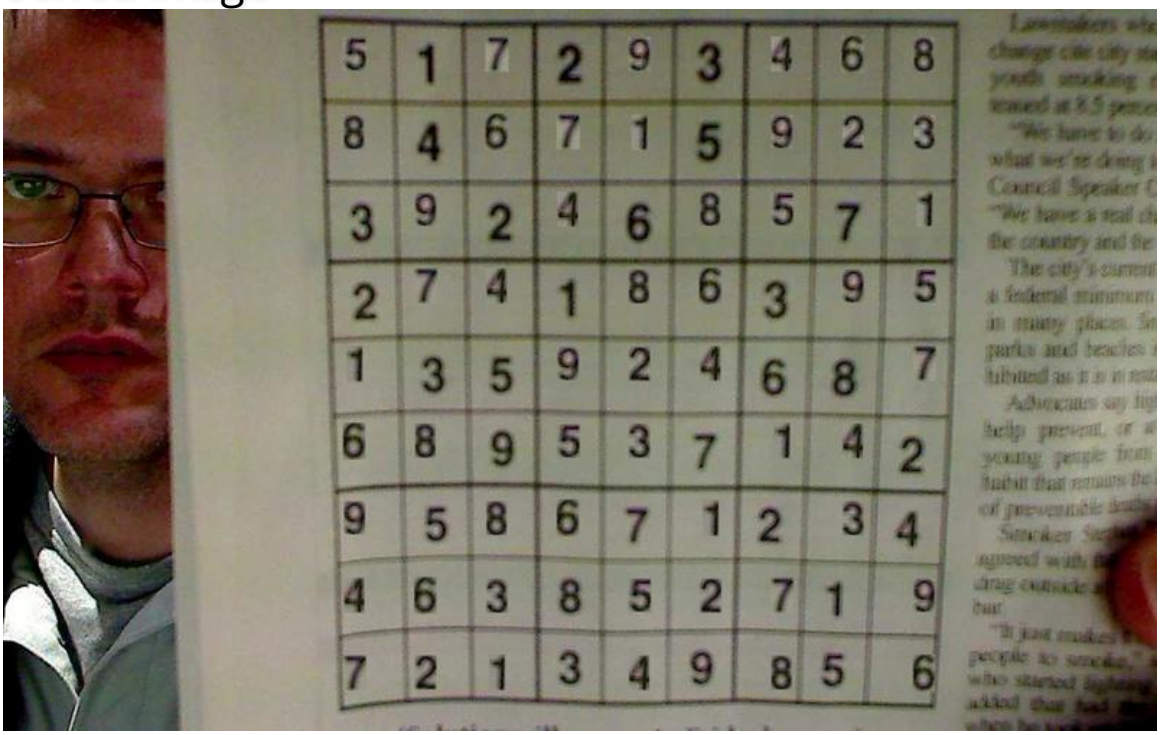
This assumes the unsolved puzzle contains all the digits!

```
numpic_width, numpic_height = numpic.size

xpaste = int(round(puzzle_offset_x \
                + (block_width * (x+0.5)) \
                - (numpic_width/2.0)))
ypaste = int(round(puzzle_offset_y \
                + (block_height * (y+0.5)) \
                - (numpic_height/2.0)))

solved_puzzle_image.paste(numpic, (xpaste, ypaste))
```

Unsolved Image



Solved Image



Areas of Improvement

This is just a start. Like everything, there is room for improvement.

1. It is sensitive to identification of digits
2. It doesn't take deformation into account
3. It could do better smoothing to blend in the numbers

Code and Contact

Eric Floehr

eric@intellovations.com

http://www.intellovations.com

Code: http://bit.ly/cvarp-pytn-code

Presentation: http://bit.ly/cvarp-pytn-talk