OpenCV Applications

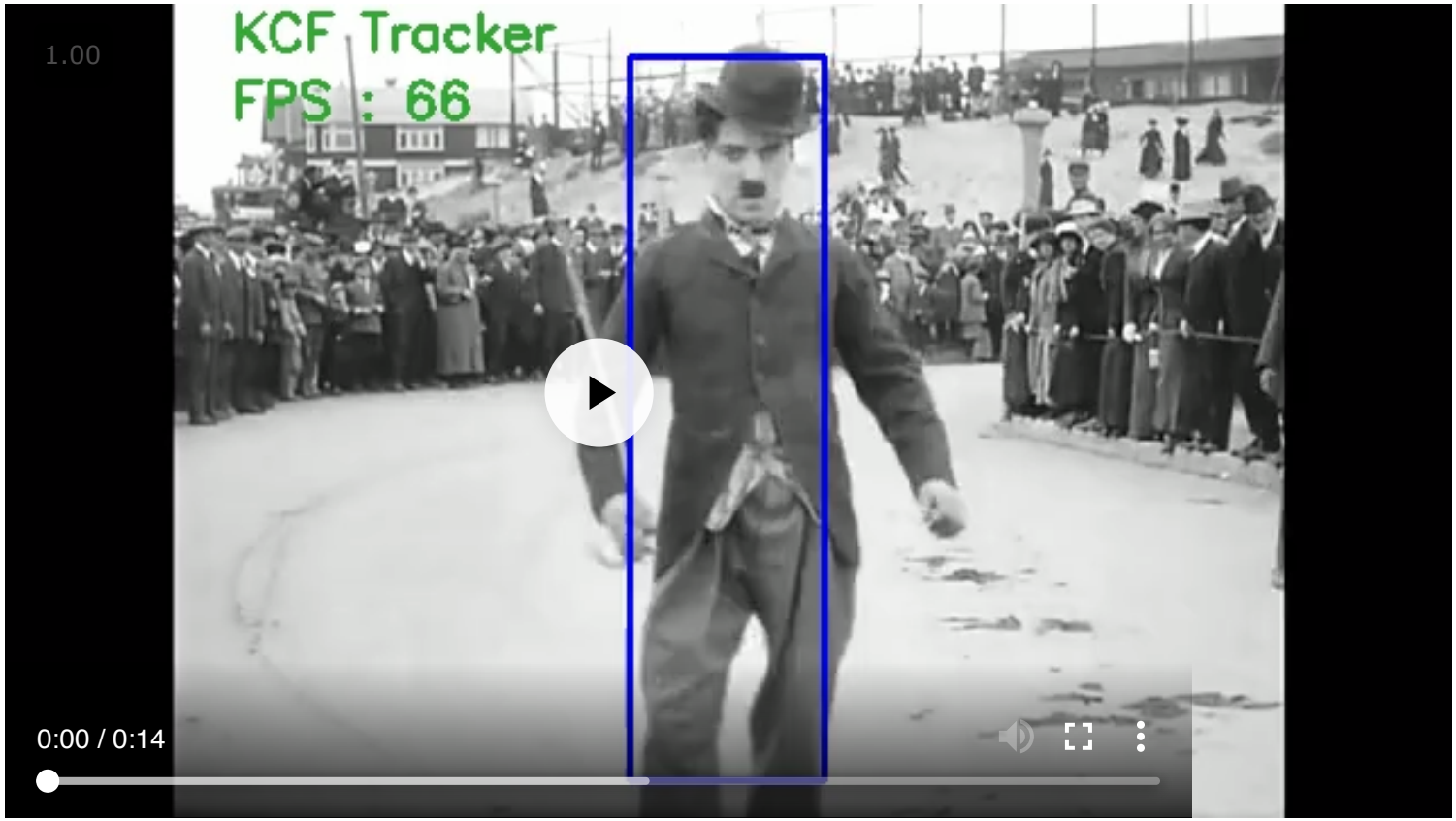
By **Satya Mallick, LearnOpenCV.com**

Image Alignment & Panorama

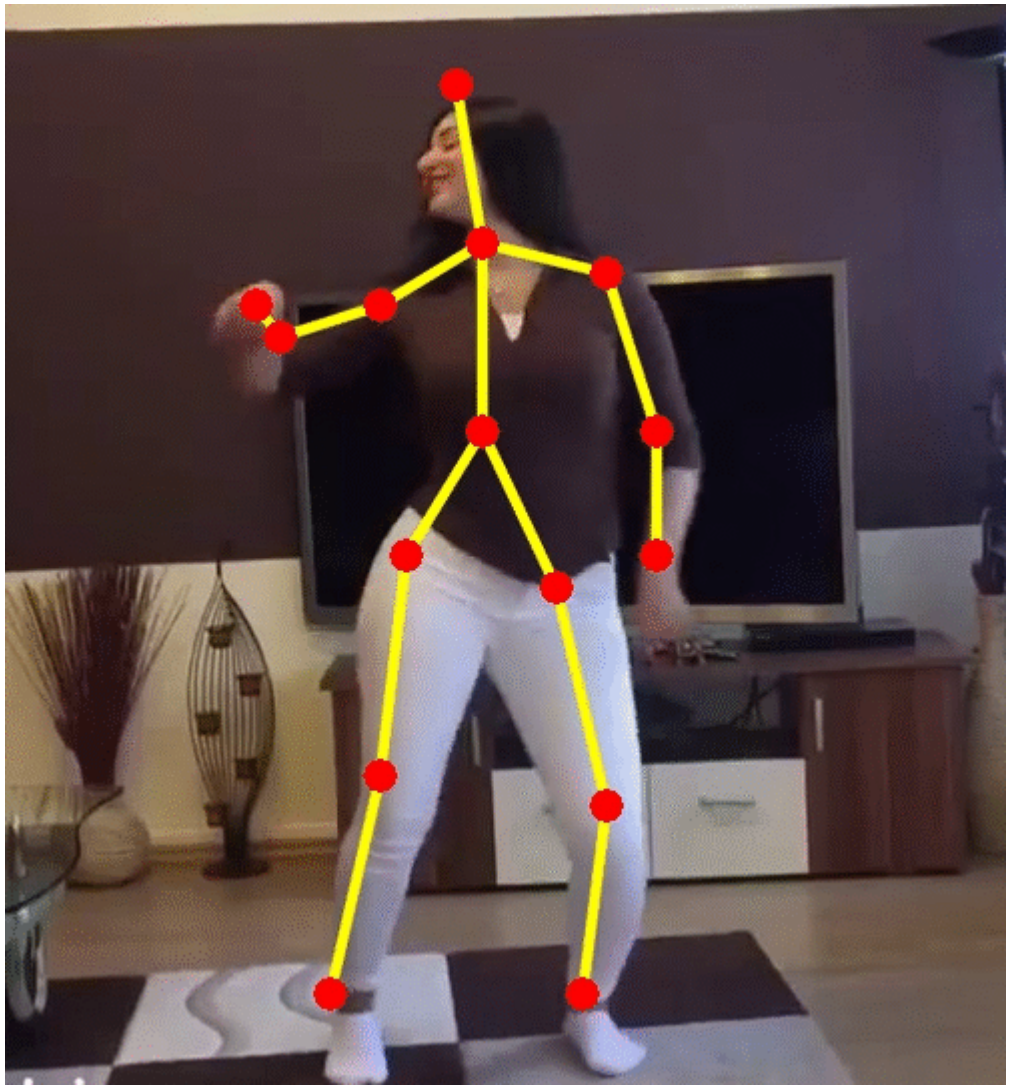
Seamless Cloning

Object Tracking

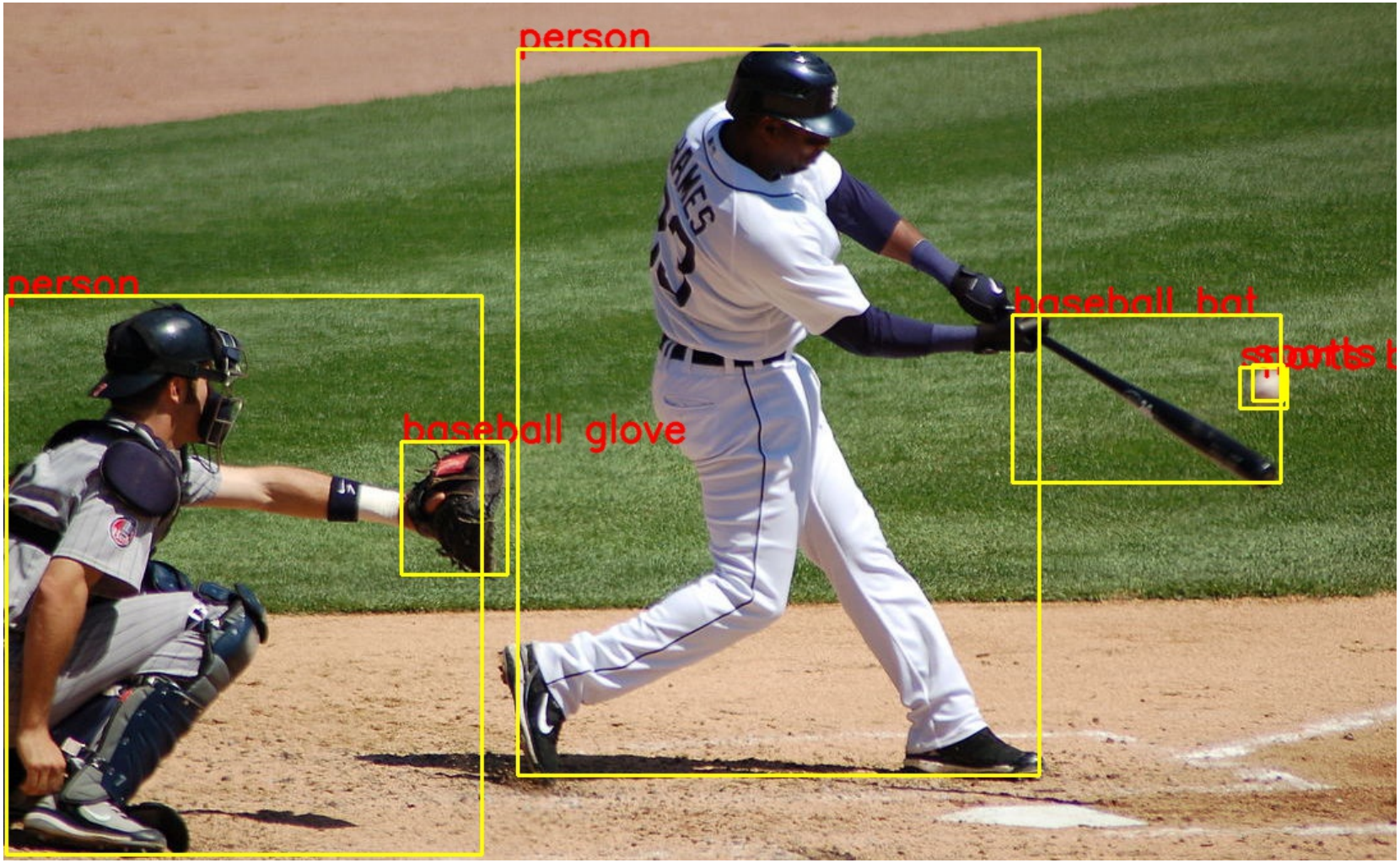
In [1]:
```
%%HTML
<video width="1024" controls>
  <source src="images/tracking.mp4" type="video/mp4">
</video>
```



Pose Estimation : Deep Learning



Object Detection : Deep Learning



High Dynamic Range Imaging

About Me

Trust me, I am a **Doctor**

- 2001 - 2006, Ph.D. University of California, San Diego.

And an **Entrepreneur**

- 2007 - 2015, Co-Founder / CTO, Sight Commerce Inc.
- 2015 - Present, Partner Big Vision LLC.

And a **Blogger**

- 2015 - Present, @LearnOpenCV.com


YOU HAVE YOUR OWN BLOG?
YOUR LIFE MUST BE ALL KINDS OF FASCINATING

Sight Commerce Inc.

Virtual Makeover

Sight Commerce Inc.

Virtual Hairstyle

Sight Commerce Inc.

Virtual Sunglasses

Sight Commerce Inc.

Virtual Jewelry

Big Vision LLC : Consulting

1. **Document analysis**
    - Digital ID tamper detection
    - Document alignment
    - OCR
2. **Fashion**
    - Visual search
    - Counterfeit detection
3. **Security**
    - Intrusion detector
    - Face recognition
    - Drone application
4. **Medical**
    - Auto refractor
    - Tissue analysis
    - Parasite counting in horse feces

LearnOpenCV.com

Course : Computer Vision for Faces (http://cv4faces.com (http://cv4faces.com))

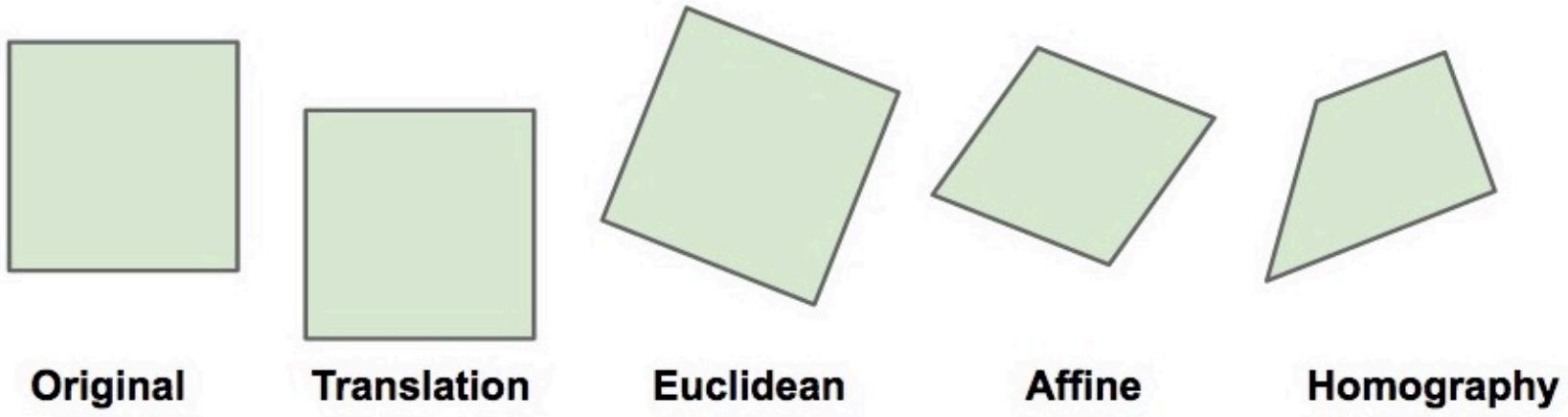
Image Alignment

Satya Mallick, LearnOpenCV.com

Align an image to a template.

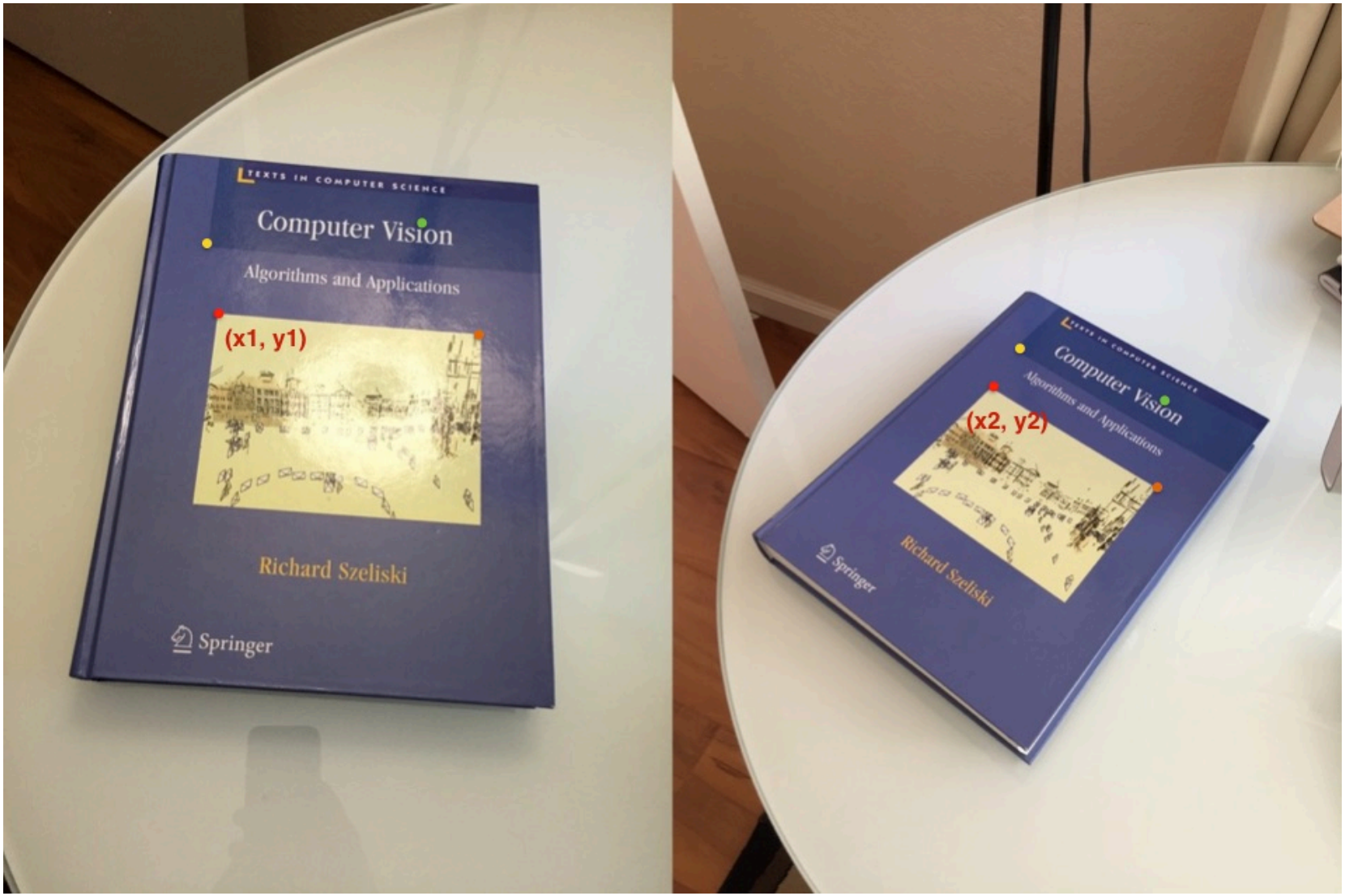
Theory

1. A **Homography** transforms a square to arbitrary quad.



Theory

1. Images of two planes are related by a **Homography**
2. We need **4 corresponding points** to estimate Homography


TEXTS IN COMPUTER SCIENCE
Computer Vision
Algorithms and Applications
(x1, y1)
Richard Szeliski
Springer
(x2, y2)

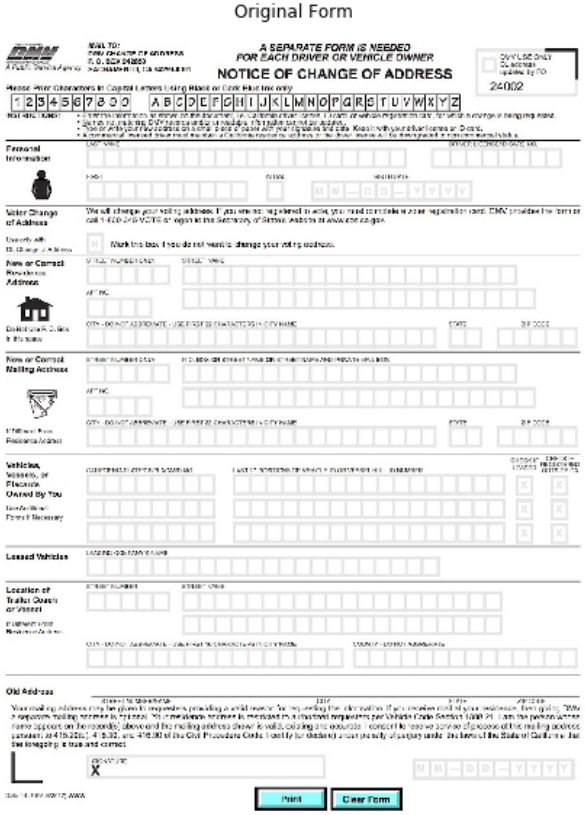
Step 1: Read Tempalate and Scanned Image

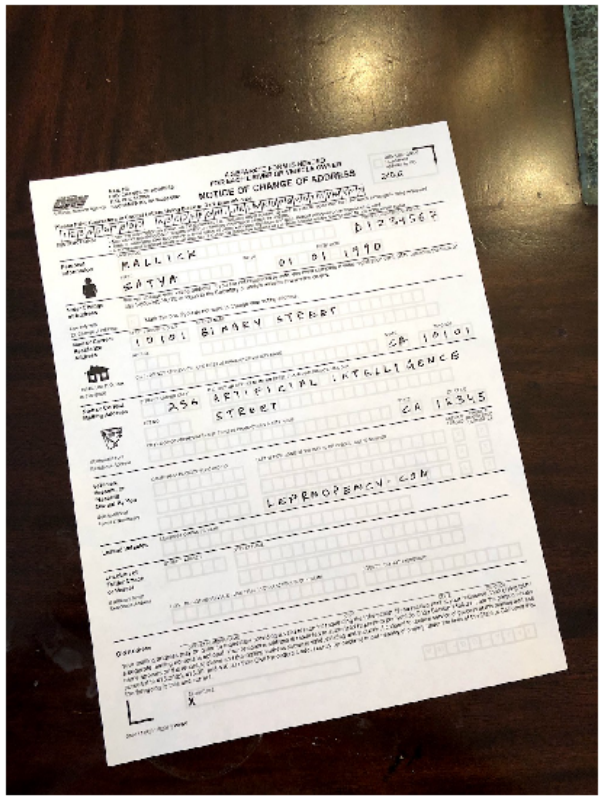
In [14]:
```
# Read reference image
refFilename = "form.jpg"
print("Reading reference image : ", refFilename)
im1 = cv2.imread(refFilename, cv2.IMREAD_COLOR)
im1 = cv2.cvtColor(im1, cv2.COLOR_BGR2RGB)

# Read image to be aligned
imFilename = "scanned-form.jpg"
print("Reading image to align : ", imFilename)
im2 = cv2.imread(imFilename, cv2.IMREAD_COLOR)
im2 = cv2.cvtColor(im2, cv2.COLOR_BGR2RGB)
```

```
Reading reference image :  form.jpg
Reading image to align :  scanned-form.jpg
```

In [24]:
```
# Display Images

plt.figure(figsize=[20,10]);
plt.subplot(121); plt.axis('off'); plt.imshow(im1); plt.title("Original Form")
plt.subplot(122); plt.axis('off'); plt.imshow(im2); plt.title("Scanned Form")
```

Out[24]:
```
Text(0.5,1,'Scanned Form')
```



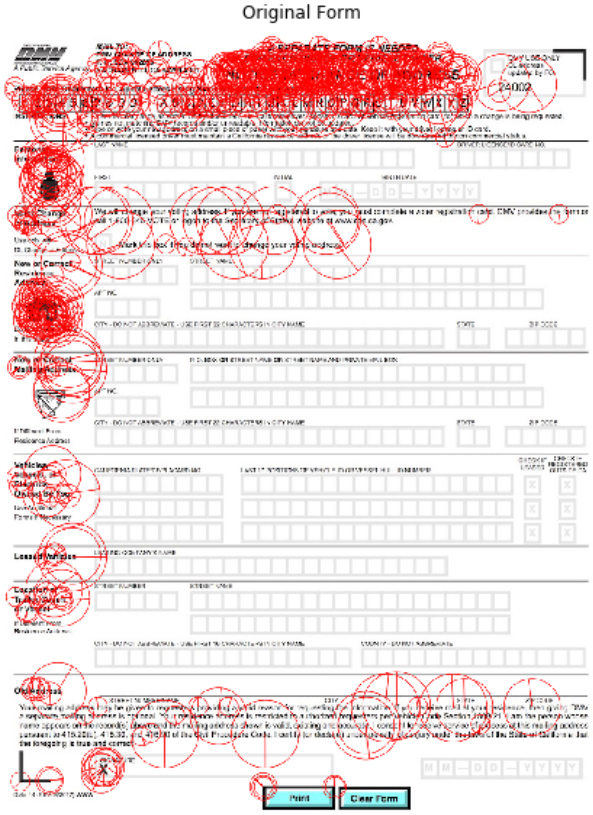


Step 2: Find keypoints in both Images

Think of keypoints as corner points that are stable under image transformations

In [16]:
```
# Convert images to grayscale
im1_gray = cv2.cvtColor(im1, cv2.COLOR_BGR2GRAY)
im2_gray = cv2.cvtColor(im2, cv2.COLOR_BGR2GRAY)


# Detect ORB features and compute descriptors.
MAX_NUM_FEATURES = 500
orb = cv2.ORB_create(MAX_NUM_FEATURES)
keypoints1, descriptors1 = orb.detectAndCompute(im1_gray, None)
keypoints2, descriptors2 = orb.detectAndCompute(im2_gray, None)

# Display
im1_display = cv2.drawKeypoints(im1, keypoints1, outImage=np.array([]), color=(255
, 0, 0), flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
im2_display = cv2.drawKeypoints(im2, keypoints1, outImage=np.array([]), color=(255
, 0, 0), flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
```

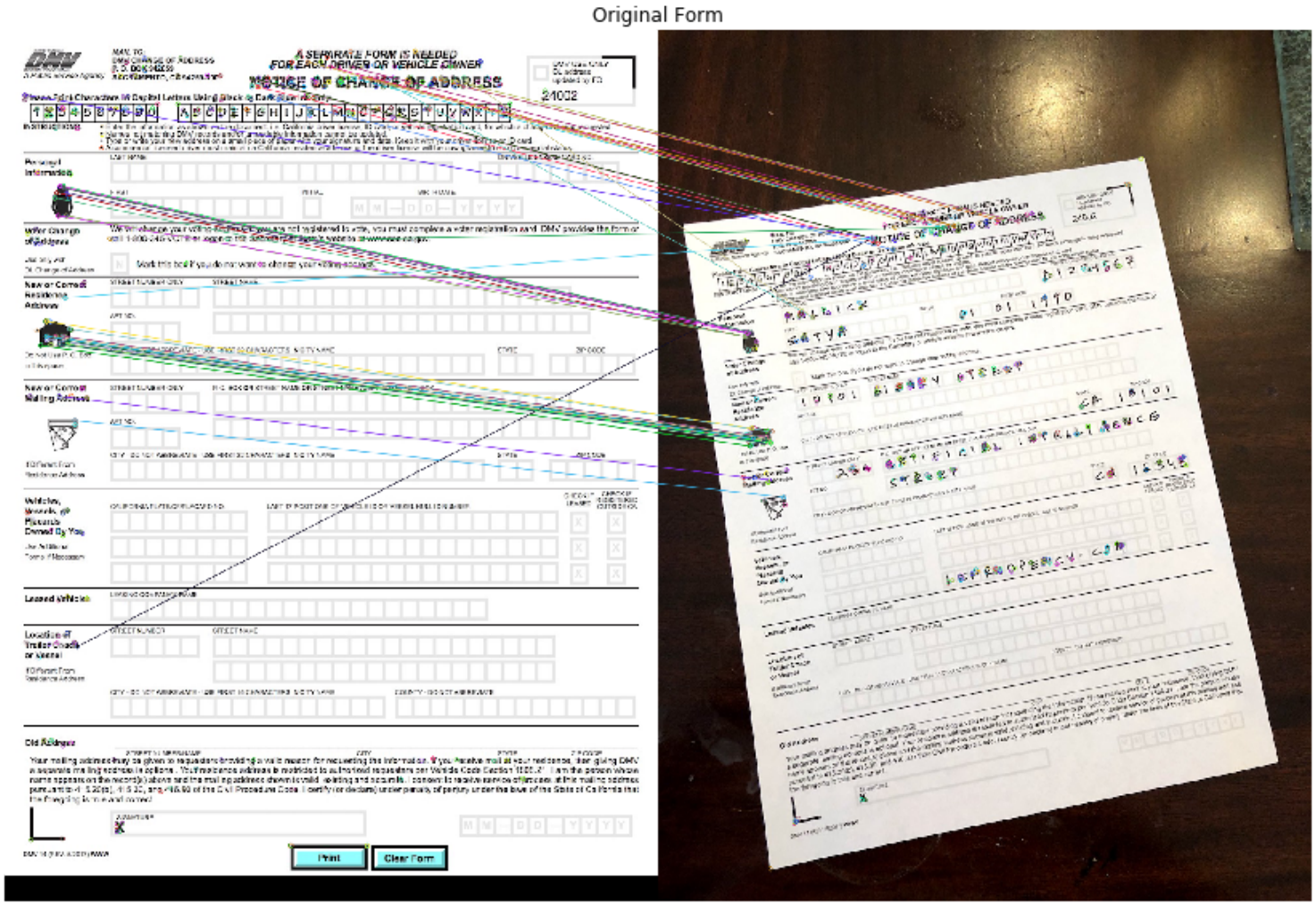
In [28]:
```
plt.figure(figsize=[20,10])
plt.subplot(121); plt.axis('off'); plt.imshow(im1_display); plt.title("Original Fo
rm");
plt.subplot(122); plt.axis('off'); plt.imshow(im2_display); plt.title("Scanned For
m");
```



Step 3 : Match keypoints in the two image

In [18]:
```
# Match features.
matcher = cv2.DescriptorMatcher_create(cv2.DESCRIPTOR_MATCHER_BRUTEFORCE_HAMMING)
matches = matcher.match(descriptors1, descriptors2, None)

# Sort matches by score
matches.sort(key=lambda x: x.distance, reverse=False)

# Remove not so good matches
numGoodMatches = int(len(matches) * 0.1)
matches = matches[:numGoodMatches]
```

In [27]:
```
# Draw top matches
im_matches = cv2.drawMatches(im1, keypoints1, im2, keypoints2, matches, None)

plt.figure(figsize=[40,10])
plt.imshow(im_matches); plt.axis('off'); plt.title("Original Form");
```



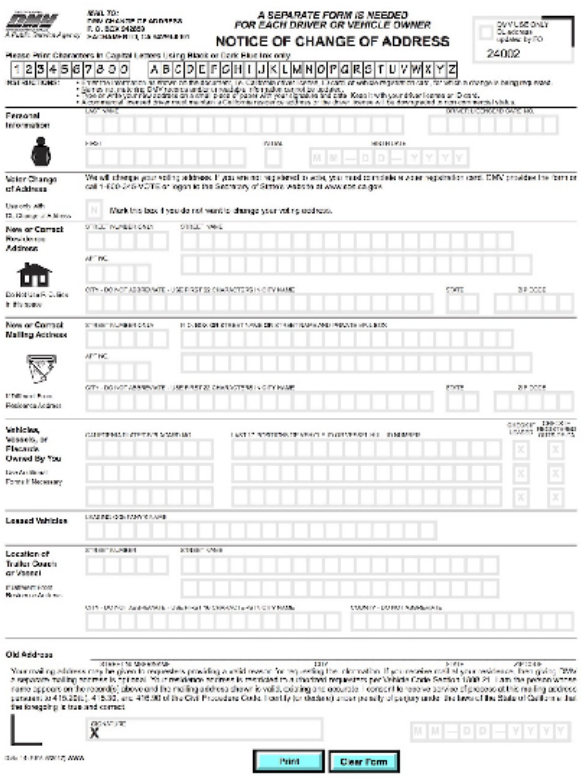
Step 4: Find Homography

In [20]:
```
# Extract location of good matches
points1 = np.zeros((len(matches), 2), dtype=np.float32)
points2 = np.zeros((len(matches), 2), dtype=np.float32)

for i, match in enumerate(matches):
  points1[i, :] = keypoints1[match.queryIdx].pt
  points2[i, :] = keypoints2[match.trainIdx].pt

# Find homography
h, mask = cv2.findHomography(points2, points1, cv2.RANSAC)
```

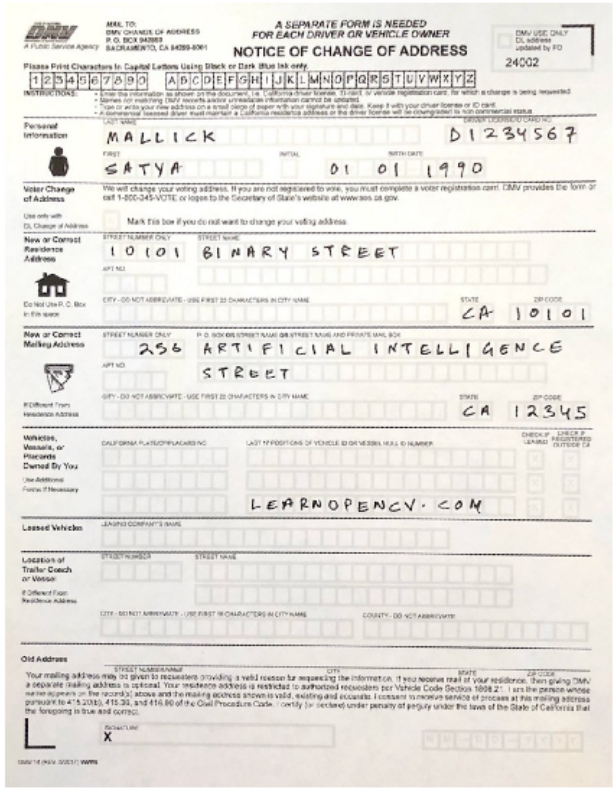
Step 5: Warp image

In [30]:
```
# Use homography to warp image
height, width, channels = im1.shape
im2_reg = cv2.warpPerspective(im2, h, (width, height))
# Display results
plt.figure(figsize=[20,10]);
plt.subplot(121); plt.imshow(im1); plt.axis('off'); plt.title("Original Form");
plt.subplot(122); plt.imshow(im2_reg); plt.axis('off'); plt.title("Scanned Form");
```

Original Form



Scanned Form



Creating Panoramas using OpenCV

Satya Mallick, LearnOpenCV.com

Steps for Creating Panoramas

1. Find keypoints in all images
2. Find pairwise correspondences
3. Estimate pairwise Homographies
4. Refine Homographies
5. Stitch with Blending

Steps for Creating Panoramas using OpenCV

1. Use the **sticher** class

In [7]:
```
# Read Images

imagefiles = glob.glob("boat/*")
imagefiles.sort()

images = []
for filename in imagefiles:
  img = cv2.imread(filename)
  img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
  images.append(img)

num_images = len(images)
```

In [8]:
```
# Display Images
plt.figure(figsize=[30,10])
num_cols = 3
num_rows = math.ceil(num_images / num_cols)
for i in range(0, num_images):
  plt.subplot(num_rows, num_cols, i+1)
  plt.axis('off')
  plt.imshow(images[i])
```






In [9]:
```
# Stitch Images
stitcher = cv2.createStitcher()
status, result = stitcher.stitch(images)
if status == 0:
  plt.figure(figsize=[30,10])
  plt.imshow(result)
```

Seamless Cloning

Satya Mallick, LearnOpenCV.com

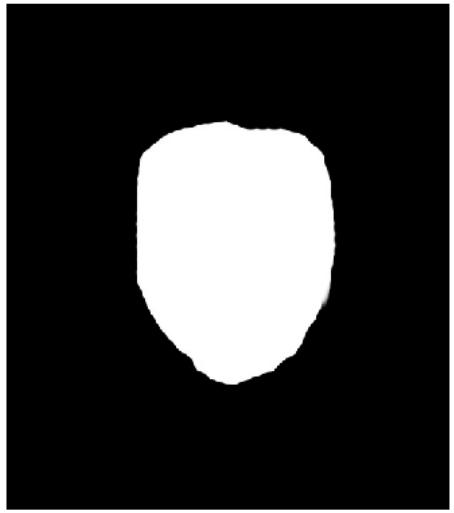
Goal

In [102]:
```
# Standard imports
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

In [114]:
```
# Read images
dst = cv2.imread("trump.jpg")
dst = cv2.cvtColor(dst, cv2.COLOR_BGR2RGB)

src = cv2.imread("obama.jpg")
src = cv2.cvtColor(src, cv2.COLOR_BGR2RGB)

src_mask = cv2.imread("obama-mask.jpg", cv2.IMREAD_GRAYSCALE)
```

In [115]:
```
plt.figure(figsize=(20,10));

plt.subplot(131); plt.imshow(dst); plt.axis('off');
plt.subplot(132); plt.imshow(src); plt.axis('off');
plt.subplot(133); plt.imshow(src_mask, cmap='gray'); plt.axis('off');
```





Simple Alpha Blending with Mask

1. The lighting in the images is very different
2. The skin tones are very different
3. The blend will look ridiculous

In [119]:
```
alpha = cv2.cvtColor(src_mask.copy(), cv2.COLOR_GRAY2RGB)
alpha = alpha.astype(np.float32) / 255.0
output_noclone = src * alpha + dst * (1 - alpha)
output_noclone = output_noclone.astype(np.uint8)
plt.figure(figsize=(7,7)); plt.imshow(output_noclone); plt.axis('off');
```

Solution?

Option 1

Perform gradient domain **Poisson blending**

1. Find the x and y gradients of the source and destination images
2. Copy the gradients from source images to the destination image
3. Integration the gradients domain with Dirichlet boundary conditions

Option 2

Use **cv2.seamlessClone** ... does all the above for you!

Find Blob Centroid

In [120]:
```
# Find blob centroid
ret, src_mask_bin = cv2.threshold(src_mask, 128,256, cv2.THRESH_BINARY)
m = cv2.moments(src_mask_bin)
center = (int(m['m01']/m['m00']), int(m['m10']/m['m00']) )
```

$$M_{ij} = \sum_x \sum_y x^i y^j I(x,y)$$

Seamless Cloning

In [121]:
```
# Clone seamlessly.
output = cv2.seamlessClone(src, dst, src_mask, center, cv2.NORMAL_CLONE)

plt.figure(figsize=(20,10));
plt.subplot(121); plt.imshow(output_noclone); plt.axis('off');
plt.subplot(122); plt.imshow(output); plt.axis('off');
```

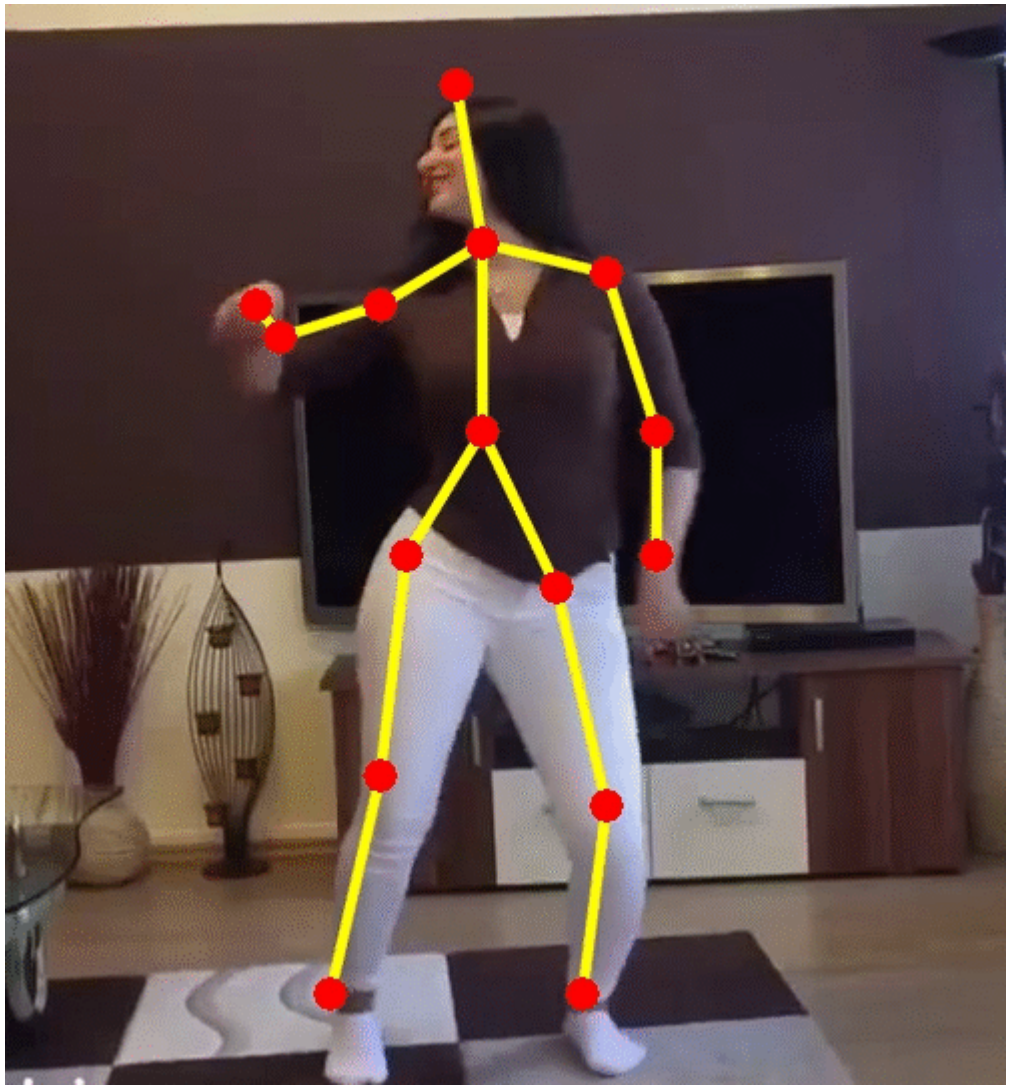
Deep Learning with OpenCV

Satya Mallick, LearnOpenCV.com

Highlights

1. Train elsewhere, perform inference in an OpenCV application
2. Supports **Caffe**, **Tensorflow**, **Torch** and **Darknet**.
3. **Supported layers** : AbsVal, AveragePooling, BatchNormalization, Concatenation, Convolution (including dilated convolution), Crop, Deconvolution, DetectionOutput (SSD-specific layer), Dropout, Eltwise (+, *, max), Flatten, FullyConnected, LRN, LSTM, MaxPooling, MaxUnpooling, MVN, NormalizeBBox (SSD-specific layer), Padding, Permute, Power, PReLU (including ChannelPReLU with channel-specific slopes), PriorBox (SSD-specific layer), ReLU, RNN, Scale, Shift, Sigmoid, Slice, Softmax, Split, TanH

Caffe Model Example : OpenPose



Load a Caffe Model

A typical Caffe Model has two files

1. **Architecture** : Defined in a **.prototxt** file
2. **Weights** : Defined in **.caffemodel** file

In [2]:
```
if not os.path.isdir('model'):
  os.mkdir("model")

protoFile = "model/mpi.prototxt"
weightsFile = "model/mpi.caffemodel"

if not os.path.isfile(protoFile):
  # Download the proto file
  !wget https://raw.githubusercontent.com/CMU-Perceptual-Computing-Lab/openpose/ma
ster/models/pose/mpi/pose_deploy_linevec_faster_4_stages.prototxt -O $protoFile

if not os.path.isfile(weightsFile):
  # Download the model file
  !wget http://posefs1.perception.cs.cmu.edu/OpenPose/models/pose/mpi/pose_iter_16
0000.caffemodel -O $weightsFile
```

In [3]:
```
nPoints = 15
POSE_PAIRS = [[0,1], [1,2], [2,3], [3,4], [1,5], [5,6], [6,7], [1,14], [14,8], [8,
9], [9,10], [14,11], [11,12], [12,13] ]
net = cv2.dnn.readNetFromCaffe(protoFile, weightsFile)
```

Read Image

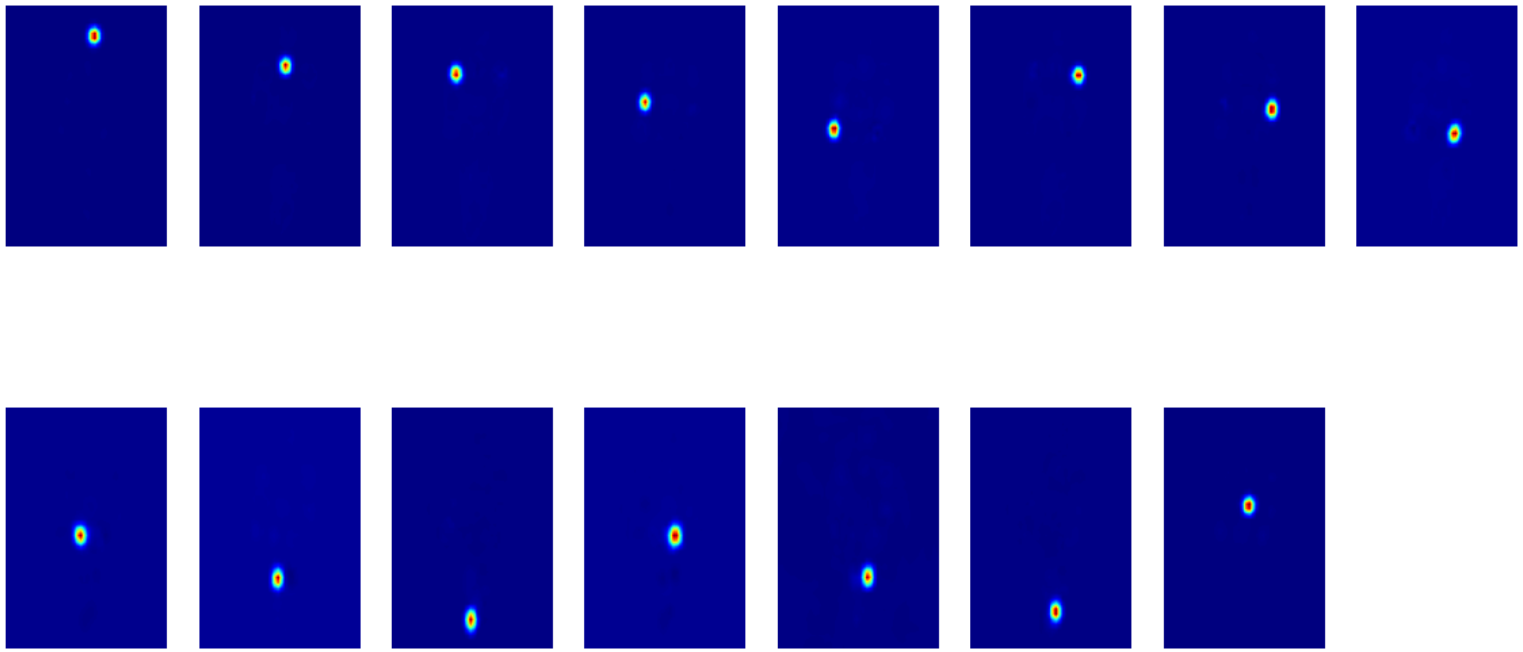
In [4]:
```
im = cv2.imread("man.jpg")
im = cv2.cvtColor(im, cv2.COLOR_BGR2RGB)
inWidth = im.shape[1]
inHeight = im.shape[0]
```

Convert image to blob

In [5]:
```
netInputSize = (368, 368)
inpBlob = cv2.dnn.blobFromImage(im, 1.0 / 255, netInputSize, (0, 0, 0), swapRB=True, crop=False)
net.setInput(inpBlob)
```

Run Inference (forward pass)

In [6]:
```
# Forward Pass
output = net.forward()

# Display probability maps
plt.figure(figsize=(20,10))
for i in range(nPoints):
    probMap = output[0, i, :, :]
    displayMap = cv2.resize(probMap, (inWidth, inHeight), cv2.INTER_LINEAR)
    plt.subplot(2, 8, i+1); plt.axis('off'); plt.imshow(displayMap, cmap='jet')
```



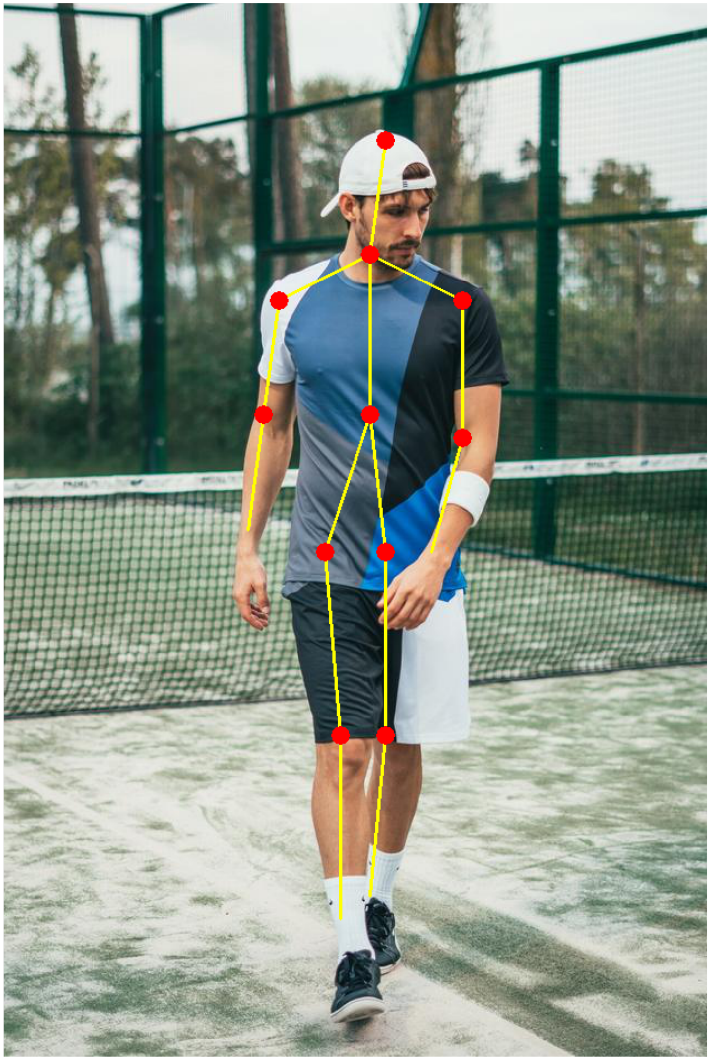
Extract points

In [7]:
```
# X and Y Scale
scaleX = inWidth / output.shape[3]
scaleY = inHeight / output.shape[2]

# Empty list to store the detected keypoints
points = []

# Treshold
threshold = 0.1

for i in range(nPoints):
    # Obtain probability map
    probMap = output[0, i, :, :]

    # Find global maxima of the probMap.
    minVal, prob, minLoc, point = cv2.minMaxLoc(probMap)

    # Scale the point to fit on the original image
    x = scaleX * point[0]
    y = scaleY * point[1]

    if prob > threshold :
        # Add the point to the list if the probability is greater than the thresho
ld
        points.append((int(x), int(y)))
    else :
        points.append(None)
```

Display Points & Skeleton

In [8]:
```
imPoints = im.copy()
imSkeleton = im.copy()
# Draw points
for i, p in enumerate(points):
    cv2.circle(imPoints, p, 8, (255, 255,0), thickness=-1, lineType=cv2.FILLED)
    cv2.putText(imPoints, "{}".format(i), p, cv2.FONT_HERSHEY_SIMPLEX, 1, (255,0,0
), 2, lineType=cv2.LINE_AA)

# Draw skeleton
for pair in POSE_PAIRS:
    partA = pair[0]
    partB = pair[1]

    if points[partA] and points[partB]:
        cv2.line(imSkeleton, points[partA], points[partB], (255, 255,0), 2)
        cv2.circle(imSkeleton, points[partA], 8, (255, 0, 0), thickness=-1, lineTy
pe=cv2.FILLED)
```

In [9]:
```
plt.figure(figsize=(50,50))
plt.subplot(121); plt.axis('off'); plt.imshow(imPoints);
plt.subplot(122); plt.axis('off'); plt.imshow(imSkeleton);
```



Deep Learning based Object Detection

Satya Mallick, LearnOpenCV.com

1. **Architecture** : Mobilenet based Single Shot Multi-Box (SSD)
2. **Framework** : Tensorflow

Download Model files from Tensorflow model ZOO

Model files can be downloaded from the Tensorflow Object Detection Model Zoo
https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detect
(https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detec

The cell given below downloads a mobilenet model

Download mobilenet model file

The code below will run on Linux / MacOS systems. Please download the file
http://download.tensorflow.org/models/object_detection/ssd_mobilenet_v2_coco_2018_03_2
(http://download.tensorflow.org/models/object_detection/ssd_mobilenet_v2_coco_2018_03_

Uncompress it and put it in models folder.

ssd_mobilenet_v2_coco_2018_03_29
|— checkpoint
|— **frozen_inference_graph.pb**
|— model.ckpt.data-00000-of-00001
|— model.ckpt.index
|— model.ckpt.meta
|— pipeline.config

Create config file from frozen graph

1. Extract the files
2. Run the tf_text_graph_ssd.py (https://github.com/opencv/opencv/blob/master/samples/dnn/tf_text_graph_ssd.py) file with input as the path to the frozen_graph.pb file and output as desired.

A sample config file has been included in the models folder

Read Tensorflow Model

In [3]:
```
# Read the Tensorflow network
net = cv2.dnn.readNetFromTensorflow(modelFile, configFile)
```

Check Class Labels

In [4]:
```
with open(classFile) as fp:
    labels = fp.read().split("\n")
print(labels)
```

```
['unlabeled', 'person', 'bicycle', 'car', 'motorcycle', 'airplane', 'bus', 'tr
ain', 'truck', 'boat', 'traffic light', 'fire hydrant', 'street sign', 'stop s
ign', 'parking meter', 'bench', 'bird', 'cat', 'dog', 'horse', 'sheep', 'cow',
'elephant', 'bear', 'zebra', 'giraffe', 'hat', 'backpack', 'umbrella', 'shoe',
'eye glasses', 'handbag', 'tie', 'suitcase', 'frisbee', 'skis', 'snowboard',
'sports ball', 'kite', 'baseball bat', 'baseball glove', 'skateboard', 'surfbo
ard', 'tennis racket', 'bottle', 'plate', 'wine glass', 'cup', 'fork', 'knif
e', 'spoon', 'bowl', 'banana', 'apple', 'sandwich', 'orange', 'broccoli', 'car
rot', 'hot dog', 'pizza', 'donut', 'cake', 'chair', 'couch', 'potted plant',
'bed', 'mirror', 'dining table', 'window', 'desk', 'toilet', 'door', 'tv', 'la
ptop', 'mouse', 'remote', 'keyboard', 'cell phone', 'microwave', 'oven', 'toas
ter', 'sink', 'refrigerator', 'blender', 'book', 'clock', 'vase', 'scissors',
'teddy bear', 'hair drier', 'toothbrush', 'hair brush', '']
```

Detect Objects

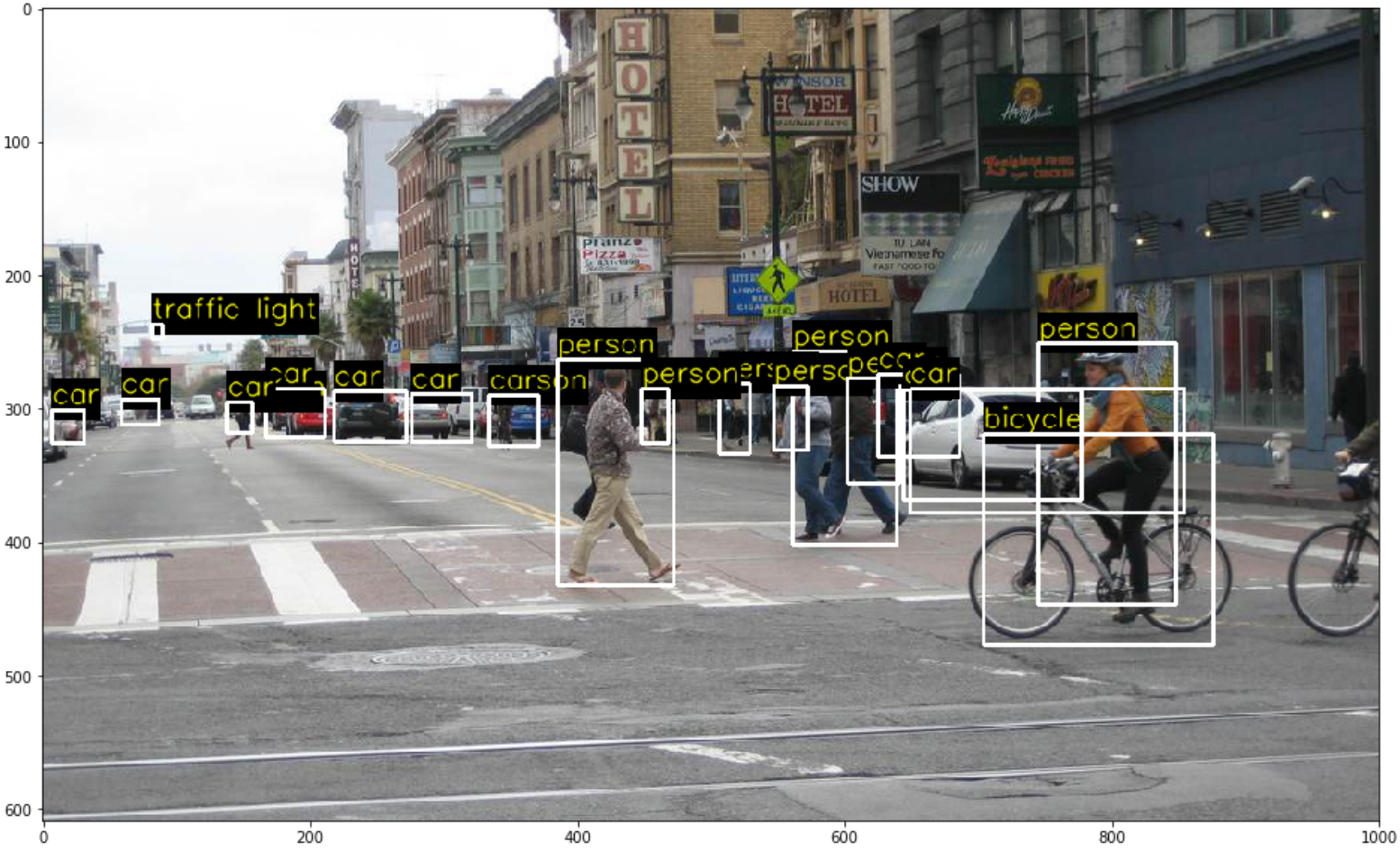
In [5]:
```
# For ach file in the directory
def detect_objects(net, im):
    dim = 300
    # Resize the image to the dimension required for SSD
    im300 = cv2.resize(im, (dim, dim))

    mean = (127.5, 127.5, 127.5)

    # Create a blob from the image
    blob = cv2.dnn.blobFromImage(im300, 1.0/127.5, (dim, dim), mean, True)


    # Pass blob to the network
    net.setInput(blob)

    # Peform Prediction
    objects = net.forward()
    return objects
```

Display Objects

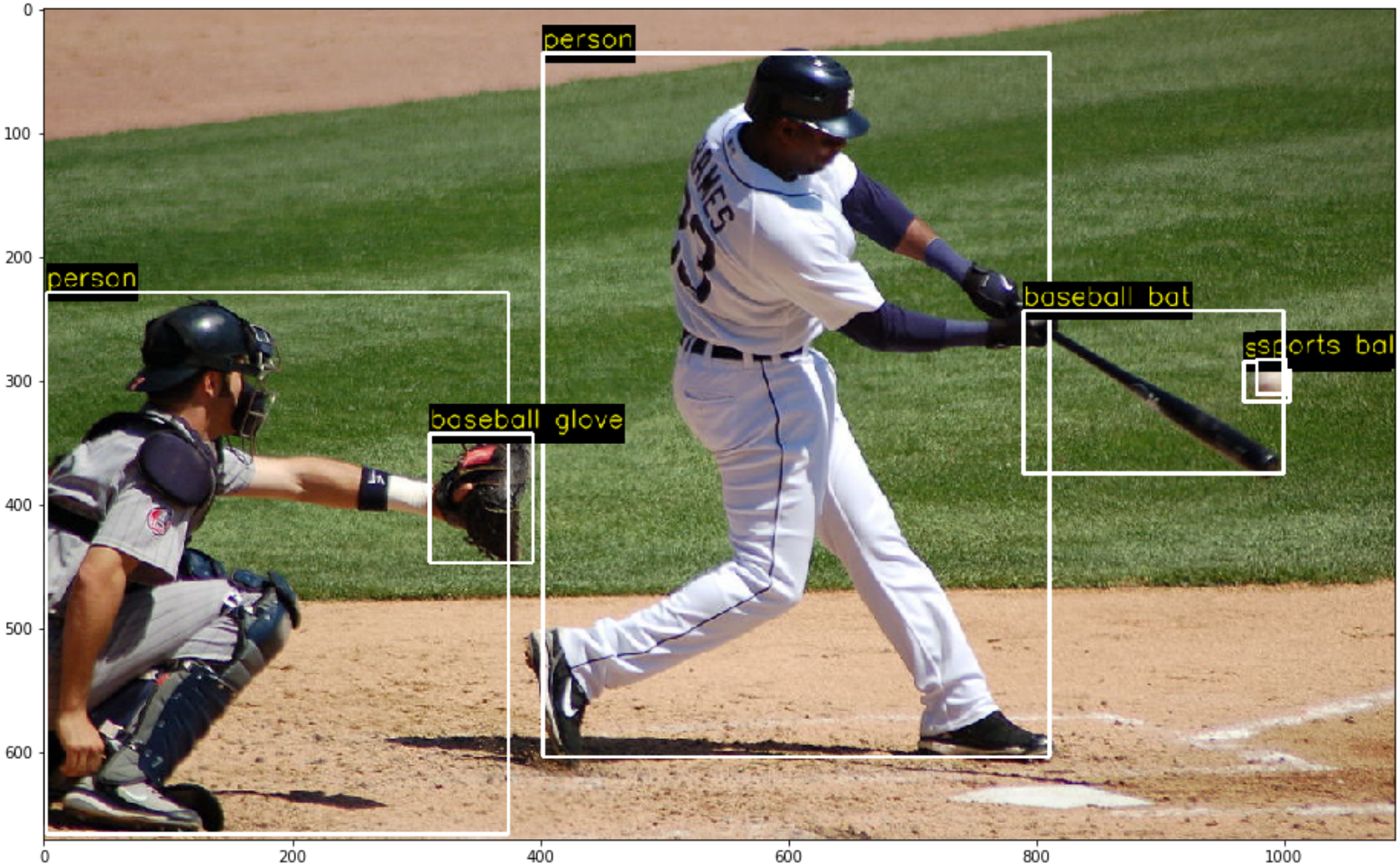
In [7]:
```
FONTFACE = cv2.FONT_HERSHEY_SIMPLEX
FONT_SCALE = 0.7
THICKNESS = 1

def display_objects(im, objects, threshold = 0.25):

    rows = im.shape[0]; cols = im.shape[1]

    # For every Detected Object
    for i in range(objects.shape[2]):
        # Find the class and confidence
        classId = int(objects[0, 0, i, 1])
        score = float(objects[0, 0, i, 2])

        # Recover original cordinates from normalized coordinates
        x = int(objects[0, 0, i, 3] * cols)
        y = int(objects[0, 0, i, 4] * rows)
        w = int(objects[0, 0, i, 5] * cols - x)
        h = int(objects[0, 0, i, 6] * rows - y)

        # Check if the detection is of good quality
        if score > threshold:
            display_text(im, "{}".format(labels[classId]), x, y)
            cv2.rectangle(im, (x, y), (x + w, y + h), (255, 255, 255), 2)

    # Convert Image to RGB since we are using Matplotlib for displaying image
    mp_img = cv2.cvtColor(im, cv2.COLOR_BGR2RGB)
    plt.figure(figsize=(30,10)); plt.imshow(mp_img); plt.show();
```

Results

In [8]:
```
im = cv2.imread('images/street.jpg')
objects = detect_objects(net, im)
display_objects(im, objects)
```

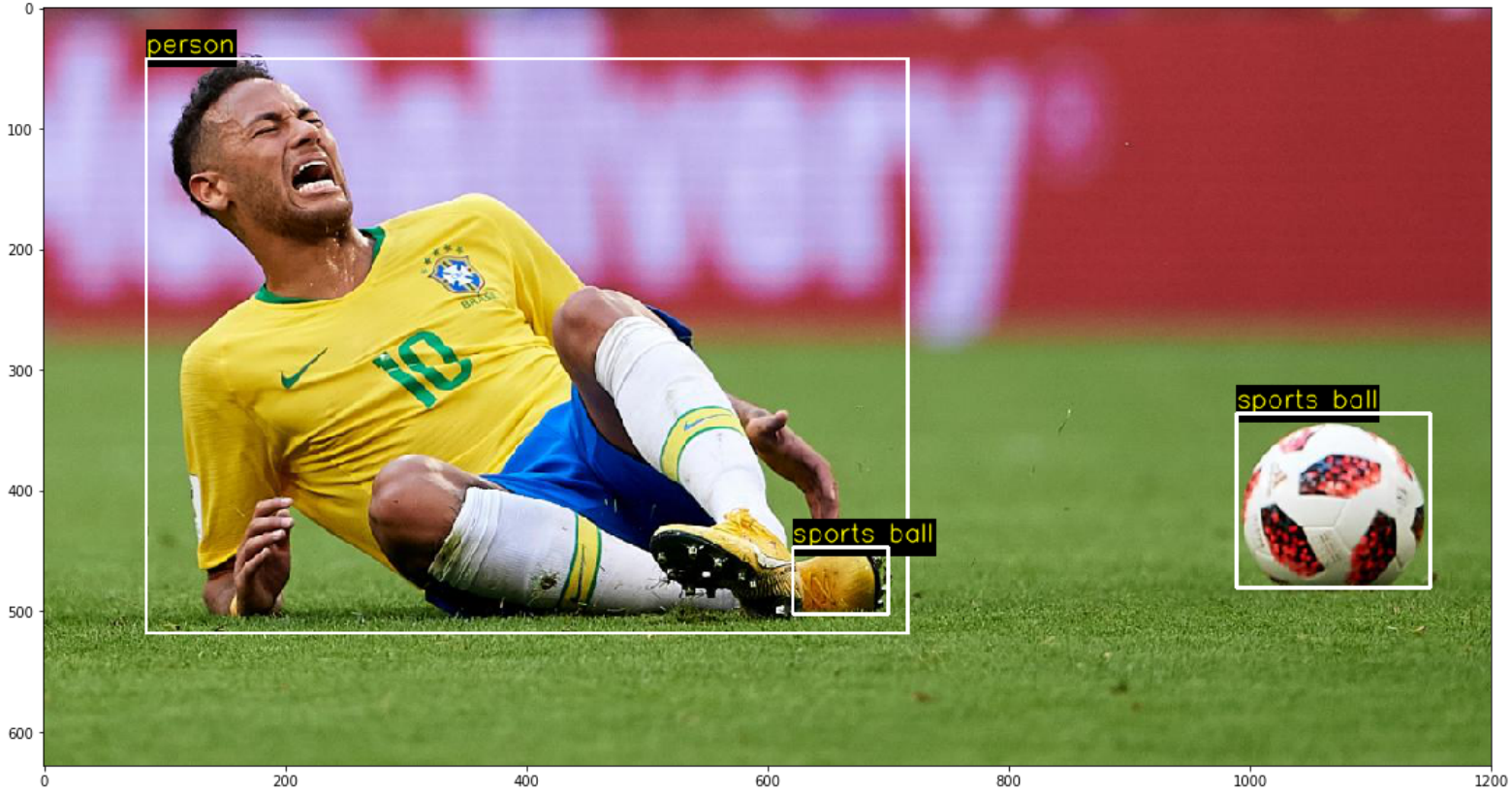


In [9]:
```
im = cv2.imread('images/baseball.jpg')
objects = detect_objects(net, im)
display_objects(im, objects, 0.2)
```



False Alarm

In [10]:
```
im = cv2.imread('images/soccer.jpg')
objects = detect_objects(net, im)
display_objects(im, objects)
```



High Dynmaic Range (HDR) Imaging

Satya Mallick, LearnOpenCV.com

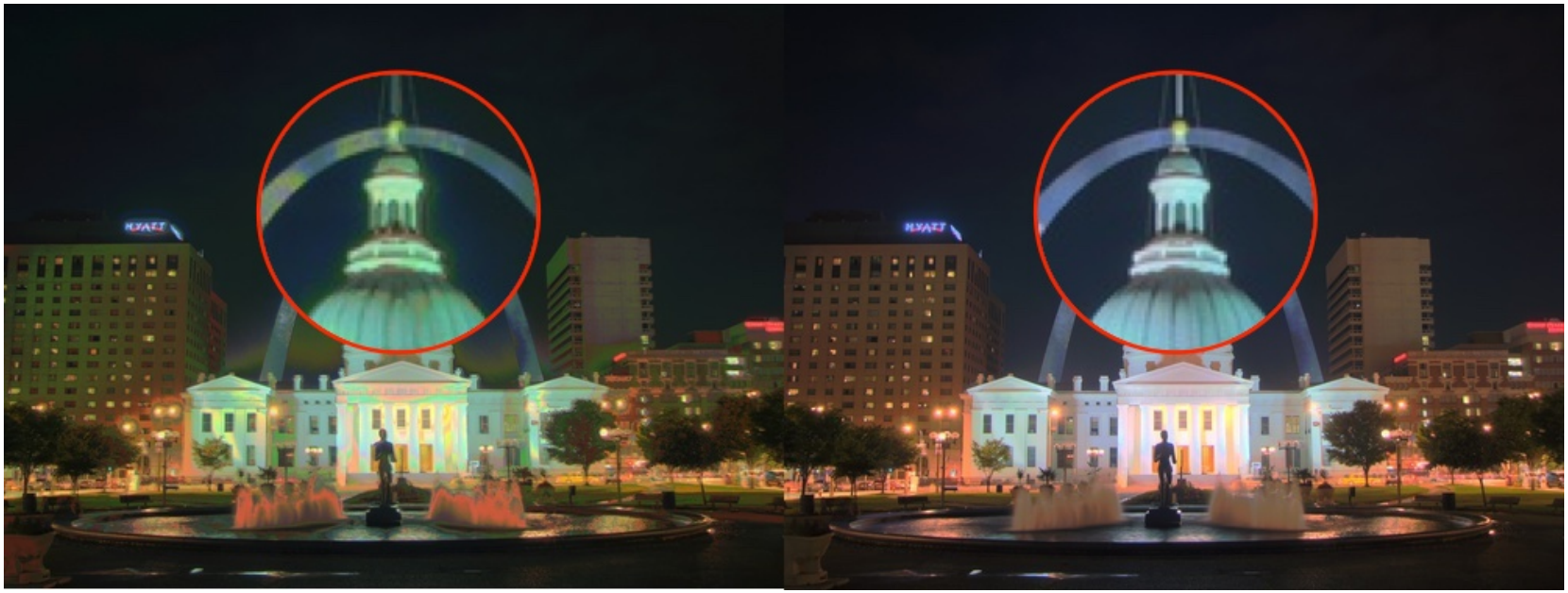
Basic Idea

1. The **dynamic range** of images is limited to 8-bits (0 - 255) per channel
2. Very bright pixels saturate to 255
3. Very dark pixels clip to 0

Step 1: Capture Multiple Exposures

In [38]:
```
import cv2
import numpy as np
import matplotlib.pyplot as plt

def readImagesAndTimes():
  # List of file names
  filenames = ["img_0.033.jpg", "img_0.25.jpg", "img_2.5.jpg", "img_15.jpg"]

  # List of exposure times
  times = np.array([ 1/30.0, 0.25, 2.5, 15.0 ], dtype=np.float32)

  # Read images
  images = []
  for filename in filenames:
    im = cv2.imread(filename)
    im = cv2.cvtColor(im, cv2.COLOR_BGR2RGB)
    images.append(im)

  return images, times
```

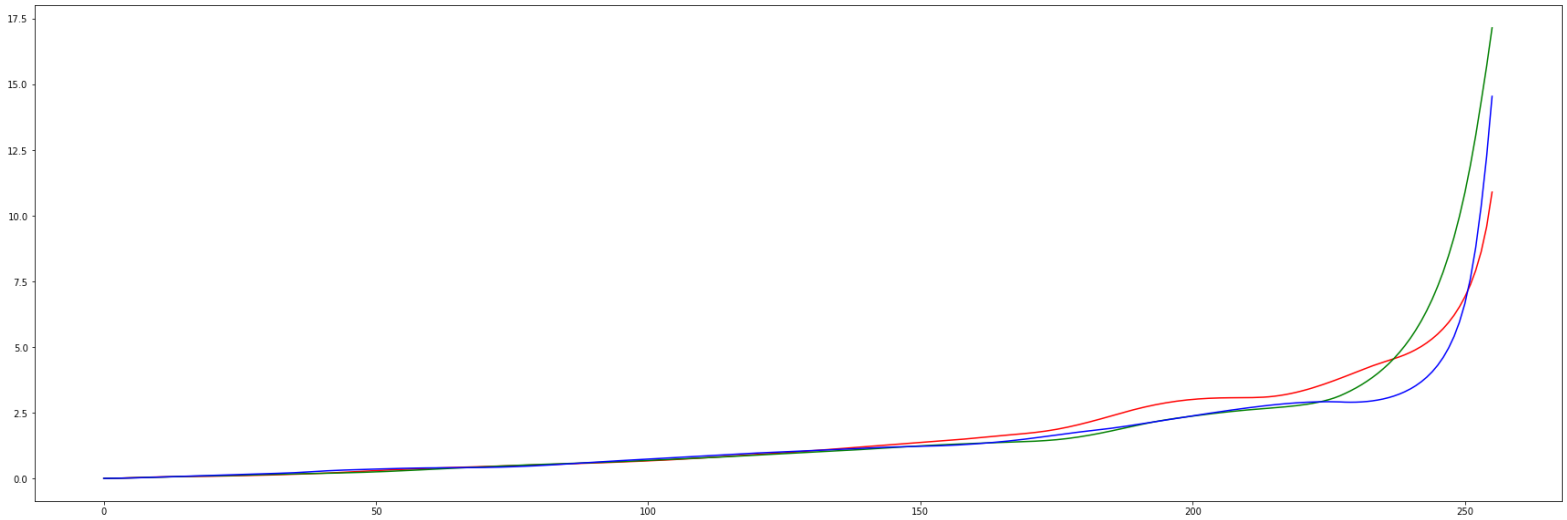
Step 2: Align Images



In [39]:
```
# Read images and exposure times
images, times = readImagesAndTimes()

# Align Images
alignMTB = cv2.createAlignMTB()
alignMTB.process(images, images)
```

Step 3: Estimate Camera Response Function

In [40]:
```
# Find Camera Response Function (CRF)
calibrateDebevec = cv2.createCalibrateDebevec()
responseDebevec = calibrateDebevec.process(images, times)

# Plot CRF
x = np.arange(256, dtype=np.uint8)
y = np.squeeze(responseDebevec)

plt.figure(figsize=(30,10))
plt.plot(x, y[:,0],'r'  , x, y[:,1],'g', x, y[:,2],'b');
```



Step 4: Merge Exposure into an HDR Image

In [41]:
```
# Merge images into an HDR linear image
mergeDebevec = cv2.createMergeDebevec()
hdrDebevec = mergeDebevec.process(images, times, responseDebevec)
```

Step 5: Tonemapping

Many Tonemapping algorithms are available in OpenCV. We chose Durand as it has more controls.

In [121]:
```
tonemapDurand = cv2.createTonemapDurand(1.2,3,1.0,1.0,1.5)
ldrDurand = tonemapDurand.process(hdrDebevec)
ldrDurand = np.clip(3 * ldrDurand,0,1)
plt.figure(figsize=(20,10)); plt.imshow(ldrDurand); plt.axis('off');
```


HYATT

Thank You!