CDTM Drones Lecture 2016

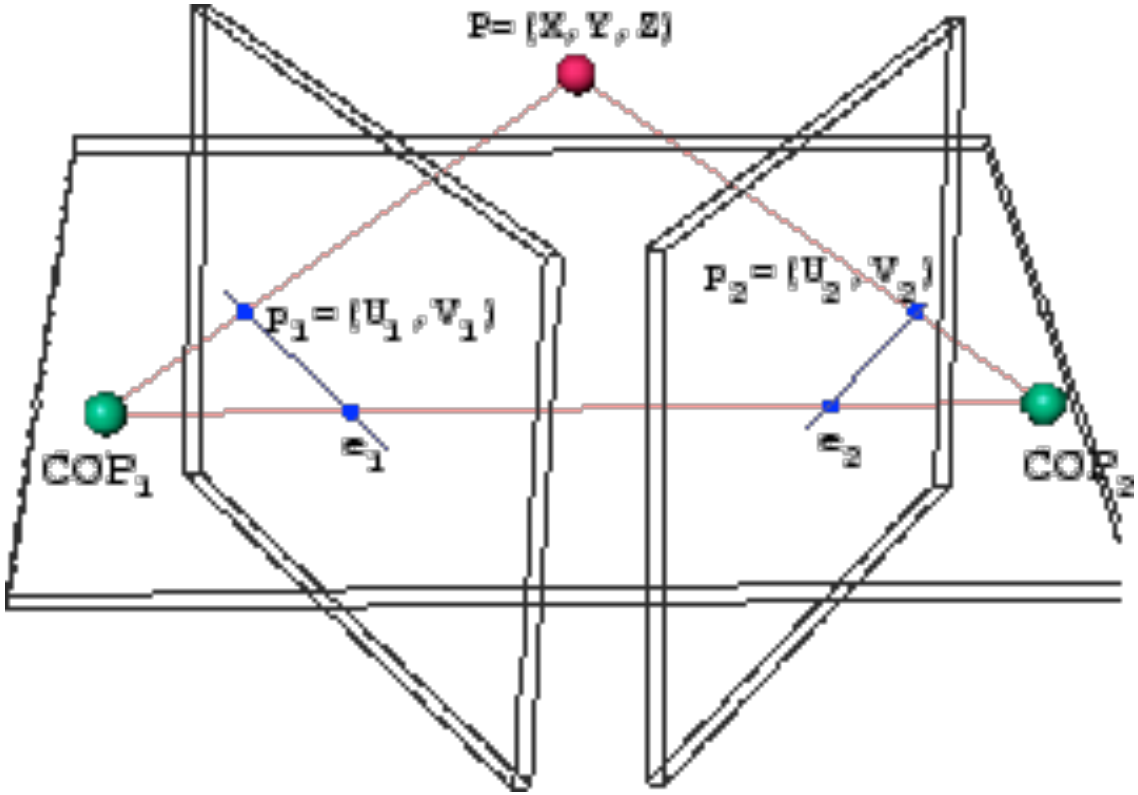
Basics of Computer Vision
Part 3
CV, Object Detection, Tracking

Seyed-Ahmad Ahmadi
Dept. Neurology, LMU Klinikum Großhadern

Outline of this part

- Brief overview/history of object detection and tracking
- Image features for object tracking
- Basic concept of machine learning for object tracking

Historic motivation: Stereoscopic 3D vision

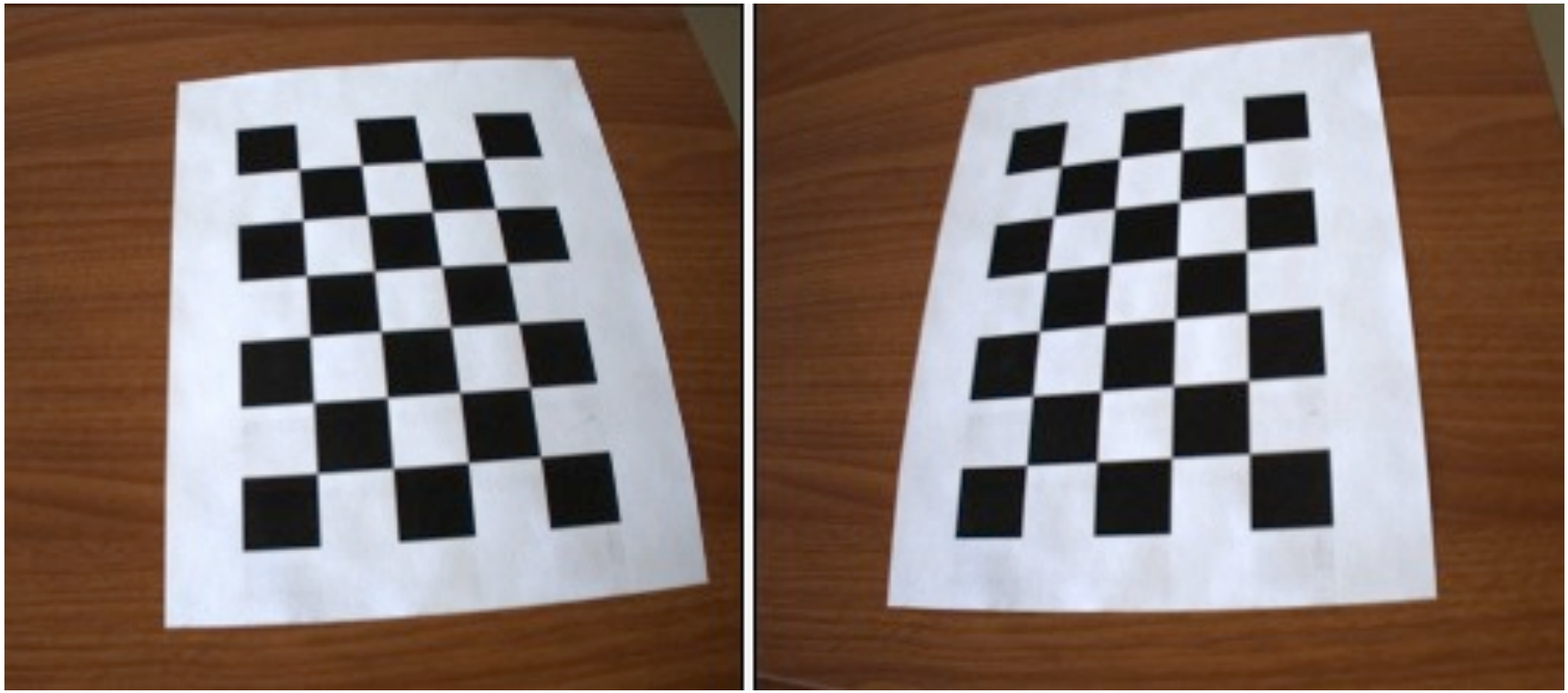


Left camera

Right camera

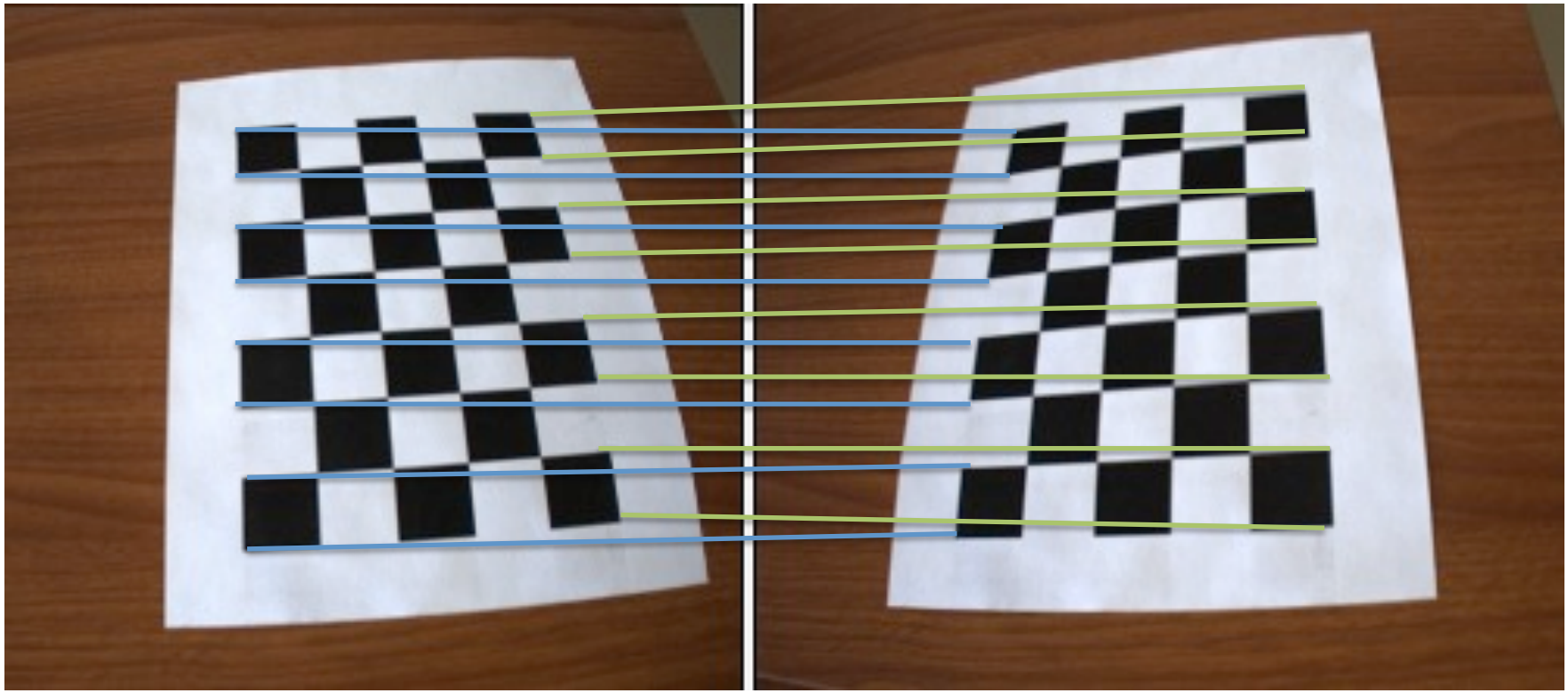
© National Instruments (NI), http://www.ni.com/white-paper/14103/de/

What the cameras see...



BoofCV real-time Java Computer Vision, http://boofcv.org/index.php?title=Example_Calibrate_Planar_Stereo

Establishing correspondence of key feature points (e.g. corners)

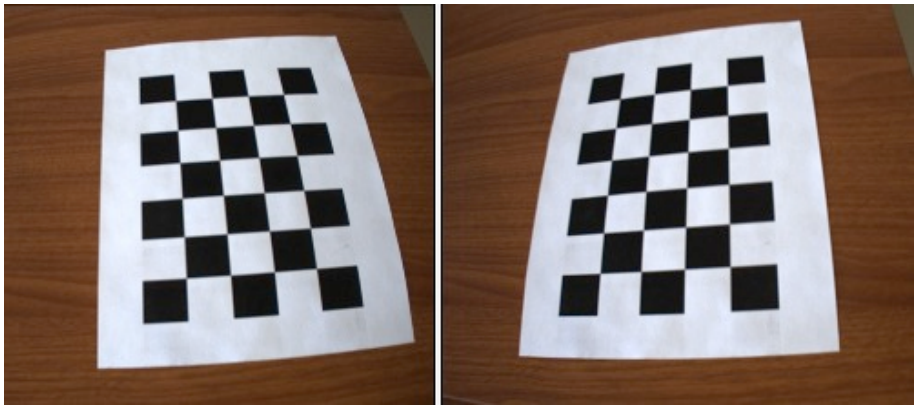


BoofCV real-time Java Computer Vision, http://boofcv.org/index.php?title=Example_Calibrate_Planar_Stereo

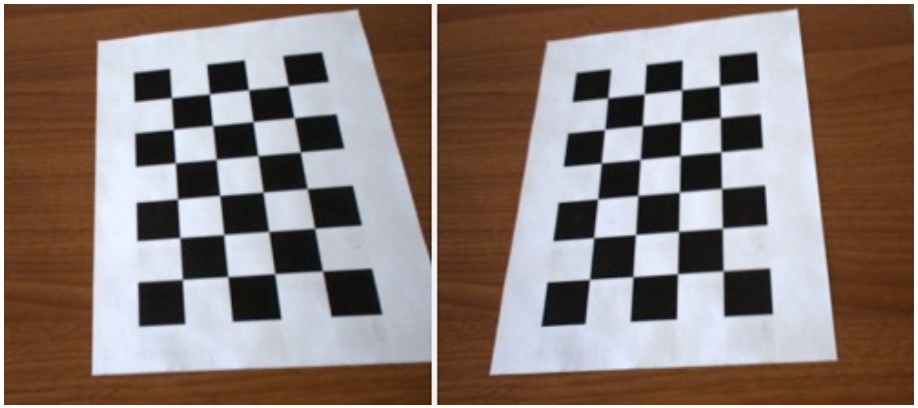
Using corresponding points, we can solve linear equations of camera parameters

- Intrinsic: focal length, image sensor format, and principal point
- Extrinsic: 3D rotation/translation from camera to world coordinates (position and heading)

Before calibration



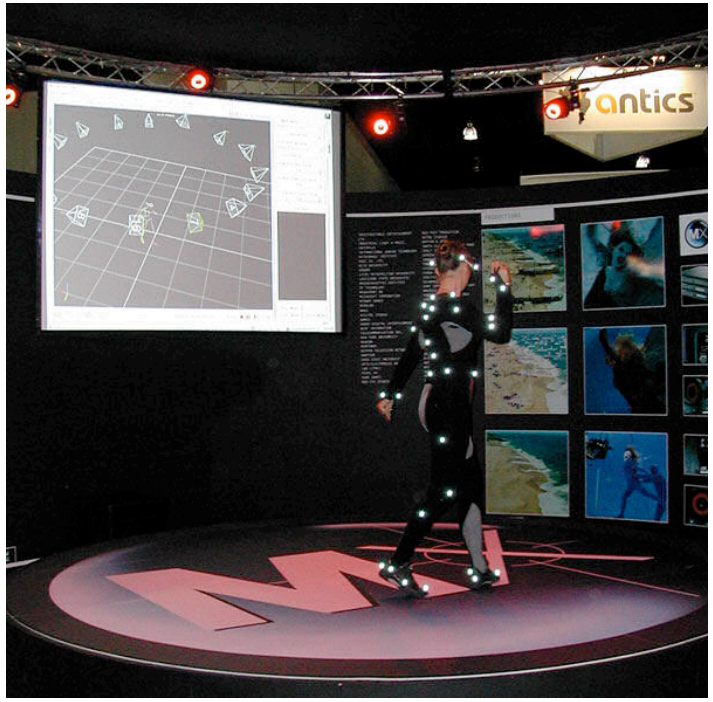
After calibration



BoofCV real-time Java Computer Vision, http://boofcv.org/index.php?title=Example_Calibrate_Planar_Stereo

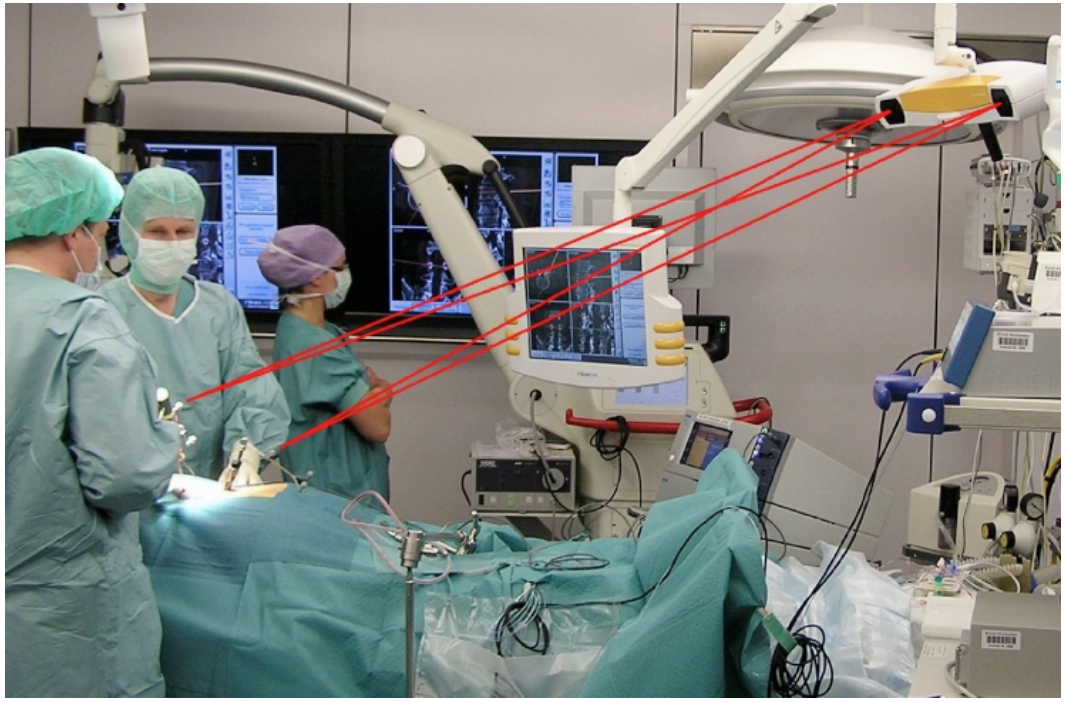
Applications of stereoscopic vision

Motion capturing
(Stereo- or multi-camera)



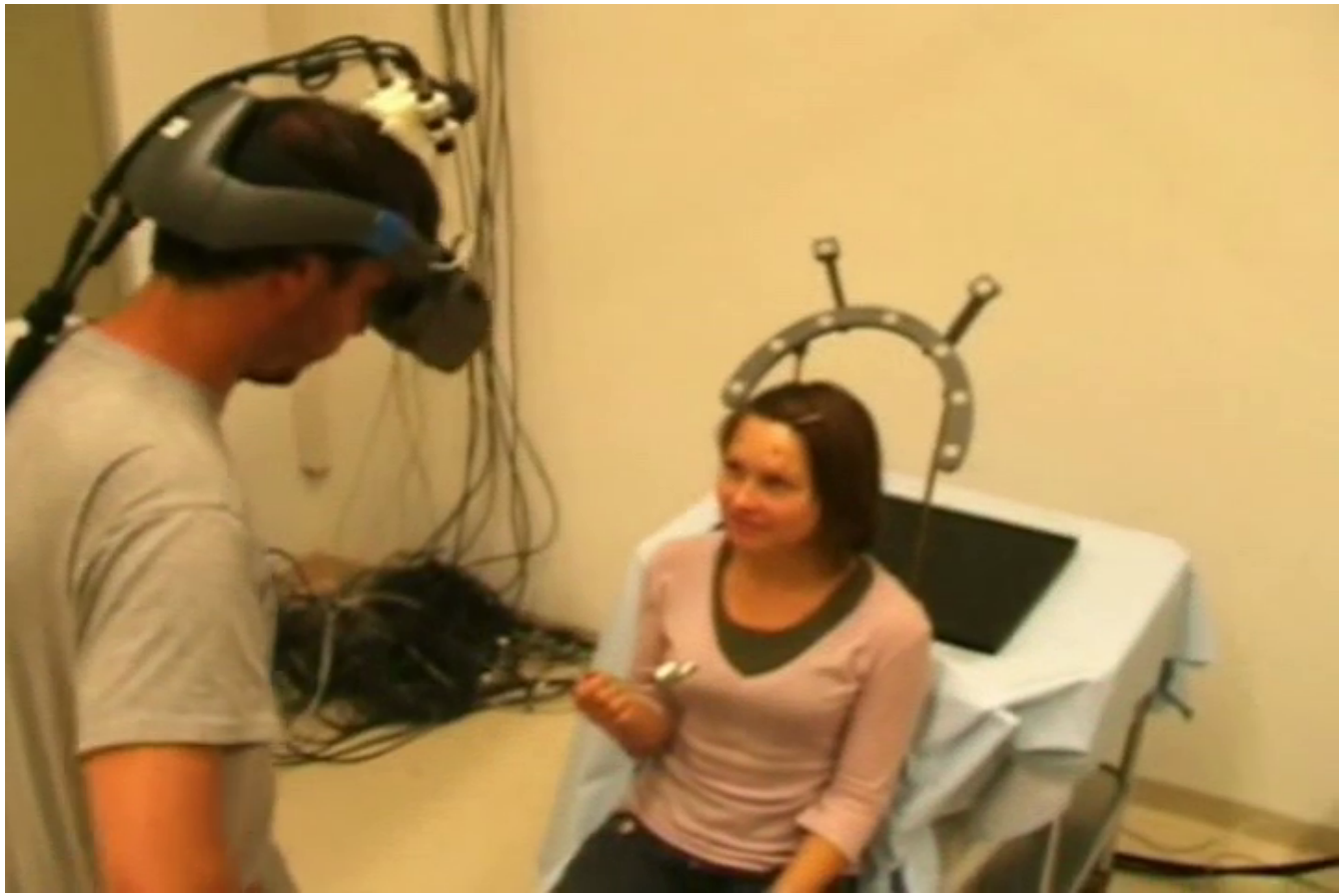
"MotionCapture", upload by T-tus
https://en.wikipedia.org/wiki/Motion_capture

Navigated surgery



https://www.scienceopen.com/document/vid/03b46b7f-ebba-4229-b17d-b300d6ce5e7e?2

Calibrated cameras allow for AR projections

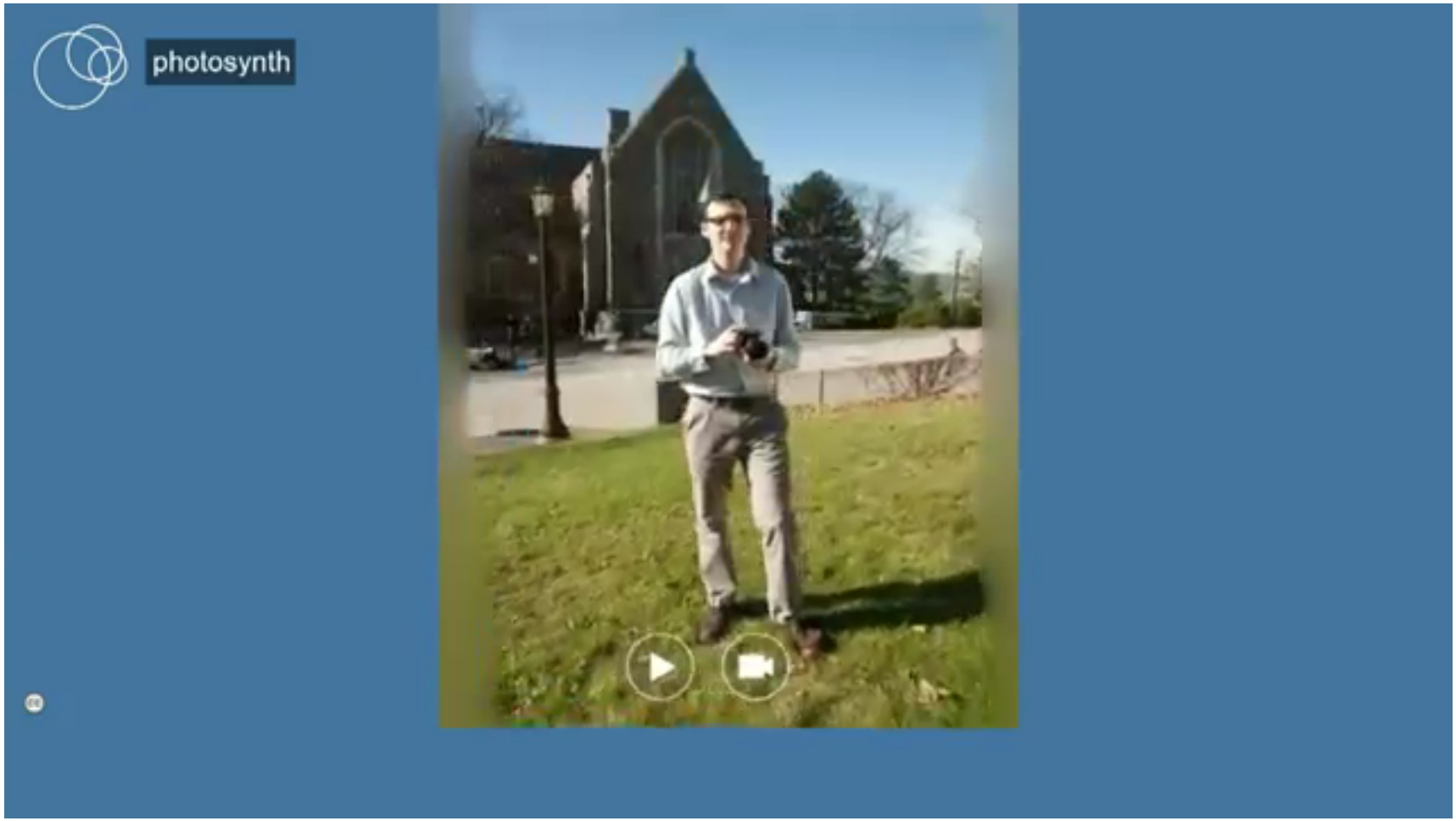


C. Bichlmeier, F. Wimmer, S.M. Heining, N. Navab, Contextual Anatomic Mimesis: Hybrid In-Situ Visualization Method for Improving Multi-Sensory Depth Perception in Medical Augmented Reality, Int. Conf. ISMAR, 2007.

More than two cameras...

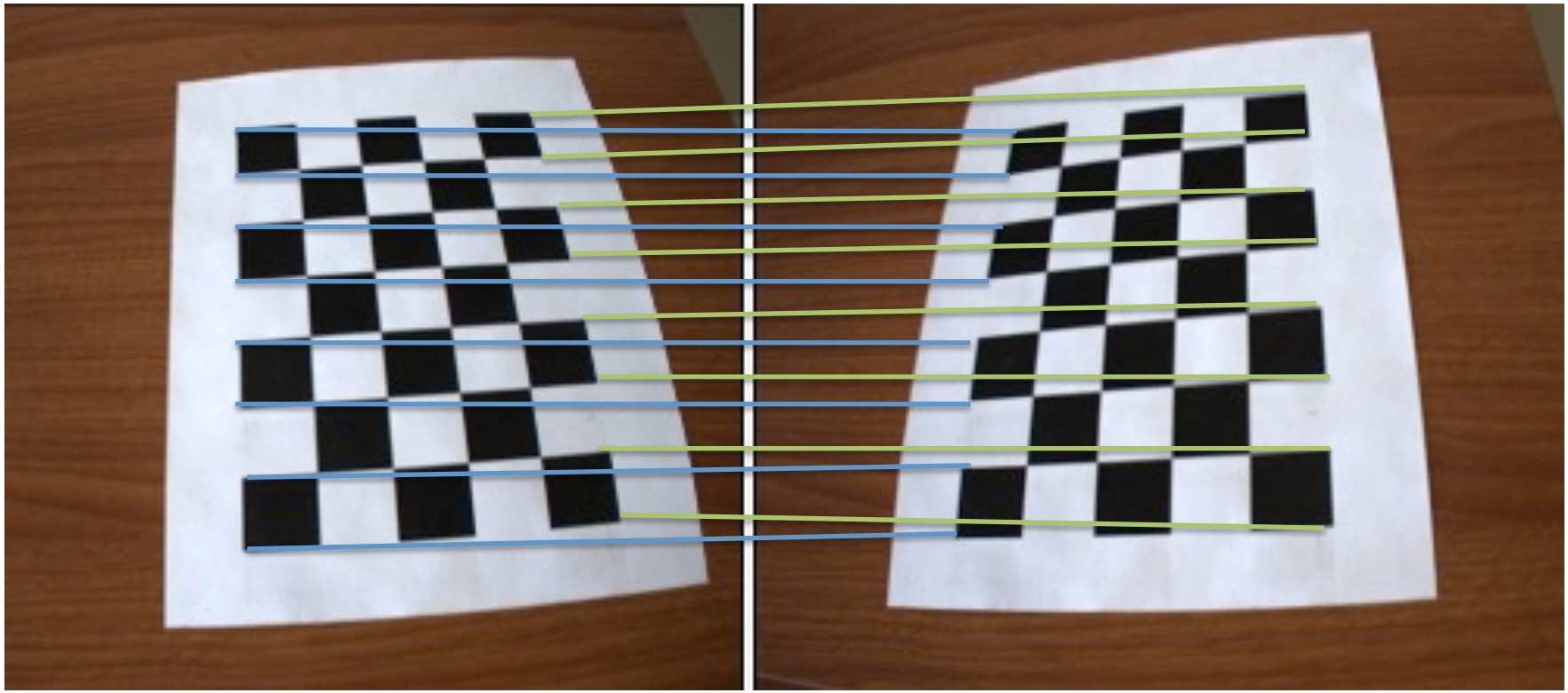
© Warner Bros., images from:
http://sabia.tic.udc.es/gc/Contenidos%20adicionales/trabajos/Peliculas/FX/ej3.html

Modern multi-view setups



©Microsoft Research, Photosynth Software, presented at TED, Dec. 2013
https://www.youtube.com/watch?v=4LxlhoemR3A

Establishing correspondence of key feature points (e.g. corners)



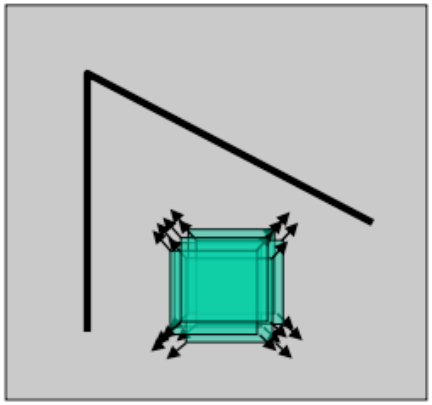
BoofCV real-time Java Computer Vision, http://boofcv.org/index.php?title=Example_Calibrate_Planar_Stereo

Features for tracking

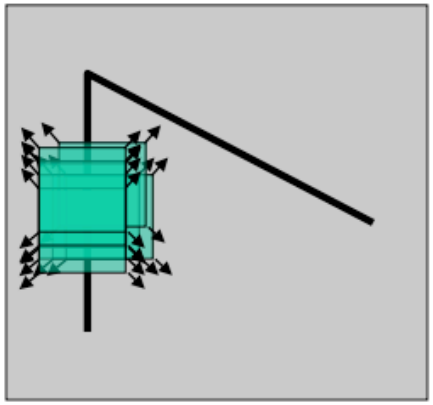
- Edges / Corners / Blobs...
  - Canny edge detector (1986, >22000 citations!)
  - Harris corner detector (1988, >11000 citations!)
  - FAST corner detector (2006, ~1800 citations)
- Shi-Tomasi
  - „Good features to track“ (1994, >6000 citations!)
- Feature detector/descriptors
  - Scale-Invariant-Feature-Transform (SIFT) (1999, ~31000 citations)
  - Speeded Up Robust Features (SURF) (2006, >6000 citations)

Harris Corner Detector

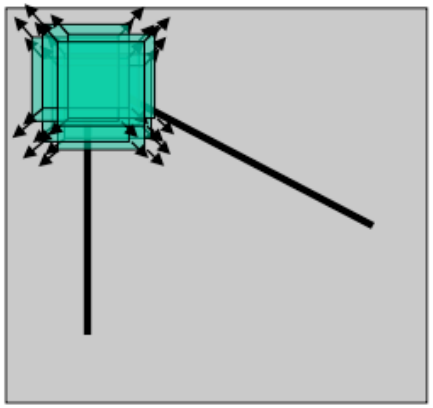
- Basic principle: corners are points of maximal change from one pixel to another



"flat" region:
no change in
all directions



"edge":
no change along
the edge direction



"corner":
significant change
in all directions

http://dsp.stackexchange.com/questions/14338/corner-detection-using-chris-harris-mike-stephens

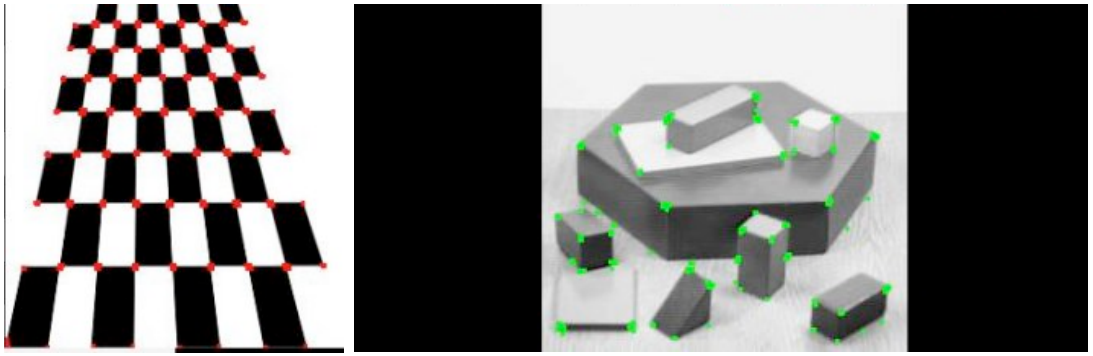
Harris Corner Detector

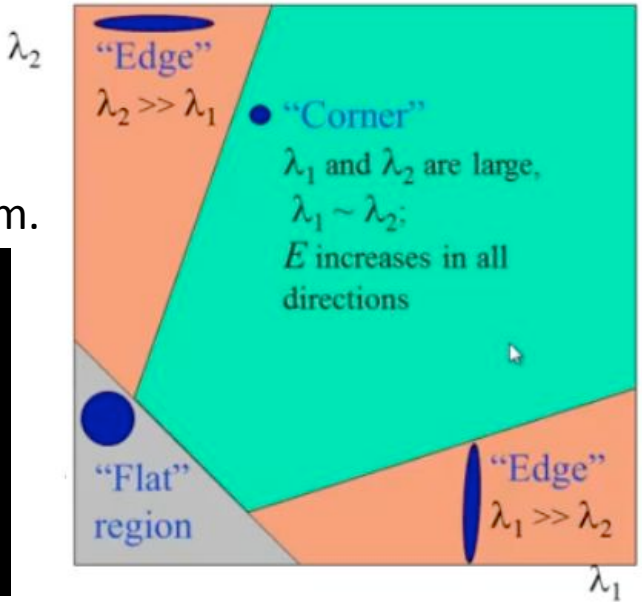
$$E(u,v) = \sum_{x,y} \underbrace{w(x,y)}_{\text{window function}} \underbrace{[I(x+u,y+v)}_{\text{shifted intensity}} - \underbrace{I(x,y)]^2}_{\text{intensity}}$$

- Maximize difference between 2 imgs:
- Ends up in an eigenvalue analysis
- In OpenCV:
- `dst = cv2.cornerHarris(gray,2,3,0.04)`

$$\underbrace{[I(x+u,y+v)}_{\text{shifted intensity}} - \underbrace{I(x,y)]^2}_{\text{intensity}}$$

Image Patchsize Sobel size Harris param.



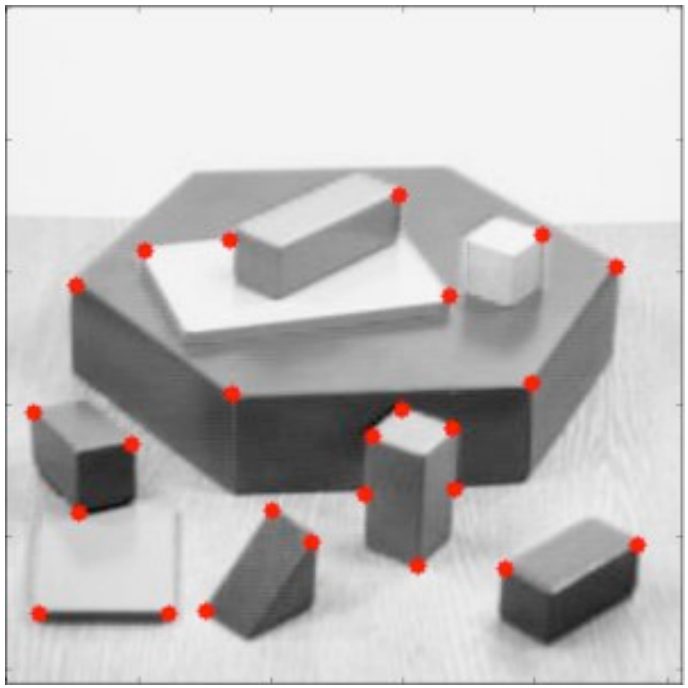


http://opencv-python-tutroals.readthedocs.org/en/latest/py_tutorials/py_feature2d/py_features_harris/py_features_harris.html

Shi Tomasi „Good Features to Track“

- In OpenCV:

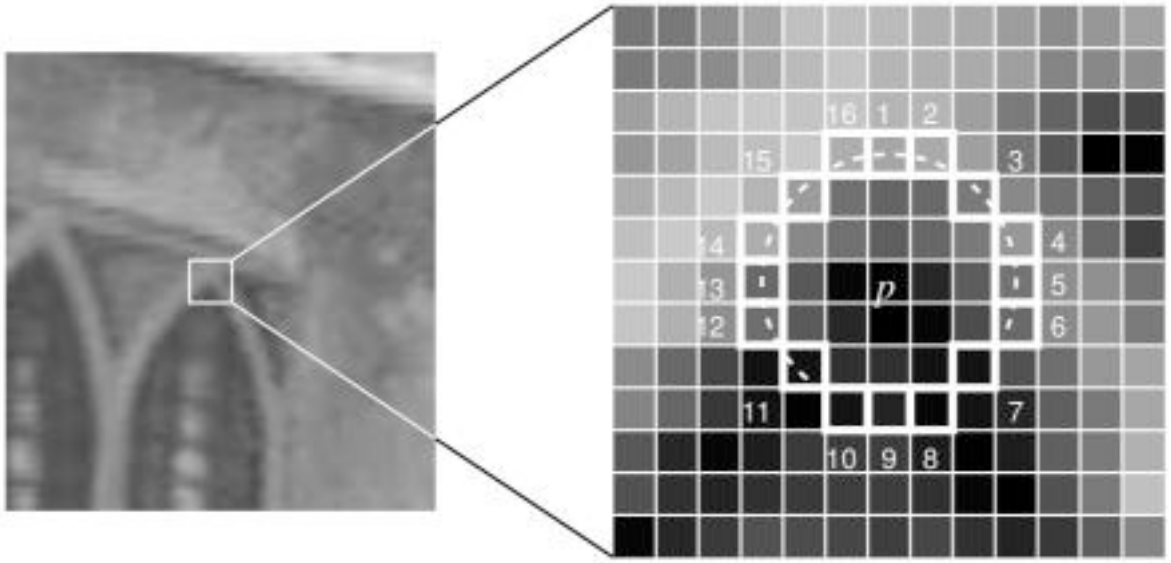
```
cv2.goodFeaturesToTrack( ...
    image, maxCorners, qualityLevel, minDistance)
```



FAST Corner Detector
(Features from Accelerated Segment Test)

```
fast = cv2.FastFeatureDetector()
kp = fast.detect(img,None)
img2 = cv2.drawKeypoints(img, kp, color=(255,0,0))
```

- Consider a circle around each pixel p
- Corner = There must be at least n brighter/darker pixels than center pixel p
- Even faster: only evaluate pixels at location 1-5-9-13



- Weak/erroneous detections are compensated by domain-specific machine learning
- Adjacent detections are compensated with non-maximum suppresion
- Several times faster than other corner detectors

http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_fast/py_fast.html

SIFT Feature Detector+Descriptor

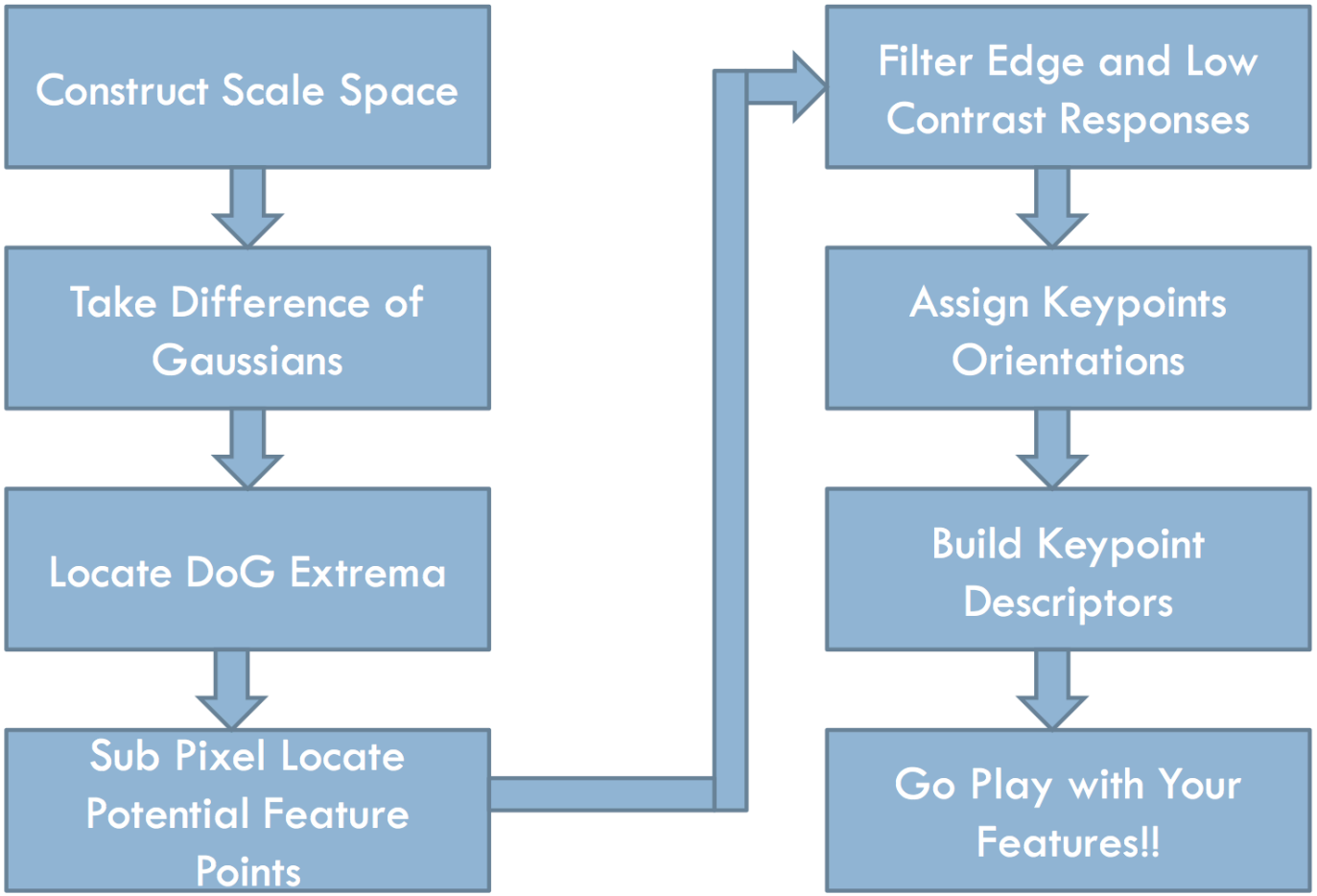
(Scale Invariant Feature Transform)

>31.000 citations!!

- Invented in 1999 by David Lowe, patented!!
- Many advantages
  - Scale invariance
  - Rotation invariance
  - Illumination invariance
  - Viewpoint invariance (to some degree)

SIFT Feature Detector+Descriptor

(Scale Invariant Feature Transform)



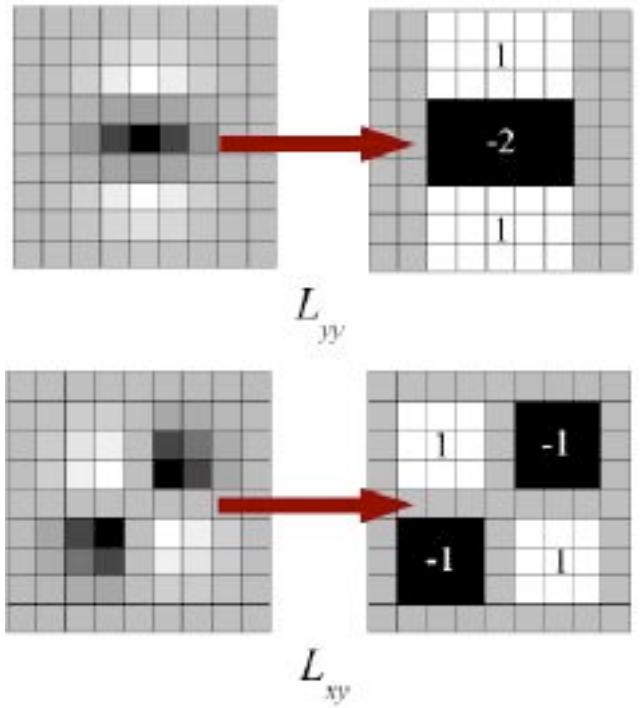
© Jason Clemons, U.Mich.
http://web.eecs.umich.edu/~silvio/teaching/EECS598/lectures/lecture10_1.pdf

SIFT in Python/OpenCV

```
# first, detect keypoints
sift = cv2.SIFT()
kp = sift.detect(gray,None)
img=cv2.drawKeypoints(gray,kp,flags= cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
cv2.imshow('features',img)

# second, compute the keypoint descriptors
kp,des = sift.compute(gray,kp)

# or, do it in one pass
kp, des = sift.detectAndCompute(gray,None)
```

SURF Feature Detector+Descriptor
(Speeded Up Robust Features)

- ~3x faster than SIFT (more suited for real-time applications)
- Similarly accurate as SIFT
- Main reason for being faster:
  - Approximates Laplacian-of-Gaussian via box filters, rather than Difference-of-Gaussians in scale-space
  - Uses local wavelet filters responses as descriptors
- SURF is good at handling images with blurring and rotation, but not good at handling viewpoint change and illumination change.



SURF in Python/OpenCV

```
# create SURF object, with Hessian threshold 400
# (the higher the threshold, the fewer keypoints are found; in practice, use 300-500)
surf = cv2.SURF()

# Find keypoints and descriptors directly
kp, des = sift.detectAndCompute(gray,None)

# Draw keypoints on image
img2 = cv2.drawKeypoints(img,kp,None,(255,0,0),4)
plt.imshow(img2),plt.show()

# Optional: compute un-oriented SURF
# Even faster than SURF, but:
# Orientation invariance only up to +/- 15°
surf.upright = True
kp = surf.detect(img,None)
```

Many more descriptors and techniques

- Histogram of Oriented Gradients (HoG)
- Local Invariant Descriptors
- BRIEF
- ORB (Oriented FAST, Rotated BRIEF)
- Bag-of-Visual-Words
- Deformable Parts Models
- Exemplar Models
- Etc. etc...
- **30-40 years of computer vision research**

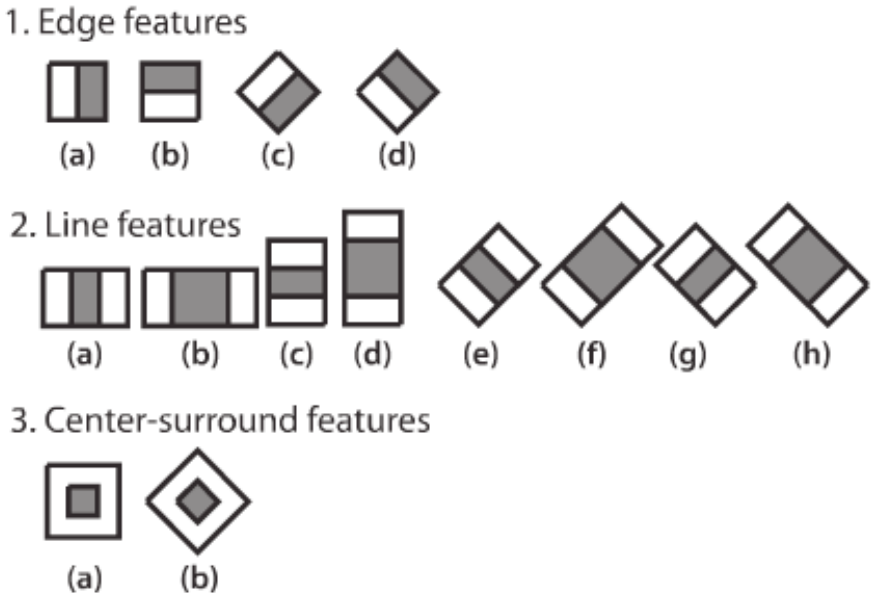
Matching of images via features

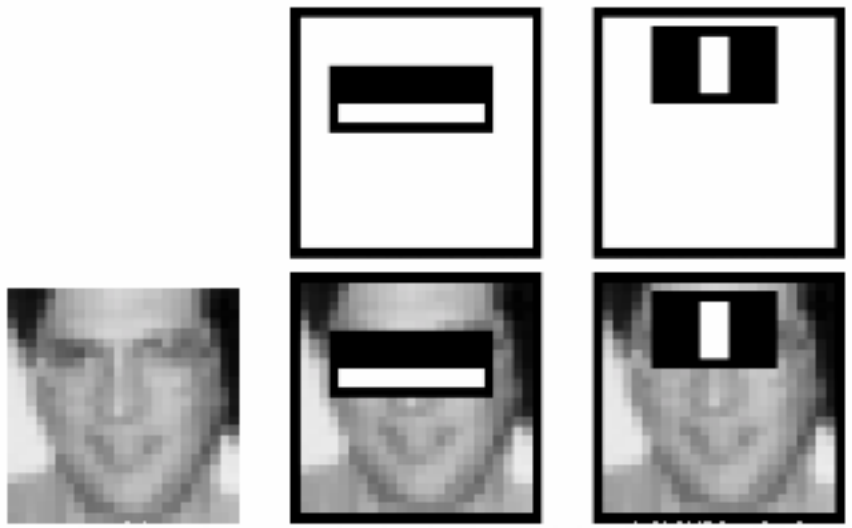
1. Find good features/keypoints in image A
2. Find good features/keypoints in image B
3. Try to match the two sets of keypoints with a transformation model
   - Robust matching despite noisy features and noisy matches
   - Specialized algorithms (e.g. Random Sample Consensus, RANSAC)

Object detection in images via machine learning

1. Image-wise object detection, no tracking
2. Extract good features at all image locations
3. Train a machine learning algorithm to find the object

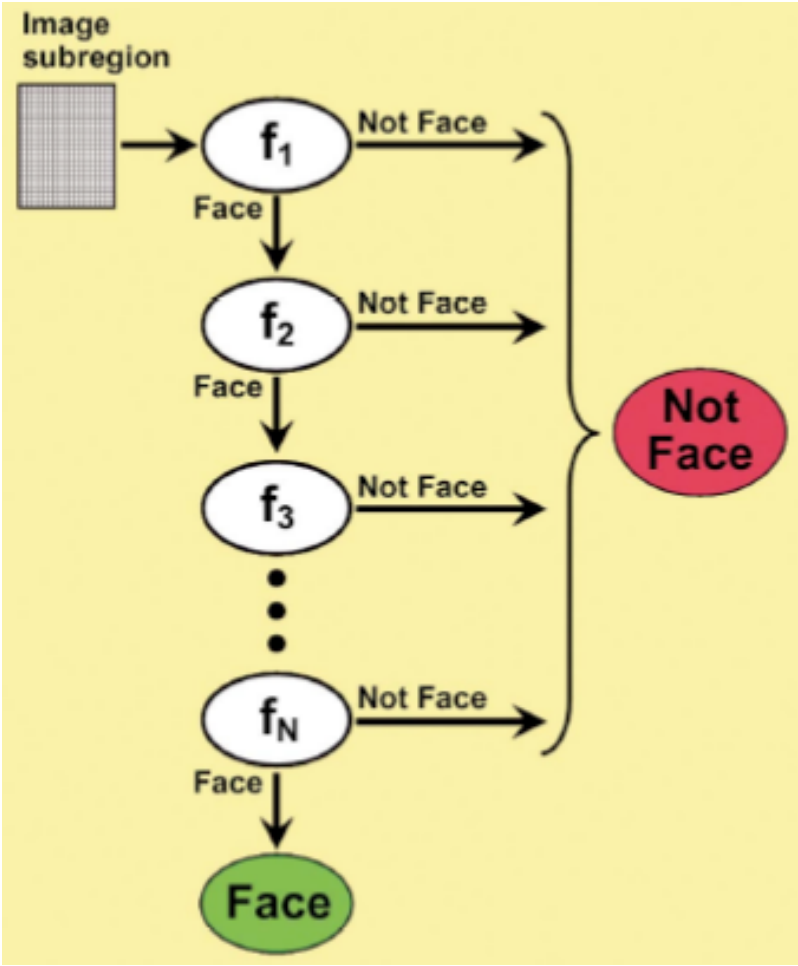
Face Detection Using Haar Cascades





- >160.000 features! How to choose best ones?
- Adaboost (Ensemble machine learning)

Cascade of Classification

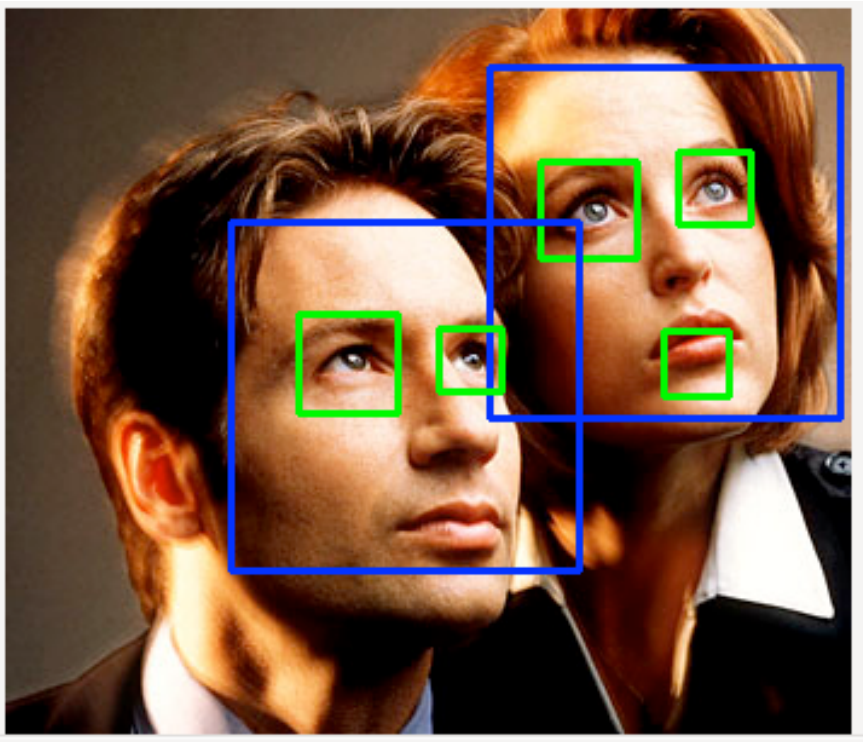


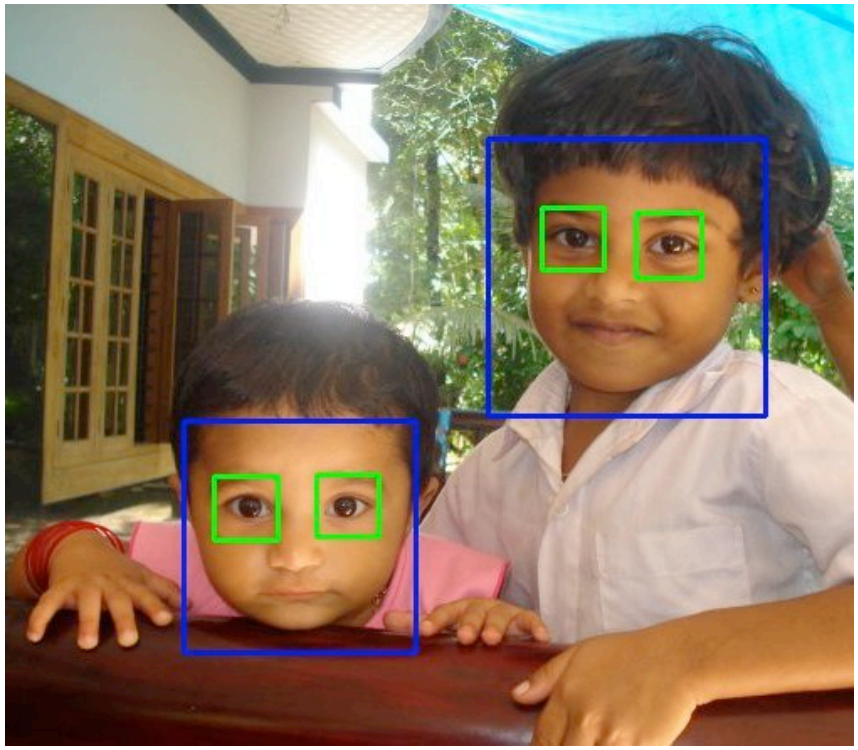

```
haarcascade_eye_tree_eyeglasses.xml
haarcascade_eye.xml
haarcascade_frontalface_alt2.xml
haarcascade_frontalface_alt_tree.xml
haarcascade_frontalface_alt.xml
haarcascade_frontalface_default.xml
haarcascade_fullbody.xml
haarcascade_lefteye_2splits.xml
haarcascade_lowerbody.xml
haarcascade_mcs_eyepair_big.xml
haarcascade_mcs_eyepair_small.xml

haarcascade_mcs_leftear.xml
haarcascade_mcs_lefteye.xml
haarcascade_mcs_mouth.xml
haarcascade_mcs_nose.xml
haarcascade_mcs_rightear.xml
haarcascade_mcs_righteye.xml
haarcascade_mcs_upperbody.xml
haarcascade_profileface.xml
haarcascade_righteye_2splits.xml
haarcascade_smile.xml
haarcascade_upperbody.xml
```

The Code

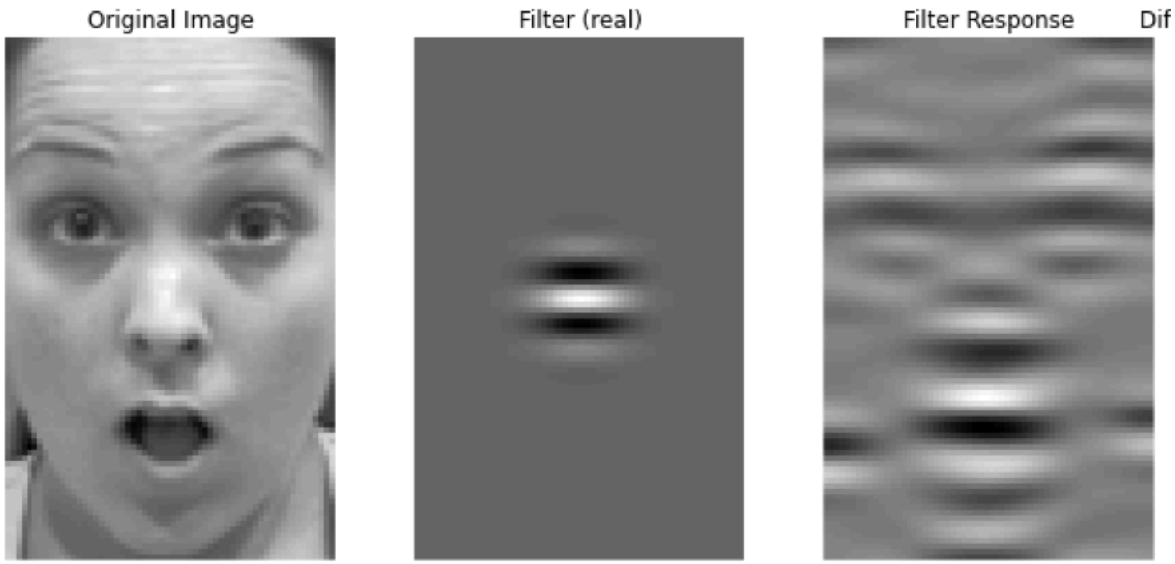
```
import numpy as np
import cv2
from matplotlib import pyplot as plt

face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')

img = cv2.imread('xfiles4.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

faces = face_cascade.detectMultiScale(gray, 1.3, 5)


for (x,y,w,h) in faces:
    cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
    roi_gray = gray[y:y+h, x:x+w]
    roi_color = img[y:y+h, x:x+w]
    eyes = eye_cascade.detectMultiScale(roi_gray)
    for (ex,ey,ew,eh) in eyes:
        cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)

cv2.imshow('img',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Results

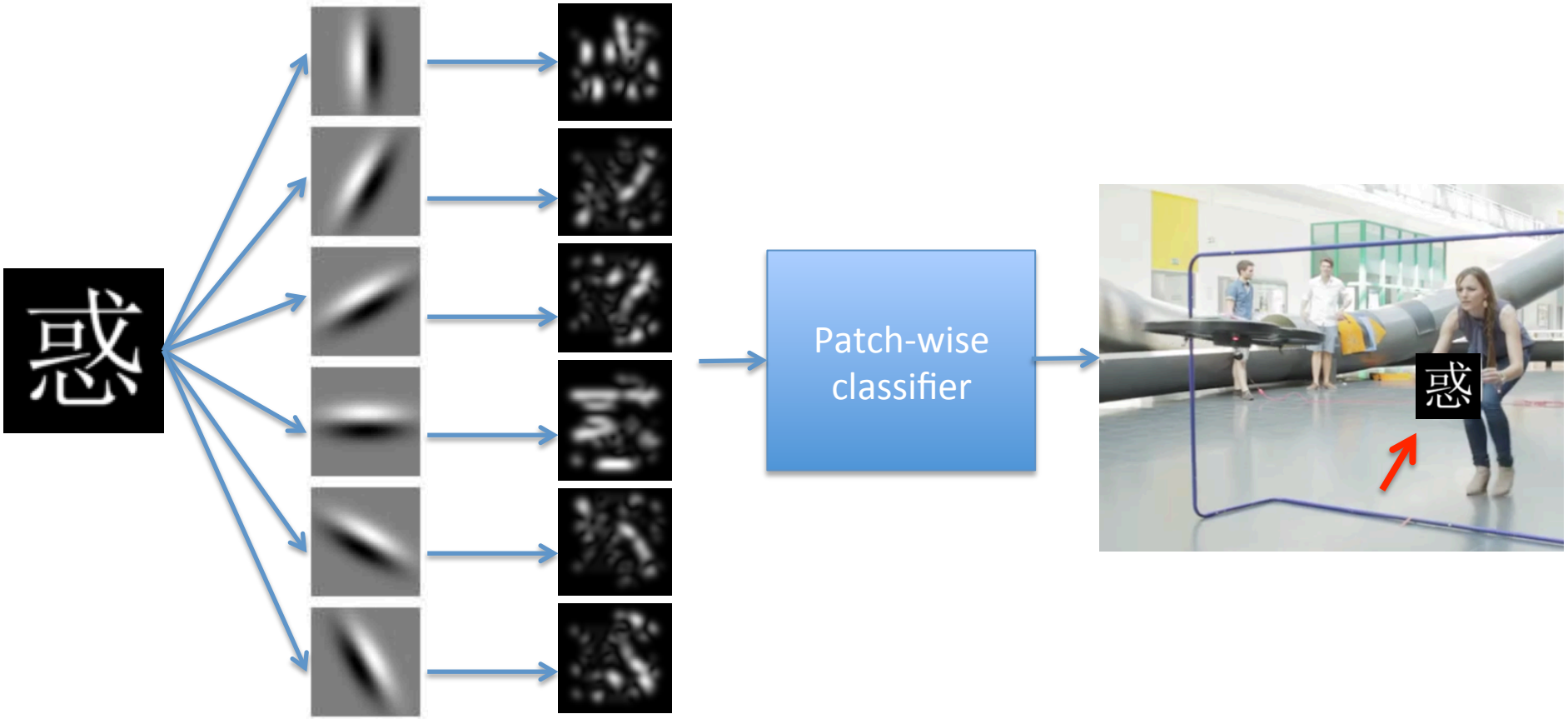




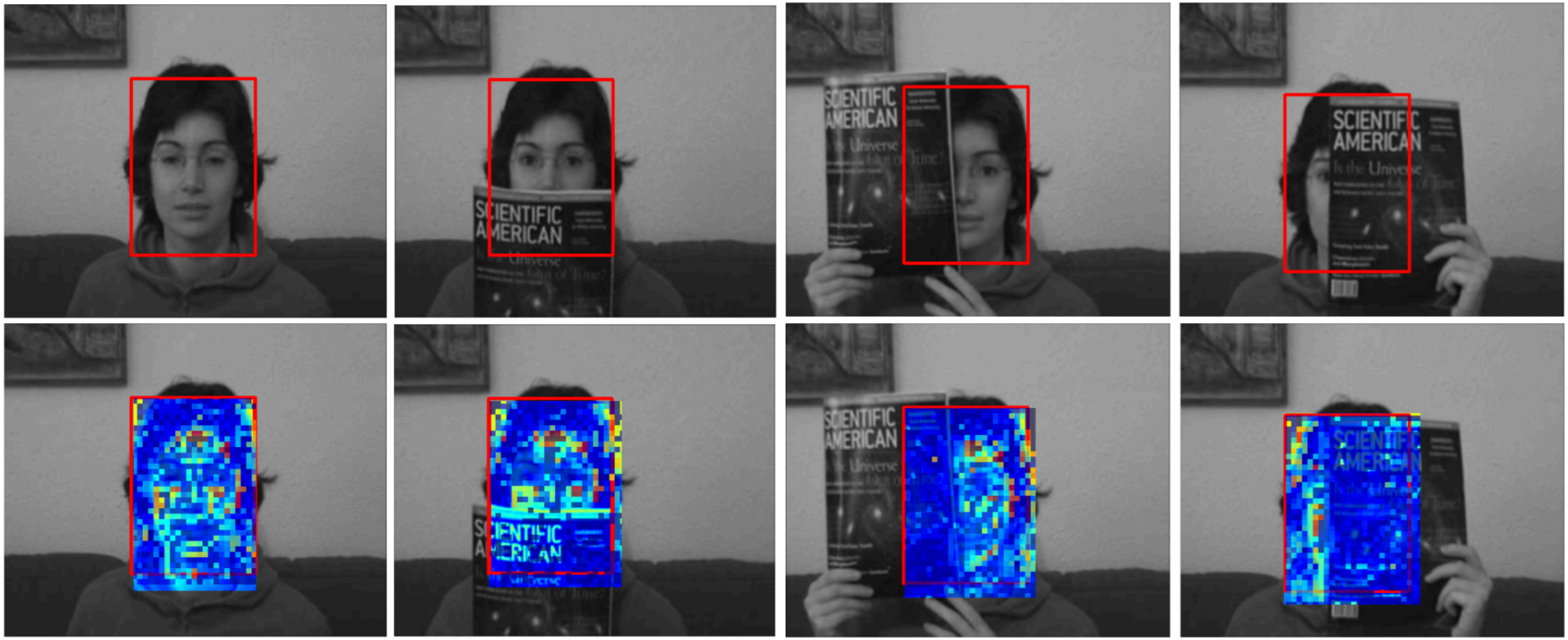
Filters can represent „good features“ in images



Gabor filter banks mimic visual cortex basic functions



Filter responses for local specificity and global robustness



Conclusion

- Computer vision was motivated by stereo 3D vision and matching of keypoints between image pairs
- Keypoints need to be identified via features
  - 30-40 years of computer vision community to find good keypoints and good matching algorithms
- Features can be extracted for each pixel via banks of filters
- Local feature responses from many filters give a good clue (e.g. to a classifier) where the object is

Questions?