CS-465 Computer Vision

Nazar Khan

PUCIT

1. Getting Started

Preliminaries

- Attendance will be marked at the beginning. If you miss it, I will NOT mark it later.
- Quizzes can be announced as well as unannounced.
- There is no such thing as a stupid question. Your questions will
  - help your class-mates,
  - make sure I do not go too fast, and
  - provide feedback for me.

Preliminaries

- We will not be following any standard textbook.
- Helpful books include
  - *Digital Image Processing* by *Gonzales and Woods*.
  - *Programming Computer Vision with Python* by *Jan Erik Solem*.
- Office Hours: MW 11:30 am till 1:00 pm or set a time via email (nazarkhan at pucit.edu.pk).
- Course webpage: `http://faculty.pucit.edu.pk/nazarkhan/teaching/Fall2018/CS465/CS465.html`

Study Tip

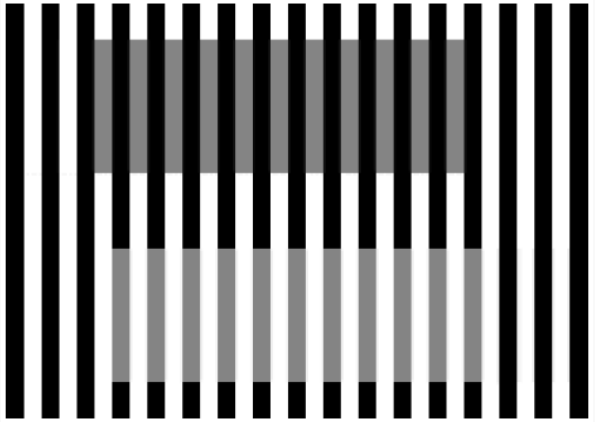
- These slides are available before class in the course folder and web page.
- Before class:
  - Print them
  - Read them
- During class:
  - Take notes on them.
- This will save you *lots of* effort after class.

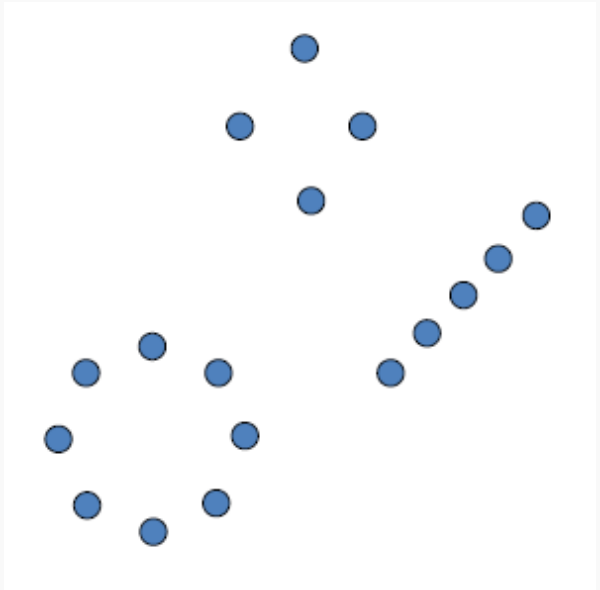
Introduction

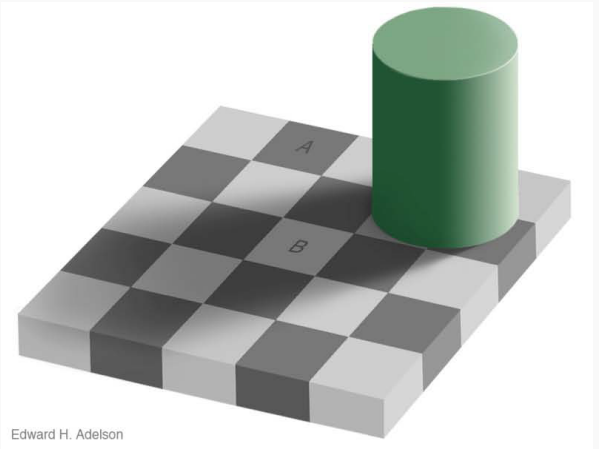
- Visual data is ubiquitous – your phone probably contains gigabytes of visual content.
- This course deals with automatically extracting meaningful information from images and videos.
- For biological brains (humans and animals), this is a trivial process.
- For computing machines, this is a *deceptively hard* process.

> What you see as a face or a car or a mountain is just an array of numbers for a computer.

CV is Deceptively Hard







CV Algorithms

- Some CV algorithms try to copy biological vision but it is too complex to understand and implement.
- Some algorithms use statistical approaches.
- Other algorithms employ geometry.

Key to understanding most algorithms lies in solid understanding of calculus and linear algebra.

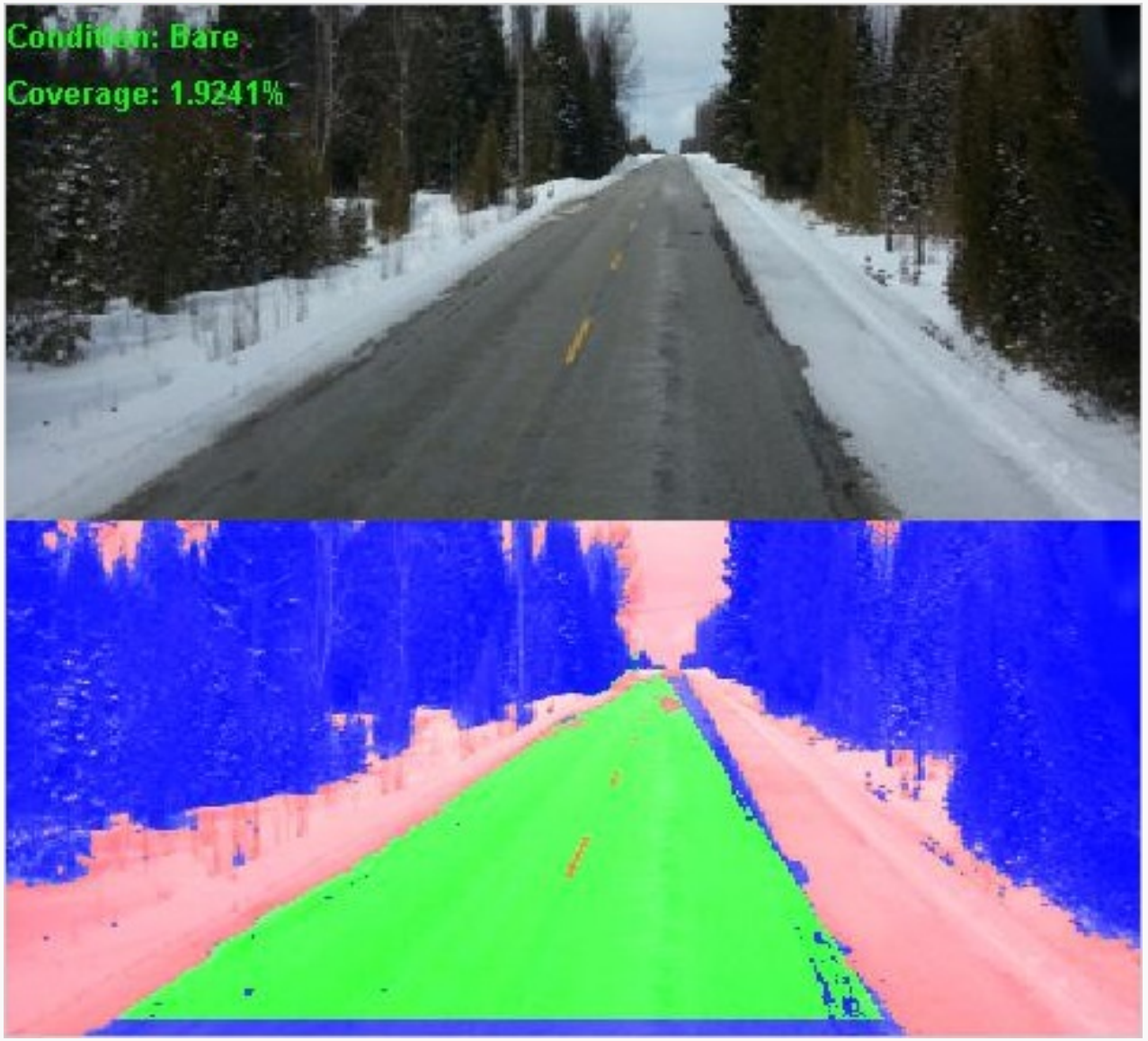
To become good in CV, you must be (or *become*) good at *mathematics* and *programming*.

Applications

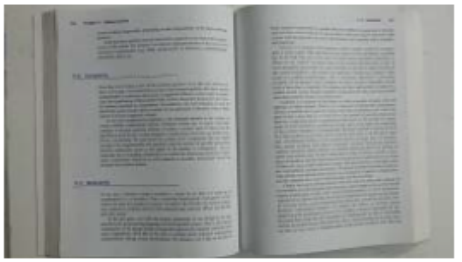
Some applications of computer vision include

- Robotic control
- Surveillance
- Augmented reality
- Medical image analysis
- Document analysis

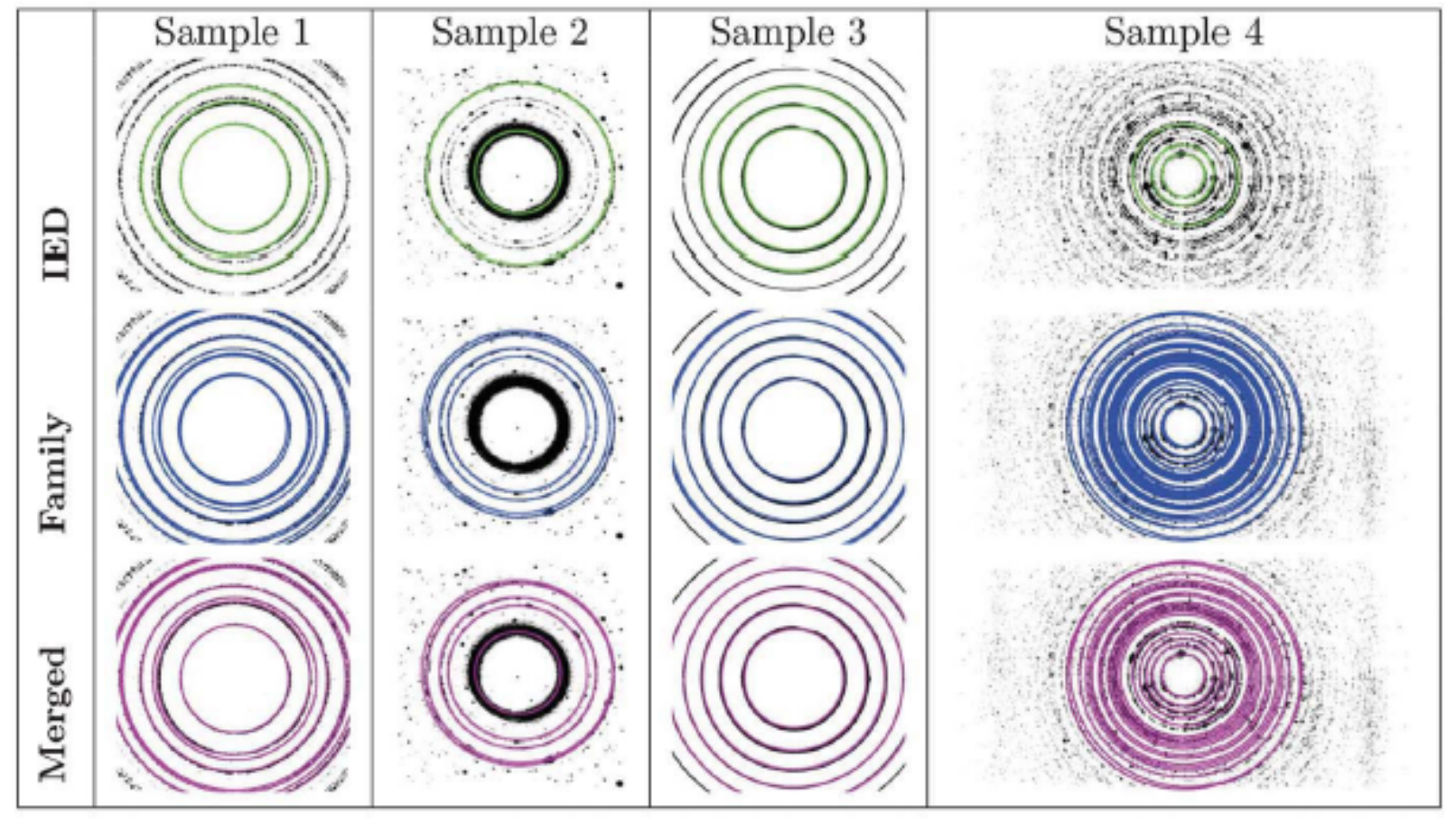
CV @ PUCIT



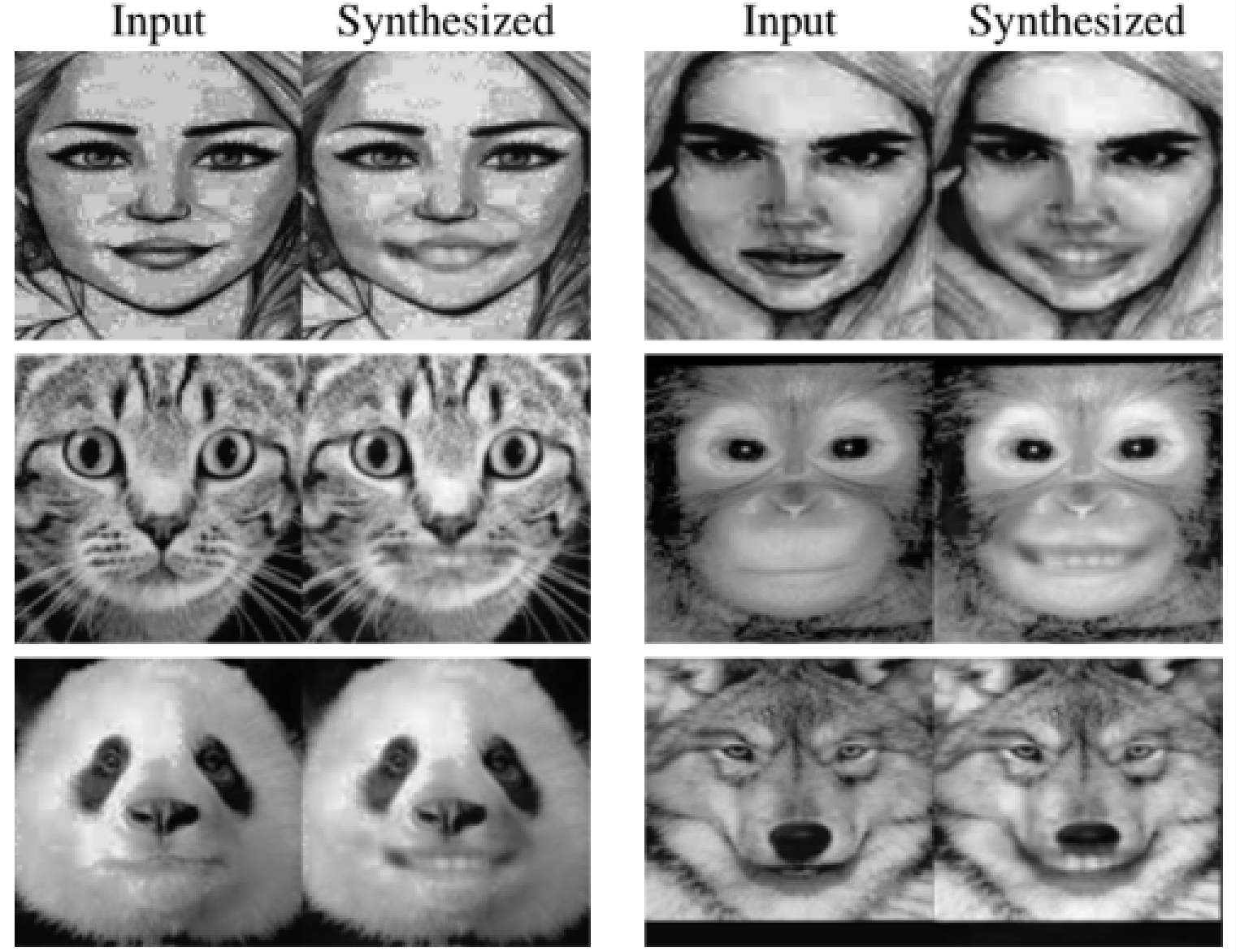
CV @ PUCIT



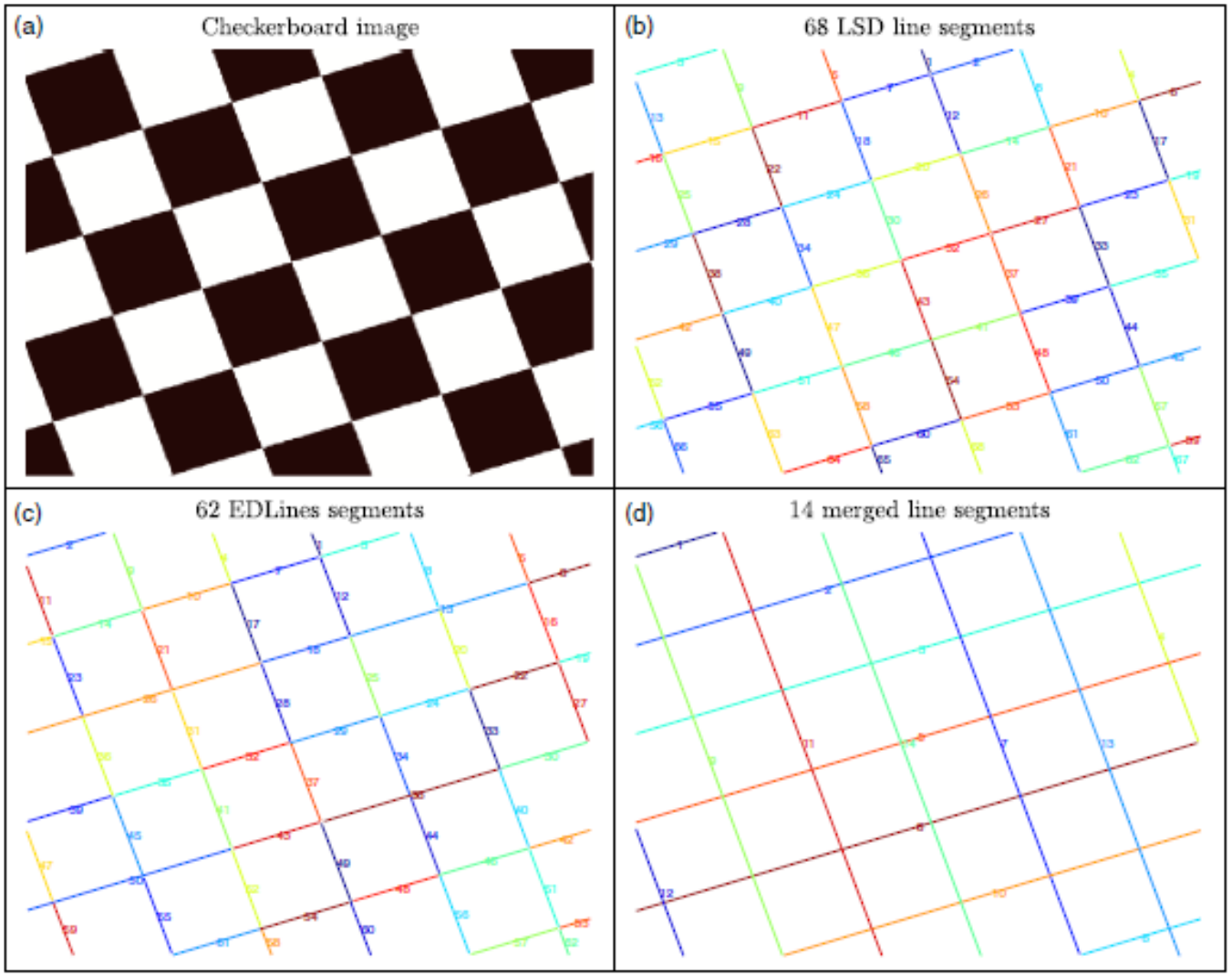
CV @ PUCIT



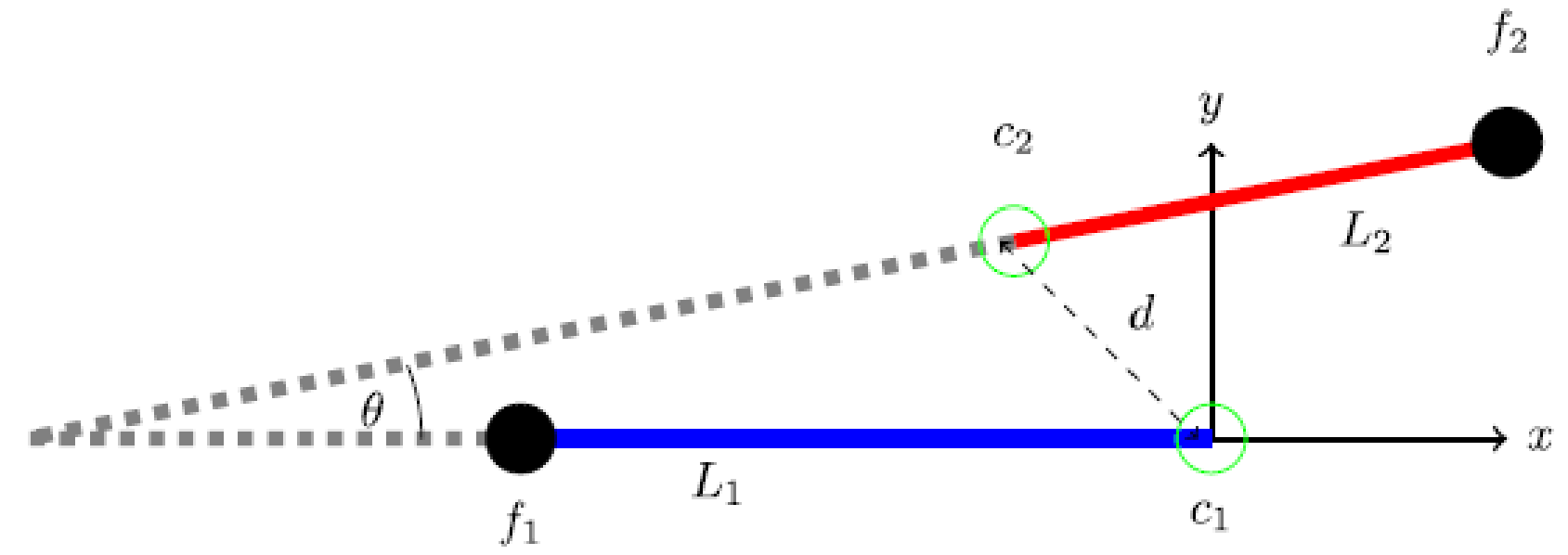
CV @ PUCIT



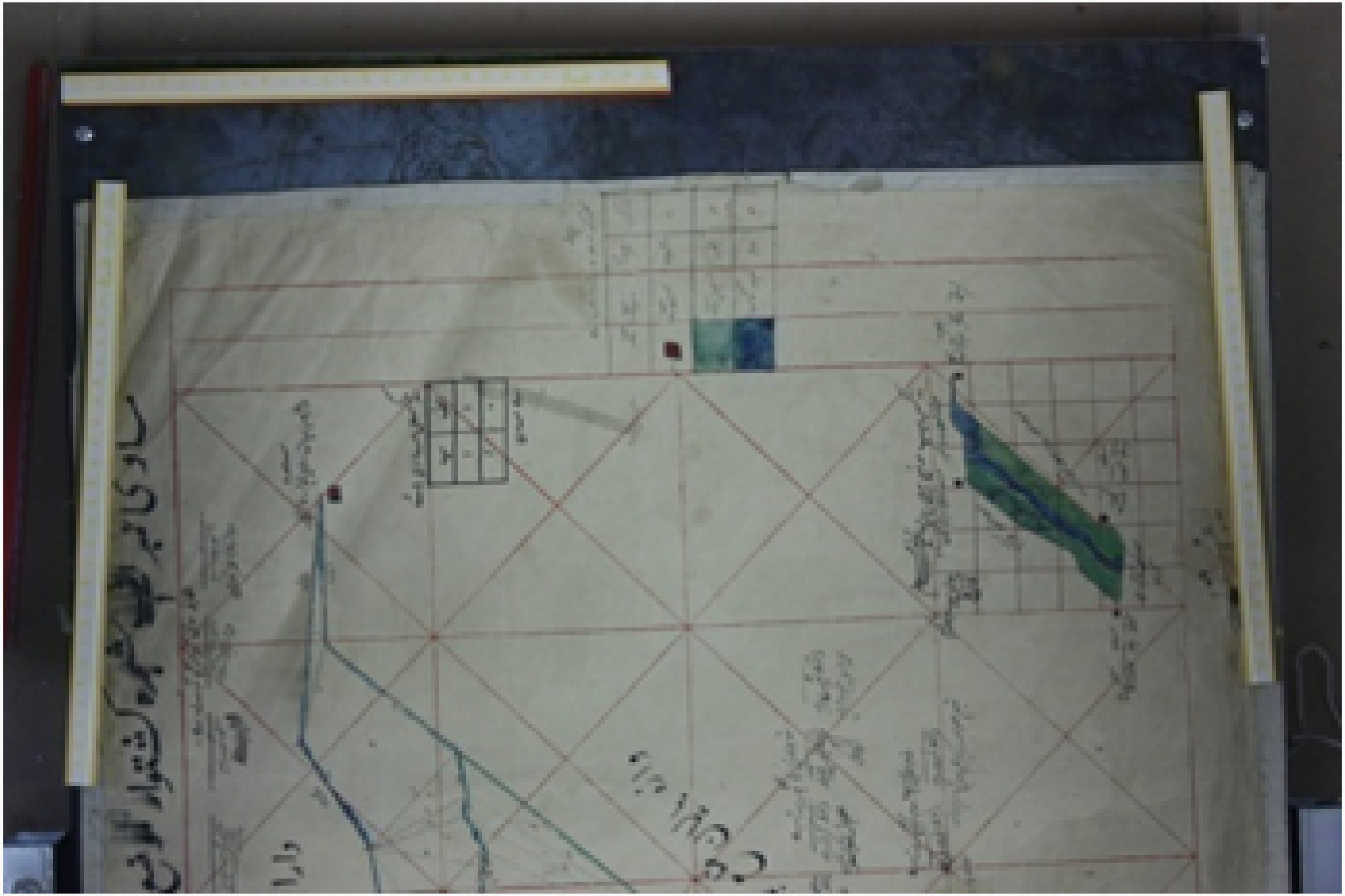
CV @ PUCIT


(a)
Checkerboard image
(b)
68 LSD line segments
(c)
62 EDLines segments
(d)
14 merged line segments

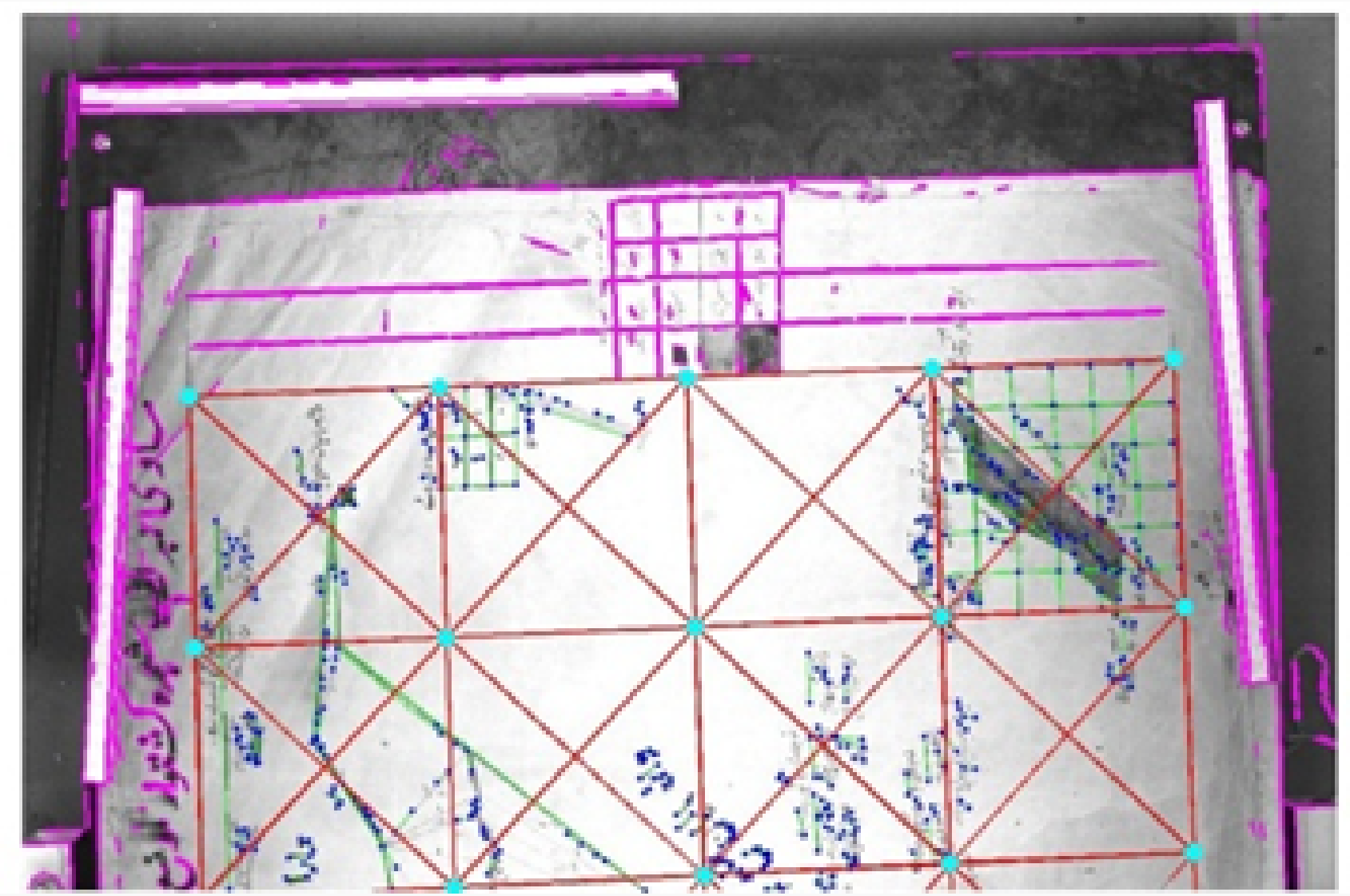
CV @ PUCIT



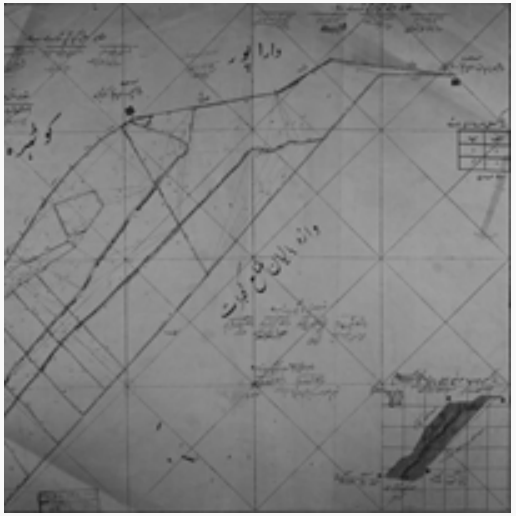
CV @ PUCIT

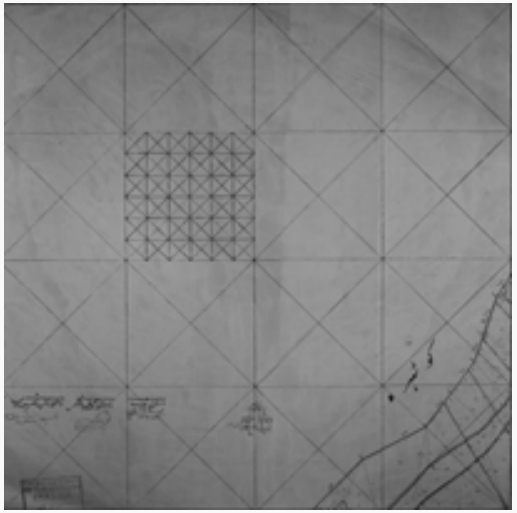


CV @ PUCIT

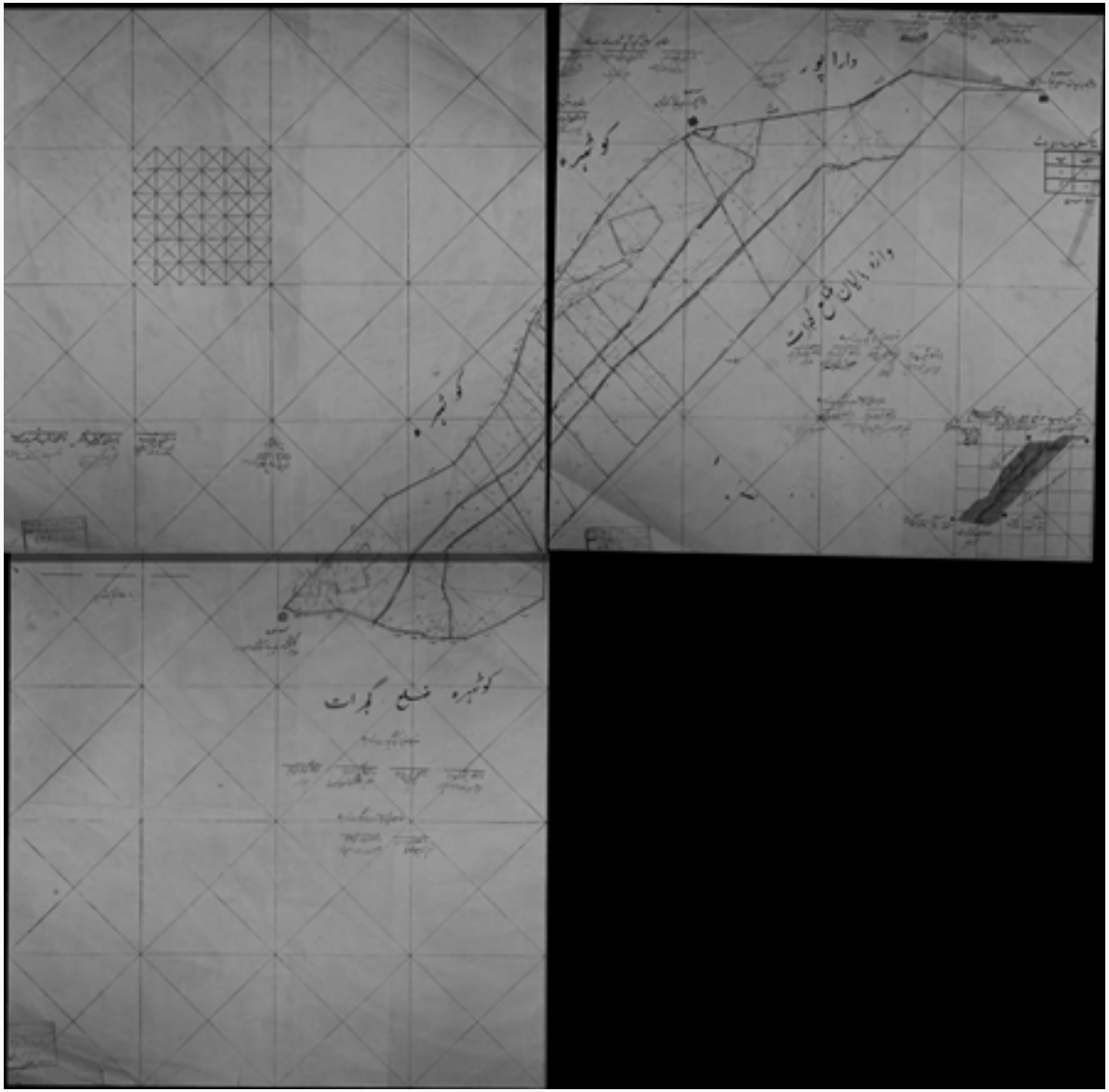


CV @ PUCIT





CV @ PUCIT



Working with images

- *On a hard-disk*, images are usually stored in unsigned 8-bit integer format (uint8).
- This means that pixel values can only be integers ranging from 0 to $255 (= 2^8 - 1)$.
  - What will happen when we subtract/add 5 from every pixel or multiply/divide every pixel by 3?
- *In RAM*, images can have any range and/or data type, *e.g.* satellite or medical imagery having floating precision values in $\mathbb{R}$.

Working with images

- The following transformation *normalizes* pixel values in any range to lie between 0 and c.

$$J(x,y) = \frac{I(x,y) - min}{max - min} \times c$$

Common choices for c are 1 for computations on images and 255 for saving images.

1. Whenever you load an image, check its range and data type.
2. If required, normalize every image into a standard range and data type before performing any computations.
3. During computations, be aware that pixel values can get out of range.
4. Normalize images into proper range before saving.

Python

- Install Python 3.6 from `www.python.org`.
- Install new Python packages from *cmd* using *pip install package_name*
- To start python instance in *cmd*, type *python*.
- To run a python program in *cmd*, type *python my_program.py*.
- IPython is an interactive Python shell that makes debugging and experimentation easier. `http://ipython.org/`.
- An extremely useful Python tutorial for Computer Vision is available at `http://cs231n.github.io/python-numpy-tutorial/`

Basic Image Handling and Processing

Reading, writing, resize, conversion, plotting, selecting pixels

```
from PIL import Image          #image handling functions
from pylab import *            #plotting functions

#image read, convert, resize, rotate, save
im = Image.open('book.png')
im.save('book.bmp')
im.convert('L').resize((640,480)).save('book_gray.jpg')
im_rotated = im.rotate(45)
im_rotated.save('book_rotated.tiff')

im = array(im)  #store image as array
imshow(im)      #plot the image
#4 points
c = [100,100,400,400]    #columns
r = [200,500,200,500]    #rows
plot(c,r,'r*')           #plot 4 points with red star-markers
plot(c[:2],r[:2],'w')    #line plot connecting first two points
plot(c[1:4],r[1:4],'b--')   #line plot connecting last three points
```

```
20  #interactive annotation
21  print('Please click 4 points')
22  x = ginput(4)   #input 4 points using the mouse
23  #draw lines between the 4 points
24  for i in range(0,len(x)-1):
25      plot((x[i][0],x[i+1][0]),(x[i][1],x[i+1][1]),'b')
26  plot((x[-1][0],x[0][0]),(x[-1][1],x[0][1]),'b')
27  print('The ', len(x), ' points that you clicked are:')
28  for i in x:
29      print(i)
30  title('Plotting: "book.png"')   #add title and show the plot
31  show()
```

Getting files in a directory

Defining a function

```
from PIL import Image
import os        #functions for interacting with the operating system

def get_imlist(path, ext):
 """ Return list of '*.ext' files in directory 'path' """
 if ext[0] != '.': ext = '.'+ext
 return ([os.path.join(path,f) for f in os.listdir(path)
          if f.endswith(ext)])

path = 'D:/nazar/CS565Python/'
ext_in,ext_out = '.tiff','.gif'
filelist =  get_imlist(path,ext_in)
for infile in filelist:
    outfile = os.path.splitext(infile)[0] + ext_out
if infile != outfile:
    try:
        Image.open(infile).save(outfile)
    except IOError:
        print("cannot convert", infile)
```

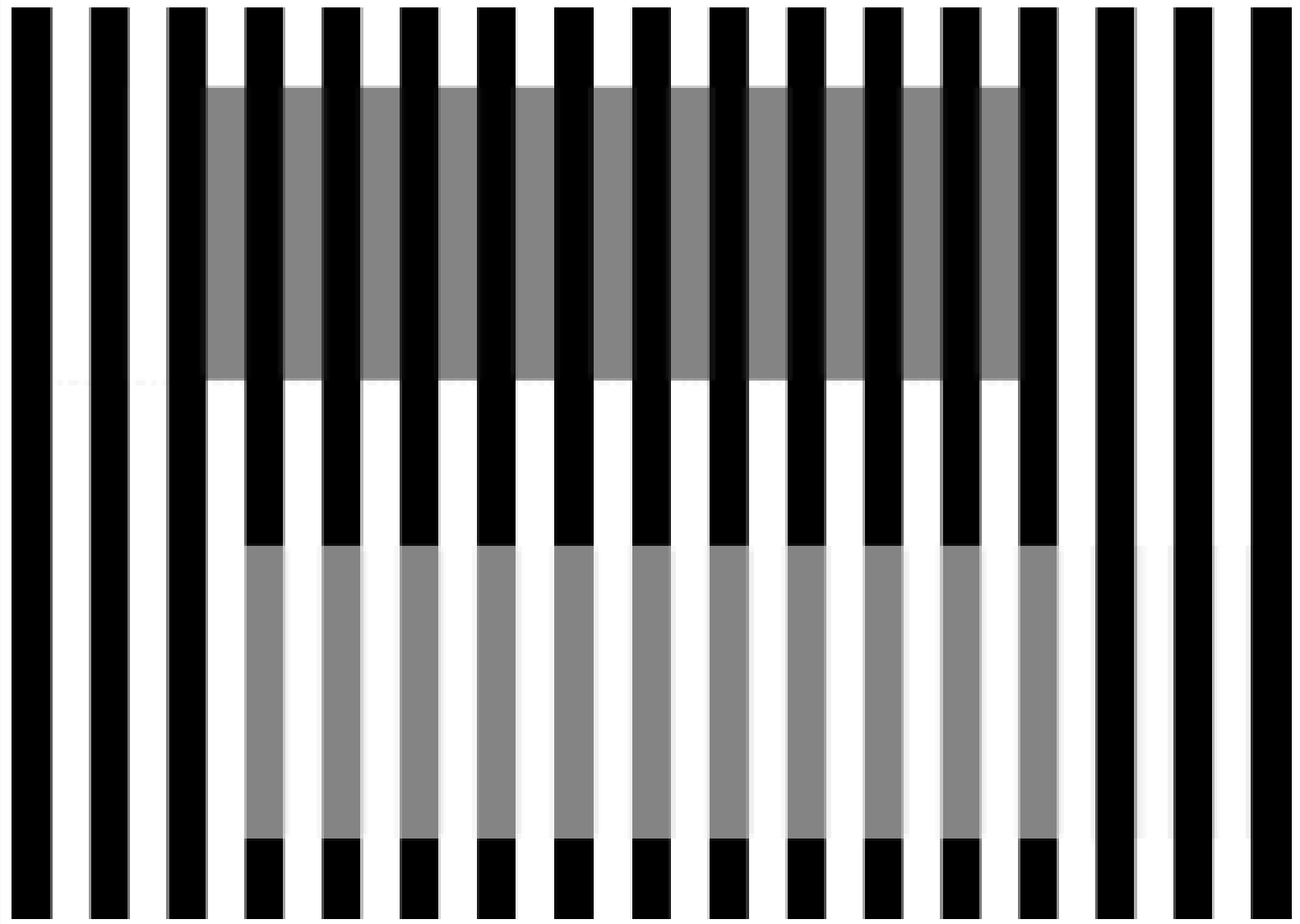
Arrays in NumPy

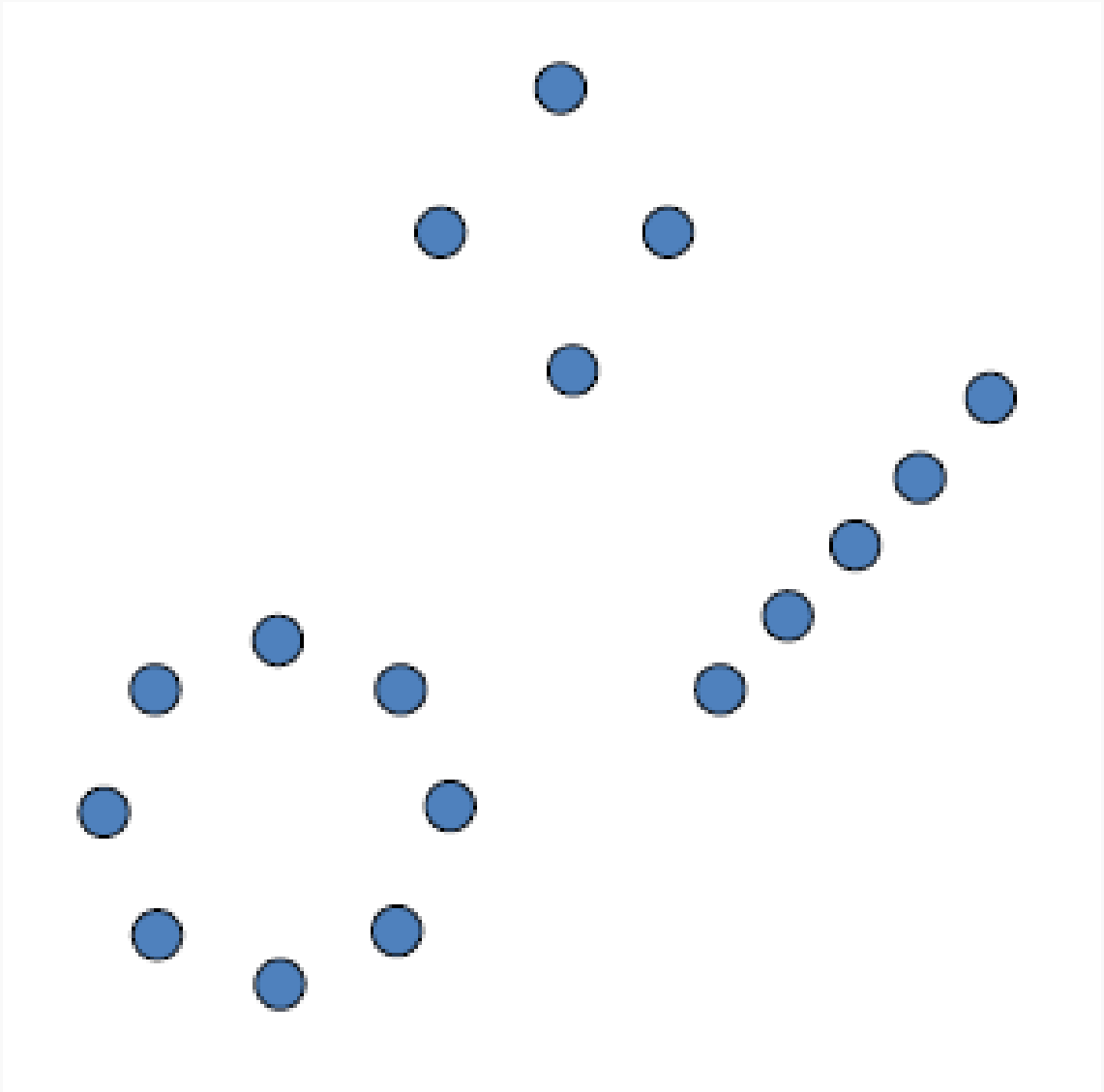
- A 3-dimensional array 'im' can be accessed as im[i,j,k].
- One very important NumPy functionality is *slicing* of arrays.

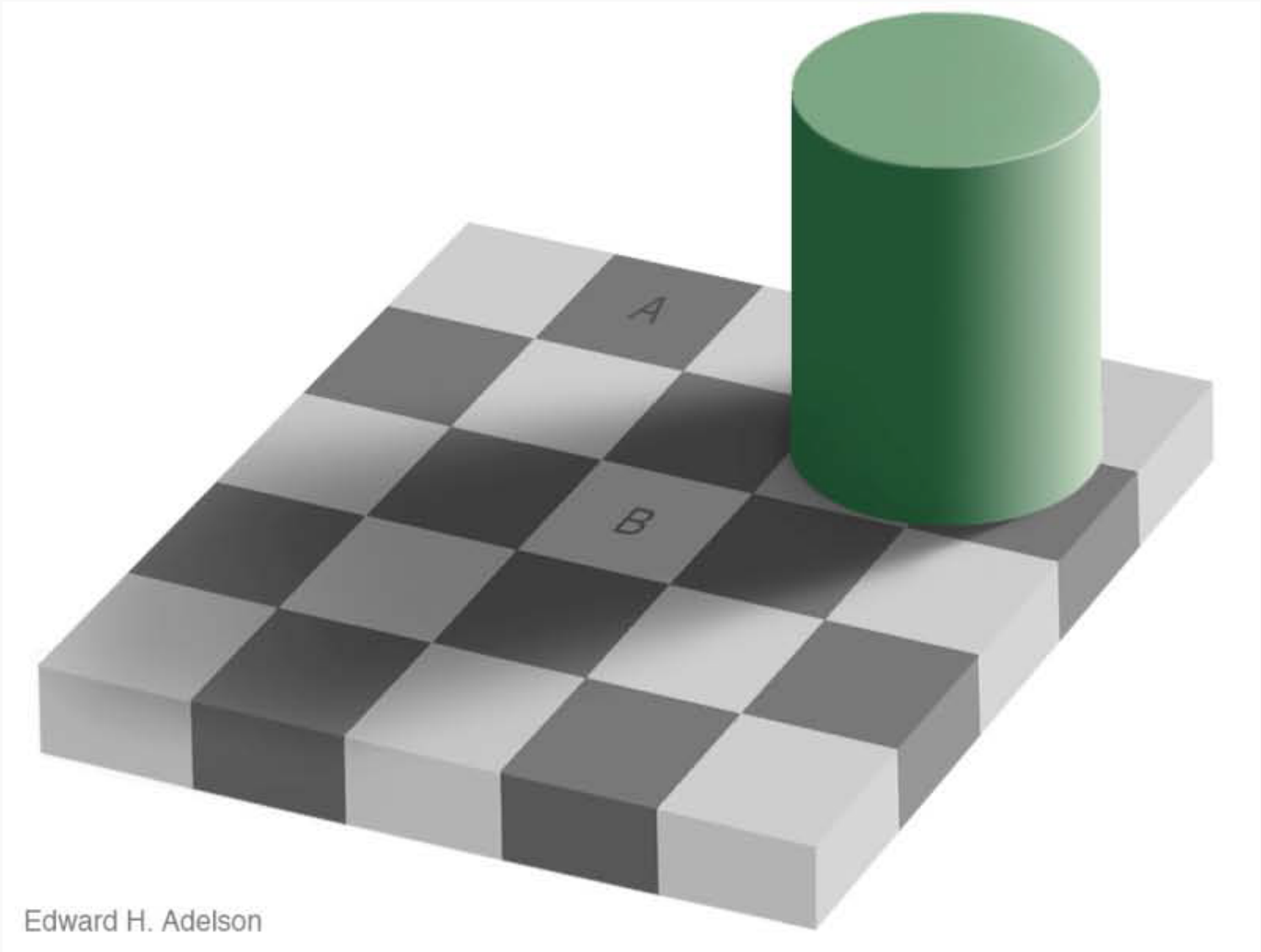
```
im[i,:] = im[j,:] #set values of row i with values from row j
im[:,i] = 100 #set all values in column i to 100
im[:100,:50].sum() #sum of first 100 rows and 50 columns
im[50:90,70:90] #rows 50-90, columns 70-90 (90-th not included)
im[i].mean() #average of row i
im[:,-1] #last column
im[-2,:] (or im[-2]) #second to last row
```

- Loops can be avoided via slicing.


بِسْمِ اللَّهِ الرَّحْمَٰنِ الرَّحِيمِ






A
B
Edward H. Adelson