INTRODUCTION TO COMPUTER VISION

Natalie Parde
parde@uic.edu

CS 594: Language and Vision
Spring 2019

Computer Vision

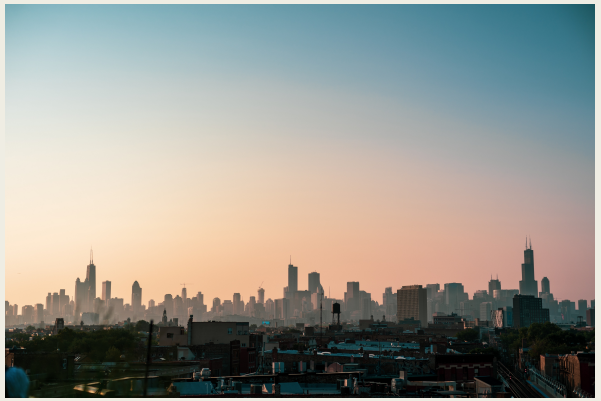
- The study of computational mechanisms that enable machines to interpret, modify, and generate visual imagery.

How are images represented in computer vision?

- Two-dimensional array of pixels
- A numeric value is associated with each pixel, corresponding to its color/shade

Types of Images

- Black and white
  - *All pixels contain a binary value*
- Grayscale
  - *All pixels contain an integer ranging from 0-255*
  - *This value corresponds to **shade** from black to white*
- Color
  - *All pixels contain three integers, each ranging from 0-255*
  - *These values correspond to **shades** of three color channels: red, green, and blue*



A few core tasks....

IMAGE PROCESSING

IMAGE SEGMENTATION

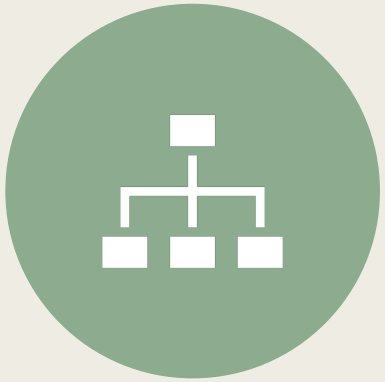


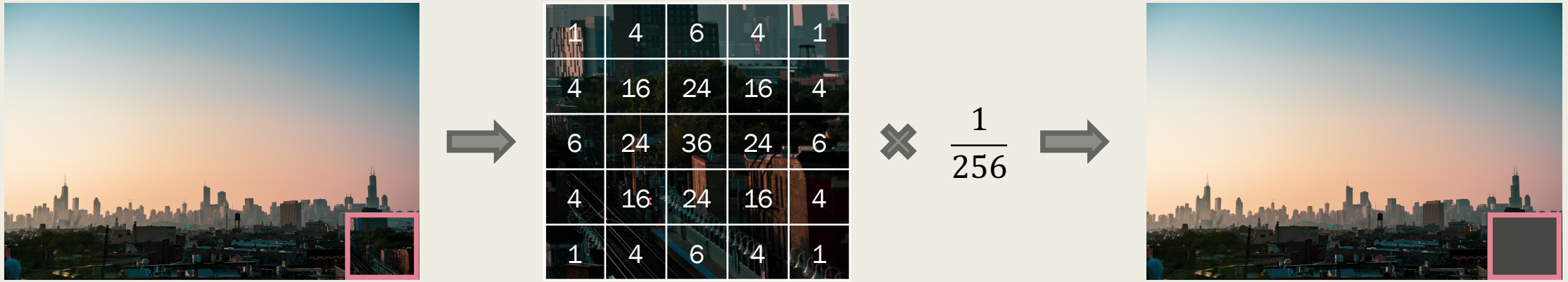
IMAGE CLASSIFICATION

Image Processing

- Performing an action or sequence of actions on an image in order to transform it or extract information from it
- Actions might include:
  - *Blurring the image*
  - *Sharpening the image*
  - *Making the image brighter*
  - *Making the image darker*
  - *Cropping the image*
  - *Resizing the image*

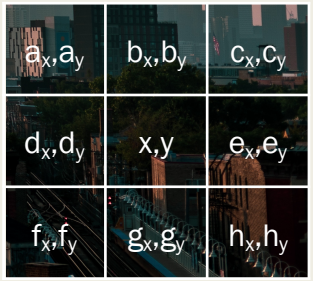
Gaussian Blur

- Blurring an image using a Gaussian function
- All pixels in a square window are multiplied by fractions corresponding to their distance from the center of the window
- The products are summed to produce the single value for the output pixel



Sobel Filter

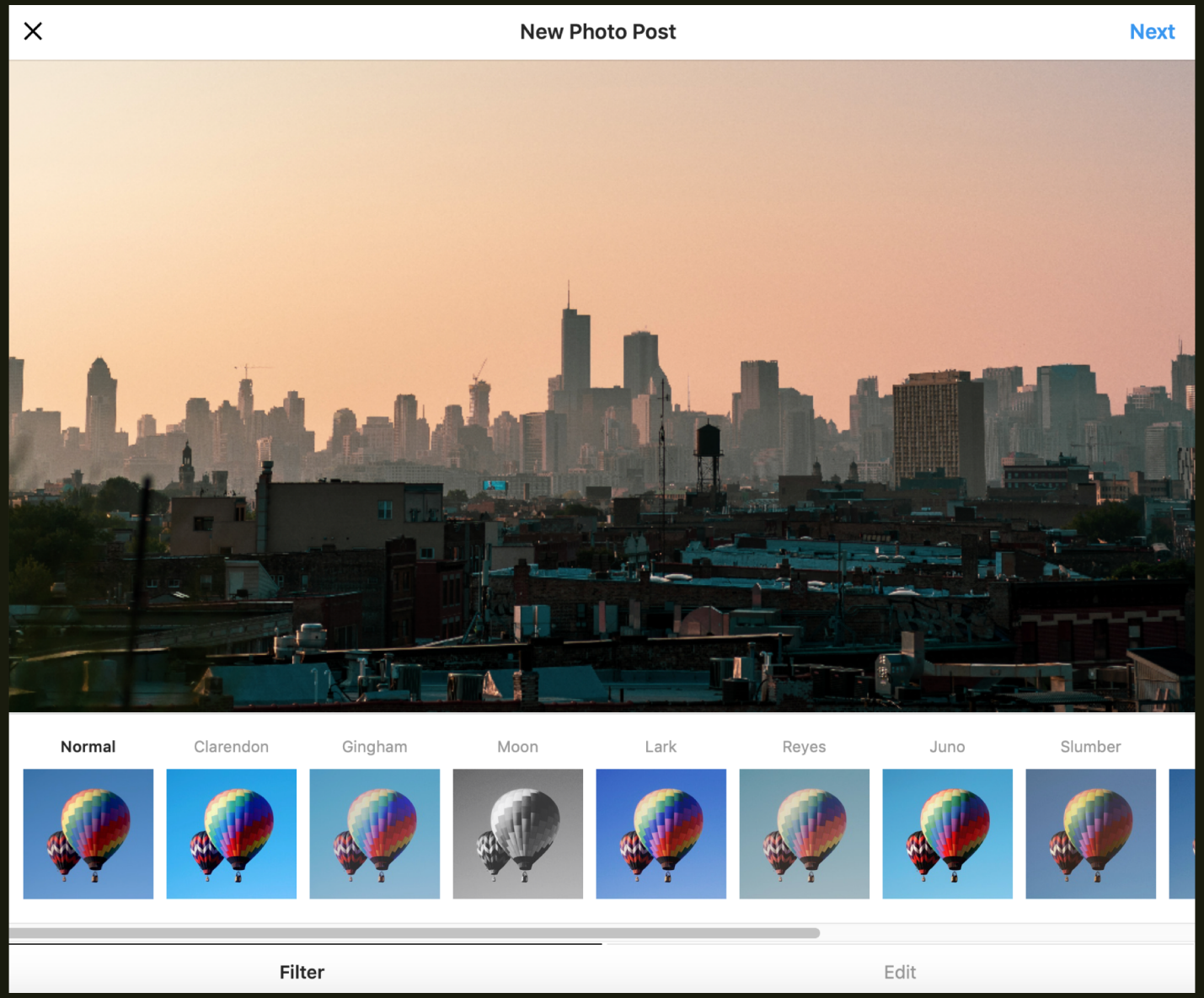
- Detecting edges by identifying areas in an image with sharp changes in color and/or intensity between two adjacent pixels
- All pixels in a square window are multiplied by two different kernels
  - *One designed to catch horizontal changes*
  - *One designed to catch vertical changes*



-1	0	1
-2	0	2
-1	0	1

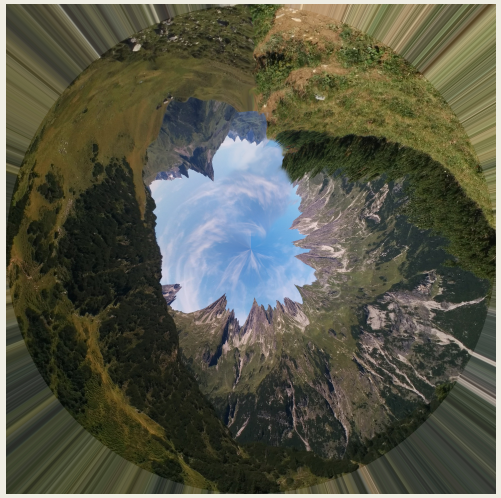
-1	-2	-1
0	0	0
1	2	1

$$x' = -a_x + c_x - 2d_x + 2e_x - f_x + h_x$$
$$y' = -a_y - 2b_y - b_y + f_y + 2g_y + h_y$$
$$\text{mag}(x', y') = \sqrt{x'^2 + y'^2}$$



One obvious application of image processing....

- Instagram filters!
- Typically a mix of blurring/sharpening and color operations

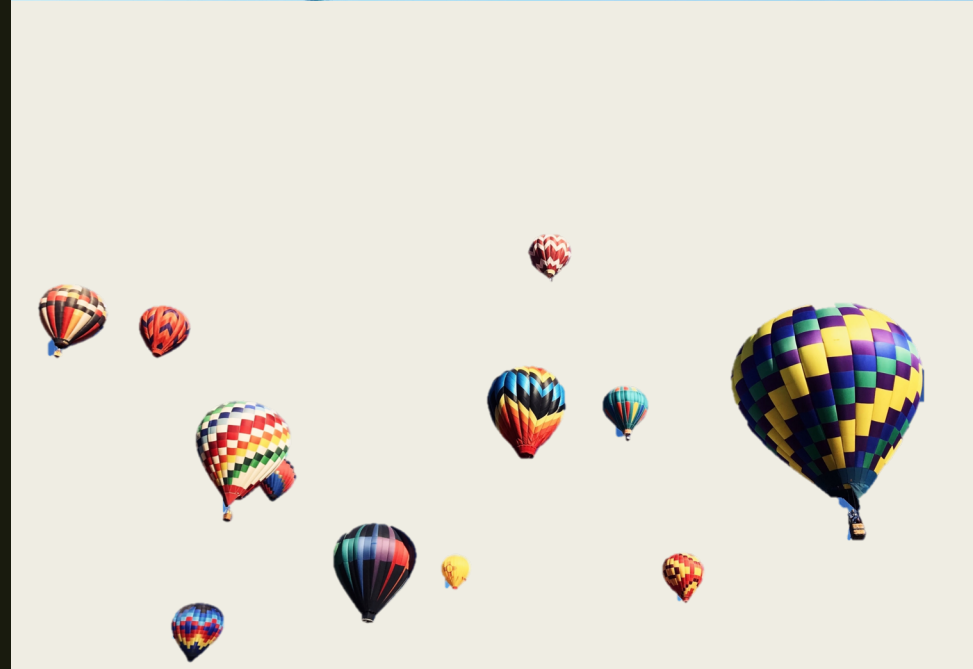


Other Applications

- Image editing
- Image stitching
- Image restoration
- Digital art

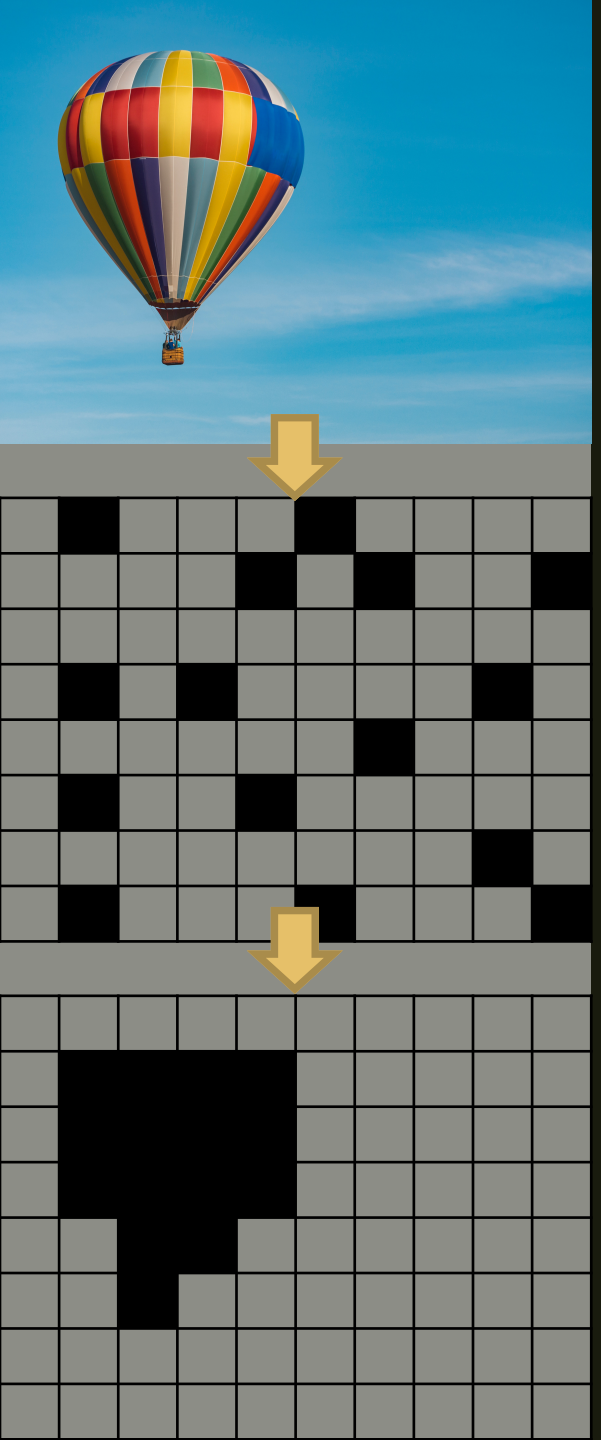
Image Segmentation

- Automatically finding the boundaries between different entities in an image
- Necessary preprocessing step for many downstream tasks!
- Two common approaches:
  - *Watershed method*
  - *K-means clustering*



Watershed Algorithm

- An image segmentation technique modeled after the way water flows from a high point (a hill or mountain) to a lower point (a river or large body of water)
  - *A **watershed** is the high point, whereby on one side water flows one way, and on the other side it flows the other way*
- Images are converted to grayscale
  - *Viewed as a topographic map, high points → light shades, low points → dark shades*
- Pixels in the image can correspond to:
  - *A local minimum*
  - *A basin that flows to the local minimum*
  - *Watershed lines*
- Goal is to detect the watershed lines, thereby segmenting the image



K-Means Clustering

- Segments images by clustering pixels into *k* groups based on their associated features
- K-Means clustering works by:
  1. *Randomly assigning each pixel to one of k clusters*
  2. *Computing the mean of each cluster*
  3. *Computing the similarity between each pixel (regardless of current cluster assignment) and each cluster mean*
  4. *Assigning each pixel to the cluster to whose mean it is most similar*
  5. *Repeating steps 2-4 until the clusters converge*
- K-Means clustering can be applied to many tasks!
- With image segmentation, additional assumptions can be added when assigning pixels to improve upon segmentation performance (e.g., neighboring pixels are highly likely to belong to the same cluster)

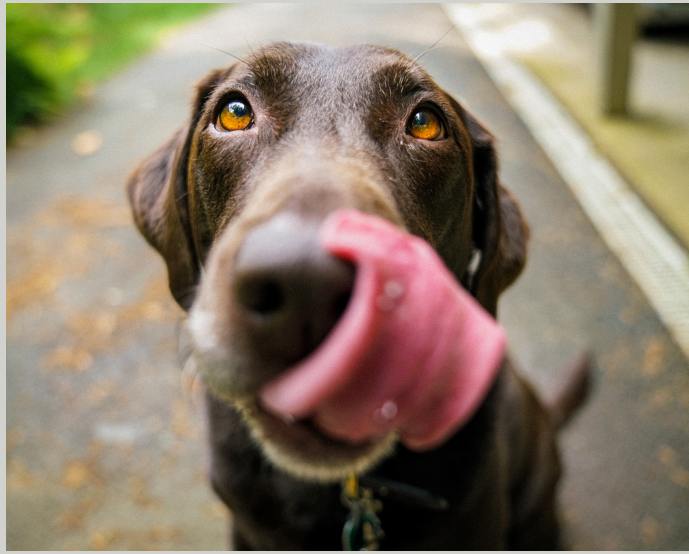


Image Classification

- Automatically assigning a category label to an image or to an entity in an image
- Why is this difficult?
  - *Different perspectives*
  - *Different sizes*
  - *Proximity to other objects*
  - *Lighting conditions*
  - *Variation among objects of the same class*

How can we solve this problem?

- Lots of data!
- Traditional machine learning
  - *K-Nearest Neighbor (classification version of K-Means)*
- Deep learning
  - *Convolutional neural networks*

?

?

Features for Image Classification

RGB Histograms

Local Binary Patterns (LBP)

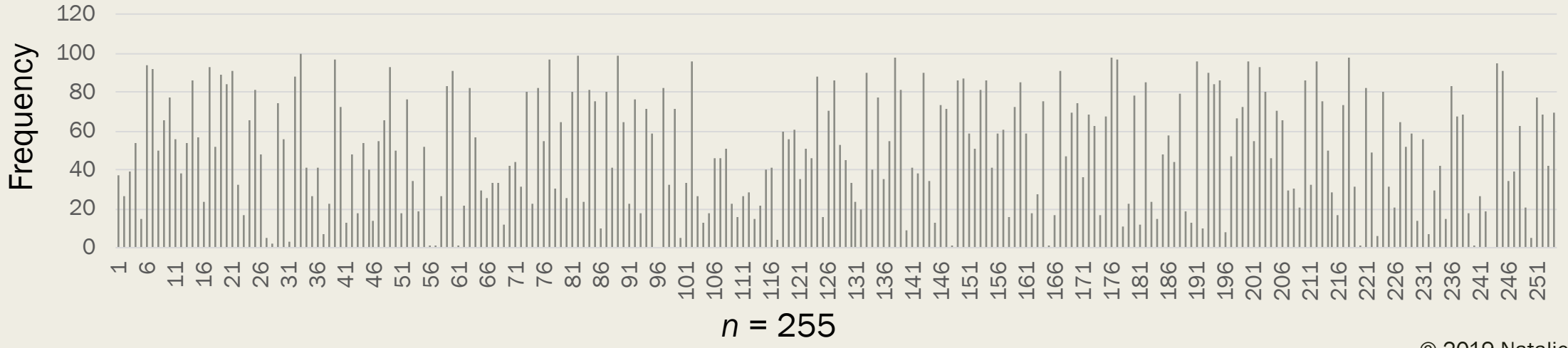
Histograms of Oriented Gradients (HOG)

Scale-Invariant Feature Transform (SIFT)

Neural Features

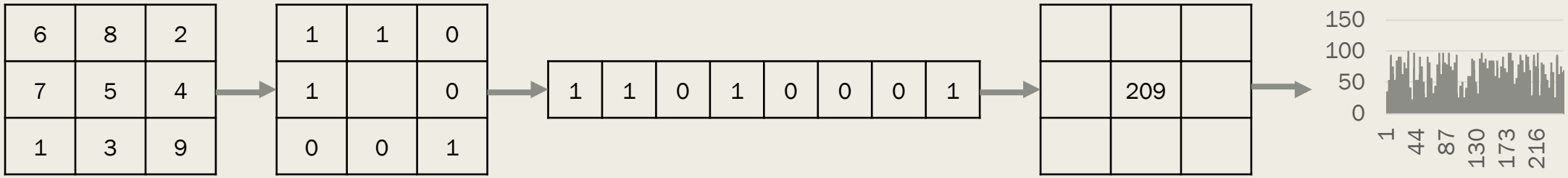
RGB Histograms

- Computed across a region, rather than a single pixel
- *N*-dimensional ($1 \leq n \leq 255$) feature vector for each color channel (red, green, and blue)
- Feature values correspond to the number of pixels in each range within the region of interest



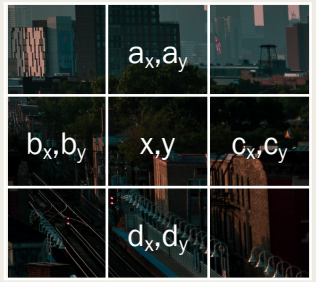
Local Binary Patterns

- Computes a local representation of texture by comparing the values for a given pixel with those of other neighboring pixels
- Sets binary values for neighboring pixels based on a threshold:
  - *Current pixel* $\geq$ *neighboring pixel* $\rightarrow$ *1*
  - *Current pixel* $<$ *neighboring pixel* $\rightarrow$ *0*
- Converts *n*-bit array of binary pixel values to decimal
- When a decimal value has been computed for all pixels, computes a histogram over all decimal values
- This histogram is the feature vector!



Histograms of Oriented Gradients

- Computes the distribution of gradient (increases or decreases in magnitude) orientations within a source image, providing us with information about the shapes of objects in the image
- Starts by computing the horizontal and vertical gradients for each pixel, along with their magnitude and direction
- Calculates histograms of the computed gradients in pixel windows of size $i \times j$
  - *Bins in the histogram correspond to angles*
  - *Values added to the bins correspond to magnitudes*
- Normalizes blocks of gradients across multiple pixel windows
- Concatenates all normalized vectors of histograms to create the final feature vector



-1	0	1

-1
0
1

$$x' = -b_x + c_x$$
$$y' = -a_y + d_y$$
$$\theta = \arctan \frac{x'}{y'}$$
$$\text{mag}(x', y') = \sqrt{x'^2 + y'^2}$$

Scale-Invariant Feature Transform

Computes **keypoints** in an image in a way that is invariant to size, rotation, perspective, and lighting conditions

Progressively blur an image using Gaussian Blur across several **octaves** (resized images that are half the previous size)

Compute the **Difference of Gaussians (DoG)** for each sequential pair of blurred images, subtracting the blurrier from the clearer of the two

Compute the minima and maxima in the DoG images by comparing each pixel with all others in its neighborhood (including pixels in the DoGs "above" and "below" it)

- *If the pixel itself is higher or lower than all other pixels in its neighborhood, it is a minimum or maximum*

Reject minima and maxima (a) with intensities below a threshold or (b) having gradients indicative of a flat region or edge (rather than a corner)

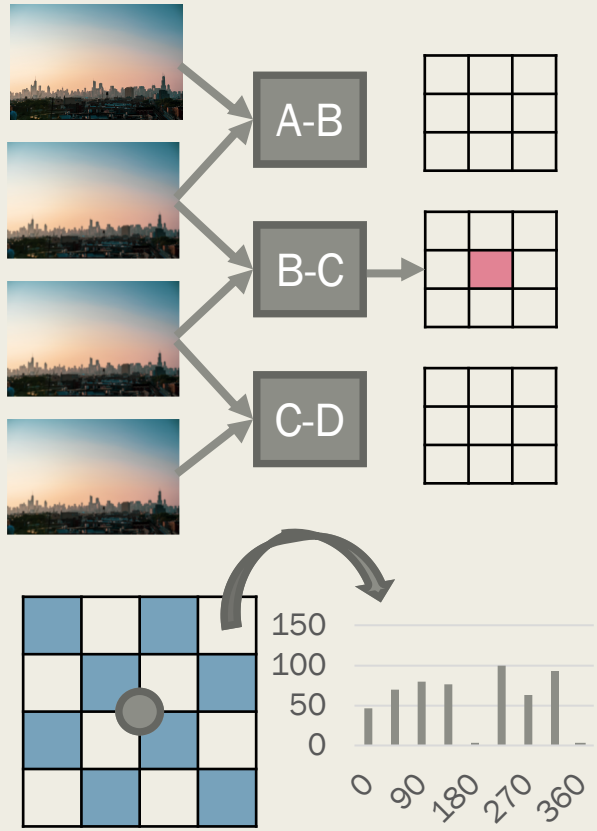
Compute the gradient and magnitude of the pixels around each keypoint (minima/maxima), and construct histograms of these gradients using 10-degree bins

- *The keypoint is assigned the orientation that serves as the "peak" in the histogram*

Generate a 16x16 pixel window around each keypoint, and break that window in sixteen 4x4 windows

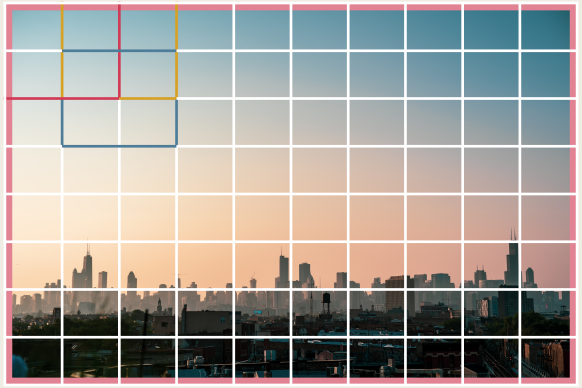
Compute the gradient and magnitude for each 4x4 window, and construct histograms using 45-degree bins ranging from 0-360 degrees

Normalize the values in those bins (subtract the keypoint's orientation from each 4x4 window, and create a maximum threshold to avoid influence from changes in lighting), producing the 128-dimensional (16 4x4 windows, 8 bins per window) feature vector for the keypoint



Neural Features

- Raw pixel values
- Pixel values in sliding windows
  - *Individual windows are fed into the neural network*
- Pixel regions
  - *Input image is first scanned for possible objects using a search algorithm, and the pixel regions comprising the possible objects are then fed into the neural network*
- The neural network automatically learns which combinations of pixels/pixel values represent different image features …the bulk of the work when using neural features shifts to tuning parameters in the network



Think, Pair, Share

- Write down three computer vision tasks for which one or more of the engineered features we discussed (RGB histograms, LBP, HOG, and SIFT) might be useful, and three tasks for which implicitly learned features might have an advantage over other feature types
- Talk about those viewpoints with a partner
  - *Did you name similar tasks?*
  - *Did you disagree about anything?*
- Share one task for which one or more of the engineered features might be useful, and one task for which implicitly learned features might have an advantage

- Timer: https://www.google.com/search?q=timer

Popular Tools for Computer Vision Tasks

OpenCV	• https://opencv.org/
scikit-image	• https://scikit-image.org/
SimpleCV	• http://simplecv.org/
Pillow	• https://python-pillow.org/

Datasets and Other Useful Resources

- Huge, curated list of computer vision books, courses, datasets, libraries, tutorials, blogs, etc.: https://github.com/jbhuang0604/awesome-computer-vision
- Main computer vision conferences:
  - *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*
  - *IEEE International Conference on Computer Vision (ICCV)*
- Datasets for various computer vision tasks:
  - *http://www.cvpapers.com/datasets.html*
  - *http://riemenschneider.hayko.at/vision/dataset/*
  - *https://www.visualdata.io/*
- ImageNet Challenge:
  - *2018 onward: https://www.kaggle.com/c/imagenet-object-localization-challenge*
  - *2017 and prior: http://image-net.org/challenges/LSVRC/2017/index*

Applications of Computer Vision

Face recognition

Emotion recognition

Biomedical image processing

Self-driving cars

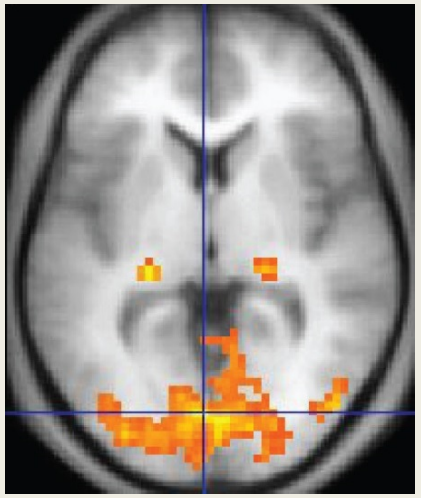
Robotics

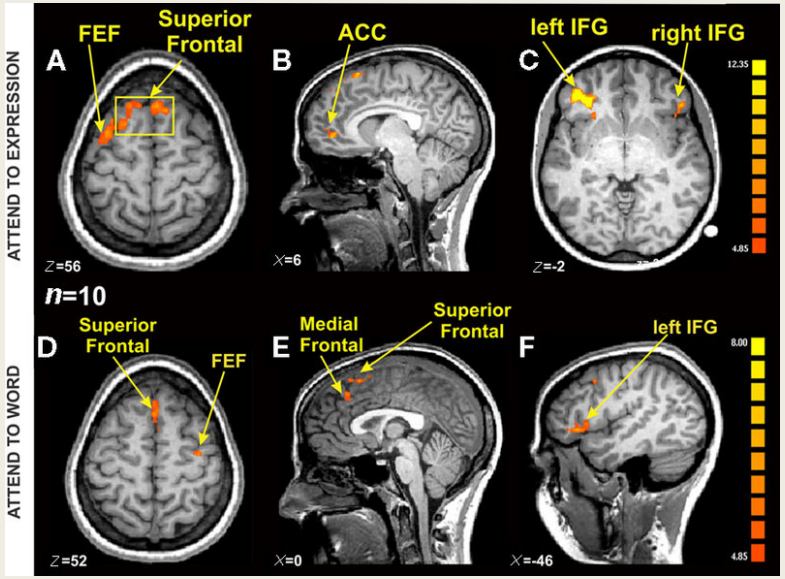
Face Recognition

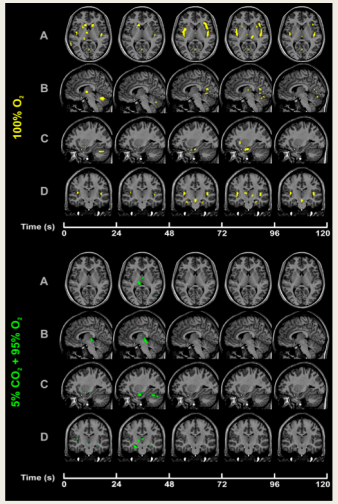
- Automatically matching new faces to known individuals
- Useful for a variety of applications:
  - *Surveillance*
  - *Passport/ID card authentication*
  - *Safety alerts*
  - *Photo tagging*
- Must be robust to changes in perspective, scale, and lighting
- Popular methods:
  - *GaussianFace:* https://github.com/jangerritharms/GaussianFace
  - *FaceNet:* https://github.com/davidsandberg/facenet
- Useful face recognition library: https://github.com/ageitgey/face_recognition

Emotion Recognition

- Automatically matching facial expressions to a set of known emotions
- Useful for:
  - *Forensics*
  - *Marketing*
  - *Social robotics*
- Open question: should emotion recognition be framed as a categorical or continuous problem?
  - *Most work to date has classified emotions into seven discrete categories: fear, anger, sadness, disgust, contempt, happiness, and surprise*
- Useful tools for emotion recognition:
  - *EmoPy:* https://github.com/thoughtworksarts/EmoPy
  - *Microsoft Face API:* https://azure.microsoft.com/en-us/services/cognitive-services/face/





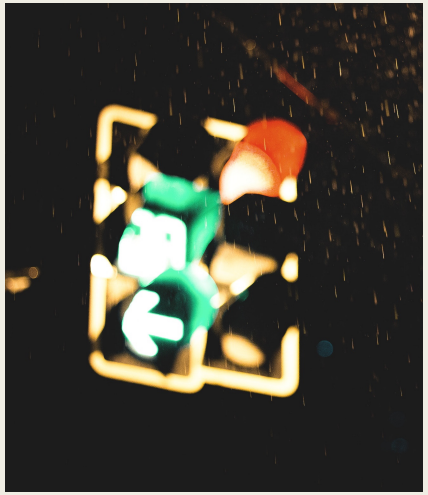


Biomedical Image Processing

Employing computer vision to aid in clinical diagnosis, anatomical modeling, and surgical guidance	Relies heavily on:	Useful tools for biomedical image processing:
• *Cancer detection* • *fMRI analysis* • *Surgical site visualization* • *Electrical source imaging*	• *Segmentation* • *Shape analysis*	• *Deep Learning Toolkit for Medical Imaging (DLTK): https://dltk.github.io/* • *Insight Segmentation and Registration Toolkit (ITK): https://itk.org/*

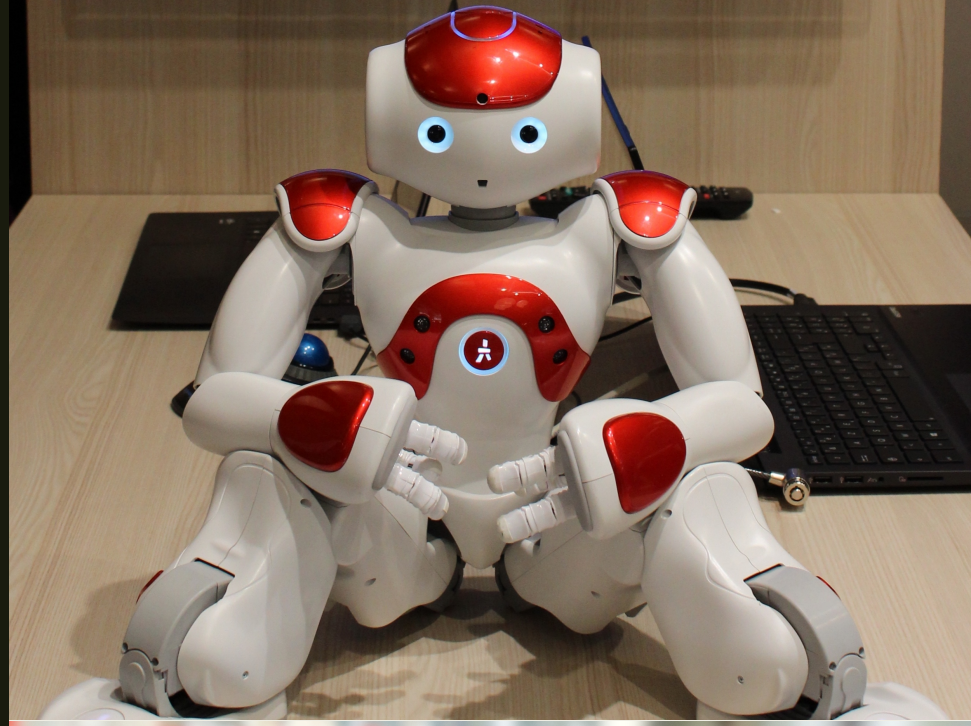
Self-Driving Cars

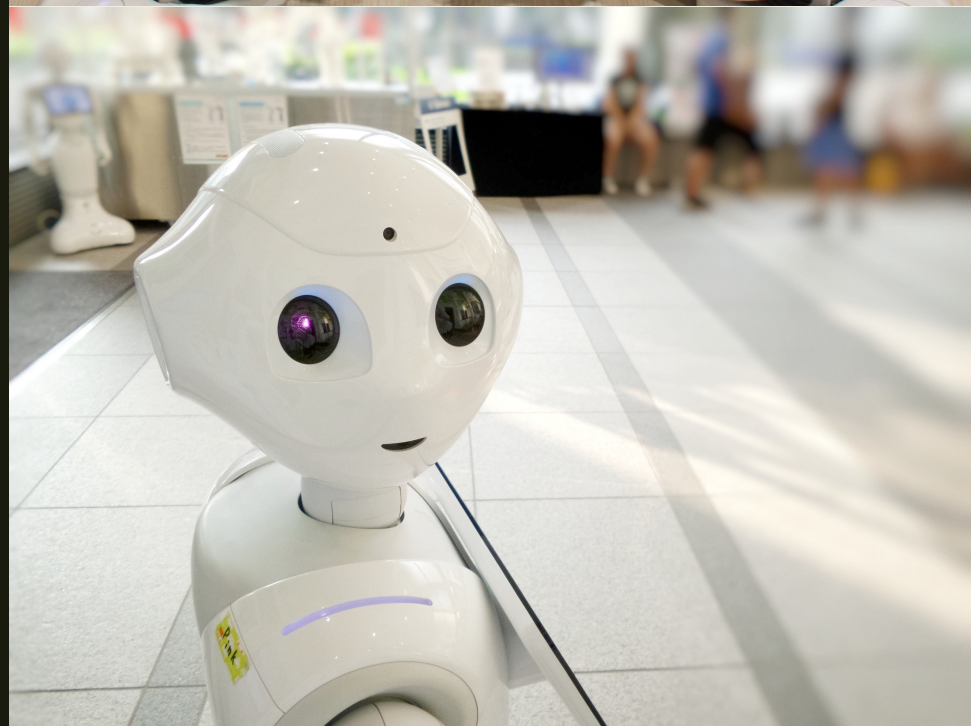
- Automatically detecting all the necessary information needed to drive a car autonomously:
  - *Lane boundaries*
  - *Traffic lights*
  - *Road curvature*
  - *Obstacles (objects, animals, other cars, or people!)*
  - *Traffic signs*
- Must track objects from one frame to another
- Must do all of this in real time, and quickly enough that other components of the car can act upon the information
- For more information about self-driving cars:
  - https://engineering.uic.edu/interdisciplinary-areas/autonomous-vehicles/
  - *Waymo self-driving cars:* https://youtu.be/B8R148hFxPw



Robotics

- Automatically interpreting visual input to facilitate navigation and enhance a robot's knowledge of its surrounding environment
  - *Can a robot move from one room to another without encountering obstacles?*
  - *Can an robot detect irregularities in items moving down an assembly line?*
  - *Can a robot in a multi-person home correctly identify the person with whom it is speaking? (Can it automatically detect that a person is in the room?)*
- Advantages that robots have over most other vision applications: motion and capacity for two-way interaction
- Useful tools for robotics applications:
  - *Robot Operating System (ROS):* http://www.ros.org/
  - *Webots:* https://www.cyberbotics.com/
  - *Simultaneous Localization and Mapping (SLAM) datasets:* https://github.com/younggguncho/awesome-slam-datasets
  - *Code for affect recognition in NAO robots:* https://github.com/thealexhong/starship
  - *Code for gesture recognition in NAO robots:* https://github.com/AravinthPanch/gesture-recognition-for-human-robot-interaction
- For further reading, check out: https://github.com/Kiloreux/awesome-robotics





Wrapping up....

- Image characteristics
- Image processing
  - *Gaussian Blur, Sobel filter*
- Image segmentation
  - *Watershed algorithm, K-Means clustering*
- Image classification
  - *Engineered and implicitly learned features*
- Tools, datasets, and other useful resources
- Applications of computer vision
  - *Face recognition, emotion recognition, biomedical image processing, self-driving cars, and robotics*