Inference tasks

- **Simple queries**: compute posterior probabilities.
- **Optimal decisions**: decision networks include utility information; probabilistic inference required for $P(\text{outcome}|\text{action, evidence})$
- **Value of information**: which evidence to seek next?
- **Sensitivity analysis**: which probability values are most critical?
Bayesian Networks

Bayesian networks are a type of probabilistic graphical model that uses Bayesian inference for probability computations.
Bayesian Networks

Bayesian networks are a type of probabilistic graphical model that uses Bayesian inference for probability computations.

- A simple, graphical notation for conditional independence assertions.
Bayesian networks are a type of probabilistic graphical model that uses Bayesian inference for probability computations.

- A simple, graphical notation for conditional independence assertions.
- **Syntax:**
  - a set of nodes, one per variable;
  - a directed, acyclic graph (link ≈ “directly influences”);
  - a conditional distribution for each node given its parents:
    \[ P(X_i|\text{Parents}(X_i)) \]
Bayesian Networks

Bayesian networks are a type of probabilistic graphical model that uses Bayesian inference for probability computations.

- A simple, graphical notation for conditional independence assertions.
- **Syntax:**
  - a set of nodes, one per variable;
  - a directed, acyclic graph (link \( \approx \) “directly influences”);
  - a conditional distribution for each node given its parents:
    \[ P(X_i | \text{Parents}(X_i)) \]
- In the simplest case, conditional distribution represented as: a *conditional probability table* (CPT) giving the distribution over \( X_i \) for each combination of parent values.
Bayesian Inference

Bayesian inference is about the quantification and propagation of uncertainty, defined via a probability, in light of observations of the system.

From Prior → Posterior
Bayesian Inference

Bayesian inference is about the quantification and propagation of uncertainty, defined via a probability, in light of observations of the system.

From Prior $\rightarrow$ Posterior

**Reminder**: A posterior probability is the probability of the event’s outcome given the data (observation).
A prior probability is the probability of the event’s outcome before you collect the data (make observations).
Bayesian Networks: Uses

- **Diagnosis:** \( P(\text{cause}|\text{symptom}) = ? \)
- **Prediction:** \( P(\text{symptom}|\text{cause}) = ? \)
- **Classification:** \( \max_{\text{class}} P(\text{class}|\text{data}) \)
- **Decision-making** (given a cost function)
Naive Bayes Classifier

- Uses the probabilities of each attribute belonging to each class to make a prediction.
- Assumes that the probability of each attribute belonging to a given class value is independent of all other attributes.
- A version of this algorithm is applied to Computer Vision and implemented in OpenCV (we will return to this algorithm later).
Computer Vision

Make computers understand images and video.
Computer Vision

Make computers understand images and video.
Computer Vision

Make computers understand images and video
Computer Vision

Make computers understand images and video
- What kind of scene?
- Where are the cars?
- How far is the building?
Why computer vision matters?

Safety

Health

Security

Comfort

Fun

Access
Applications of Computer Vision

“Face Recognition”
“Pose Estimation”
“Body Tracking”

“Speech Reading”
“Palm Recognition”
“Car Tracking”
Compact representation for image data in terms of a set of components.
Segmentation

- Compact representation for image data in terms of a set of components.
- Components share “common” visual properties.
Segmentation

- Compact representation for image data in terms of a set of components.
- Components share “common” visual properties.
- Properties can be defined at different level of abstractions.
Segmentation

- **Tokens**
  - whatever we need to group (pixels, points, surface elements, etc.).
Segmentation

- **Tokens**
  - whatever we need to group (pixels, points, surface elements, etc.).

- **Bottom up segmentation**
  - tokens belong together because they are locally coherent.
Segmentation

- **Tokens**
  - whatever we need to group (pixels, points, surface elements, etc.).

- **Bottom up segmentation**
  - tokens belong together because they are locally coherent.

- **Top down segmentation**
  - tokens belong together because they lie on the same object.
What is Segmentation?

Clustering image elements that “belong together”

- **Partitioning**
  - Divide into regions/sequences with coherent internal properties.

- **Grouping**
  - Identify sets of coherent tokens in image.
An open source BSD licensed computer vision library
- Patent-encumbered code isolated into “non-free” module (SIFT, SURF, some of the Face Detectors, etc.)

Available on all major platforms
- Android, iOS, Linux, Mac OS X, Windows

Written primarily in C++
- Bindings available for Python, Java, even MATLAB (in 3.0).

Well documented at http://docs.opencv.org

Source available at https://github.com/Itseez/opencv
OpenCV

**Image Processing**

- Filters
- Transformations
- Edges, contours
- Robust features
- Segmentation

**Video, Stereo, 3D**

- Calibration
- Pose estimation
- Optical Flow
- Detection and recognition
- Depth
Load an image from the disk, display it on our screen, and write it to file in a different format.
OpenCV: Pixel

- **Grayscale**: each pixel has a value between 0 (black) and 255 (white). Values between 0 and 255 are varying shades of gray.
OpenCV: Pixel

- **Grayscale**: each pixel has a value between 0 (black) and 255 (white)
  - values between 0 and 255 are varying shades of gray.

- **Color**: pixels are normally represented in the RGB color space
  - one value for the Red component, one for Green, and one for Blue,
  - each of the three colors is represented by an integer in the range 0 to 255,
  - how “much” of the color there is.
OpenCV: Coordinate System

- The point (0, 0) corresponds to the upper left corner of the image
- x value increases as we move to the right
- y value increases as we move down
OpenCV: Image Representation

- OpenCV represents images as NumPy arrays (matrices).
- NumPy is a library for the Python programming language that provides support for large, multi-dimensional arrays.
- To access a pixel value, we need to supply the $x$ and $y$ coordinates of the pixel.
- OpenCV actually stores RGB values in the order of Blue, Green, and Red.
Images

How to input or output an image?
How to input or output an image?

**Load Image**
Read image from disk.

```
cv::imread(filename, 0/1);
```

0: read as grayscale image
1: read as color image

**Save Image**
Write image to disk.

```
cv::imwrite(filename, im);
```

**Visualize Image**
Show image in a window.

```
cv::imshow(title, im);
```

*Note: if CV_32FC1, the gray value range is 0 to 1. Everything above 1 is white and everything below 0 is black.*

**Waitkey**
Waits n milliseconds for user input.

```
cv::waitkey(n);
```

*If n == -1, it waits forever.*

*Note: There must be a waitkey to show the image.*
Drawing Primitives

\[ \text{cv::line}(\text{im}, p_1, p_2, \text{color}, \text{thickness}); \]
\[ \text{cv::Point}(x, y) \]

\[ \text{cv::circle}(\text{im}, c, r, \text{color}, \text{thickness}); \]
\[ \text{CV_RGB}(r, g, b) \]
rectangle = np.zeros((300, 300), dtype = "uint8")
cv2.rectangle(rectangle, (25, 25), (275, 275), 255, -1)
Examine every pixel in the input images:

- `cv2.bitwise_and` (used in masking example): if both pixels have a value $> 0$, the output pixel is set to 255 in the output image, otherwise it is 0.

- `cv2.bitwise_or`: if either of the pixels have a value $> 0$, the output pixel is set to 255 in the output image, otherwise it is 0.

- `cv2.bitwise_xor`: same as OR, with a restriction: both pixels are not allowed to have values $> 0$.

- `cv2.bitwise_not`: pixels with a value of 255 become 0, pixels with a value of 0 become 255.
OpenCV

1. Load an image from the disk, display it on our screen, and write it to file in a different format.
2. Access and manipulate pixels.