1. Overview

Computer Vision

- Low Level: Processing and Feature Extraction
  - image denoising/ deblur
  - edge/corner detection
- Mid Level: Analyzing local structure, 3D reconstruction
- High Level: Understanding

Tasks:

- data acquisition
- image processing, feature extraction(low)
- analyze local structures and 3D reconstruct(mid-high)
- understanding(high)
- generation
- serving embodied agents

2. Classic Vision

2.1 Images as Function

An image is a function $f: R^2 \to R^M$

2.1.1 Filters

Convolution

Discrete: $h[n]=(fg)[n]=\displaystyle\sum_{m=-\infty}^{\infty}f[m]g[n-m]$ Continuous: $(fg)(x)=\displaystyle\int_{t=-\infty}^{\infty}f(t)g(x-t)dt$

2.2 Edge Detection

Criteria:

- $Precision = \frac{TP}{TP+FP}$ (TP: true positive)
- $Recall = \frac{TP}{TP+FN}$

Smoothing by a Low-Pass Filter

2.2.1 Canny Edge Detector

Edge: image intensity change significantly along one direction, and almost no change along orthogonal direction
hyperparameters:

- σ in Gaussian filter
- maxVal and minVal in hyteresis thresholding

2.2.2 Non-Maximal Suppresion (NMS)

- Bilinear Interpolation
- Simplified Version

2.2.3 Hysteresis Thresholding

Use a high threshold(maxVal) to start edge curves and a low threshold(minVal) to continue them

2.2.4 Edge Linking

Using the direction infomation and the lower threshold to grow edges

2.3 Keypoint Detection

Corner: image gradient has 2 or more dominant directions

Harris Corner Detector

1. Image derivatives
2. Square of derivatives
3. Rectangle window of Gaussian filter
4. Corner response function: $\theta = g(I_x^2)g(I_y^2) - [g(I_xI_y)]^2 + \alpha[g(I_x^2) + g(I_y^2)]^2 - t$
5. NMS
   TODO:example

Properties: equivariant with translation and rotation, but not invariant to scale

2.4 Fitting

2.4.1 Least Square Method

point: (x_i, y_i), line: $ax + by = d$
find (a, b, d) to minimize $E = \sum_{i=1}^{n}(ax_i + by_i - d)^2$
$Ah = 0$, A:data, h:model parameters

2.4.2 RANSAC

Idea: find a line that has the largest supporters
RANSAC Loop:

1. Randomly select a seed group of points
2. Compute transformation from seed group
3. Find inliners
4. If inliners are sufficent, recompute LSE of all inliners

Prob. failure: $(1 - w^n)^k$, w: frac of inliners, n: points needed to define hypo, k: samples num

- Pros:
  - General, suited for a wide range of line fitting problems
  - Easy to complement and calculate failure rate
- Cons:
  - Only handles moderate rate of outliners

2.4.3 Hough Transformation

RANSAC: voting in original space
Hough Transformation: voting in parameter space, can handle a high rate of outliners
TODO: add an example

3. Machine Learning

Outline:

1. Set up the task
2. Prepare the data (labeled dataset)
3. Build a model (neural network)
4. Decide fitting object (loss)
5. Perform fitting (training)
6. Testing (evaluating on test data)

3.1 Multilayer Perceptron

Idea: Stacking linear layers and nonlinear activations

3.1.1 Classification function with MLP

1. Initialization: randomly generate the weights
2. Forwarding
3. Gradient Descent: update weights

3.1.2 Backpropagation

Chain rule : *Downstream gradient* = *Upstream gradient* $\times$*Local gradient* TODO: Add hw example, matrix form

3.1.3 Activation Function

Sigmoid, tanh, **ReLU**, leaky ReLU, Maxout

3.1.4 Probelm

- Flatten an image to vector: expensive
- Flatten operation breaks local structure

3.2 Convolutional Neural Network

3.2.1 Convolution Layer

Idea: Convolve the filter with the image, *slide over the image spacially, computing dot product Size(example):

- input: 32x32(pixel), 3(channels)
- filter: 5x5(pixel), 3(channels)
- convolve with 6 filters, output: 28x28x6

Considering Stride and Padding: $(N+2P-F)/S+1$

Parameters(example):

- 10 filters, 5x5x3 (1 for bias)
- 760 parameters

3.2.2 Pooling

Idea: downsampling, makes representation smaller

3.2.3 Comparison: MLP and CNN

1. Parameters:

- Input: $W_1 \times H_1 \times C$; output: W_2, H_2, K
- FC: $W_1 H_1 W_2 H_2 C K$
- CNN: $F^2 C K$ (and K bias)
  - Sparse connectivity and parameter sharing
    Teoplitz Matrix for Convolution

2. Expressiveness

- FC
  - a superset of CNN, so should be more expressive
  - But changes dramatically to small shift and rotation, leading to optimazation problem
- CNN:
  - Parameter sharing = Equivariance with Translation
  - Conv + Pooling: invariance to small rotation and translation

3.3 CNN Training

Mini Batch SGD Loop:

- Sample a batch of data
- Forwardprop through the gragh, get loss
- Backprop, get gradient
- Update parameters using the gradient

3.3.1 Data preparation

zero-centered and normalized
input always positive/negative: zigzag path
After norm: less sensitive to small changes in weights, easier to optimize

3.3.2 Weight initialization

1. Small random numbers: problem with deeper networks All activation tend to be 0, no learning
2. (Relatively) big random numbers(0.1->0.5) All activation saturated, no learning
3. Xavier Init: For conv layer: /sqrt(filter_size^2*input_channel) If change tanh to ReLU, activation collapses to 0(because Xavier assumes 0 mean)
4. He Init: *sqrt(2/Din)

3.3.3 Set a Loss Function

3.3.4 Start Optimization

1. Optimizer
    1. GD / SGD Problem with GD: local minima or saddle point
       Problem with SGD: very slow progress along shallow dimention, jitter along steep dimention
    2. SGD + Momentum $v_{t+1} = \rho v_t + \nabla f(x_t)$
       $x_{t+1} = x_t - \alpha v_{t+1}$
       ρ gives friction: typically 0.9
    3. Adam Momentum + Bias_Correction + AdaGrad/RMSProp
2. Learning rate
    1. Too low: undershoot; Too high: overshoot
    2. LR Schedule: high at beginning, decay latter Try Cosine schedule, less hyper para
    3. Batch size increased by N, also scale initial LR by N

3.4 Underfitting and Overfitting

3.4.1 Underfitting

Usually caused by limited model capacity or unsatisfying optimization

1. Batch Norm
    1. Train Mode: Learnable scale and shift γ, β
    2. Test Mode: Becomes a linear operator
    3. Pros and Cons:
       smooths the loss landscape
       behaves differently between training and testing, buggy, and would be random if training batch is small --> Layer Norm, Instance Norm, Group Norm

2. Skip Link
    1. Deep Network: Optimization problem
    2. Residual Link: provide bypath for gradient bp
    3. Promote flatter minimizers

3.4.2 Overfitting

Generazation Gap: Usually caused by imbalance between data and model

1. Data Augmentation
   Apply changes to data with label unchanged
   Position aug: scaling, chopping, flipping... Color aug: brightness, contrast...
2. Regularization
   Push against fitting data too well
   $L(W) = \frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), y_i) + \lambda R(W)$, data loss + regularization
   Common: L1, L2 regularization
   BatchNorm as Regularization, may not need dropout
3. Dropout
   Each forward pass, randomly set some neurons to 0
   At test time, all neurons activated

3.5 Image Classification

3.5.1 Nearest Neighbor Classifier

1. Non-parameteric
2. Distance metric: L1(Manhattan), L2(Eucidean)
3. NN & KNN
4. Problems
    1. Pixel distance: too sensitive to little changes
    2. Very slow at test time

3.5.2 CNN Classifier

parameteric method
For image classification, the most widely used paradigm is **Softmax classifier and Cross Entropy Loss**

1. Network Structure Softmax classifier(multinominal logistic regression): want to interpret raw scores to probabilities
   a generalization of logistic function to multiple dimentions, also a generalization of Sigmoid
2. Loss Function
    1. Negative Log-likelihood Loss(NLL)
    2. Distance between 2 distributions: KL Divergence
    3. From KL divergence to Cross Entropy
       $D_{KL}(P||Q) = H(P) - H(P, Q)$, P is ground truth distribution, so H(p) is a constant

   Cross Entroy Loss: $L_{CE} = H(P,Q) = -\sum P(x)log\frac{P(x)}{Q(x)}$

3. CNNs for Image Classification
    1. VGGNet
        1. Calculation of effective receptive field
        2. Small filters, deep network: fewer parameters
    2. ResNet

3.6 Segmentation

Goal: identify groups of pixels that go together

3.6.1 Grouping-based Segmentation

1. Clustering: group together similar data points and represents them with a single token
2. K-Means clustering
    1. Loop: assign and update
    2. Sensitive to initialization, Need to choose K, unsupervised
3. Summary
    1. A mid-level vision task
    2. A bottom-up approach
    3. Don't care about semantics
    4. unsupervised

3.6.2 Semantic Segmentation

1. Dense labeling problem: per-pixel classification
2. Using Fully Convolution Without downsampling, too expensive at original resolution
3. Auto-Encoder
    1. Information bottleneck: Reducing redundant information via dimention reduction
    2. Non-Learnable Upsampling: Unpooling "Nearest Neighbor", "Bed of Nails"(Max Unpooling)
    3. Learnable Upsampling: Transpose Convolution
    4. Advantage of bottleneck
        1. Lower memory cost
        2. Larger receptive field, better global context
4. UNet Structure
5. Summary
    1. A top-down approach
    2. Bottleneck structure
    3. Skip link
        1. Assist final segmentation
        2. Avoid memorizing

3.6.3 Evaluation Metrics

1. Pixel Accuracy
2. Intersection over Union(IoU)

1. IoU = $\frac{target \cap prediction}{target \cup prediction}$
2. IoU Loss: $L_{IoU} = 1 - IoU$
3. mIoU: get a mean IoU over all classes

4. 3D Vision

4.1 Camera Model

1. Pinhole Camera
    1. aperture too big: blur
    2. aperture too small: less light passes through
2. Lens Camera
    1. lens focuses light on the film
    2. center & focal point
    3. in focus & out of focus
    4. Issues: radial distortion

4.2 Camera Intrinsics

1. Offset $(x, y, z) \to (f\frac{x}{z} + c_x, f\frac{y}{z} + c_y)$
2. From metric to pixel $(x, y, z) \to (\alpha\frac{x}{z} + c_x, \beta\frac{y}{z} + c_y)$, $\alpha = fk, \beta = fl$, k,l(pixel/m)
3. Homogeneous Coordinate System
    1. $E \to H$: $(x, y) \to (x, y, 1)$
    2. $H \to E$: $(x, y, z) \to (x/w, y/w)$
    3. $(\alpha\frac{x}{z} + c_x, \beta\frac{y}{z} + c_y) \to (\alpha x + c_x z, \beta y + c_y z, z)$
    4. Projective Transformation $(\alpha x + c_x z, \beta y + c_y z, z) \to [\alpha \quad 0 \quad c_x \quad 0\,0 \quad \beta \quad c_y \quad 0\,0 \quad 0 \quad 1 \quad 0][x\ y\ z\ 1\ ]$
    5. $P' = MP = K[I, 0]P$, K: camera matrix
    6. Camera skewness: 5 degrees of freedom

4.3 Camera Extrinsics

1. 3D Translation $T = [T_x\ T_y\ T_z\ ], P' \to [R \quad 0\,0 \quad 1\ ]_{4\times4}[x\ y\ z\ 1\ ]$
2. 3D Rotation $R = R_x(\alpha)R_y(\beta)R_z(\gamma), P' \to [I \quad T\ 0 \quad 1\ ]_{4\times4}[x\ y\ z\ 1\ ]$
3. 3D Translation and Rotation $P' \to [R \quad T\ 0 \quad 1\ ]_{4\times4}[x\ y\ z\ 1\ ]$
4. World Reference System $P' = K[R, T]P_w$
5. One-point perspective & Weak perspective projection
    1. Weak perspective: simpler math, acurate when object is small and distant
    2. Pinhole perpective is more accurate for modeling 3D-to-2D mapping

4.4 Camera Calibration

Goal of calibration: estimate camera intrinsics and extrinsics from one or multiple images

4.5 Representations

4.5.1 Depth Image

1. Stero Sensers
    1. A single value channel filled by depth value, 2.5D
    2. Machanism: estimate correspondence, compute disparity and turn it into depth
    3. Advantages:
        1. Robust to the illumination of direct sunlight
        2. Low implementation cost
    4. Disadvantage: finding correspondences is hard and erroneous
2. Structured Light
    1. belongs to active steroscopic approaches
    2. Advantage: simplify correspondence preblem
    3. Drawbacks
        1. Near field
        2. Indoor
3. Time-of-Flight Sensor(ToF)
    1. dToF(future)
    2. iToF(classic 3D imaging)

4.5.2 Voxel

1. $H \times W \times D$, can be indexed, but expensive
2. Not a surface representation

4.5.3 Mesh

Surface mesh: A piece-wise linear surface representation
Both a geometric and surface representation

- Mesh is a graph {vertices, edges}
- Faces are triangles(if triangle mesh)
- $V = v_1, v_2, ... \sub R^3$
- $E = e_1, e_2, ... \sub V \times V$
- $F = f_1, f_2, ... \sub V \times V \times V$

Data stuctures: Triangle list, indexed face set
Compute mesh geodesic distance

4.5.4 Point Cloud

1. Point Cloud
    1. N*3, irregular and orderless data
    2. A light weight geometric representation
2. Limitations
    1. Not a surface representation
    2. How to sample

3. Sampling Strategy
    1. Uniform Sampling
        1. easy to implement
        2. Issue: irregularly spaced
    2. Farthest Point Sampling(FPS)
        1. NP-hard, but has greedy approximation
        2. Fast Sample + Select
4. Distance Metric between Point Clouds
    1. Chamfer distance: sum of closest distance, insensitive to sampling
    2. Earth Mover's distance: sum of matched closest distance, sensitive to sampling

4.5.5 Implicit Field

1. Implicit Field
    1. both an implicit geometric and surface representation
    2. can convert into mesh
    3. examples: SDF, UDF, occupany network
2. Signed Distance Function(SDF)
    1. Interior, Exterior and Surface
    2. Extract Zero iso-surface
        1. Classical Solution: 2D Marching Square

4.6 3D Deep Learning

4.6.1 Point Networks

1. PointNet
    1. Problem: N orderless points, need to be N! permutation invariant
    2. Solution: construct symmetric functions by neural network
        1. Point wise MLP
        2. Local embedding, then Global Feature
    3. Advantages
        1. Light weight and fast
        2. Robust to data corruption
    4. Drawbacks
        1. No local context for each point
        2. Global feature depends on absolute coordinate. Hard to generalize to unseen scene configurations
2. PointNet++
    1. Recursively apply pointNet at local regions