Automatic 3D reconstruction via object detection and 3D transformable model matching

CS 269 Project
Junhua Mao and Lunbo Xu
UCLA
Motivation

- We have good model for object detection
  - HOG template matching
  - SIFT feature matching
  - Deformable part based model
  - Deep convolutional network
Motivation

- We also have good model to 3D reconstruct target object
  - 3D deformable surface
  - Image-Assisted Modeling from Sketches
  - 3-Sweep: Extracting Editable Objects from a Single Photo
Goal & Steps

• Built a system that can:
  o Detect the locations of the objects we want to find
  o Match the 3D transformable object model to the detected objects
  o Edit the image by rotating, scaling and translation of the objects

• Steps to build the system:
  o Get the template image of the target object
  o Extract image features of the template image
  o Detect the objects in the testing images
  o Estimate the initial 3D model of the detected objects
  o Match the transformable 3D model to the detected objects

• User input:
  o Template object image and template 3D model
Image Template

- Traditional object detection
  - Lots of training images
  - Objects: large variance

- Our system
  - One training image
  - Approximately same object

- HOG (Histogram Of Gradient)
  - Cell size: 30
  - Feature dimension: 31

- Color Histogram
  - HSV color space
  - 8 bins for Hue channel, 4 bins for saturation channel, 2 bins for value channel
  - L1 normalization
  - 64 dimension feature

Original image  HOG Template visualization
Testing

- Deformable Template Feature matching with sliding window
  - Multiple scales (20 scales)
  - Allow the small deformation of the sliding window
  - Apply non-maximal suppression to get rid of duplicate detection

Original image

HOG scale 1

HOG scale 9

HOG scale 18
Non-Maximal Suppression

Intersection-Over-Union (IoU)
More detection results
Interface of two part

- Output of detection part (input of 3D reconstruction part)
  - Original image
  - Image without target objects (apply image inprinting on original images)
  - Edge maps for detected objects
  - Distance map of every pixel to its nearest edge pixel
    - Calculate fast using distance transform
  - Texture
  - Estimated initial 3D model parameters (e.g. radius of the bottom and top circle of the cup, the orientation)
**Cup Model**

- Output of detection part (input of 3D reconstruction part)
  - Shape: \( r_0, r_1, h = 2 \times \sqrt{x^2+y^2+z^2} \)
  - Direction: \( <x, y, z> \)
  - Position: \( (x_0, y_0) \), the coordinate of the center
  - Flat Shading
  - Texture: output from detection step
    - outside wall
    - inside & bottom
• Image Size \((W \times H)\)
• Perspective Projection
• near = 1
• far = \(\tan(90^\circ - \text{fovy}/2) \times H/2\)
• put the cups' centers at the far/2 plane, i.e. \((x_0, y_0, -\text{far}/2)\)
Estimation

• Output from detection part:
  o For each cup:
    • Bounding box: (xb, yb, wb, hb);
    • Template id determine the estimated initial parameter (e.g. direction and shape for cup)
      o \(<x', y', z'>, r_0', r_1'\>
  
• Image Size \((W \times H)\)
• Draw cups at far/2 plane
• Initialization:
  \[x_0 = \frac{(xb + wb / 2 - W / 2)}{2};\]
  \[y_0 = \frac{(yb + hb / 2 - H / 2)}{2};\]
  \[<x,y,z> = \frac{<x', y', z'>}{2};\]
  \[r_0 = \frac{r_0'}{2};\]
  \[r_1 = \frac{r_1'}{2};\]
Optimization - Naive algorithm

- Enumerate each parameter:
- Try \( \text{init} + (-3\Delta, -2\Delta, -\Delta, 0, \Delta, 2\Delta, 3\Delta) \)
  - 7 parameters, 7 choices each, \( O(7^7) \)
- Project \( 32 + 16 + 8 + 8 \) representative points on the far plane
- Minimize \( \sum \| (X_i, Y_i) - (\text{its nearest edge point}) \|^2 = \sum \text{dis}[X_i][Y_i]^2 \)
  - (\text{dis}: distance map, an output of detection part)
- \( O(7^7 \times (32+16+8+8) + WH) \)
**Optimization - EM algorithm**

- EM algorithm: Expectation–maximization algorithm
- minimize $E = \Sigma \|(X_i, Y_i) - (\text{its nearest edge point})\|^2$
- $\theta = (X_0, Y_0, x, y, z, r_0, r_1)$
- $X_i = f_{X_i}(\theta)$
- $Y_i = f_{Y_i}(\theta)$
- while ($E > \text{threshold}$) {
  - ① calculate $(X_i, Y_i)$ for all $i$
  - ② fint the nearest edge point for each of $(X_i, Y_i)$
  - ③ formulize $\frac{\partial E}{\partial \theta}$
  - ④ Gradient Descent
- }
- future work!
Results
Image editing

- Translation
- Rotation
- Scaling
- Show by demo
Thank you