Grasping

The process of grasping

1. grasp synthesis
    - predict the grasp pose
2. Inverse kinematics
3. motion planning
    - get a geometrically valid path
4. control
    - ensures the motion follows the planned trajectory

Grasp Pose

1. 4-DOF grasp
   used in a top-down grasp manner. 3 translation DOF and 1 rotation DOF.
2. 6-DOF grasp
   3 translation DOF and 3 rotation DOF.

Open-Loop Grasping

- For known objects (works better)
    - annotated object grasp pose in CAD coordinate system
    - 6D object pose estimation
    - transform grasp pose to robot base coordinate system
- For unknown objects
    - directly predicting grasp pose

Math Prerequisite

1. Orthogonal Procrustes

1. problem formulation
   we have $M \in \mathbb{R}^{n \times p}$ and $N \in \mathbb{R}^{n \times p}$ and we want to find $A \in \mathbb{R}^{p \times p}$ such that $\hat{A} = argmin_{A \in \mathbb{R}^{p \times p}} ||M - NA||_F^2$ subject to $A \in SO(p)$
2. solution
    - SVD decomposition: $M^T N = UDV^T$
    - we can find the rotation matrix A by $A = VU^T$

2. ICP(Iterative Closest Point) algorithm

1. problem formulation
   given two point clouds P and Q, we want to find a rigid transformation T that minimizes the distance between corresponding points.
2. solution
    - for each iteration:
        - find the closest point pairs between P and Q
        - compute the rotation matrix R using Orthogonal Procrustes $R = argmin_{R \in SO(3)} ||(P - \bar{P}) - R * (Q - \bar{Q})||_F^2$
        - compute the translation vector $t = \bar{Q} - R * \bar{P}$

   - update $P = R * P + t$
3. Advantages

- Simple, no need for point cloud segmentation or feature extraction.
- When the initial estimate is good, it has decent accuracy and convergence.

4. Disadvantages

- High computational cost for finding the nearest corresponding points (can be reduced by downsampling dense point clouds or using small-sample matching for faster iterations).
- Only considers point-to-point distances, lacking utilization of point cloud structural information.
- Highly dependent on the accuracy of the initial estimate.

5. Variants
   - Point-to-plane ICP
   - Plane-to-plane ICP

6D Object Pose Estimation

Case1: Instance Level

1. use a neural network to predict the 6D object pose

it's straightforward to represent 3DOF translation using xyz but how to represent 3DOF rotation?

- Euler Angle: 3D
- Axis-Angle: 3D
- Quaternion: 4D
- Rotation Matrix: 9D
  which one is the most suitable for training?

Basically, we need something that has bijective mapping to $SO(3)$, and it changes continuously with the object pose.

- the first requirement is to garentee that all rotations can be represented
- the second requirement is to make training easier
  Actually, we can prove that to satisfy these two requirements, we at least need 5DOF rotation represention.

Euler Angle

- discontinuity problem
  consider rotation in x-y plane, the angle we need to predict is θ. The model output is continuous to the change in input, so when the groud truth rotation changes from 0 to 2π, the model output will also change from 0 to 2π. Then at 0 degree, the model output can be ambiguous.

Axis-Angle

- discontinuity problem
  - at $\theta = 0$, each axis represents identity rotation
  - at $\theta = \pi$, each axis and it's negation represent the same rotation, which become discontinuous

Quaternion

- discontinuity problem
  - Double coverage: q and $-q$ represent the same rotation, this will cause ambiguity in the model output.
  - Assume we constrain q to be at the upper semi-sphere, we should also just include half the equator plane. Then at the two division point on the equator, the model output will be ambiguous.

6D Representation

- Only predict the first two columns of the rotation matrix

- apply Gram-Schmidt process to orthogonalize the remaining three columns
  - Column 1 is normalized
  - Column 2 only preserves the perpendicular component to column 1 and is then normalized
  - Column 3 is determined by cross product of column 1&2
- problematic when design loss function(supervision), because different coloumn in the rotation matrix have different importance(because of Gram-Schmidt process).

9D Representation

- apply singular value decomposition (SVD) orthogonalization to map the Euclidean output of neural networks to $SO(3)$:

$$\mathbf{R} = \mathbf{U}\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & det(UV)\end{bmatrix}\mathbf{V}^T$$

NOTE: The 9D representation and 6D representation performs consistently better than other representations. But we can choose other representations based on the specific application. For example, if we need to predict $\delta\theta$, then the rotation will be close to identity, so quaternion representation works fine; if the expected output contains all possible rotations, then we should use 9D or 6D representation.

Refinement

we can use ICP(Iterative Closest Point) algorithm to refine the predicted object pose. Actually, the model has given a rough but reasonable pose estimation, so we can use ICP to heuristically refine the pose.

2. rotation fitting

Predict object coordinate or correspondence and then solve rotation：

- Prepare the training data:
  - transform the ground truth CAD model using R and T
  - render the object. The color for each vertex should be coordinate in the CAD coordinate system.
  - then we have the ground truth CAD coordinate of each pixel.
- Training a model to map pixel on RGB image to 3D coordinate on CAD model
  - supervision is the distance between predicted and ground truth 3D coordinate(in CAD coordinate system) of each pixel
- fit the rotation based on the corresponding
  - this step does not need neural network, we actually have analytic solution for this problem.

how to solve the rotation?

1. problem formulation
   we have $M \in \mathbb{R}^{n\times p}$ and $N \in \mathbb{R}^{n\times p}$ and we want to find $A \in \mathbb{R}^{p\times p}$ such that $\hat{A} = argmin_{A\in\mathbb{R}^{p\times p}}||M - NA||_F^2$ subject to $A \in SO(p)$
2. solution
   - SVD decomposition: $M^T N = UDV^T$
   - we can find the rotation matrix A by $A = VU^T$
3. refinement
   - SVD is sensitive to outliers, so we can use RANSAC to refine the solution

3. Limitations

• Instance-specific
• Can’t generalize to novel objects
• Needs CAD model

Case2: Category Level

NOCS (Normalized Object Coordinate Space)

1. (rotation normalization) align object orientations: for example, for all the mark cups, we define that the cup opening faces the top, and the handle faces the left is the regular orientation.
2. (translation normalization): zero-center the objects
3. (scale normalization): uniformly normalize the scales (to [-0.5, 0.5]x[-0.5, 0.5]x[-0.5, 0.5])

Method

1. Using network to predict NOCS map from RGB image
2. Depth is backprojected to point cloud
3. Every points in point cloud exactly correspond to a pixel in RGB image, which is related to a certain point in NOCS map. Now we have a dense correspondence map.
4. Use umeyama algorithm to predict the rotation, translation and scale(the scale of the diagonal of NOCS coordinate frame) of the object. Now we have a 7D pose representation.
5. To predict the bounding box of the real object, we still need to predict the scale along each axis. We done this by simply find the maximum absolute value along each axis in the NOCS map and scale it by the predicted scale.
   - we don't need to use |max-min| as the range in NOCS, because the NOCS map is zero-centered. Though we don't have complete NOCS points due to partial observation, we are sure the max absolute value along each axis in NOCS can be achievd on the negative direction of the axis.

---

synthetic data

1. supervised dara
   - action label: teleoperation
   - pose/grasp label
2. unsupervised data
   - RL

try to solve domain-gap

1. domain randomization
2. use real data for co-training

Grasp Data Sythesis

Force Closure and Form Closure: What is a successful grasp?

- In physics,
  - given a set of frictional contacts acting on a body, it is in force closure if the positive span of the wrench cones is the entire wrench space
  - if a rigid body is fully immobilized by a set of rigid stationary fixtures, we say it is in form closure
- The conditions for force closure are identical to the conditions for first-order form closure if all contacts are frictionless.
- When planning a grasp by a robot hand, force closure is a good minimum requirement. Form closure is usually too strict, requiring too many contacts.
- Simply, successful grasp <= force Closure <= form closure

NOTE: If friction coefficient is 0 and a grasp configuration is in force closure, then it must be in form closure.

- mathematically speaking, the force that can be applied to a contact point without sliding will form a cone whose apex angle if $arctan(\mu)$. For simplicity, we usually sample six vectors on the surface of the cone to represent the cone, and any feasible force will be the positive span of these vectors. Assume the concact number is N, then we will have a matrix:

$$\mathbf{M} = \begin{bmatrix} f_{1x} & f_{2x} & \cdots & f_{6Nx} \\ f_{1y} & f_{2y} & \cdots & f_{6Ny} \\ f_{1z} & f_{2z} & \cdots & f_{6Nz} \\ L_{1x} & L_{2x} & \cdots & L_{6Nx} \\ L_{1y} & L_{2y} & \cdots & L_{6Ny} \\ L_{1z} & L_{2z} & \cdots & L_{6Nz} \end{bmatrix}$$

  where $f_{i(x,y,z)}$ is the force vector for a sampled contact force on the cone and $L_{i(x,y,z)}$ is the torque on the object. We will regard a grasp to be force closure if any desired force f can be represented by the positive span of the column vectors in $\mathbf{M}$.

Synthesis Process

1. Annotate real data
2. Generate synthetic data with grasp label
   i. sample grasp candidate with objects geometry
   ii. evaluate if the grasp is force closure
      - method1: evaluate them with physical simulator by changing the direction of gravity
      - method2: using mathematical representation of friction cone and force closure.
   iii. label grasp that is force closure as positive

Existing Datasets

Object Datasets: without grasp annotation

- ShapeNet
- ModelNet
- Objaverse-XL(10M+ 3D objects)

Grasp Datasets: with grasp annotation

- ACRONYM: synthetic data
- GraspNet-1Billion: real data
   i. Scan Real Object to get CAD model
   ii. Synthetic force closure grasps for each CAD model
      - sample grasp point on point cloud
      - sample in-plane rotation and gripper depth
      - under different friction coefficient
      - evaluate if the grasp is force closure
   iii. Place real objects randomly in different scene
   iv. Manually done 6D pose match
   v. project synthesised grasps to world coordinate system
   vi. Filter out grasps in syntheticed grasps that is collision free with other objects, background and so on.

Grasp Detection

visual input representations

1. Voxel Grids

- pros: explicity geometry, insensitive to depth error
- cons: limited by volume resolution, a lot of adundant information

2. Point Clouds

- pros: explicity geometry, light weight as points only exist on the surface
- cons: sensitive to depth error and sensor noise

3. RGB

- feature: implicity geometry

Metrics

- Success Rate
   - the ratio of successful grasp executions
- Percent cleared
   - the percentage of objects removed during each round

- Planning time
    - the time between receiving input and returning grasps

Grasp Detection Pipeline

1. DexGraspNet:
    1. where to grasp
        - use sparse convolutional neural network to extract the local geometric features of the scene
        - predict objectiveness and graspness
    2. palm qpos is regarded as multi-modal
        - use diffusion model to regress distribution of the palm qpos
    3. finger DOFs(θ) given palm qpos is regarded as nearly uni-modal
        - use MLP to regress θ
2. Where2Act
    1. predict affordance
    2. predict manipulation method(push, pull, etc.)

Summary of Vision-based Open-Looped Approaches

- Use vision inputs to predict
    - object pose (CAD model needed, grasp annotation needed)
    - or grasp pose (no CAD model and grasp anno.)
    - or affordance (slightly going beyond grasping)
- Motion planning to reach, heuristic to grasp/manipulate
- Limited to certain predefined manipulation (bottleneck is the heuristics)
- Usually not closed-loop, but can be closed-loop