

图像直接为每一个像素预测其在 NOCS 中的对应点 (x, y, z) ，随后将其与 RGBD 重建得到的点云信息进行配准。这里根据像素关系，可以天然形成数量相同的变换前后的坐标对，所以不再需要找到最近邻（Correspondence）。而后，我们可以直接用 Umeyama 算法（和 ICP 去除最近邻匹配的后半段类似）来重建得到 7 DoF 物体位姿 (s, R, t)

- 输入 RGBD 图像，提取 RGB 信息，使用 Mask R-CNN 获得 ROI (Region of Interest)，分割物体
- 预测每个像素的 NOCS 空间坐标 (x, y, z) ，得到 **NOCS Map**
- 将 NOCS Map 的点反投影（Back Projection）到三维空间中，得到点云数据 **q_i**
- 通过 NOCS Map 和 Depth 图像得到的点云数据，进行 Pose Fitting，利用 Umeyama 算法，计算得出物体的 7DoF 位姿（缩放 + 旋转 + 平移）。缩放系数的计算就是简单的用 NOCS Map 的各轴向长度与物体实际点云各轴向作了一个除法。而反过来计算 Bounding Box 的时候，则利用了 NOCS 建模时令物体中心处在原点从而具有的对称性，以预测出的 NOCS Map 各轴向最大绝对值乘 2 再乘缩放系数作为了 Bounding Box 的各轴向尺寸

额外引入 NCOS 而不是直接 NN 预测原始点然后结合 Depth 直接回归 6DoF 位姿的原因：

- 实验效果好
- 将问题分解为 2D → 3D 映射 + 3D → 3D 几何优化，更直观
- NOCS 方法充分利用了形状 / 几何先验，提升了对未见物体的泛化能力。

Synthetic data 合成数据：训练 NOCS 网络需要大量标注数据，但真实数据标注成本高、泛化性差，所以需要合成数据进行训练。然而存在 Sim2Real Gap，导致模型在真实世界性能下降

Mixed Reality Data：将合成前景物体叠加到真实背景上，易于获取大量带 NOCS 标签的数据。问题：合成前景与背景 分界太过明显，从而导致分割的 Mask R-CNN 学习到的经验难以应用到真实世界

Co-Training：结合图像分割领域收集的真实数据集与合成数据集来一同对 Mask R-CNN 进行 **混合训练**，但前者不参与后续的 NOCS 映射训练，只为分割提供监督信号

后续处理：对于预测得到的位姿，有时候还需要 Refinement，比如之前介绍的 ICP 算法，也可通过神经网络（合成数据训练）完成

Form Closure：纯几何约束，不依赖摩擦力，最稳固

Force Closure：依赖摩擦力，通过接触力抵抗任意 Wrench（力 + 力矩），也即可以让物体产生的任何加速度 **a** 和角加速度 **α**

Form Closure ⊆ Force Closure ⊆ Successful Grasp，反例：1. 双指尖夹纸 2. 托起

摩擦锥 (Friction Cone)：定义了在静摩擦系数 μ 下，接触点不滑动的力的方向范围（与法线夹角最大值 $\alpha = \arctan \mu$ ）

Force Closure 形式化：定义抓取矩阵（Grasp Matrix） $F = \begin{bmatrix} \mathcal{F}_1 & \cdots & \mathcal{F}_j \end{bmatrix} \in \mathbb{R}^{n \times j}$ ， $n = 3$ 或 6 ， $j = k \times C$

其中， C 是接触点（摩擦锥）的数量， k 是为了近似每个摩擦锥所使用的力旋量数量（也即用多少面体锥来近似摩擦锥）。

- 抓取矩阵必须满秩： $\text{rank}(F) = n$
- 原点在凸锥内部（小正数下限保证稳定性）： $Fk = 0$ for some $k \in \mathbb{R}^j$, $k_i \geq \epsilon > 0$ for all i

GraspNet-1B：获得物体模型 → 在表面采样抓取位姿 → Force Closure 筛选 → 场景生成与物体位姿标注 → 变化抓取位姿到场景中，进行碰撞检测 → 多视角渲染扩增数据集。意义：证明了基于 3D 几何的模型能有效学习抓取，但大规模多样性需要合成数据。

摩擦系数 μ ：GraspNet 数据集实际上为从低到高不同的 μ 值都进行了筛选并存储了标签。 μ 值越低，对抓取的要求越高（更接近 Form Closure），这样的抓取在低摩擦表面上更可能成功。训练时可用小 μ ，泛化性强。

Grasp Detection：从输入（点云 / TSDF / 图像）预测抓取位姿，包括位置、朝向、夹爪宽度、质量分数。

输入模式：往往 3D 几何通常优于 2D 信息，因其直接包含形状信息。

VGN (Volumetric Grasping Network) 输入：3D TSDF；架构：U-Net（3D CNN）；输出：预测三个体素网格，即对于输出网格中的每一个体素，网络预测 抓取质量（Grasp Quality/Score）、抓取朝向（Grasp Orientation）、抓取宽度（Grasp Width）（防止夹爪过宽向外碰撞）

特点：**依赖几何信息**、不依赖纹理、Sim2Real 效果好（优势在于 TSDF 对传感器噪声不敏感；现实物理效应（力闭合、变形）可能优于仿真；可以工程优化摩擦系数问题），**Sim2Real 甚至可以是负的！**

评估指标：通常是与物体无关、非任务导向的 **抓取成功率 (Success Rate)**、**清理率 (Percentage Cleard)**、**规划时间 (Planning Time)**

后处理：**高斯平滑**（抓取评分非阶跃）、**距离掩膜**（限制夹爪在 TSDF 的区域）、基于质量的 **NMS**

VGN 的局限性：依赖多视角构建完整的场景 TSDF、精度受体素大小限制、高分辨率计算所需的内存成本高

GraspNet 成功的本质：

- 点云的优良性质：准确、轻量、效率高、精度高
- 架构：**端到端网络、多阶段设计**（先预测抓取分数，选高的做候选继续预测接近方向等），每个阶段都有监督信号，稳定
- 泛化性**：局部性与平移等变性
 - 局部性**：Cylinder Grouping 聚合，依赖候选点周围的局部几何信息判断，而不太关心场景中其他远处的物体。
 - 平移等变性 (Translation Equivariance)**：类似二维情形，模型学习到的几何模式识别能力不随物体在空间中的位置变化而失效。

Cylinder Grouping：在候选点附近，沿着接近方向定义一个圆柱体区域，聚合该区域内所有点的特征。这个聚合后的特征被用来预测最佳的旋转角和深度 / 宽度。

GraspNet 的核心在于 **学习局部几何特征 (Local Geometric Features) 与抓取成功的关系**，泛化性强

抓取的条件生成模型：替代检测方法，直接学习抓取位姿的分布（解决多峰分布）；尤其适用于高自由度灵巧手；常用条件扩散模型，基于局部几何特征生成抓取。

DexGraspNet：合成数据（Synthetic Data）+ 深度学习

- 场景理解：预测每个点 **抓取可能性 (Graspness)**，是否是 **物体 (Objectness)**
- 局部特征：不用全局特征（关联性弱、泛化性差），选择 Graspness 高的地方附近的点云，提取局部特征（**几何信息**）
- 条件抓取生成模块：**条件生成处理 (T, R) 多峰分布**，然后采样后直接预测手指形态 θ

问题：仅处理包覆式抓取（Power Grasp），没处理指尖抓取（Precision Grasp）；主要使用力封闭抓取；透明（Transparent）或高反光（Highly Specular/Shiny）物体有折射（Refraction）/ 镜面反射（Specular Reflection），导致点云质量差。

ASGrasp：深度修复，合成数据 + 监督学习。**域随机化**、多模态立体视觉、立体匹配（Stereo Matching）。

Affordance 可供性：指一个物体所能支持或提供的交互方式或操作可能性，哪个区域、何种方式进行交互。

Where2Act：大量随机尝试 + 标注。学习从视觉输入预测交互点 a_p 、交互方向 $R_{z|p}$ 和成功置信度 $s_{R|p}$ 。**VAT-Mart**：预测一整条操作轨迹。

利用视觉输入进行预测：

- 物体位姿 (Object Pose)**：需要模型、抓取标注。
- 抓取位姿 (Grasp Pose)**：直接预测抓取点和姿态，无模型或预定义抓取。
- 可供性 (Affordance)**

启发式 (Heuristic) 规则：预抓取 Pre-grasp，到附近安全位置再闭合，避免碰撞

- 操作复杂度有限**：难以处理复杂任务，受启发式规则设计限制。
- 开环执行 (Open-loop)**：规划一次，执行到底，闭眼做事。高频重复规划可近似闭环。

3 Policy

策略学习：学习 $\pi(a_t|s_t)$ 或 $\pi(a_t|o_t)$ ，实现 闭环控制。

Behavior Cloning：将 $D = \{(s_i, a_i^*)\}$ 视为监督学习任务，学习 $\pi_\theta(s) \approx a^*$ 。

Distribution shift：策略 π_θ 错误累积，访问训练数据中未见过的状态 $(p_\pi(s)$ 与 $P_{\text{data}}(s)$ 不匹配)，策略失效。

- 改变 $P_{\text{data}}(o_t)$** ：扩充专家数据的轨迹，使其能够覆盖策略执行过程中可能出现的状态空间。主要是要学会**纠偏**。DAgger：从（传统算法）最优解中获取；从教师策略中学习（有 **Privileged Knowledge**）
- 改变 $p_\pi(o_t)$** ：更好地去拟合专家路线，避免偏离。

Dataset Aggregation (DAgger)：训练 $\pi_t \Rightarrow$ 用 π_t 执行 (**Roll-out**) 收集新状态 \Rightarrow 查询专家在此状态下的 $a^* \Rightarrow D \leftarrow D \cup \{(s, a^*)\} \Rightarrow$ 重新训练 π_{i+1} 。但是**出错才标注，也会影响准确性**。

遥操作数据 (Teleoperation)：贵，也存在泛化问题。

非马尔可夫性：引入历史信息，但可能过拟合，**因果混淆 (Causal Confusion)**。

Multimodal behavior 多峰行为：同时存在多个可行解；NN 直接回归会出现平均行为。处理：学习分布，而不是预测单一确定性动作，如 GMM、Diffusion、VAE、AR 等。

Multi-task Learning：不单独学习每一个任务；将目标也作为条件，即**目标条件化 (Goal-conditioned)**： $\pi(a|s, g)$ ，共享数据和知识。但 g 也有分布偏移问题。

IL 局限性：依赖专家数据、无法超越专家、不适用于需要精确反馈的高度动态 / 不稳定任务。

Offline Learning：固定数据集学习，无交互。**Online Learning**：边交互边学习。

策略梯度定理： $\nabla_\theta J(\theta) = \mathbb{E}_{r \sim p_\theta(\tau)}[\nabla_\theta \log p_\theta(\tau)R(\tau)]$ ， $\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \nabla_\theta \log p_\theta(\tau^{(i)})R(\tau^{(i)})$ ， $\nabla_\theta \log p_\theta(\tau) = \sum_{t=0}^{T-1} \nabla_\theta \log p_\theta(a_t|s_t)$ ，**奖励函数无需可导**。证明： $\nabla_\theta p_\theta(\tau) = p_\theta(\tau)\nabla_\theta \log p_\theta(\tau)$ ， $p_\theta(\tau) = p(s_0)\prod_{t=0}^{T-1} \pi_\theta(a_t|s_t)p(s_{t+1}|s_t, a_t)$ ，取对数消无关。

环境模型：包括状态转移概率 $p(s_{t+1}|s_t, a_t)$ 和奖励函数 $r(s_t, a_t)$

- Model-Free**：不需要知道环境模型
- Model-Based**：利用神经网络学习环境模型

REINFORCE： $\hat{g} = \frac{1}{N} \sum_{i=1}^N \left[\left(\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t^{(i)}|s_t^{(i)}) \right) R(\tau^{(i)}) \right]$ ，按照整条轨迹的总回报 $R(\tau^{(i)})$ 加权，**On-Policy**。BC 是平权。优点： \hat{g} 无偏。缺点：高方差（训练不稳定，收敛缓慢）、样本效率低（On-policy）。

如果想让 BC 足够好（避免 Distuibution Shift）：1. 正确覆盖所有的完美轨迹，且你训练的模型能够正确地 follow 这些轨迹；2. 对各种 error 的 corner case 都有挽回来的部分覆盖，但不要有导致 error 发生的部分

On-Policy：数据来自当前策略。效果好，**样本效率低**，每次都得重新采样。**Off-Policy**：数据来自不同策略。**样本效率高**，可能不稳定。

Reward-to-Go：降方差，用未来回报 $\hat{Q}(s_t, a_t) = \sum_{t'=t}^T r_{t'}$ 加权梯度。认为一个动作只对未来的奖励负责。

Baseline：降方差，减去 a_t 无关状态基线 $b(s_t)$ ， $\hat{Q}(s_t, a_t) - b(s_t)$ 加权。梯度无偏。证明： $\mathbb{E}[\nabla_\theta \log p_\theta(\tau)b] = \int p_\theta(\tau)\nabla_\theta \log p_\theta(\tau)b d\tau = \int \nabla_\theta p_\theta(\tau)b d\tau = b\nabla_\theta \int p_\theta(\tau) d\tau = b\nabla_\theta 1 = 0$

最优基线： $b^* = \frac{\mathbb{E}[(\nabla_\theta \log p_\theta(\tau))^2 R(\tau)]}{\mathbb{E}[(\nabla_\theta \log p_\theta(\tau))^2]}$ ，证明：令 $g(\tau, b) = \nabla_\theta \log p_\theta(\tau)(R(\tau) - b)$ ， $\text{Var}[g(\tau, b)] = \mathbb{E}[g(\tau, b)^2] - (\mathbb{E}[g(\tau, b)])^2$ ， $\mathbb{E}[g(\tau, b)^2] = \mathbb{E}[(\nabla_\theta \log p_\theta(\tau))^2 (R(\tau) - b)^2]$ ，两边对 b 求导令其等于 0。

均值基线： $b = \frac{1}{N} \sum_{i=1}^N R(\tau^{(i)})$ ，使用蒙特卡洛算法，不同的 b 的选择的确会影响采样计算出的 $\nabla_\theta J(\theta)$ 近似值，但是这是由于采样不足， N 不够大造成的。

状态价值函数 $V^\pi_\theta(s_t) = \mathbb{E}_{r \sim p_\theta(\tau)} \left[\sum_{t'=t}^T \gamma^{t'-t} r_{t'} \middle| s_t \right] = \mathbb{E}_{a_t \sim \pi_\theta(\cdot|s_t)}[Q^{\pi_\theta}(s_t, a_t)]$ ：表示从状态 s_t 开始，遵循策略 π_θ 之后所能获得的期望（折扣）Reward-to-Go 回报，它只依赖于状态 s_t 和策略 π_θ 。

动作价值函数 $Q^\pi_\theta(s_t, a_t) = \mathbb{E}_{r \sim p_\theta(\tau)} \left[\sum_{t'=t}^T \gamma^{t'-t} r_{t'} \middle| s_t, a_t \right] = r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim P(\cdot|s_t, a_t)}[V^{\pi_\theta}(s_{t+1})]$ ：表示在状态 s_t 采取动作 a_t 后，再遵循策略 π_θ 所能获得的期望（折扣）Reward-to-Go 回报，它依赖于状态 s_t 、动作 a_t 和策略 π_θ 。

Advantage 基线： $A^\pi_\theta(s_t, a_t) = Q^\pi_\theta(s_t, a_t) - V^\pi_\theta(s_t) = r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim P(\cdot|s_t, a_t)}[V^{\pi_\theta}(s_{t+1})] - V^\pi_\theta(s_t)$ ，动作相对平均的优势，可替换 $R(\tau^{(i)})$ 做权重，即 $\nabla_\theta J(\theta) = \mathbb{E}_{(s_t, a_t) \sim \pi_\theta}[\nabla_\theta \log \pi_\theta(a_t|s_t)A^\pi_\theta(s_t, a_t)]$ 。估计值： $\hat{A}(s_t, a_t) = r(s_t, a_t) + \gamma \hat{V}(s_{t+1}) - \hat{V}(s_t)$ （暴力地对期望 $\mathbb{E}_{s_{t+1} \sim P(\cdot|s_t, a_t)}[V^{\pi_\theta}(s_{t+1})]$ 进行蒙特卡洛估计）

估计 $V(s_t)$ ：1. 蒙特卡洛， $\hat{V}(s_t) = \frac{1}{N} \sum_{i=1}^N \sum_{t'=t}^T \gamma^{t'-t} r(s_{t'}, a_{t'})$ ；2. 神经网络（监督学习）： $\hat{V}(s) = \hat{V}_\phi(s)$ ， $\mathcal{D} = \{(s_i, t, r(s_i, t, a_{i,t}) + \gamma \hat{V}_\phi^\pi(s_{i,t+1}))\}$ ，其中， $s_{i,t}$ 是在第 i 条轨迹、时刻 t 遇到的状态。

Bootstrap 自举：使用基于当前函数估计的值 $\hat{V}_\phi^\pi(s_{i,t+1})$ 来更新 同一个函数在另一个点 $s_{i,t}$ 的估计 $\hat{V}_\phi^\pi(s_{i,t})$

Actor-Critic：还是 $\nabla_\theta J(\theta) = \mathbb{E}_{(s_t, a_t) \sim \pi_\theta}[\nabla_\theta \log \pi_\theta(a_t|s_t)A^\pi_\theta(s_t, a_t)]$

- Actor (演员)**：指策略网络 $\pi_\theta(a_t|s_t)$ ，负责根据状态 s_t 做出动作决策，决定此步的 $r(s_t, a_t)$ 进而影响 $A(s_t, a_t)$
- Critic (评论家)**：指价值网络 $(V_\phi(s_t)$ 或者 $Q_\phi(s_t, a_t)$ ， ϕ 表示其参数)，负责评估 Actor 所处的状态 s_t 或采取的动作 a_t 的好坏（即估计 V 值或 Q 值，进而计算优势 A 值）

在训练完成后，真正推理（干活）的时候，不用 Critic，只用 Actor。

Batch AC：收集一批完整轨迹或转换数据后，统一更新 C / A 。梯度估计更稳定，但更新频率低。

Online AC：每一步交互（**极小批量**）后，立即更新 C / A 。更新快，数据利用率高，但梯度估计方差较大。

A / C 可共享网络的底层部分，增加参数效率，但训练可能更复杂，且一般效率劣于分开时。

即使在 Online AC 中，也常常收集一个小批量数据来更新 Critic \hat{V}_ϕ^π 和 Actor θ ，因为这有助于稳定学习过程，降低梯度估计的方差。

Parallelization：多 worker 采样，提速增稳。并行又可分为**同步 (Synchronous)** 和**异步 (Asynchronous)**。同步并行存在同步点，整体速度受限于最慢的 worker。异步并行则没有同步点，会更快。