认知系统研究所
慕尼黑工业大学电气与计算机工程系

用于机器人视觉感知的两种基于图像分割的物体检测器的实现与讨论

王润聪
runcong.wang@tum.de2024 年 4 月 6 日

摘要--本报告记录了两种物体检测解决方案（与 CLIP 相结合的任意分割方案和与机器人 RGB-D 摄像头点云处理相结合的 Yolact 方案）的实施情况，比较了这两种解决方案之间的差异，并讨论了基于分割的视觉感知与基于边界框的视觉感知之间的优缺点。

1 导言

视觉感知是机器人的一个重要方面。它包括物体检测、物体识别机器人与检测到的进行交互。 物体检测算法与点云处理之间的良好连接至关重要。如何高效、精确地匹配检测到的物体和相应的点云一直是个大问题。本报告介绍了将 Anything[1][2] 与 CLIP[3][4] （SAM+CLIP）和 Yolact（You Only Look At CoefficienTs）[5][6] 结合的两种物体检测器 Segment 在 ROS 环境中的应用。此外，还对 SAM+CLIP Yolact 进行了比较，并对基于分割的视觉感知（使用 SAM+CLIP 或 Yolact）和基于边界框的视觉感知进行了比较 sual感知（使用 YOLOv2）。

任何分段

Segment Anything 是 Meta 的一个项目，旨在为图像分割建立一个基础模型。该模型就是 Segment Anything Model（SAM）。SAM 是一个用于图像分割的预训练、可提示、零镜头模型，这意味着它也能很好地处理未见项目。

SAM 包括

1. 图像编码器，它使用 MAE 预训练的视觉变换器、
2. 提示编码器，包括点、方框、文本和掩码提示、
3. 掩码解码器，将图像嵌入、提示嵌入和输出标记映射到掩码中。

其结构如图 1 所示。高权重图像编码器输出图像嵌入，然后可以通过各种输入提示进行高效查询，以最快的速度实时生成对象掩码。对于涉及多个对象的模糊提示，SAM 可以输出多个有效的掩码和相关的置信度分数 [1]。

SAM 可以通过点、边界框和文本（文本提示编码器决定采用 CLIP，论文[1]中提到过，但尚未实现）进行提示。图 2 显示，SAM 可以用一个点生成所有可能的遮罩。它还可以用两个点来指定要分割。的项目图 3 显示，SAM 还可以使用边界框作为提示。此外，它还可以灵活地结合点和边界框提示来指定对象。

无论是在输入端还是输出端，SAM 都具有可扩展性。SAM 可以接收来自其他系统的提示，例如来自其他物体检测器的边界框。它生成的掩码也可用于其他系统。总之，SAM 是一种灵活、可提示的零点模型。值得我们去探索它的应用。

Yolact

YOLACT（You Only Look At CoefficienTs）是一种快速的单级实例分割模型。与 Mask R-CNN 和 FCIS 等其他方法不同，Yolact 以 SSD 和 YOLO 级别的速度执行实例分割。图 4 显示了 COCO 上各种实例分割方法（包括 Yolact）的速度-性能权衡[5]。Yolact 是速度最快的实例分割模型，其 mAP 也不低。与 SAM 不同，Yolact 表示

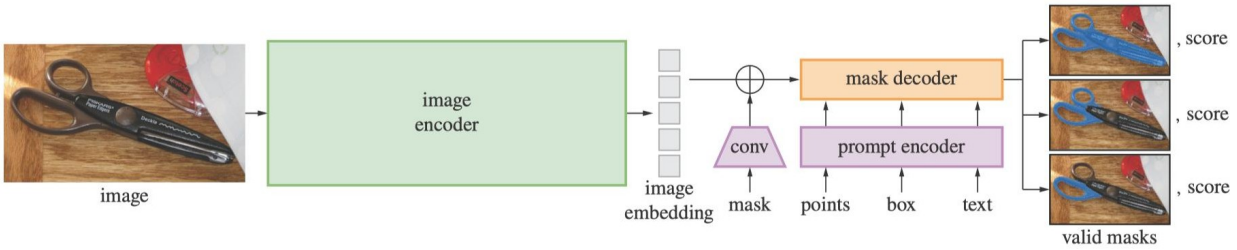

图 1 分段内容模型（SAM）概览。

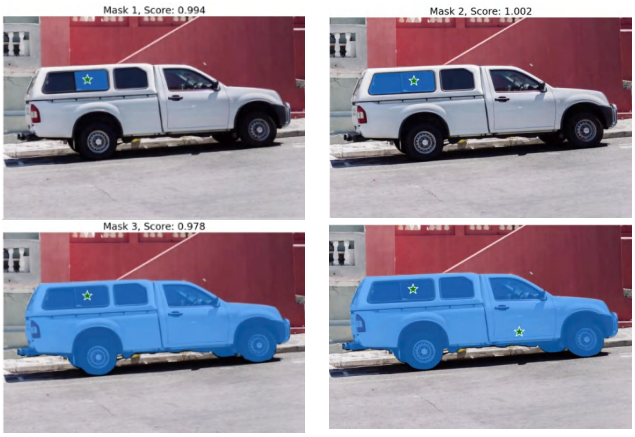

图 2 SAM 点提示

图 3 SAM 边框提示以及边框和点的组合提示

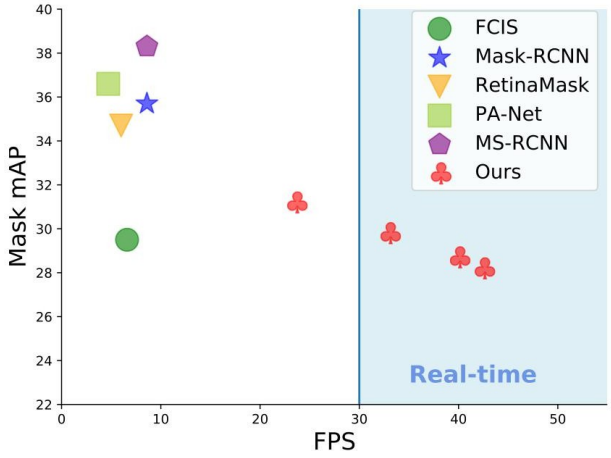

图 4 COCO 上各种状态分割方法的速度-性能权衡。

传统 "图像分割模型是完全受监督的，灵活性较差。

2 相关作品

近年来，物体识别算法发展迅速。许多算法已经被应用到机器人的视觉感知部分。Robocup@Home 机器人大赛是一个专注于家用机器人领域的机器人大赛，吸引了众多机器人团队参加。他们使用的物体识别算法是非常有意义和代表性的。

Hibikino-Musashi 队是比赛中最著名、最出色的队伍之一，在过去的比赛中应用了 GoogLeNet、YOLO、YolactEdge 等技术。

[7][8][9][10][11]。CATIE 团队除了 YOLO 或 YOLOv8 之外，还选择了定制掩膜（Custom maskrcnn）[12][13][][1415]。单发探测器（SSD）也很流行[16][17]。其中一些解决方案有点过时，如 SSD，其性能与 YOLOv2 相似[18]。有些团队也对 YOLOv5 等之前的 YOLO 模型感到满意[19]。但总的来说，YOLO 系列的物体检测器非常普及。除 YolactEdge 外，上述对象检测器都提供了边界框来确定对象的位置。目前仍有少数团队在使用提供物品分段的物体检测器。TRIALS 小组 [20] 使用的是 Mask R-CNN。homer@Unikoblenz 团队提到他们使用了一种基于 SegNet 的像素语义分割方法，名为 HomeNet [21]，但他们在 2021 年改用了 YOLOv3 [22]。

我们可以看到，基于分割的物体检测器仍然是次要的，较少应用于机器人。基于边界框的物体检测器仍占大多数。

3 方法

本节将解释如何在 ROS 中实现 SAM+CLIP 和 Yolact 的技术细节。SAM+CLIP 和 Yolact 的流程相似。对于 SAM+CLIP，首先要将 Segment Any- thing 与 CLIP 串行连接。然后，它应能在 ROS 环境中使用。对于 Yolact，由于 Yolact 本身可以识别物品，只要将有用的信息以 ROS 消息的形式发布，对点云处理部分稍作修改即可完成任务。

3.1 SAM+CLIP

3.1.1 SAM 和 CLIP 的连接

该连接的基本流程是，首先在 SAM 模型中处理来自相机的图像，然后 SAM 模型将返回二进制项目掩码，其大小与原始图像及其边界框相同。项目将根据二进制掩码和边界框进行裁剪。图 5 是相机拍摄的原始图像帧。图 6 是由 SAM 自动分割的原始图像。图 7 显示部分分割。

图 5 相机拍摄的 RGB 原始图像

CLIP 将根据给定自然语言的所有这些掩码的得分对片段进行排序，并选择得分最高的片段。图 8 展示了给定自然语言 "咖啡 "的所有候选掩码，这些掩码的得分都高于一定的阈值，而 "咖啡 "中的 "咖啡 "则高于阈值。

图 6 对 RGB 图像进行分割

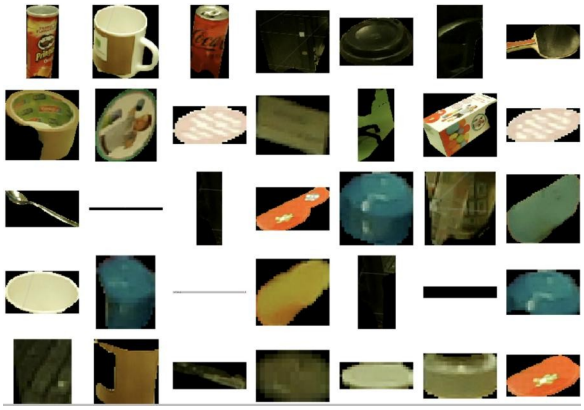

图 7 裁剪过的项目（部分）

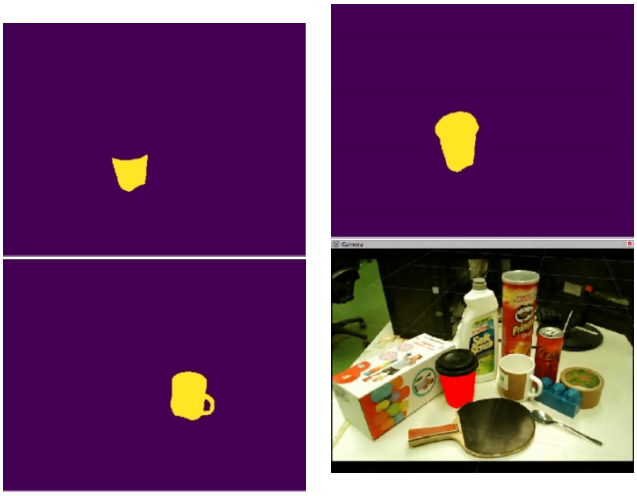

图 8 合格候选人和最终结果

最终结果是得分最高的掩码。然而，在实际操作中，得分最高的片段并不总是正确的，这一点将在后面讨论。

如果不需要使用 CLIP 模型，只需要 SAM 的结果就足够了，则可以跳过 CLIP 部分，直接使用 SAM 的分段结果。

SAM 可以通过 ROS 媒介直接发布。

流程图如图 9 所示。SAM 从相机获取原始 RGB 图像并为 CLIP 生成遮罩，然后 CLIP 根据文本提示对所有遮罩进行排序，并选择得分最高的遮罩进行进一步处理。如果不需要 CLIP，则可以不加选择地传输所有遮罩，以构建所有物品的点云。

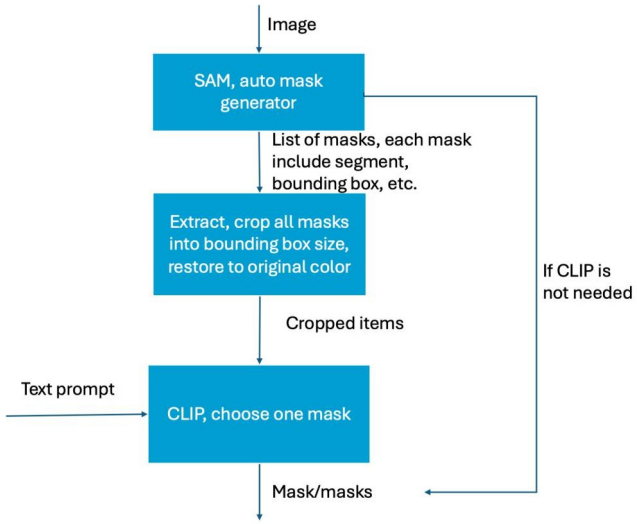

图 9 SAM+CLIP 流程图

3.1.2 将 SAM+CLIP 纳入 ROS

SAM+CLIP 任务在所选程序段被时完成。SAM+CLIP 必须集成到 ROS 中，使其他模块也能访问结果。定义一个独特的 ROS mes- sage 是必要的。Segment Anything 的掩码生成函数将返回一个掩码列表，其中每个掩码都是一个字典，包含有关掩码的各种数据。键值为[2]：

- 分割：面具、
- area：遮罩的面积，单位为像素、
- bbox：XYWH 格式的掩码边界框、
- prediction_iou：模型自己对掩膜质量的预测、
- point_coords：产生此掩码的采样输入点、
- 稳定性分数：衡量掩码质量的附加指标、
- crop_box：用于生成此掩码的图像裁剪，格式为 XYWH。

因此，可以在此基础上创建 ROS 消息。为了实现与 Segment Anything 中的掩码样式类似方式（即掩码列表），我们不妨为单个掩码创建一个包含上述所有关键字的信息，并为这个单个掩码信息创建一个数组。，创建一个名为 "singlemask "的信息，其形式如下

- int16 maskid：当前掩码的 ID、
- int16[ ] shape：掩码的形状、
- int64[ ] 分割：二进制掩码、
- int32 area、
- int32[ ] bbox、
- float64 predicted_iou、
- float32[ ] point_coords、
- float64 稳定性分数、
- int16[ ] crop_box。

然后形成一个嵌套 "sin-glemask "的名为 "maskID "的信息：

- singlemask[ ] maskID.

值得注意的是，将所有内容转换成消息形式有点麻烦。ROS 消息不支持 Numpy 数组，而 Numpy 数组正是 Segment Anything 返回的掩码形式。因此，原始的 Numpy 数组必须扁平化为一维列表。这就是为什么 "singlemask "信息中也包含掩码形状的原因。

3.1.3 点云处理

与点云相关的 ROS 主题在会话开始时订阅。然后存储原始点云数据。与常规的点云处理不同，无需对点进行预处理、
即过滤或降低点云的采样率。原因在于 RGB 图像和点云是像素对应的。根据分段直接对原始点云进行像素切割更为简单。

同样，掩码按摩 "maskID "也应转换为二维矩阵形式。我们在 C++ 中创建了一个自定义的数据结构 "singlemask"，它可以帮助重整单个掩码的内容。而 "singlemask "的向量则用来表示所有的掩码。

现在可以根据掩码对点云进行切割。由于 RGB 图像和点云在像素上是对应的，因此对像素进行迭代是切割点云的好方法。为了加快计算速度，利用遮罩的边界框，直接从边界框的一角开始操作，可以避免大部分不必要的迭代，因为遮罩只占整个图像的一小部分。因此，切割点云的主要过程如下：

- 遍历边界框中的点
- 检查掩码是否包含这一点
- 计算点云索引
- 将点添加到输出云中

在点云处理过程中，当掩码总数超过 5 个时，面积最大的掩码将被视为被移除。因为这个掩码通常是背景掩码，不包含有用信息。

图 10 和图 11 分别描述了来自 RGB-D 摄像机的原始点云和经过 SAM 分割后的处理点云。桌面的遮罩具有最大的分段，因此被移除。

图 10 来自 RGB-D 摄像机的原始点云

3.2 Yolact

对于集成到 ROS 的部分，现有的封装器[23]是一个不错的选择。该封装器可创建

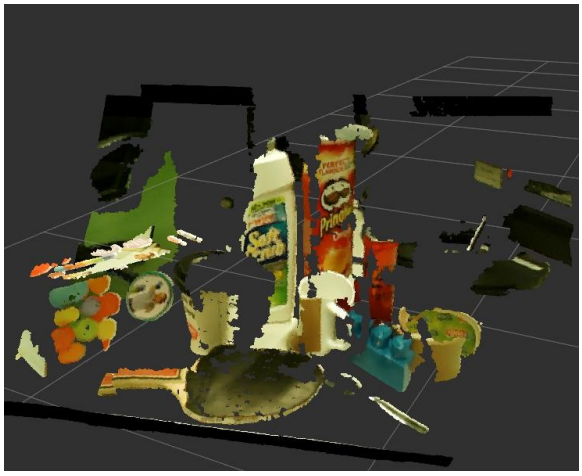

图 11 分割后的处理点云

封装器是一种 ROS 消息，用于发布 Yolact 的对象识别结果，包括类名、得分、边界框和掩码。但是，直接使用封装器还不够，因为在点云处理部分，需要与原始图像形状相同的二进制掩码，而封装器会将打包、裁剪成边界框大小的列表作为掩码发布。因此，对原始包装器进行了修改，现在的掩码信息是原始图像形状的二进制掩码。此外，封装器还采用了 rqt_reconfigure，这样就可以轻松更改配置。在修改后的封装器中，还加入了 "所选类别 "这一附加功能，以便选择特定的目标项目。因此，自行定义的 ROS 消息和点云处理必须进行相应修改。如果选择掩码选项不是 "无"，则修改后的点云处理将检查每个掩码类别是否与所选类相同，并只处理所选类别。

图 12 显示了显示修改后的 rqt_reconfigure 界面，其中增加了 "chosen_class "选项，而图 13 则了选定类别为 "cup "的点云的可视化效果。

4 讨论

本节将讨论 SAM+CLIP 和 Yolact 的优缺点。此外，还将讨论基于分割的视觉感知与基于边界框的视觉感知之间的差异。

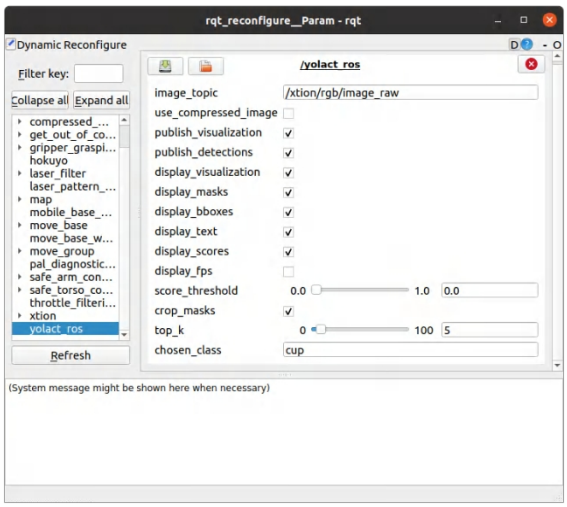

图 12 Rqt_reconfigure 工具新增 "chosen_- class "选项

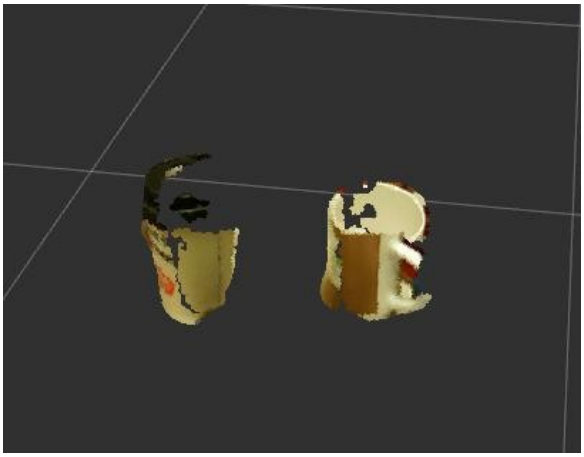

图 13 选定类别 "杯子 "的处理点云

4.1 SAM+CLIP 和 Yolact

4.1.1 磁盘/内存使用量和计算时间

SAM 的模型比其他对象保护模块更大。默认的 SAM 模型 "vit_h "占用 2.4GB 存储空间，而 "vit_l "和 "vit_b "占用分别为 1.2GB 和 358MB。这里的 CLIP 模型也需要约 500MB。不过，磁盘存储空间是最容易忽略的方面。SAM 模型 "vit_h "需要约 10GB 的 VRAM，在 RTX4090 上运行约需 8 秒，而在 GTX1080ti 上运行则需 20 秒。这表明运行该模型需要一台高性能计算机。因此，该模型无法在老式或小型计算机上运行。

相比之下，Yolact 要轻便得多。一个约 200MB 的模型就能完成令人满意的工作。此外，它只需占用 1GB VRAM，对老设备非常友好。此外，它只需要 1GB VRAM，对老设备非常友好。

计算速度也会影响点云处理的速度。裁剪后点云只有在物体检测器发送新片段后才会更新。因此，在 SAM+CLIP 生成目标片段之前，机器人可能不得不等待很长时间，以确保能正确执行抓取操作。

而 Yolact 能以 30FPS 的速度检测物体，因此点云可以更新。

4.1.2 稳定性和精确性

SAM 可以稳定地生成图像中每个物品的片段。虽然它不能只给出整个物品的一个完整片段，但它能提供所有可能的重叠解决方案。例如，如果我们对一个瓶子进行分段，分段可以是瓶子的顶部、标记、整个瓶子、瓶子以及瓶子前面的杯子，如图 14 所示。因此，段的数量远远多于项的数量。这么多的片段为 CLIP 提供了更多的选择，因此 CLIP 可以根据所提供的文本提示找到最合适的片段。

SAM 自动生成掩膜的可调参数：有几个可调参数可以控制点的采样密度，以及去除低质量或重复掩膜的阈值。此外，它还可以在图像的裁切部分生成片段，因此对于小物体也有很好的表现。不过，在实际应用中，默认配置已经能提供令人满意的结果。例如，如果我们增加每边的采样点，就会生成更多的片段，而这些片段可能是多余的，会增加计算时间。如果我们减少每侧的采样点，减少的片段数量可能无法覆盖整个项目。图 15 显示，减少分段后，项目更加零碎，尤其是图像左侧的方框。尽管如此，如果想获得完美的定制效果，对参数进行微调并没有坏处。可调整参数的完整列表和相应的参数说明见附录。

另一方面，Yolact 无法生成稳定的片段，除了概率超高的项目。概率较低的项目会在 Yolcat 的标记和未标记之间切换。图 16 展示了 Yolact 两种检测框架的比较。从图中可以看出，一些概率较高的项目

图 14 瓶子的所有可能分段

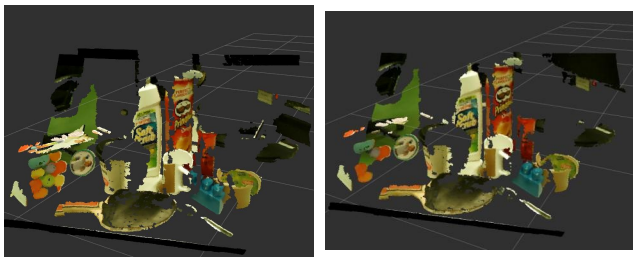

图 15 经过去故障配置（左）和更严格配置（右）后的裁剪点云对比

可能出现的情况仍然被标记，比如两个杯子和一个勺子。但其他项目，如桌子和 "书"（应为盒子），则会来回切换。

与 SAM 不同，Yolact 对项目分割并不那么精确。我们以图 16 中的桌子为例。一旦标记了桌子，它的分段就几乎覆盖了桌子上的所有东西，甚至地面的某些部分也包括在内，而 SAM 可以更细致地对桌子进行分段。如果指定桌子为所选类、

图 16 Yolact 的两帧检测对比

Yolact 会返回包括项目在内的整个表格，而 SAM+CLIP 则会返回带有许多孔洞的表格，因为孔洞应该是项目，而不是表格的一部分。这可能会导致点云截取不准确，尤其是在项目遮挡的复杂场景中。指定类别的片段可能包含其他项目，因此相应点云的形状与项目的实际形状不符。

4.1.3 零点检测器与完全监控检测器的对比

除了分割精度，SAM+CLIP 还能对未见模式进行预测。

CLIP 在一个庞大的图像-文本对数据集上接受训练，这有助于它学习各种视觉概念及其与文本描述的关联。通过这种训练，CLIP 可以理解和解释各种文本提示，并将其作为一种语言来使用。

即使对于从未明确接受过培训的班级来说，也可以用相关的视觉内容对他们进行引导。在图 5 的图像中，文字提示 "可乐"、"饮料"，甚至句子 "我渴了 "可能指的是同一个可乐罐，而 "杯子"、"马克杯"、"咖啡 "可能指的是同一个马克杯。

相比之下，Yolact 是在一个特定的数据集上进行训练的，该数据集具有预定义的类别。它学会根据这些类别来检测和分隔物体，其识别物体的能力仅限于它所训练的类别。在图 16 的例子中，桌子必须是。如果选择的类别是其他类型的桌子，就不会收到反馈。书"（应为盒子）也表明，即使片段是正确的，但错误的分类也会导致难以正确拾取物品。此外，如图 13 所示，当选择的类别是 "杯子 "时，Yolact 本身无法进一步区分这两个杯子。

不过，CLIP 也有其局限性。如图 17 所示，CLIP 可能会选择一个错误但高度相关的分段，而不是正确的分段。

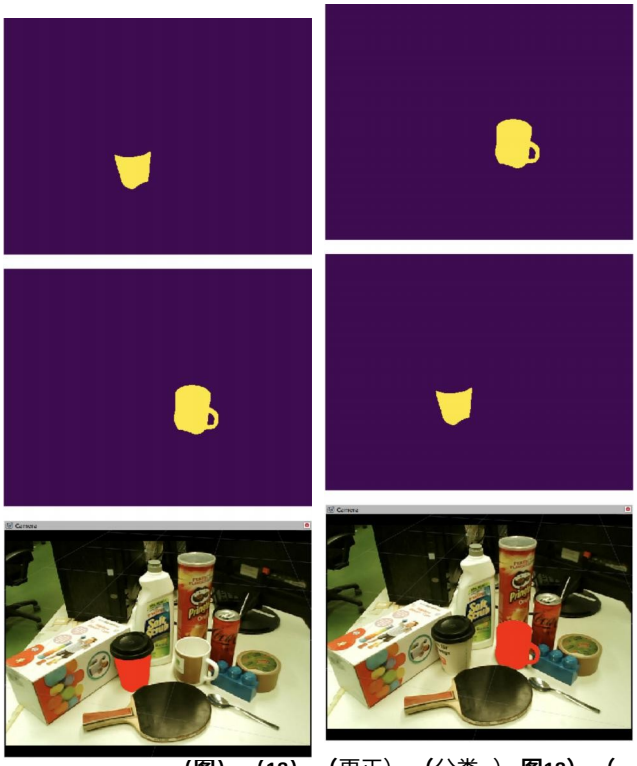

图 17 错误的分类 **（图） （18）** （更正） （分类--） **图18） （**更正） 分类--） **(图19**

文字提示为 "杯子 "的提示

用文字提示进行分类 "带柄马克杯"

最上面的面具是得分最高的。但第二个掩码应该是正确答案。减轻这种限制的方法之一是，可以去掉

如图 18 所示，CLIP 可以更详细地刻划项目，并帮助 CLIP 锁定正确的线段。现在，两个相关片段的排序已经颠倒，可以选择正确的片段了。这正是 Yolact 的优势所在，因为 Yolact 无法进一步分离两个杯子。

4.2 基于分割和边界框的视觉感知之间的差异

基于分割的视觉感知可以直接对点云进行像素处理，而基于边框的视觉感知则不同，它通常需要一系列更复杂的操作。由于检测器只能给出物品的边界框，因此很难将相应的点云与类别相匹配。在处理点云时，可以

- 对点云进行降采样和过滤、
- 使用 RANSAC 对表格和物品的点云进行分离和聚类、
- 计算聚类分离实体的中心点、
- 利用相机固有矩阵，将中心点投影到二维图像平面上、
- 比较集群的中心点和边界框、的中心点
- 将集群分配到最接近的边界框中。
- 聚类中的每个点都根据其相关边界框的类别进行标注，而二分法则将类别名称映射到特定的标签或 ID 上。

回顾基于分割的点云处理过程

- 遍历边界框中的点
- 检查掩码是否包含这一点
- 计算点云索引
- 将点添加到输出云中。

基于边界框的视觉感知要复杂得多。如图 19 所示，由于下采样和滤波，点云的分辨率也没有基于分割的视觉感知高。这可能会导致细节丢失。

此外，在许多物品遮挡的复杂场景中，要想

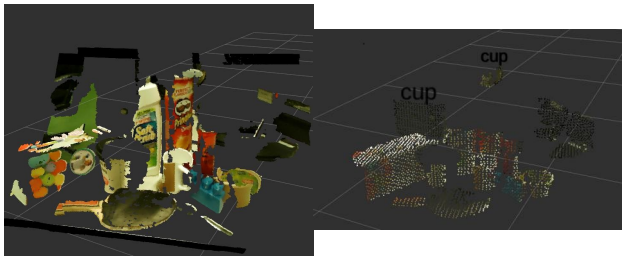

图 19 点云分辨率对比（左：SAM 后的点云，右：YOLOv2 后的点云）

RANSAC 来寻找解决方案。而将边界框与聚类中心点相联系的方法可能会造成误差。

此外，由于点云是以数字的方式进行标注的，因此在类别和数字之间建立映射关系也很重要，这就使其缺乏灵活性，更难以检测到未预先定义的对象，因为 YOLOv2 是一种完全受监督的检测器，点云数字与类别之间的映射关系也必须预先定义。

5 结论

与 Yolact 相比，Segment Anything 在精确度和零镜头方面具有巨大优势，而 Yolact 在实时图像分割方面表现更好。

基于分割的物体检测器在点云处理方面具有精度高、操作简单等优点，但这些优点的代价是计算量大。基于边框的视觉感知在处理点云时需要更多繁琐的步骤，但如今基于边框的视觉感知仍占主要地位。有必要开展进一步研究，探讨基于分割的视觉感知为何不那么普遍。

参考资料

[1] A.Kirillov、E. Mintun、N. Ravi、H. Mao、C. Rol-land、L. Gustafson、T. Xiao、S. Whitehead、A. C. Berg、W.-Y.Lo, P. Dollár, and R. Girshick, "Seg- ment anything," *arXiv:2304.02643*, 2023.

[2] "Segment Anything Github." 访问日期：2024- 03-22 https://github.com/facebookresearch/segment- 任何东西。

[3] A.Radford, J. W. Kim, C. Hallacy, A. Ramesh、G. Goh, S. Agarwal, G. Sastry, A. Askell、P.Mishkin、 J. Clark、 G. Krueger、 和 I.Sutskever, "Learning transferable visual mod- els from natural language supervision," 2021.

[4] "CLIP Github 。 " 访 问 时 间 ： 2024-03-29 https://github.com/openai/CLIP。

[5] D.Bolya, C. Zhou, F. Xiao, and Y. J. Lee, "YOLACT: Real-Time Instance Segmentation," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, (Seoul, Korea (South)), pp.

[6] "Yolact Github。"访问时间：2024-03-29 https://github.com/dbolya/yolact。

[7] S.S. Hori, Y. Ishida, Y. Kiyama, Y. Tanaka、Y.Kuroda, M. Hisano, Y. Imamura, T. Hi- maki, Y. Yoshimoto, Y. Aratani, K. Hashimoto、G. Iwamoto, T. Morie, and H. Tamukoh, "Hibikino-Musashi@Home 2017 Team De- scription Paper," 2017.

[8] Y.Ishida, S. Hori, Y. Tanaka, Y. Yoshi- moto, K. Hashimoto, G. Iwamoto, Y. Aratani、K.K. Yamashita, S. Ishimoto, K. Hitaka, F. Ya- maguchi, R. Miyoshi, K. Miyoshi, S. Ishimoto, K. Hitaka, F. Ya-maguchi.Miyoshi, K. Honda, Y.Honda, Y.安倍晋三 Y.Kato, T. Morie, and H. Tamukoh, "Hibikino-Musashi@Home 2018 Team Description Pa- per," Nov. 2022.评论：8页，5图，RoboCup@Home。

[9] Y.田中 Y.Ishida、 Y.安倍晋三 T.小野 K. Kabashima、 T.Sakata、 M. Fukuyado、F.Muto, T. Yoshii, D. Kamimura, K. Nakamura、Y.Nishimura, and H. Tamukoh, "Hibikino-Musashi@Home 2019 Team Description Pa- per," 2019.

[10] T.Shiba, T.Ono, S.Tokuno, I.内野 M.Okamoto, D.Kanaoka, K.高桥 K.K. Tsukamoto, Y. Tsutsumi, Y. Nakamura、Y.Fukuda, Y. Hoji, H. Amano, Y. Kub- ota, M. Koresawa, Y. Sakai, R. Takemoto、K.Tamai, K. Nakahara, H. Hayashi, S. Fuji- matsu, A. Mizutani, Y. Mizoguchi, Y. Yoshim- itu, M. Suzuka, I.Suzuka, I.M. Suzuka, I. Matsumoto, Y. Y.Yano、Y.Tanaka, T. Morie, and H. Tamukoh, "Hibikino-Musashi@Home Team Description Pa- per," Nov.评论：arXiv 管理员注：与 arXiv:2005.14451 有大量文字重叠，arXiv:2006.01233。

[11] T.Shiba, A. Mizutani, Y. Yano, T. Ono、S.Tokuno, D. Kanaoka, Y. Fukuda, H. Amano、M.Koresawa, Y. Sakai, R. Takemoto, K. Tamai、K.Nakahara, H. Hayashi, S. Fujimatsu, Y. Mi-

zoguchi, M. Anraku, M. Suzuka, L. Shen、

K.K. Maeda、F. Matsuzaki、I. Matsumoto、K. Mu-rai、

K. Isomoto 、 K. Minje 、 Y. Tanaka 、 T. Morie 和 H. Tamukoh，"Hibikino-Musashi@Home 2023 团队说明文件"，2023 年 10 月。

[12] R.R. Fabre, B. Albar, C. Dussieux, L. Joffroy、C.Pinet, J. Simeon, and S. Loty, "CATIE Robotics @Home 2019 Team Description Pa- per," 2019.

[13] R.R. Fabre, B. Albar, C. Dussieux, L. Jof- froy, J.-B. Horel, Z. Li, A. Perrot, C. Pinet, L. Jof- froy, J.-B.Horel, Z. Li, A. Perrot, C. Pinet、

S.Pouyet, F. Larrue, and S. Loty, "CATIE Robotics @Home 2020 Team Description Pa- per," 2020.

[14] S.Delpeuch, B. Albar, A. Chauvel, C. Laigle、S.S. Gamardes, S. Pouyet, L. Saint-Germain、F.Larrue, L. Joffroy, and S. Loty, "CATIE Robotics @Home 2023 Team Description Pa- per,"

[15] S.Delpeuch, P.-M.Ancele、T. Arsicaud、C. Dor-moy、C. Jaureguiberry、C. Pinet、J.-N.Barthas、

F.Larrue 和 S. Loty，"CATIE 机器人 @Home 2024 团队介绍文件"。

[16] P.P. T. Aquino-Junior、F. M. Aidar 和 G. N. Marostica，"RoboFEI@Home Team Descrip- tion Paper for RoboCup@Home 2023：法国 版"。

[17] M. Andries, M. Barange, M. Bouabdelli、C.Buche, D. N. Do, Y. He, L. Li, M. Neau、

T.Ung、S. Wang 和 C. Y. J. Wong，"机器人 Breizh 2023 团队说明文件"。

[18] Y.Kang, "Research on SSD base network," *IOP Conference Series：材料科学与工程*》，第 768 卷，第 072031 页，2020 年 3 月。

[19] T.Ribeiro, F. Gonçalves, M. Luís, E. Fernan- des, and R. Silva, "LAR@Home 2023 团队说明文件"。

[20] C.C. Tsuji, D. Komukai, M. Shirasaka, H. Wada、T.T. Omija, A. Horo, D. Furuta, S. Yamaguchi、S.Ikoma, S.Tsunashima, M.小林 K.Ishimoto, Y. Ikeda, T. Matsushima, Y. Iwa- sawa, and Y. Matsuo, "TRAIL Team Description Paper for RoboCup@Home 2023," Oct.

[21] R.Memmesheimer, I. Mykhalchyshyna, N. Y. Wettengel, T. Evers, L. Buchhold, P. Schmidt、

N.N. Schmidt、I. Germann、M. Mints、G. Ret-tler、C. Korbach 、 R. Bartsch 、 T. Wei-land 和 D. Paulus ，"RoboCup 2019 - homer@UniKoblenz（德国）"，2019 年。

[22] D.Müller, N.Wettengel, 和 D. Paulus、*homer@UniKoblenz: 年度最佳阵容。*

RoboCup 虚拟家庭开放平台 League 2021，第 283-290 页。01 2022.

[23] "yolact_ros Github。"访问时间：2024-03-29 https://github.com/Eruvae/yolact_ros。

任何自动屏蔽发生器可调参数的段落

model (Sam) 用于掩码预处理的 SAM 模型。

points_per_side（int 或 None） 沿图像一边采样的点数。如果为 "无"，"point_grids "必须提供明确的点采样。

points_per_batch (int) 设置模型同时运行的点数。数量越多速度越快，但会占用更多 GPU 内存。

pred_iou_thresh（float） 使用模型预测的掩膜质量，过滤阈值为 [0,1]。

stability_score_thresh（float） 一个过滤，取值范围为 [0,1]，用于衡量掩码在用于对模型的掩码预测进行二值化的截点发生变化时的稳定性。

stability_score_offset（浮点） 计算稳定性分数时移动截止值的量。

box_nms_thresh (float) 非最大抑制用于过滤重复掩码的盒式 IoU 截止值。

crop_n_layer（int） 如果 >0，将在裁剪图像时再次运行掩码预测。决定要运行的层数。

crop_nms_thresh（float） 非最大抑制使用的方框 IoU 截止值，用于过滤不同作物之间的重复掩码。

crop_overlap_ratio（浮点） 设置作物重叠的程度。重叠度会随着作物层数的增加而减少。

crop_n_points_downscale_factor (int) 缩放因子 的 的缩放因子 数量的 的缩放因子。 的缩放因子。

point_grids （list(ndarray) 或 None）用于采样的显式点网格，归一化为 [0,1]。

min_mask_region_area (int) 后处理阈值 以 删除 小 的后处理阈值。

output_mode (str) 返回掩码的形式。选项包括 "二进制掩码"、"未压缩掩码 "或 "可压缩掩码"。