6. 基于 Kafka 的旋转框自动标注平台搭建

6.1 现有开源旋转框标注软件及标注方法研究

目前最受欢迎的数据标注软件主要集中水平框标注：如 Label Studio[24]、LabelIMG[25]和 CVAT[26]。相比之下，由于旋转目标检测发展的较晚，开源旋转框标注软件的选择则局限得多，较受欢迎的有 RolabelImg 和 Labelme。

其中 RolabelImg 基于 Python 中的 PyQt 库而建立且专注于旋转框的标注，由于其安装较为容易，是目前 Github 上获得收藏数最多的旋转框标注软件。其标注一个旋转框总共可以拆分为 3 步：首先，在待标注目标上绘制一个水平框并设置目标的名称；其次，使用相应的快捷键进行方向的调整；最后，使用鼠标对旋转框进行大小和位置的调整。具体绘制过程如下图 6-1 所示。

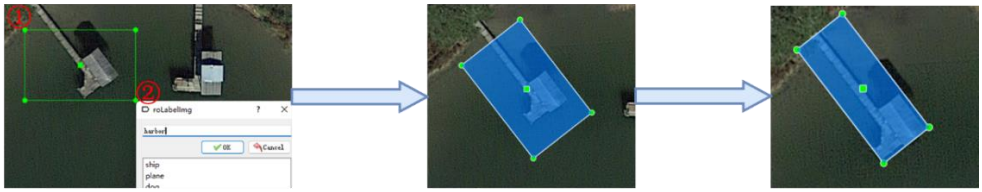

图 6-1 RolabelImg 绘制旋转框过程

与 RolabelImg 不同，LabelImg 支持适用于图像、文本、超文本、音频、视频和时间序列数据的标注，是集大成者的标注软件。对于其图像部分，LabelImg 仅支持水平框与多边形标注，不直接支持旋转框标注。但网络上有网友分享的“十字标注法”[27]亦可实现使用 LabelImg 标注旋转框。“十字标注法”主要使用的为多边形标注工具，具体标注过程同样可分为三步：首先，在待标注目标的周围交叉标注 5 个点，使标注图形呈交叉封闭状态；其次，将标注好的文件进行导出并使用最小外接矩形计算器将标注图像转为旋转框；最后，将转换后的文件再次导入 LabelImg 中进行位置的微调（无法对方向进行微调）。具体标注过程入下图所示。

图 6-2 LabelImg 绘制旋转框过程

经过上述标注过程的深入剖析，可以明显观察到当前开源的旋转框标注软件在易用性方面存在显著不足。这些软件往往操作复杂，标注方法不够直观，甚至给人一种舍近求远之感。这些问题不仅增加了标注工作的难度和耗时，更在一定程度上阻碍了旋转目标检测技术的进一步发展和应用。

6.2 旋转框绘制方法设计

为了解决以上旋转框标注方法的不足，本文基于长边定义法[28]设计了一种更为方便的旋转框标注方法。具体定义如下图所示 6-3 所示，其中 A、B 两点坐标(x_A, y_A)、(x_B, y_B)为初始时标注的坐标信息。

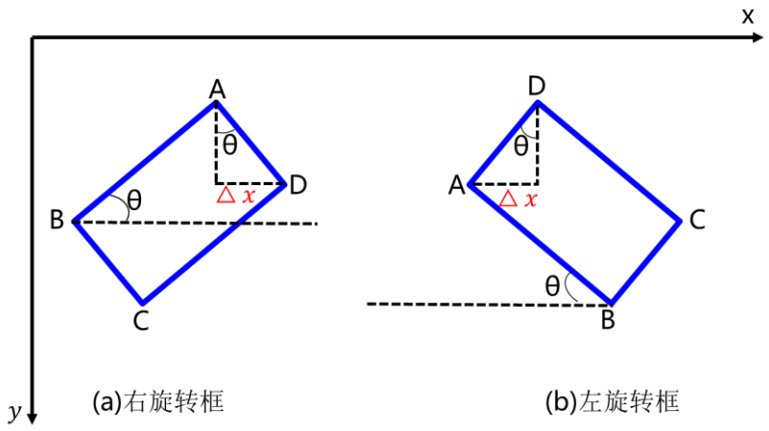

图 6-3 长边定义法构造旋转框

显然，左、右两种旋转框之间存在着对称关系。为便于阐述，本文将聚焦于右旋转框，演示如何通过已知的 A、B 两点坐标精确地计算出 C、D 两点的坐标。

首先，根据两点坐标计算出θ角的正弦值、余弦值和正切值：

$$HD = |x_2 - x_1| \tag{6-1}$$

$$VD = |y_2 - y_1| \tag{6-2}$$

$$distance = \sqrt{HD^2 + VD^2} \tag{6-3}$$

$$Sin\theta = VD/distance \tag{6-4}$$

$$Cos\theta = HD/distance \tag{6-5}$$

$$Tan\theta = Sin(\theta)/Cos(\theta) \tag{6-6}$$

上述式子中，HD 为两点间水平距离、VD 为两点间垂直距离。

其次，根据角度互余的关系计算 C、D 两点相对 A、B 两点的垂直坐标变化值Δh：

$$\Delta h = \Delta x \times Tan(\theta) \tag{6-7}$$

其中Δx为初始拟定值，表示 C、D 两点相对 A、B 两点的水平坐标变化值,本文将其设定为长边的二分之一。

最后，可得出 C、D 两点的坐标(x_C, y_C)和(x_D, y_D)。

$$(x_C, y_C) = (x_B + \Delta x, y_B + \Delta h) \tag{6-8}$$

$$(x_D, y_D) = (x_A + \Delta x, y_A + \Delta h) \tag{6-9}$$

通过以上步骤，即可仅凭一条包含方向信息的长边来精确确定整个旋转框的位置和方向。

值得注意的是，旋转框的旋转是通过对中心点进行旋转来实现的，这一中心点的坐标$(x_{\text{center}}, y_{\text{center}})$可以通过以下方式计算得出：

$$(x_{center}, y_{center}) = ((x_A + x_C)/2, (y_A + y_C)/2) \tag{6-10}$$

接下来，以中心点坐标为基准，计算 4 个顶点(x_i, y_i)在旋转α度后的位置坐标。

首先，计算 4 个顶点相对于当前旋转框中心点的偏移量。

$$(offset_{ix}, offset_{iy}) = (x_i - x_{center}, y_i - y_{center}) \tag{6-11}$$

按照三角函数关系式，不难得出旋转过后四个顶点的相对横坐标x_i'表示如下：

$$x_i' = Cos(\alpha) \times offset_{ix} + Sin(\alpha) \times offset_{iy} \tag{6-12}$$

同理，旋转后四个顶点的相对纵坐标y_i'表示如下：

$$y_i' = -Sin(\alpha) \times offset_{ix} + Cos(\alpha) \times offset_{iy} \tag{6-13}$$

综上，将相对坐标与中心点坐标相加即得到相应顶点旋转过后的绝对坐标：

$$(x_i'', y_i'') = (x_i' + x_{center}, y_i' + y_{center}) \tag{6-14}$$

6.3 旋转框绘制方法演示及比较

利用上述定义设计的旋转框绘制方法，可将其绘制过程归纳为三个主要步骤。首先，根据目标的长边特性，绘制一条倾斜的直线（程序会自动补全为相应方向的旋转框）并为其选择相应的标签。其次，为了更精确地定位目标，对绘制的旋转框进行方向上的微调，以确保其与实际目标的方向一致。最后，进行旋转框大小上的微调，以更贴近目标的实际尺寸。具体绘制方法如下图所示：

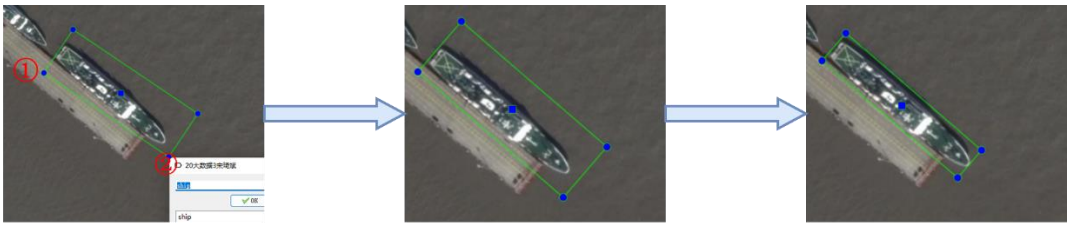

图 6-4 本文设计的旋转框绘制方法展示

根据上图的展示，可以清晰地看出，相较于 RolabelImg 和 LabelImg 的“十字标注法”，本文所设计的旋转框绘制方法更为直观且直接。该方法无需任何额外的繁琐步骤，使得标注过程更加高效且精准，为旋转目标检测任务提供了更为便捷和准确的标注手段。具体比较如下表 6-1 所示：

表 6-1 本文标注方法与其它方法的比较

标注软件名称	标点数量	方向调整幅度	大小调整幅度	借助外部软件
RolabelImg	2	较大	较大	无
LabelImg	5	无法调整	微调	最小外接矩计算
本文标注方法	**2**	**微调**	**微调**	**无**

值得一提的是，在本文设计的旋转框标注软件初稿（未集成深度学习模型与其它功能）完成后立即将其发布于 CSDN 等开源社区，以供广大研究者使用和交流。即便在未获得 CSDN 官方推荐展现的情况下，该文章[29]仍然获得了 10 个收藏与 7 个点赞。这一积极反馈

从侧面证实了本文设计的标注方法确实帮助到了部分研究者，并有效证明了本文提出的旋转框标注方法的可用性和实用性。

图 6-5 CSDN 文章后台数据展示

6.4 标注平台搭建

6.4.1 数据传输模块搭建

本文依托分布式 Kafka 系统，实现了数据在前后端之间的实时传输功能。作为一个核心的消息系统，Kafka 利用 ZooKeeper[30]来进行集群的协调和元数据的管理，两者服务器启动如下图 6-6 所示。在处理流式数据时，Kafka 系统展现出显著的优势：首先，它成功构建了实时数据流通道，该通道确保了系统与应用程序之间数据的可靠存储；其次，Kafka 系统支持实时数据流应用程序的构建，并能对实时数据流进行高效的转换与应用[31]。

(a)启动Zookeeper服务器

(b)启动Kafka服务器

图 6-6 启动 Zookeeper 和 Kafka 服务器

本文选用 Kafka 的原因是其能够解决大量实时数据流的通道传输。具体的，Kafka 允许多个生产者进程并行地在多个服务器节点上执行图片目标的实时推理任务，最终将产生的目标位置、目标类别等和图片地址等信息组合同步发送至 Kafka 集群。随后，消费者进程从集群中拉取算法推理的结果，进行进一步的将其整理、分析与保存，并将其实时的显示在标注平台中。由于图片的推理需要利用 GPU 等硬件，本文仅将生产者布置在单节点上。具体数据传输示意图如下图所示。