# 基于最大生成树的无序图像拼接算法

余涛(122106222833)谢宝烨(122106222828)崔志龙(122106222829)

南京理工大学 计算机科学与工程学院

**摘要：** 图像拼接是计算机视觉中的重要任务，用于将多个图像合成为一个更大的图像。在许多应用中，如全景摄影、虚拟现实和卫星图像拼接，图像拼接技术发挥着关键作用。然而，当输入图像是无序的时，即没有给定它们的正确顺序，图像拼接任务变得更加具有挑战性。为了克服这个问题，本报告提出了一种基于最大生成树的无序图像拼接算法，以实现高质量的图像拼接。在拼接过程中，我们还引入了非极大性抑制方法来筛选特征点，以提高拼接的质量和减少冗余的特征提取过程。

我们的算法利用最大生成树的概念来解决无序图像拼接的排序问题。最大生成树是一个连接所有图像的图结构，其中图像之间的相似度或连续性被用作边的权重。通过构建这个图并找到其中的最大生成树，我们可以得到图像之间的最佳排序。这个排序可以指导后续的拼接步骤，确保图像之间的连续性和一致性。

**关键字：**图像拼接；非极大抑制；最大生成树

## 1. 引言

图像拼接是计算机视觉中的重要任务，旨在将多个图像合成为一个更大的图像。它在许多领域中具有广泛的应用，包括全景图拼接、虚拟现实、医学影像处理等。传统的图像拼接算法通常要求输入图像是有序的，这对于实际应用中无序的输入图像来说是一个挑战。为了克服这个问题，本报告提出了一种基于最大生成树的无序图像拼接算法，旨在自动对无序输入图像进行排序和拼接，以实现高质量的图像拼接结果。

在我们的算法中，最大生成树被引入用于解决无序图像拼接的排序问题。最大生成树是一个连接所有图像的图结构，其中图像之间的相似度或连续性被用作边的权重。通过计算图像之间的相似性，并构建最大生成树，我们可以找到一种最佳的图像排序方式。这种排序方式能够确保拼接后的图像具有良好的连续性和一致性，从而提高拼接结果的质量。

为了提高拼接的准确性和降低冗余，我们还采用了非极大值抑制方法对特征点进行筛选。特征点是图像中的显著特征，对于计算图像之间的变换关系至关重要。通过应用非极大值抑制，我们可以选择具有较高响应值的特征点，并抑制其周围的非极大值。这种筛选方法可以保留最具代表性的特征点，减少冗余信息，从而提高拼接结果的精度和稳定性。此外，我们通过优化算法参数，进一步改进了非极大值抑制的性能，使其在保持有效性的同时提高筛选速度。

通过将最大生成树算法和改进的非极大值抑制方法相结合，我们的无序图像拼接算法可以自动对无序输入图像进行排序和拼接，从而获得高质量的图像拼接结果。该算法在处理多图像拼接问题时具有重要的实际应用价值，提高了图像拼接的效率并开阔了图像拼接的应用范围。

## 2. 改进算法介绍

### 2.1 最大生成树算法

最大生成树（Maximum Spanning Tree）是图论中的一个概念，它是一种连接图中所有节点的树结构，并且树中边的权重之和最大。最大生成树在许多领域中有着广泛的应用，如网络设计、图像处理、最优路径规划等。

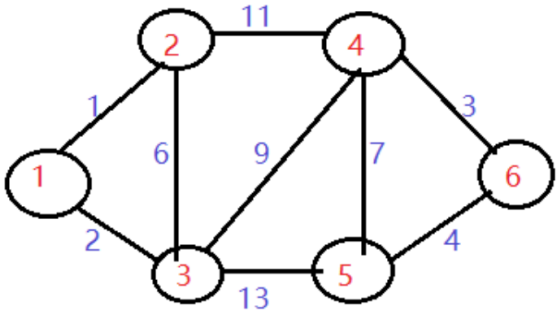



图 1 图节点示例

在最大生成树中，图的节点表示对象或位置，边表示节点之间的连接关系或关联性，权重则表示边的相关度或重要性。最大生成树的目标是选择一组边，使得这些边形成的树具有最大的总权重。

最常用的算法来构建最大生成树是 Kruskal 算法和 Prim 算法。Kruskal 算法是基于贪心策略的算法，它首先将图中的边按照权重从小到大排序，然后依次选择边，加入到最大生成树中，直到树中包含了所有的节点。Prim 算法也是一种贪心算法，它从一个初始节点开始，逐步扩展最大生成树，每次选择与当前树相连的边中权重最大的边，直到包含了所有的节点。

图像配准是将多幅图像的信息整合到一幅图像中的过程。此过程往往包括图像配准、变形、融合过程，其核心思想在于将多幅图像中表示相同结构信息的像素通过坐标转换，放在同一坐标系统中的同一空间位置。当且仅当两幅图像之间存在相似的结构信息时，配准才有可能成功。目前公开的图像配准手段中，往往只涉及两幅图像之间的配准或已知顺序的图像配准，而当需要参与配准的图像较多时，不能假定任意两幅图像之间存在相似的信息，从而产生图像配准顺序问题，顺序不佳可能直接导致配准错误甚至配准失败。为此，我们提出了一种基于最大生成树的配准顺序确定策略。


图 2 基于最大生成树匹配顺序

如图 2 所示，我们计算输入图像之间匹配的特征点对，将其作为权重构建一个无向完全图。此后，使用 Prim 算法构建最大生成树，即选择边权和最大的图像作为中心图像，随后选取边权最大的图像加入图像集合，直至所有图像加入集合。依据图像加入顺序，我们便得到了图像的合成顺序。我们的方法可以有效避

免图像漏配，在更多图像的合成任务中得到更完整的全景图像。

```
def get_hist_matched_img(self, tc3, tc1):
    if self.hist_matched_img is None:
        _, _, channel = self.img.shape
        self.hist_matched_img = np.zeros_like(self.img)
        if channel == 3:
            x, y, hist = self.get_histogram()
            for i in range(channel):
                cdf_img = np.cumsum(hist[i]) / np.sum(hist[i])
                cdf_ref = tc3[i]
                to = []
                for j in range(0, x[i]): to.append(j)
                for j in range(x[i], y[i]):
                    tmp = abs(cdf_img[j - x[i]] - cdf_ref)
                    tmp = tmp.tolist()
                    index = tmp.index(min(tmp)) + x[i]
                    to.append(index)
                for j in range(y[i], 256): to.append(j)
                to = np.array(to, dtype=np.uint8)
                self.hist_matched_img[:, :, i] = to[self.img[:, :, i]]
        else:
            x, y, hist = self.get_histogram()
            # hist = [util.hist_equalize(copy.deepcopy(hist[0]), 0.025)]
            cdf_img = np.cumsum(hist) / np.sum(hist)
            cdf_ref = tc1[0]
            to = []
            for j in range(0, x[0]): to.append(j)
            for j in range(x[0], y[0]):
                tmp = abs(cdf_img[j - x[0]] - cdf_ref)
                tmp = tmp.tolist()
                index = tmp.index(min(tmp)) + x[0]
                to.append(index)
            for j in range(y[0], 256): to.append(j)
            to = np.array(to, dtype=np.uint8)
            self.hist_matched_img[:, :, 0] = to[self.img[:, :, 0]]

    # cv2.imshow('origin', self.img)
    # cv2.imshow('out', self.hist_matched_img)
    # cv2.waitKey()
    return self.hist_matched_img
```

### 2.2 非极大值抑制

非极大性抑制（Non-Maximum Suppression，简称 NMS）是一种常用的计算机视觉技术，用于筛选特征点或边界框，以去除冗余信息并保留最显著的特征。

它的原理是在局部区域内选择具有最大响应值的特征点，并抑制周围的非极大值。

在图像拼接中，非极大性抑制可以用于筛选融合后的重复特征点。在拼接过程中，不同图像可能存在相似的特征点，例如边缘、角点等。当图像进行拼接时，这些特征点可能会出现重叠或重复的情况。为了避免重复特征点对拼接结果的干扰，需要进行非极大性抑制来选择最具代表性的特征点。

在应用非极大性抑制时，首先需要对融合后的特征点进行排序，按照其响应值或其他评估指标进行降序排列。然后，从响应值最高的特征点开始，依次遍历每个特征点。对于当前特征点，检查其周围的邻域，如果存在比它更大的响应值，则将当前特征点抑制，否则保留。

图像特征点能够反映图像本质特征，能够标识图像中目标物体。用较少的特征点来表示图像关系信息是完成图像精准匹配的关键。在图像拼接中，非极大性抑制可以帮助去除融合后的重复特征点，从而提高拼接结果的准确性和稳定性。它能够保留最显著的特征点，减少冗余信息，并避免特征点的重叠或重复现象。

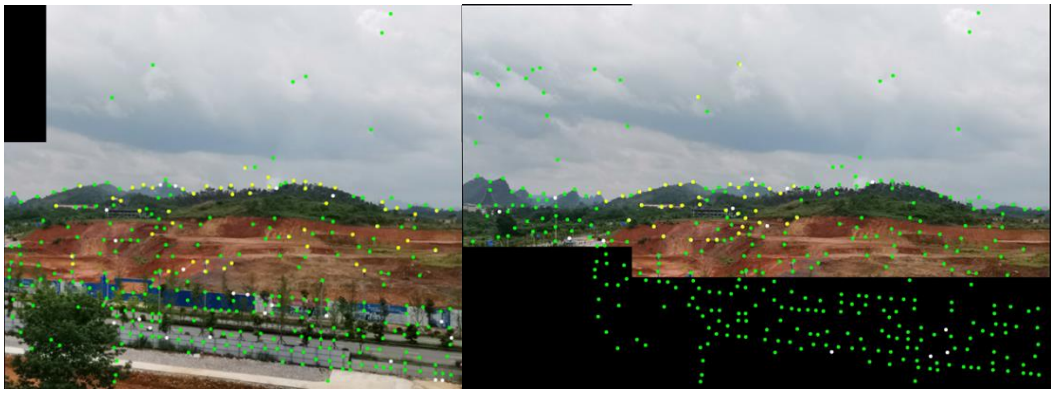
通过将最大生成树算法和非极大性抑制方法相结合，可以在图像拼接过程中实现对融合后的重复特征点的筛选。最大生成树确定了图像之间的顺序，而非极大性抑制则进一步优化了特征点的选择，确保拼接结果的质量。这种综合应用能够提高无序图像拼接的准确性和稳定性，得到更好的拼接效果

对特征提取网络中输出的特征点进行非极大值抑制，主要实现两个目标：（1）剔除局部区域中较低置信度粘连的特征点，（2）加速算法处理流程，在毫秒级别完成非极大值特征点剔除。为此我们采用图像放缩细节减少的思想，在特征点坐标维度进行操作。定义图像宽和高分别为 L 和 H，图像特征点坐标 $FP=\{(x_t,y_t)|0 \le x_t \le H, 0 \le y_t \le L, t \in [1,size]\}$，size 为图像特征点个数，每个特征点的竖直方向和水平方向最小感受野半径为$Dx_t, Dy_t$，特征点最小感受野半径为 C，设定算法循环次数 num，放缩比例数组$Z = \{Z_i | Z_i \ge 2 * C + 1, i \subset [1,num]\}$，则可以得出特征点的感受野范围即为：

$$\begin{cases} Dx_t = \min(x_t - \min\limits_{i=1,i\le num}(\lfloor x_t/Z_i \rfloor * Z_i), \max\limits_{i=1,i\le num}((\lfloor x_t/Z_i \rfloor + 1) * Z_i - 1) - x_t) \\ Dy_t = \min(y_t - \min\limits_{i=1,i\le num}(\lfloor y_t/Z_i \rfloor * Z_i), \max\limits_{i=1,i\le num}((\lfloor y_t/Z_i \rfloor + 1) * Z_i - 1) - y_t) \\ (x_t, y_t) \in FP \end{cases}$$

式中 min 为取元素最小值运算符，$\lfloor \rfloor$为向下取整运算符。

为了使上述方法的精度能与传统非极大值抑制算法相当，我们利用逐步迭代的思想优化图像特征点，通过变换不同的放缩比例来扩大特征点非极大值抑制感知区域，只需要设定迭代次数为 3 次，就能达到非常卓越的效果。图 3 展示了我们的特征点非极大值抑制图例，其中绿色点为被保留的特征点，黄色点为被筛去的特征点。从(b)中可已看出，我们的方法完好地保留了融合图像的特征点。



(a) (b)

图 3 特征点非极大值抑制图例

## 3. 图像拼接算法流程设计

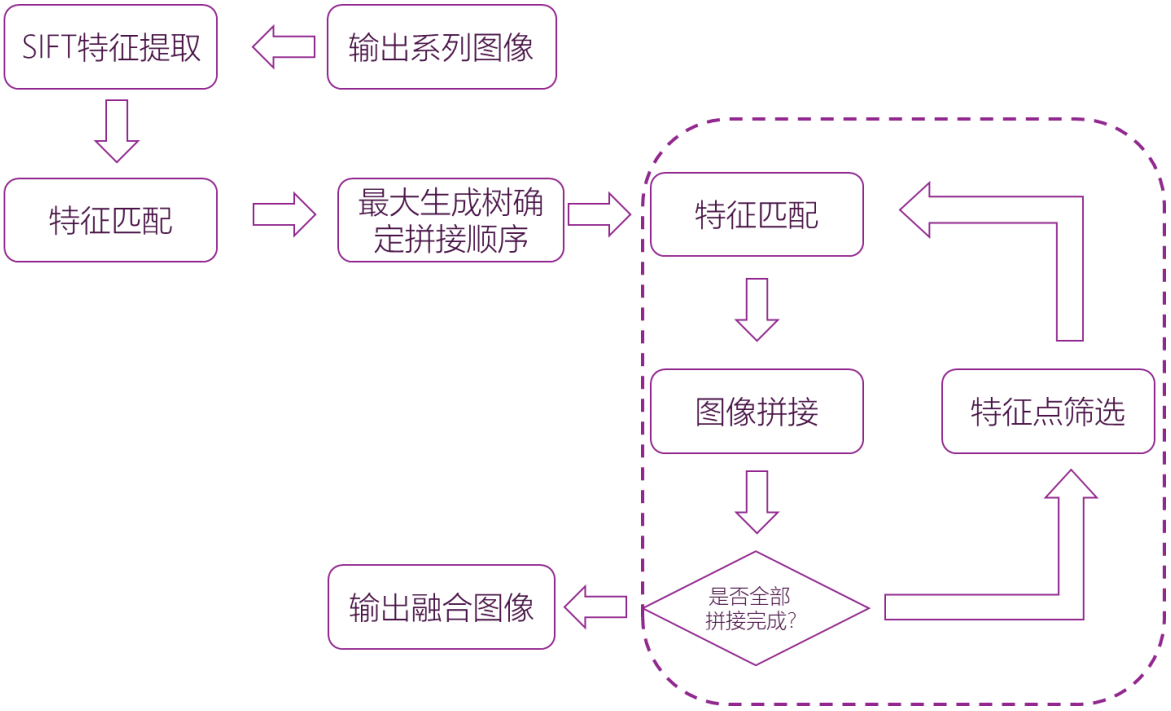



图 4 整体算法流程

改进后的整体算法流程如图 4 所示，首先通过特征提取器提取出输入图像的特征点，再两两进行特征匹配，并计算每两张图之间匹配的特征点数量，以此来作为权重构建无向图。使用 Prim 算法生成最大生成树，并以此确定拼接的顺序。之后便按照顺序进行逐个拼接，并在每次拼接后对融合图像的特征点进行非极大性抑制，来筛选出较优的特征点。

```
        if self.img.shape.__len__() > 2:
            gray_img = cv2.cvtColor(self.img, cv2.COLOR_BGR2GRAY)
            # gray_img = cv2.cvtColor(self.img, cv2.COLOR_BGR2GRAY)
        else:
            gray_img = self.img
        mask = util.getMask_morphologyEx(gray_img, 400)
        h, w = self.img.shape[:2]
        mask = cv2.resize(mask, (w, h))
        if self.img.shape.__len__() > 2:
            self.img[:, :, 0] = np.where(mask == 1, 0, self.img[:, :, 0])
            self.img[:, :, 1] = np.where(mask == 1, 0, self.img[:, :, 1])
            self.img[:, :, 2] = np.where(mask == 1, 0, self.img[:, :, 2])
        else:
            self.img = np.where(mask == 1, 0, self.img)
        bottom, top, left, right = util.calc_blackborder_with_mask(mask)
        self.top = bottom
        self.left = left
        self.img = self.img[bottom:top, left:right]
        self.img = util.img_resize(self.img, tar_size=self.config['tar_size'])
        if self.img.shape.__len__() == 2:
            self.img = np.expand_dims(self.img, axis=2)
        # if self.img.shape[2] > 1:
        #         self.img = util.color_histogram_match(self.img, ImageGroup.img_ref)

        h, w = self.img.shape[:2]
        mask = mask[bottom:top, left:right]
        mask = cv2.resize(mask, (w, h))
        # self.mask = mask
        self.weight = self.get_weight_with_template(h, w, mask)
        # self.weight = self.get_weight_with_contrast(self.img)
        self.weight *= (1 - mask)
        self.weight = np.expand_dims(self.weight, axis=2)
        # if self.img.shape.__len__() > 2:
        #         self.weight = cv2.cvtColor(self.weight, cv2.COLOR_GRAY2BGR)
        # cv2.imshow('img',self.img)
        # cv2.imshow('weight',self.weight)
        # cv2.waitKey()
        return cv2.resize(gray_img[bottom:top, left:right], (w, h))
```

# 4. 改进后目标检测算法及测试

## 4.1 水平图平移测试

以下为水平平移图像的拼接效果展示：首先，我们将图 5 所示的原图像进行

裁剪，使其分离为五张无序的图像，载入模型中获得拼接后的图像如图 6 所示。

图 5 原图像

图 6 拼接后图像

可以看出，整体的拼接效果不错，只有山的部分存在有一部分重影，可能是特征匹配算法没有良好的匹配群山的特征点导致。

### 4.2 角度偏移图像拼接

首先，我们手机拍摄三张不同和位置角度的连续图片如图 7 中所示，并将其输入进算法中进行输出：

图 7 不同拍摄角度的三张图像

图 8 拼接后图像

从图 8 中可以看出，在处理单个角度时效果较好，但在三张图像进行仿射变换时栏杆位置出现了一定程度的畸变。

## 5. 结束语

在实验结果中，可以看到我们的模型可以通过最大生成树算法和图之间的特征点匹配个数，有效识别打乱顺序后的图片，按照一定的顺序进行拼接。并且在水平平移的情况下拼接效果良好。

并且，在加入了非极大性抑制算法后，避免了对融合后图像再次提取特征的冗余操作，加快了图像拼接的速度。

当存在角度偏移的情况下，图像在进行仿射变换的时候仍然会存在难以完全匹配的问题。在未来可以通过更加优异的单应性矩阵估计方法来改善该问题。