第十一届数学中国数学建模网络挑战赛

地址：数学中国数学建模网络挑战赛组委会　　网址：www.tzmcm.cn

电话：0471-4969085　　邮编：010021　　Email：service@tzmcm.cn

第十一届“认证杯”数学中国

数学建模网络挑战赛

承　诺　书

我们仔细阅读了第十届“认证杯”数学中国数学建模网络挑战赛的竞赛规则。

我们完全明白，在竞赛开始后参赛队员不能以任何方式（包括电话、电子邮件、网上咨询等）与队外的任何人（包括指导教师）研究、讨论与赛题有关的问题。

我们知道，抄袭别人的成果是违反竞赛规则的，如果引用别人的成果或其他公开的资料（包括网上查到的资料），必须按照规定的参考文献的表述方式在正文引用处和参考文献中明确列出。

我们郑重承诺，严格遵守竞赛规则，以保证竞赛的公正、公平性。如有违反竞赛规则的行为，我们接受相应处理结果。

我们允许数学中国网站(www.madio.net)公布论文，以供网友之间学习交流，数学中国网站以非商业目的的论文交流不需要提前取得我们的同意。

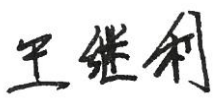
我们的参赛队号为：2281

参赛队员　(签名)　：

队员 1： 

队员 2： 

队员 3： 

参赛队教练员 (签名)： 

参赛队伍组别（例如本科组）：研究生组

第十一届数学中国数学建模网络挑战赛
地址：数学中国数学建模网络挑战赛组委会 网址：www.tzmcm.cn
电话：0471-4969085 邮编：010021 Email：service@tzmcm.cn

第十一届“认证杯”数学中国

数学建模网络挑战赛
编 号 专 用 页

参赛队伍的参赛队号：（请各个参赛队提前填写好）：2281

竞赛统一编号（由竞赛组委会送至评委团前编号）：

竞赛评阅编号（由竞赛评委团评阅前进行编号）：

第十一届数学中国数学建模网络挑战赛

地址：数学中国数学建模网络挑战赛组委会　　网址：www.tzmcm.cn

电话：0471-4969085　　邮编：010021　　Email：service@tzmcm.cn

2017 年第十一届“认证杯”数学中国数学建模网络挑战赛第一阶段论文

题　　目　基于形位特征的机械零件加工过程中的位置识别

关 键 词　二值图像、位置识别、形位特征、最大连通域、图像膨胀、腐蚀

摘　　要：

本文以工业制造自动生产线中的零件位置为研究对象，基于二值图像的数学原理，分别考虑单一零件、多个零件在一个图像的环境，分析零件轮廓及零件间间隙包含的数学规律，建立多个数学模型，制定相应计算流程，解决单一零件、多个零件在生产线上位置识别的问题，满足一定生产场合下的零件自动搬运要求。

本文的主要创新点：本文在处理包含单一零件的图像时，提出了一种利用零件形状特征，包含图像膨胀、图像最大连通域求解等算法的建模方法，该方法计算速度快，效率高；进一步地，提出一种不需要考虑零件形状特征的通用模型，该方法通用性高，覆盖范围广；在处理包含多个零件的图像时，提出了一种通过零件间位置特征和图像分割算法构建单一零件环境的建模方法，该方法易于实现、简单高效。

本文完成的主要工作总结如下：

针对问题一，在第一种建模方法中，首先以计算量少为原则确定出基准位置，并确定了表述位置的参数（X，Y，α），然后充分利用矩阵和圆的形状特征，采用图像膨胀技术，并引入物理学概念“形心”来提取特征点，最后以零件特征点与基准特征点距离最小为优化目标求解出位置参数——（202.6573,296.6206,119.58°）。

为了提高通用性，针对问题一，提出第二种建模方法：首先，在坐标系中重建出具体尺寸的零件轮廓图作为基准；其次，以零件轮廓的所有点与基准的所有点距离最小为优化目标求解出位置参数——（295.4292,203.5155,-62.28°）。

用两种模型求解之后，采用时间复杂度、空间复杂度等指标对上述两种方法进行了效率对比，得出第一种建模方法效率高的结论。

针对问题二，充分利用图像中零件无重叠这一位置特征，采用分割算法，人为构建单一零件图像环境，进行图像分割前，采用图像腐蚀算法处理边界间隙，最后调用问题一的模型分别求解出两零件的位置信息——（405,236,150.33°）和（200,300,120.57°）。

求解之后，对各个模型与求解方法进行了评价，并给出了改进建议。

最后，针对模型的应用前景与比赛第二阶段的深化研究可能性，深度开展了对五类问题的展望。

参赛队号：2281

所选题目：C 题

参赛密码 _____

（由组委会填写）

第十一届数学中国数学建模网络挑战赛
地址：数学中国数学建模网络挑战赛组委会 网址：www.tzmcm.cn
电话：0471-4969085 邮编：010021 Email：service@tzmcm.cn

Abstract

In this paper, the location of parts in the industrial automatic production line is taken as the research object. Based on the mathematical principle of binary image，several mathematical models are established through analyzing relative mathematical law of the outline of parts, considering a single part or multiple parts in an image. Corresponding calculation processes are set up to solve the problem of position recognition for a single part or multiple parts, satisfying the requirements of automatic parts handling under certain production conditions.

The main innovation of this paper: based on part shape features, a modelling method including algorithms of image expansion and a maximum threshold is proposed for dealing with images of a single part, which is fast and efficient. For high generality and wide coverage, a universal model based on simulated annealing algorithm is presented. For more complex problem, a method based on image segmentation is employed for extracting a single part from images with multiple parts.

The main work completed in this paper is summarized as follows:

In the first method modeling for the question one, Determined the reference position with the principle of less calculation, and Determined the parameters （X，Y，α） which express the position. Extracted the the characteristic points which used the shape characteristics of matrix and circle and the technique of image expansion and The concept of centroid of physics. In Final, Solved the parameters of position （202.6573,296.6206,119.58°） as the optimization objective of the minimum distance between feature points and benchmark points.

In order to improve the generality, the second kinds of modeling methods are proposed to solve the problem. First, the part profile of the specific size is rebuilt in the coordinate system as the reference. Secondly, the location parameters （295.4292,203.5155,-62.28°） are solved by the minimum distance between all points of the part contour and all the points of the reference.

After solving the above two methods, Two methods are compared with norm time complexity and space complexity, and conclusion is drawn that the first modeling method is efficient.

For the problem 2, according to the non-overlapping feature of parts in the image, image segmentation is employed combining image erosion and expansion for extracting a single part from images with multiple parts. Finally, the model of the problem 1 is called for solving the position information of the two parts respectively (405,236,150.33°) and（200,300,120.57°）.

After solution, each model has been evaluated for improvement suggestion.

Finally, based on the application prospect of the model and the possibility of further research in the second stage, we have carried out a deep prospect.

Keywords： Binary image，Position recognition，Feature of shape and position，Maximum threshold，Image expansion，Image erosion.

目　录

1 问题重述

1.1 问题提出的背景

在原始的生产过程中，人类通过眼睛来认识世界，最终由运动系统执行动作任务。在这个过程中，眼睛是人体视觉系统的一部分，它通过光线传播接收外界信息，然后在视网膜上进行光学成像，接着由视觉接收器将光信息转化为视网膜的神经活动电信息，最后通过视神经纤维把这些信息传送入大脑，由大脑获得图像感知，继而做出判断，给诸如手之类的运动系统发出动作执行的命令，从而完成生产工作任务。由上述描述可知，人体视觉系统处理的是图像信息，图像是对外界客观存在的事物的直观而具体的描述，是人类视觉信息的重要载体，只有正确地分析图像信息，才能对运动系统发出正确的指令，最终正确地完成工作。

现今社会都提倡用机器代替人来工作，机器的整个工作过程是与人的工作过程类似的。在机器智能生产过程中，利用摄像头来模拟人眼成像，利用“计算机视觉”[1-3]来模拟人眼视觉系统模拟人眼活动、自动的辨识图像目标、分析图像行为，最终由计算机给诸如机械手之类的执行机构发出动作执行的命令，从而完成生产工作任务。由上述描述可知，在智能化生产过程中，正确地分析图像信息同样是重要的。

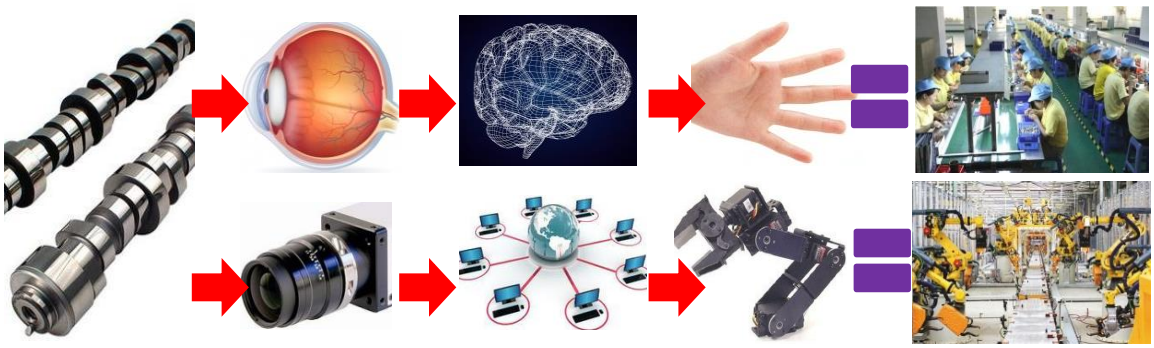


图 1.1 人与机器工作过程类比示意图

近些年来，随着全球计算机技术日新月异的发展和普及，人类已经进入了多媒体信息化时代，以计算机技术为驱动的智能化趋势使得智能化生产的覆盖面积越来越广，以节省人力、提高生产效率为需求的智能化生产线的建设范围越来越广、发展越来越迅速，大到核电[4]、汽车[5]，小到手机[6]、家电[7]，不同程度的出现了智能化生产线。

一方面，智能制造是对现有的制造业的提升，包括缩短开发周期、降低成本、提升效率等。可以在产品设计阶段就模拟出该产品的整个生命周期，从而更有效，更经济、更灵活的组织生产，实现了产品开发周期最短，产品成本最低，产品质量最优，生产效率最高的保证。另一方面，智能制造将会推动制造业发展出全新的制造模式，包括柔性制造、生物制造、绿色制造、分形制造等，这种以消费者为导向的，以需定

产的方式更受投资人青睐，同时也能推动目前在电子商务领域兴起的“C2B”“C2P2B”等模式的完善。

综上，其市场潜力之巨大可见一斑，大量的数据也证明了这一点——图 1.2 展示了 2010 至 2016 年全球智能制造产值规模的测算值，图 1.3 展示了 2010 至 2017 年我国智能制造行业产值规模情况，从图中可见，智能制造在国内外均呈现上升趋势。结合当前全球智能制造的发展现状和发展趋势，有关机构保守估计，未来几年全球智能制造行业将保持 10%左右的年均复合增速，预计到 2022 年全球智能制造的产值将达到 1.51 万亿美元左右[8][9]。智能制造趋势向好，而基本所有的智能化生产都对图像信息的处理提出不同程度的需求。

以此为背景，结合智能化生产对图像处理技术的需求，图像信息处理的质量优劣、速度快慢、精度高低等性能的重要程度在智能化生产系统应用的广泛程度的背景下，经济意义越发彰显——面对全球化经济中智能化生产线的发展状况，解决其中图像信息处理的关键问题既是全人类所迫不及待的，又是国家、企业经济命脉所关联的。

总之，智能化生产的发展会对图像信息处理技术提出了更多、更严格的要求，解决智能生产中图像信息处理的问题背景意义重大。

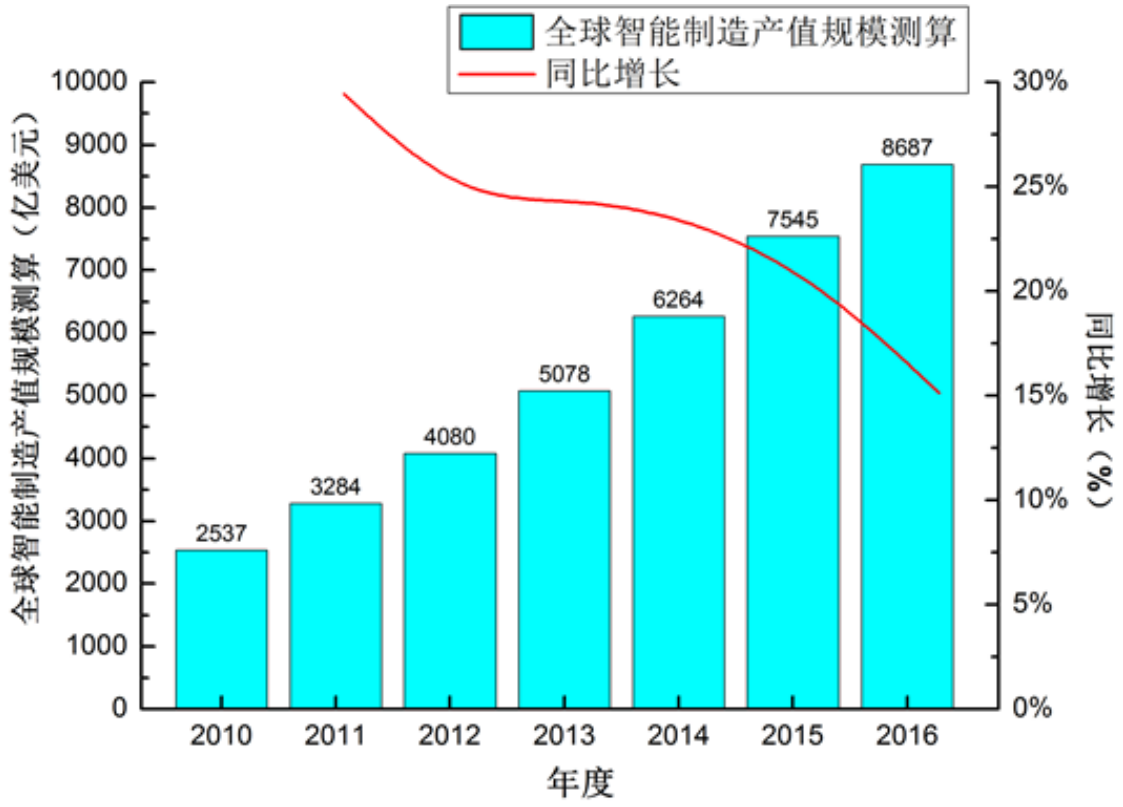


图 1.2 2010-2016 年全球智能制造产值规模测算

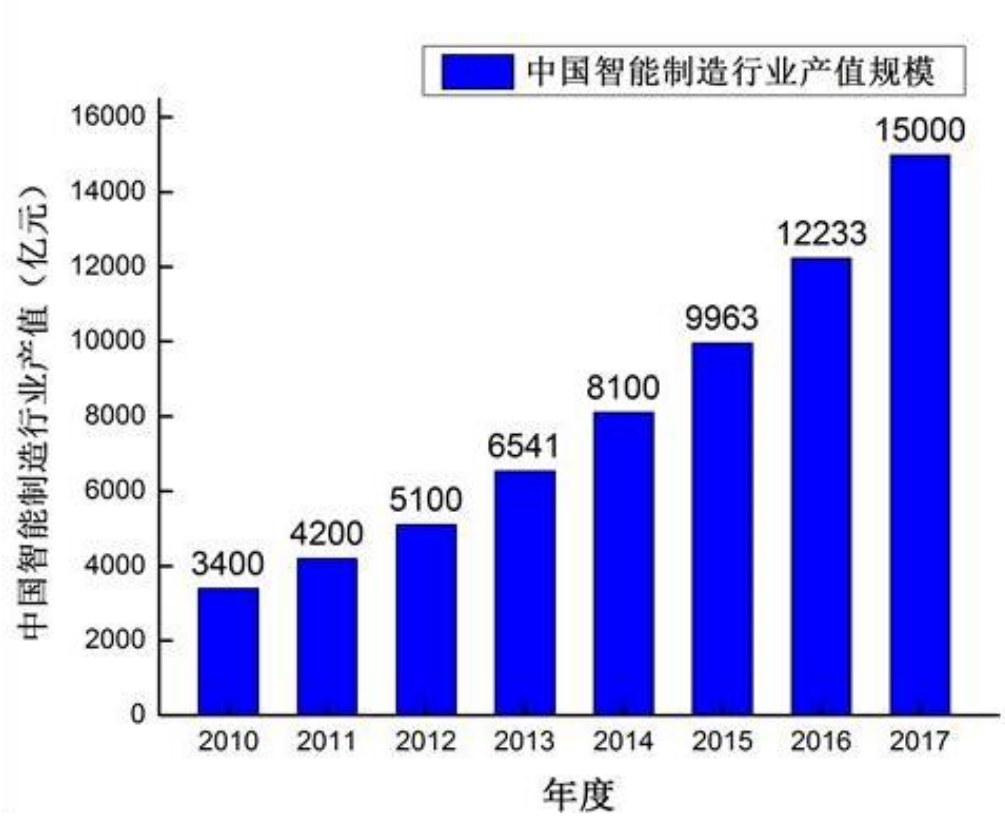


图 1.3 2010-2017 年中国智能制造行业产值规模情况

1.2 问题的提出

在研究过程中，图像信息处理技术在智能生产的应用中存在一些问题，包括轮廓测量的准确性、外观检测的难度、字符的准确识别、条码的识别以及定位信息的测量等。

其中，作为图像信息处理的一部分，位置识别应用广泛、作用显著，特别是在智能化生产中加工零件的定位方面。同时，在零件加工过程中，位置识别也存在一些问题，具体表现在难以定位、定位不准确、定位速度慢等方面，所以在零件位置相对于初始位置有变动的情况下，可能导致零件拾取的失败、加工效率的降低甚至加工的失败。所以，位置识别技术的不足会难以满足智能化的需求。

在传统生产中，可认为纠正零件位置，让生产继续进行，而在智能化生产中，只有通过图像拾取、分析图像特征、进行位置识别，再通过相应算法，才能实现零件位置的纠正或者零件的准确拾取。

现今研究人员已经提出了很多位置定位的方法，使得这一技术广泛应用于自动化生产线上。但是，这些方法都是针对具体生产线上的特定产品而开发的，如果受到外部变动的影响、零件的更换、实时性要求提高等问题，方法的普适性还不够强。因此，对具体形状零件的准确定位的研究具有个性特征，有进一步深入研究的必要。

1.3 待解决的问题

依据上述背景及问题，本次竞赛的 C 题通过给定一个具体形状的零件着重提出要解决机械零件加工过程中的位置识别的问题：

在工业制造自动生产线中，在装夹、包装等工序中需要根据图像处理利用计算机自动智能识别零件位置，并由机械手将零件自动搬运到特定位置。某零件轮廓如图 1.4 所示，图 1.5 表示零件搬运前后的位置示意图。待解决问题包括：

待解决的问题一：根据附件 DATA1（题目给定）中给出的零件轮廓数据，请建立数学模型，识别计算出给定零件的位置坐标，并分析评价求解零件位置的算法是否快速高效。

待解决的问题二：问题 1 讨论的是单个零件放置于平面操作台上的情况。有时我们需要处理多个零件显示在同一图像中的情况，请根据附件 DATA2（题目给定）中的数据，建立数学模型，识别出不同零件的位置。

本文将针对不同的问题，分别建立模型，逐一解决上述零件位置变动下的问题。

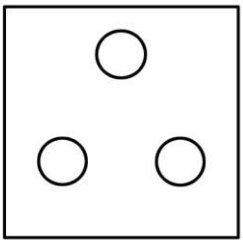


图 1.4 零件轮廓示意图

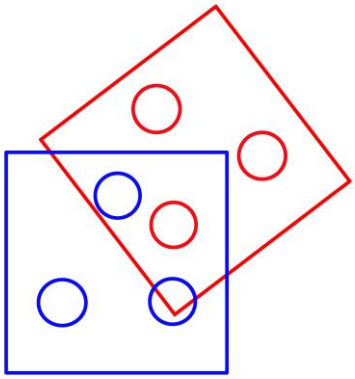


图 1.5 零件搬运前后位置示意图（红色为搬运前，蓝色为搬运后）

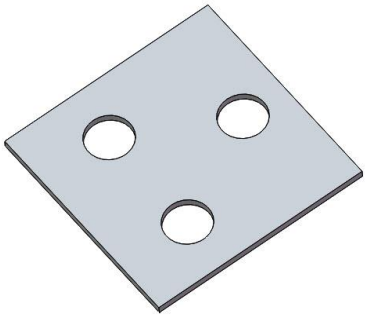


图 1.6 零件三维示意图

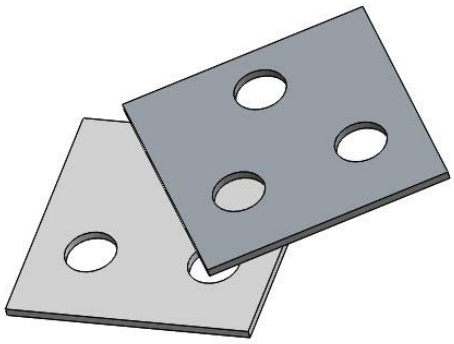


图 1.7 多个零件三维示意图

2 术语和定义

专有术语	定义及说明
二值图像	指将图像上的每一个像素只有两种可能的取值或灰度等级状态，人们经常用黑白、B&W、单色图像表示二值图像。本文中的二值轮廓用 0 和 1 表示。
形位特征	零件的形状特征：轮廓的形状，矩形，圆等； 零件间的位置特征：零件之间接触、重叠等。
二值图像的连通性	对于集合 S 存在一条通路的条件是，通路的像素的某种排列，相邻像素满足某种邻接关系。例如点 p 到点 q 之间有 n 个像素点，且相邻像素点都满足某种邻接。则 p 和 q 间存在通路。如果通路首尾相连，则称闭合通路。S 集合中的一点 p 只存在一条通路，则称为一个连通分量，如果 S 只有一个连通分量，则称为一个连通集。对于 R 为一个图像子集，如果 R 连通的，则称 R 为一个区域。对于所有不连接的 K 个区域，其并集构成了图像的前景，该并集的补集称为背景。
连通域	连通区域一般是指图像中具有相同像素值且位置相邻的前景像素点组成的图像区域。连通区域分析是指将图像中的各个连通区域找出并标记。连通区域分析是一种在图像分析处理的众多应用领域中较为常用和基本的方法。在需要将前景目标提取出来以便后续进行处理的应用场景中都能够用到连通区域分析方法，通常连通区域分析处理的对象是一张二值化后的图像。
最大连通域	上述概念的最大值[10]。
二值矩阵	二值图像分析中所有的像素只能从 0 和 1 这两个值中取，因此在 MATLAB 中,二值图像用一个由 0 和 1 组成的二维矩阵表示，这个二维矩阵称为二值矩阵。
形心	面的形心就是截面图形的几何中心，对于密度均匀的实物体，质心和形心重合。有限个点总存在几何中心，可以通过计算这些点的每个坐标分量的算术平均值得到。这个中心是空间中一点到这有限个点距离的平方和的惟一最小值点。
图像膨胀和腐蚀	膨胀和腐蚀这两种操作是形态学处理的基础,许多形态学算法都是以这两种运算为基础的。 膨胀和腐蚀的主要用途：消除噪声；分割出独立的图像元素，在图像中连接相邻的元素；寻找图像中明显的极大值或极小值区；求出图像的梯度。

图像膨胀	膨胀：求局部最大值； ①定义一个卷积核 B，核可以是任何的形状和大小，且拥有一个单独定义出来的参考点-锚点(anchorpoint)；通常和为带参考点的正方形或者圆盘，可将核称为模板或掩膜； ②将核 B 与图像 A 进行卷积，计算核 B 覆盖区域的像素点最大值； ③将这个最大值赋值给参考点指定的像素； 因此，图像中的高亮区域逐渐增长。
图像腐蚀	腐蚀：局部最小值(与膨胀相反)； ①定义一个卷积核 B，核可以是任何的形状和大小，且拥有一个单独定义出来的参考点-锚点(anchorpoint)；通常和为带参考点的正方形或者圆盘，可将核称为模板或掩膜； ②将核 B 与图像 A 进行卷积，计算核 B 覆盖区域的像素点最小值； ③将这个最小值赋值给参考点指定的像素； 因此，图像中的高亮区域逐渐减小。
特征圆	图像中识别到的类似圆但又不确定为圆的图像。
特征矩形	图像中识别到的类似矩形但又不确定为矩形的图像。
特征提取	使用计算机提取图像信息，决定每个图像的点是否属于一个图像特征。特征提取的结果是把图像上的点分为不同的子集，这些子集往往属于孤立的点、连续的曲线或者连续的区域。
轮廓外特殊点	本文中的“轮廓外特殊点”是指除去零件轮廓之外，在二值图像中显示的其他具有干扰特征的散点。
模拟退火算法	模拟退火算法是一种通用概率演算法，用来在一个大的搜寻空间内找寻命题的最优解。模拟退火的出发点是基于物理中固体物质的退火过程与一般组合优化问题之间的相似性。模拟退火算法是一种通用的优化算法，其物理退火过程由加温过程、等温过程、冷却过程这三部分组成。
角点检测	https://baike.baidu.com/item/%E8%A7%92%E7%82%B9%E6%A3%80%E6%B5%8B/2872134?fr=aladdin
Hough	Hough 变换是一种使用表决原理的参数估计技术。其原理是利用图像空间和 Hough 参数空间的点－线对偶性，把图像空间中的检测问题转换到参数空间。

注：其他未说明术语的含义结合正文及行业语言理解

3 问题分析

3.1 整体分析

依据题目所给的条件和信息及要求解的两个问题，整体来讲两个问题研究的都是零件位置识别的问题，但是问题二比问题一的应用环境更为复杂、限定条件更多。具体讲，本题中的位置识别的问题是针对简单的二值图像的位置识别的问题，属于本文背景意义中所提及的图像处理的一种类型——一种较为简单的类型。换句话说，本题的具体任务是处理二值图像。

3.2 具体问题分析

3.2.1 问题一的分析

依据题目一，首先需要对附件 DATA1 的数据进行读取。通过 matlab 的 Load 和 Imshow 命令，可知，data1 包含了一个 420×560 的二值矩阵，二值矩阵可生成一个二值图像，展示了某零件的轮廓信息，直观上讲，该零件为一带有三个孔的矩形板，外轮廓近似于正方形，三个孔的圆心可构成近似的正三角形，图像中有较多的轮廓外特殊点。

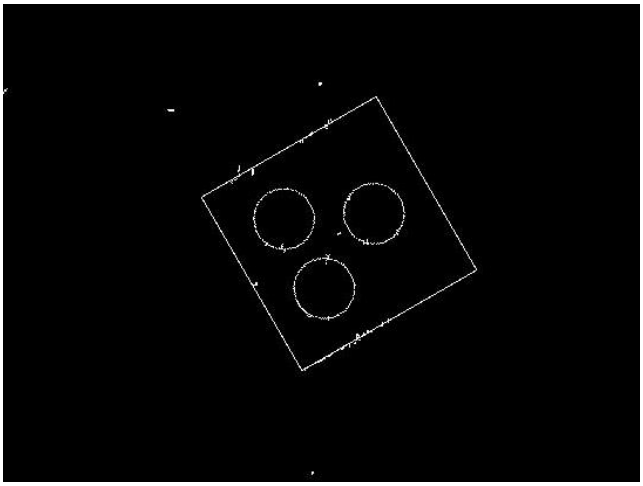


图 3.1 DATA1 的读取结果

为了解决零件位置识别的问题，首先，需要寻找表征位置的参数，题目中的零件轮廓是平面图像信息，所以可用两个方向的移动和一个方向的转动三个自由度来表征位置信息，其次，由于位置信息具有相对性，所以需要人为建立位置基准，再次，零件与位置基准之间的关系不是单一点与点之间的位置关系，而是整个轮廓与基准轮廓之间的位置对应关系，简单说：在搬运零件时，并非使得一个点重合即可，而是必须整个轮廓重合，所以，需要分析轮廓的特征（其实在实际生产工作中，也是针对特定零件的具体形状的识别和搬运方案）。通过分析轮廓特征，找出能表述该轮廓的特征点，只要零件和基准的特征点重合，则位置识别成功。最后，则需要针对寻找到的特征点建立具体的数学模型，从坐标变换的角度建立模型计算出位置信息。

另外，应题目一要求，需要考虑算法的速度和效率，毕竟工业生产中，也会对此提出要求。对于这一要求，需要建立表征算法速度和效率的模型，对其进行评价。

进一步的，在自我评价之后，需要对模型进行改进，通过简化模型、去除多余步骤等方式，提高算法速度。比如：考虑需要不需要证明轮廓的形状为具体什么形状；需不需要去除图中的轮廓外特殊点等。

3.2.2 问题二的分析

同样的，依据题目二，首先需要对附件 DATA2 的数据进行读取。通过 matlab 的 Load 和 Imshow 命令，可知，data2 包含了一个 522×560 的二值矩阵，二值矩阵可生成一个二值图像，图像展示了两个与问题一中同样的零件放置于同一操作平台上的轮廓信息，从直观上讲，两个轮廓的位置不同，但轮廓并没有重叠的部分。图像中只有少量的轮廓外特殊点。这些信息为求解问题二是利好的，因为只需要做到两个零件的分割即可利用问题一的模型求解位置信息。

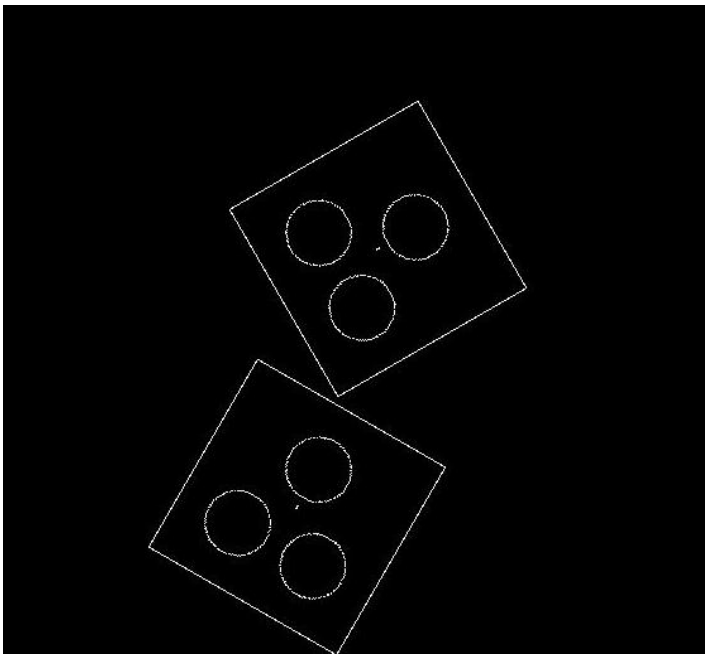


图 3.2 DATA2 的读取结果

明显的，问题二是问题一的应用，但是在应用之前，需要通过数学模型和算法将两个零件进行有效分离或者分类，这样才能应用问题一的方法对各个零件进行位置信息获取。

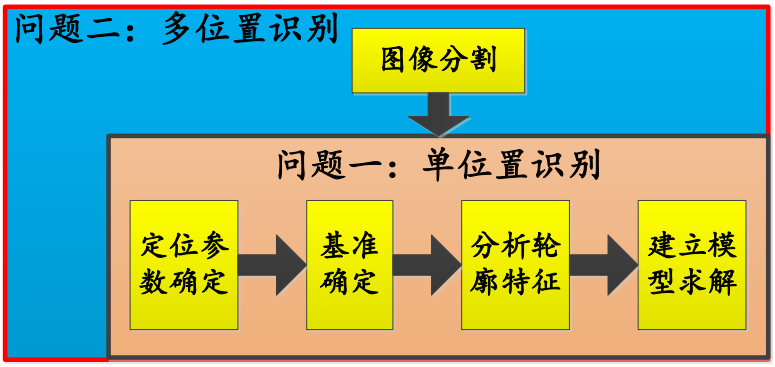


图 3.3 问题间的关系

4 模型假设

全文假设	➢ 假设一：仅考虑二值矩阵构成的图像。 ➢ 假设二：给定的轮廓都是平面二值图像。 ➢ 假设三：位置识别只考虑平面上的形状、位置参数，其他参数不予考虑。 ➢ 假设四：鉴于题目中未给定位置识别的基准位置，基准位置人为定义。
问题二的补充假设	➢ 假设五：问题二中的零件与问题一中的零件相同。 ➢ 假设六：问题二中的基准与问题一中的相同。 ➢ 假设七：问题二中的两个零件不互相接触。 ➢ 假设八：问题二中的两个零件在同一平面上。

注：其他未说明假设可结合行业默认假设理解

5 符号说明

注：本文中的符号只代表局部意义，因此不在此同一说明，各个符号的含义见正文模型。

6 二值图像处理的理论基础

6.1 基础理论

本题目的实质是处理二值图像，所以，本文中的求解过程也是基于二值图像理论进行的。

数字图像处理中，为了对图像更好地进行下一步处理，有时把图像进行二值化，

变成二值图像[11-15]，即在图像中，灰度等级只有两种，也就是说，图像中的任何像素不是 0 就是 1，再无其他过渡的灰度值。

二值图像的简单性是它的一大优点，因为它不像多灰度级图像有很多灰度值，它只有两个，但就是这两个灰度值在描述物体时更简单，如果一个灰度值表示“是”的概念，那么另一个灰度值就表示“非”的概念。两个灰度级也可以把图像的轮廓等几何特征很好的显现出来，保证了图像形状不变，内容不失真。实际生活中，在比较两幅图像的相同之处和不同之处时，比如比较医学上的 X 光照片，确定病情的变化时，保持二值图像的简单性就很重要了。当然，有时这也是缺点，因为两个灰度级不能描述清楚复杂图像的细节。

二值图像能够吸引我们的另一个优势就是存储量小，两个灰度级通常都用 0 和 1 来表示，那么一个单位就代表一个像素点的灰度值[16]，再对图像进行存储的时候，存储量会减少很多，节省大量的空间。而且在图像处理过程中效率也会明显提高，也正是因为这样，在许多实用图像的处理过程中，很多时候都是把图像转化为二值图像作为输入数据。

6.2 方法综述

提及二值图像，“连通域”必是息息相关的，二值图像的运算、求解、识别等方方面面均涉及到连通域的求解与标记。连通域标记方法的发展现状侧面反映着二值图像处理方法的现状。当然，连通域求解也是本文解题思路中的核心。

关于二值图像连通区域标记算法第一个提到的是由 Fu Chang, Chun-Jen Chen 和 Chi-Jen Lu 等人设计的轮廓跟踪连通域标记算法，该算法通过一次光栅扫描之后的标记和两次内外连通区域的扫描来确定一个连通域。该算法有很好的线性特性，对于各种类型的二值图像随着图像大小不断增加算法所需要的处理时间也呈稳定的线性增长，是二值图像连通域标记问题处理的一个典型算法。而由 J. Martin-Herrero 等人设计的混合对象标记算法，将光栅扫描和队列结合使用，该算法也得到了较好的效果。等价标记类算法是一类算法,这类算法都是先采用一次光栅扫描扫描图像并加以标记，而后根据第一次扫描的结果，再进行一次光栅扫描对第一次扫描过的已经标记的像素进行等价类划分，最终划分为各个等价类即连通域。该类算法的第一步都是进行光栅扫描和临时标记，第二步则采用了不同的方法实现等价类划分，较为典型算法有以下几种。T. Goto, Y. Ohta, M. Yoshida 等人设计的高速组件标记算法，该算法的第二步采用类似图的邻接矩阵的方式划分等价类，采用遍历邻接矩阵的方法标记出等价类。K. Wu, E. Otoo, K. Suzuki 等人设计的优化二段式连接组件标记算法，该算法在第二步采用了类似有向图的连通性的划分方式，将各个等价类映射为一个图上的点，如果等价则相互连接构成环，最终一个环就是一个划分的等价类，实现了划分等价类的目的。何立风等人设计的二值图像连通域快速标记算法，在第二步中将各个等价类映射为集合相互等价的集合之间进行集合合并操作并以最小的标记作为代表标记，最终实现了快速等价类划分的目的。通过测试，该算法在随着各类图像的不断增大的情况下有着更好的线性特性，算法较为稳定，是同一类算法中效率最高的算法。另外有一些国内

外学者结合 FPGA（Field-Programmable Gate Array）芯片设计了标记算法，这一类算法结合硬件电路设计和程序编写处理效率较高。

重庆大学覃方涛等人用 GPU 加速来实现二值图像的标记算法，有论文发表，也对效率进行了一些分析，但是通过对该算法的进一步研究发现该算法的实现有潜在的缺陷，是一个不够完善的算法，有一些测试案例不能覆盖，并且效率不高。中国空空导弹研究院王静等人设计了一种比较适合于高速运动目标实时跟踪的标记的标记算法。北京科技大学徐正光等人设计了一种适合于连通域小于 1000 像素的图像实现连通域标记的算法。东北大学的张云哲等人设计了使用轮廓跟踪和行程记录技术并用的称为 RCL（Run And Contour Based Labeling Algorithm）的标记算法，该算法效率和鲁棒性都较高。四川大学的刘奇琦等人设计了一种基于游程编码的连通域标记算法。中科院长春光机所的周跃等人设计了适合于对红外弱小目标的检测的基于标号回传的连通域标记算法，该算法具有占用内存小，实现简单等优点。另外华侨大学的陈柏生等，山东农业大学的刘贤喜等和华东理工大学的刘关松等都设计了一些新的二值图像连通域标记算法。

以上内容[17-33]都为本问题的解决提供了间接的思路、直接的工具、有效的方法。

7 模型的建立与求解

7.1 问题一

7.1.1 方法一（考虑零件形状特征的最大连通域求解法）

（1）基本思想

要确定零件位置，首先要确定出基准位置，基准位置以计算量较少为原则选定，基准选定时，要注意 matlab 程序数据读取是从矩阵左上角开始读取的，所以要注意图形坐标应与其的对应关系。基准建立之后，只要求出零件相对于基准两个方向的移动和一个方向的转动（这里最终以 X,Y,α 表示），则完成零件位置识别。

这种计算方法中，需要充分利用零件的具体形状信息，即本文利用了零件自身的**形状特征**，首先要充分观察零件的轮廓信息，假定了外轮廓为矩形，内部孔为圆形。因此，建模时也考虑了是否需要建立判断外轮廓是否形状的模型。思考结果是：没必要判断。因为求解结果即可证明假设是正确的，假设不正确并不能得出求解结果。

同样的，考虑到图像中有轮廓外特殊点，考虑了是否会影响位置信息的计算，或者是否需要专门去除。思考结果同样是：没必要的。一则，在求解最大连通域时，会去除一部分；二则，最后零件与基准的重合计算是基于特征点的重合，轮廓外特殊点部参与计算，所以不会也影响结果。

无论是形状证明还是轮廓外特殊点的去除，只会徒增运算时间，因此建模时均不

予考虑。

（2）模型建立及处理效果

1）在图片上建立笛卡尔坐标系

为方便编程，以图片左上点为原点，向下为 X 轴正向，向右为 Y 轴正向，1 个像素点的距离为 1，具体如图 7.1 所示。

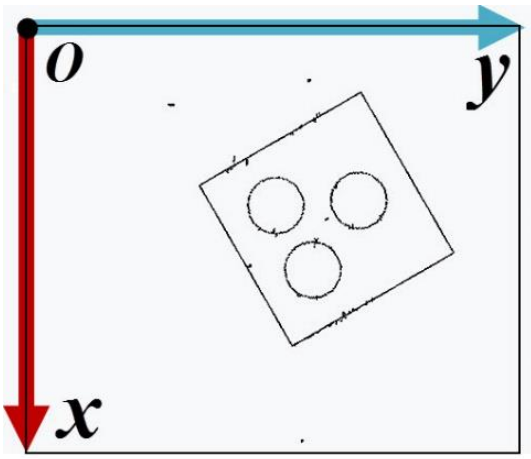


图 7.1 零件坐标系

定义搬运后零件的位置为 $(250,200,-\pi)$，其中，$(250,200)$是零件特征矩形外轮廓的形心 O_c 在所示坐标系上的位置，$(-\pi)$ 是零件的角度位置（绕 O_c 逆时针旋转为正，顺时针旋转为负），其效果如图 7.2 所示。从而建立了零件最终基准。

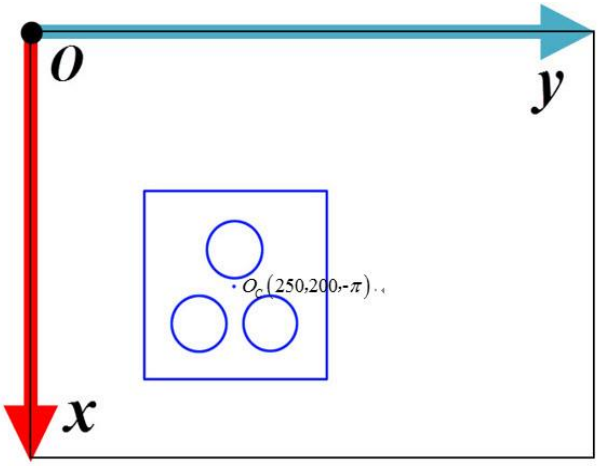


图 7.2 基准位置

2）对离散轮廓膨胀处理

对零件轮廓数据初步分析易知，特征矩形或特征圆有些地方是离散的。若直接拾取其连通域，只会得到特征圆的一小段轮廓，故先需将离散圆连通，此处采用膨胀处理[34]。膨胀处理的原理如下所述。

定义一个结构元素对二值特征圆图像进行平滑遍历，将特征圆的特征点作为参考点，将结构元素与二值特征圆进行卷积，使结构元素赋值在二值特征圆参考点进行“或”操作，从而得到经过膨胀处理后的轮廓特征矩阵，表达式如下：

$$\mathrm{X}\otimes\mathrm{S}=\left\{(\mathrm{x},\mathrm{y})\middle|\mathrm{S}_{\mathrm{xy}}\uparrow\mathrm{X}\right\} \tag{1}$$

式中，X 表示二值特征圆矩阵，S 表示结构元素，S_{xy} 表示结构元素 S 平移到二值特征圆矩阵 X 中的 (x,y) 位置。

其效果如图 7.3 所示。

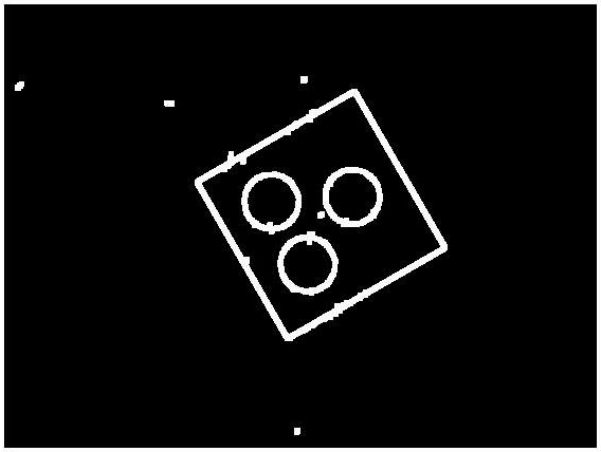


图 7.3 图像膨胀效果

3）拾取零件中的最大连通域之一

根据求解图像中的最大连通域[35]，拾取给定零件中的外边框即零件的特征矩形外轮廓，效果如下图 7.4 所示，剩余轮廓由之前两图相减可得，公式如下：

$$I_3 = I_1 - I_2 \tag{2}$$

式中，I_1 是原始图像的二值矩阵，I_2 是特征矩形外轮廓的二值矩阵，I_3 是剩余轮廓的二值矩阵。

同时求解出特征矩形的形心 $C_3(I_{3X_C}, I_{3Y_C})$，轮廓形心公式如下：

$$I_{3X_C} = \left(I_{3X_1} + I_{3X_2} + \cdots + I_{3X_k}\right) = \sum_{i=1}^{k} I_{3X_i} / k \qquad (3\text{-}1)$$

$$I_{3Y_C} = \left(I_{3Y_1} + I_{3Y_2} + \cdots + I_{3Y_k}\right) = \sum_{j=1}^{k} I_{3Y_j} / k \qquad (3\text{-}2)$$

式中，I_{3X_C} 表示特征矩形的形心的 X 轴坐标，I_{3Y_C} 表示特征矩形的形心的 Y 轴坐标, i 表示特征矩形的行数，j 表示特征矩形的列数。其效果如图 7.5 所示。

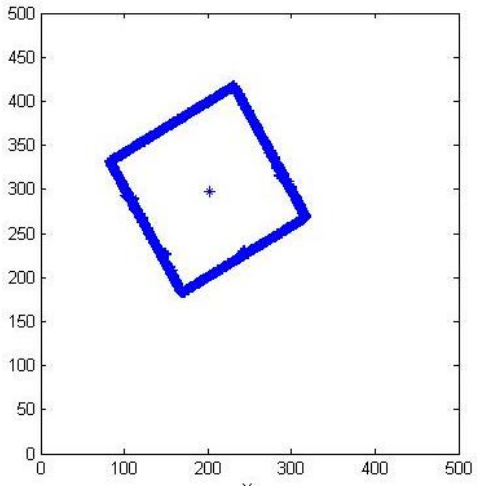


图 7.4 外边框拾取

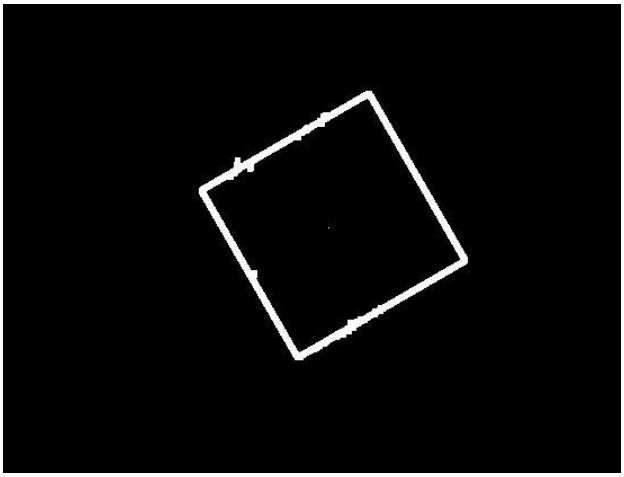


图 7.5 外边框拾取效果图

4）依次拾取零件中其他特征结构

在剩余轮廓中重复之前的步骤 2，根据给定零件的特征，总共重复 3 次，拾取到零件轮廓中的三个膨胀的特征圆轮廓，之后将膨胀特征圆轮廓的二值矩阵与原始图像二值矩阵对比，拾取出原始图像中的特征圆轮廓，其拾取公式如下：

$$I_5(i,j) = \begin{cases} 1, & I_4(i,j) + I_1(i,j) = 2 \\ 0, & I_4(i,j) + I_1(i,j) < 2 \end{cases} \qquad (4)$$

式中，I_1 是原始图像的二值矩阵，I_4 是膨胀特征圆轮廓的二值矩阵，I_5 是原始图像中特征圆轮廓的二值矩阵。

同时求解出各个特征圆的形心，其效果如下图 7.6 所示。

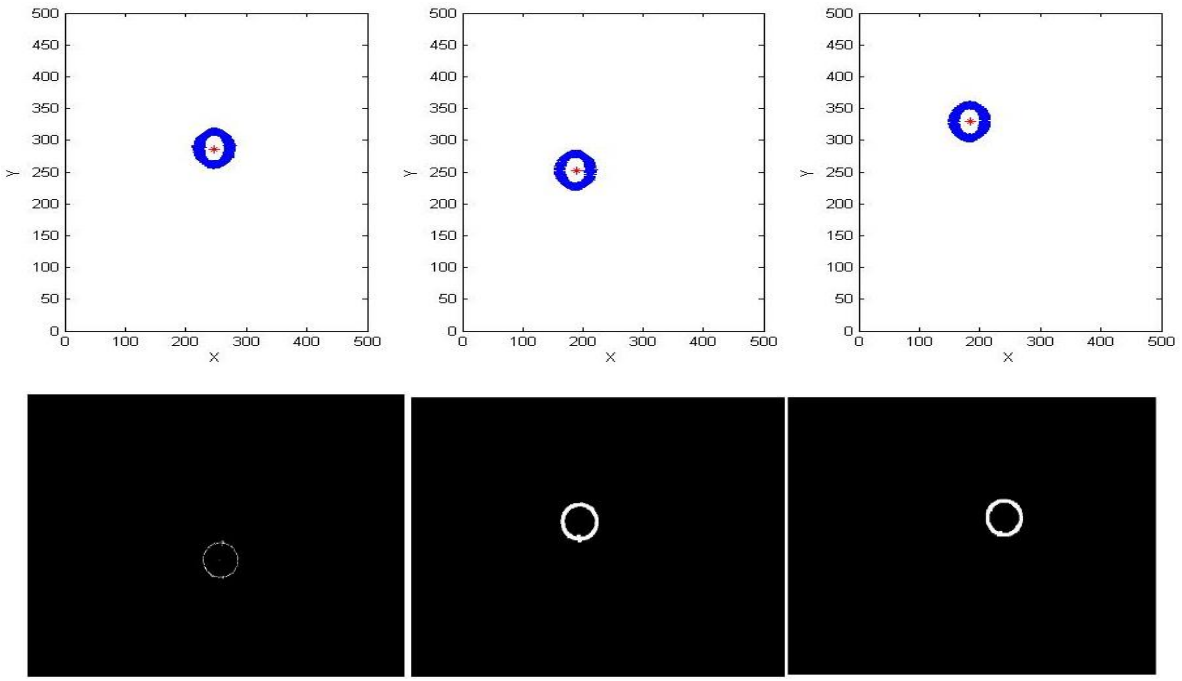


图 7.6　圆特征拾取

5）定义零件位置判别特征点

观察定位后的零件可知，零件是非对称的，而上面求解出的三个特征圆的形心位置和矩形的形心位置并不能最终确定图形角度位置。因此，需在矩形边框上取一特征点作为其角度位置判别特征点。特征矩形的边只有两种状态：水平或垂直。因此，只需矩形任一边框的中点作为其角度位置判别特征点即可。若其角度位置判别特征点落在 X 轴或 Y 轴上，即可证明其在水平或垂直状态。选取效果如下图 7.7 所示，红色点即为角度位置判别特征点。其与特征矩阵形心的连线命名为角度位置判别特征线。

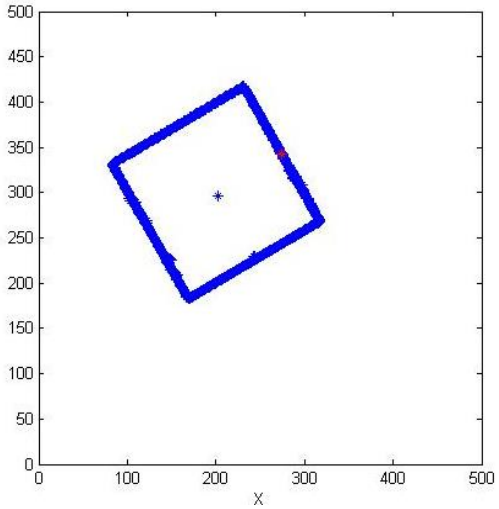


图 7.7　角度位置判别特征点拾取

6）建立确定角度位置的目标函数

为便于观察和计算，将原始图像中特征矩形的形心移动到原点后，将坐标系逆时针旋转 90°，其原始图像示意图如下图 7.8 所示，搬运后零件示意图如图 7.9 所示。搬运后的零件角度位置定义为 π 。

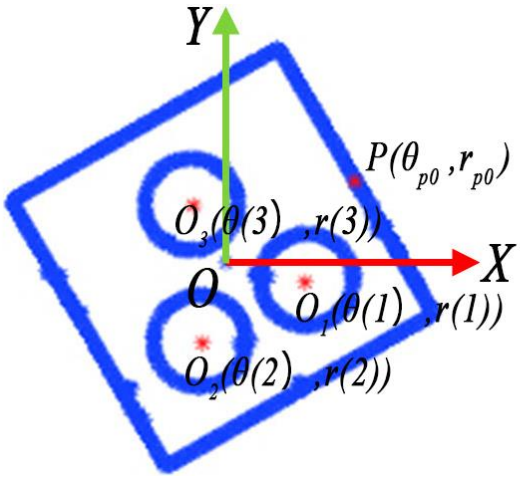


图 7.8 原始位置示意图

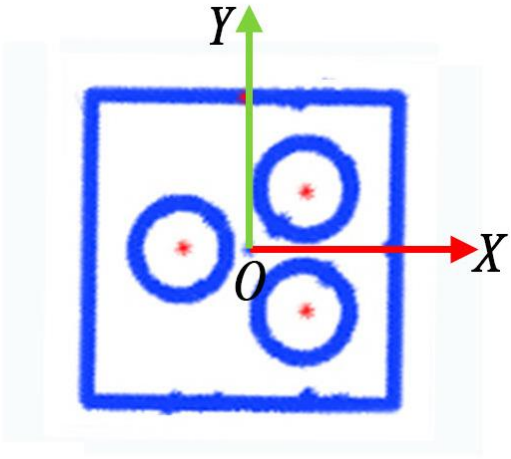


图 7.9 搬运后位置示意图

在其中一个特征圆的圆心转到 X 轴负半轴的同时，特征矩形上的角度位置判别特征点离 X 轴或 Y 轴的距离最小即为确定零件角度位置的目标函数，具体公式如下：

$$\min f(i)=\left|\theta_p(i)-\frac{k\pi}{2}\right| \tag{5}$$

$$\text{s.t.}\begin{cases}\theta_p(i)=\theta_{p0}-\theta(i)\\ -\dfrac{\pi}{4}<\theta_p(i)-\dfrac{k\pi}{2}\le\dfrac{\pi}{4}\\ i=1,2,3\end{cases}$$

式中，$f(i)$ 是第 i 次转动后角度位置判别特征线与 X 轴或 Y 轴的最小夹角，θ_{p0} 为角度位置判别特征点的角度位置，$\theta(i)$ 为第 i 个特征圆的角度位置，$\theta_p(i)$ 为第 i 次转角后 p 点的角度位置，k 为整数。

上述目标函数的求解流程详见下图 7.10 所示。

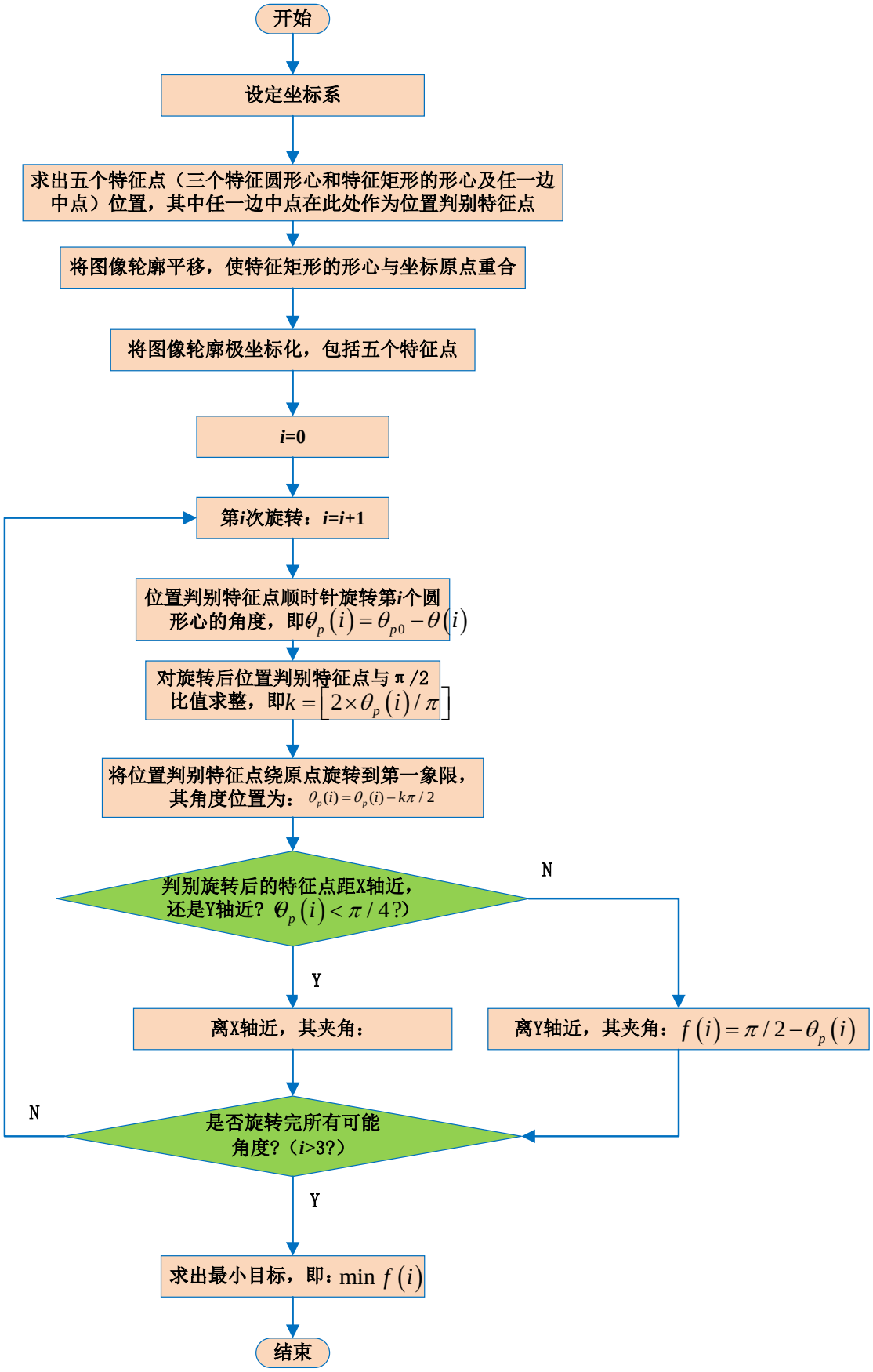


图 7.10 角度位置的目标函数求解流程图

7）零件位置

对上面的目标函数求解，在 $i=p$ 时，目标函数最小，即原始零件的角度位置为 $\theta(p)$。综上，零件的位置坐标为 $(x_c, y_c, \theta(p))$。

将零件旋转 $\pi-\theta(p)$ 后再移动(x_0-x_c， y_0-y_c)即可到达搬运后的位置。此处旋转角度若为正，代表逆时针旋转；若为负，代表顺时针旋转。

搬运后的效果图见下图 7.11 所示，这里定义搬运后的零件位置为 $(250, 200, -\pi)$。

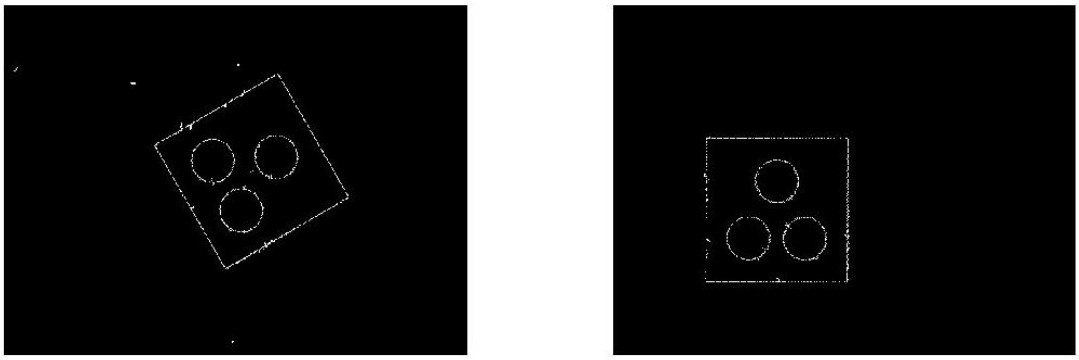


图 7.11 零件搬运前后效果图

为显示效果，将原始零件和搬运后的零件放入一个图中，并摆正坐标系，见下图 7.12 所示。

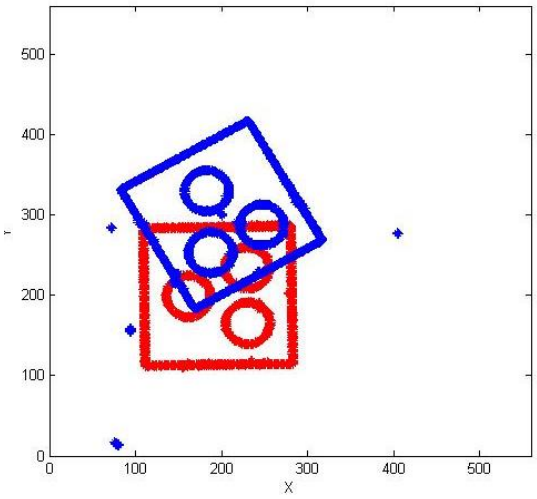


图 7.12 零件搬运前后效果对比图
（红色为摆放位置，蓝色为原始位置）

运行程序，零件的原始位置信息求解结果如下表：

表 7.1 问题一零件位置信息

模型	X	Y	α （180°/π）
零件位置	202.6573	296.6206	2.0860
最终摆放位置	250	200	-π

整个问题一的建模过程的流程图详见下图 7.13 所示。程序源代码参见附录一。

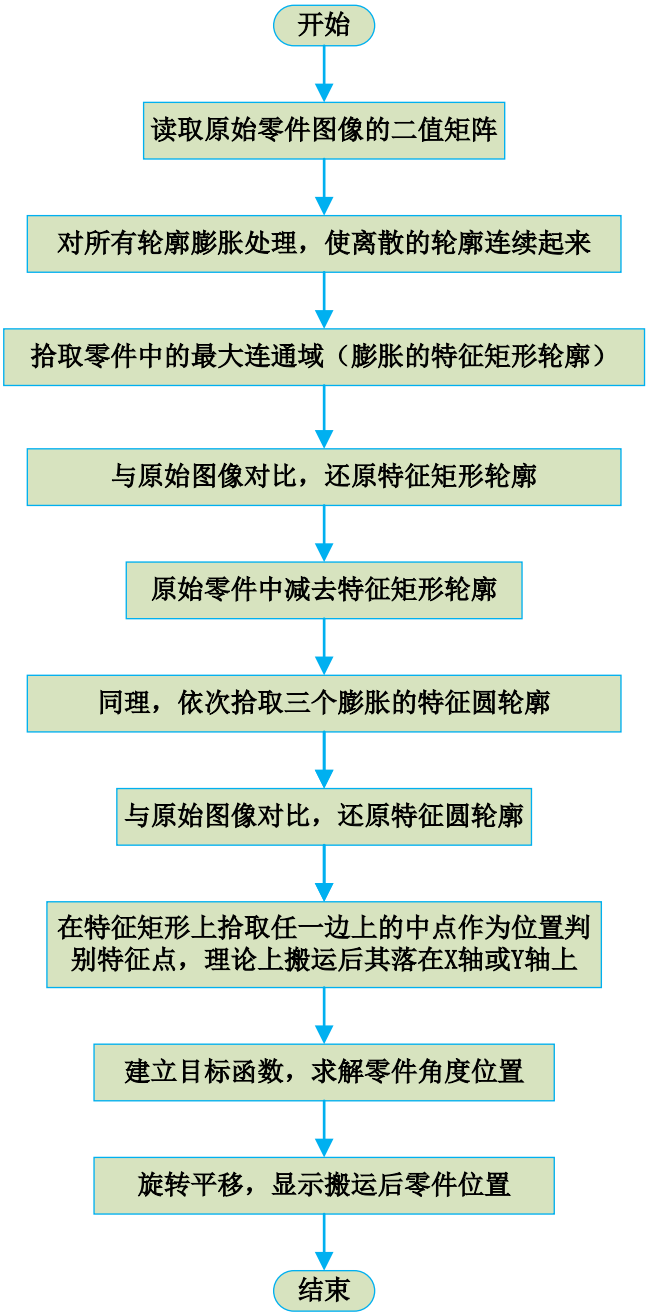


图 7.13 问题一的建模过程

7.1.2 方法二（不考虑零件轮廓的通用方法）

（1）基本思想

第一步:构建零件基准——重建出具体尺寸的零件轮廓图，作为零件识别的模板数据。

第二步:建立目标函数——依次计算数据 DATA1 中每个轮廓点到基准所有轮廓点中的最小距离，以所有最小距离的平方和最小为目标，以位移 X 和位移 Y 以及旋转角 α 为设计变量，建立最优化模型，优化求解该模型。

（2）模型及处理效果

1）重构零件轮廓，做出基准；

2）读取 DATA1 的数据，找出 DATA1 中值为 1 的点的坐标；

3）构建以求解移动后的 DATA1 轮廓与基准轮廓之间的最小距离为目标函数的数学模型；

$$\min f(\mathrm{i}) = \sum \left| \left(X(\mathrm{i})' - x(\mathrm{i}) \right)^2 + \left(Y(\mathrm{i})' - \mathrm{y}(\mathrm{i}) \right)^2 \right|$$

$$s.t. \begin{cases} i\text{是基准或零件轮廓上数值为1的点的个数} \\ (X(\mathrm{i})', Y(\mathrm{i})') = \mathrm{g}(X(\mathrm{i}), Y(\mathrm{i}),\ \mathrm{x}_0,\ \mathrm{y}_0,\ \alpha_0) \\ \mathrm{g}(\mathrm{z})\text{是坐标变换公式} \\ \mathrm{x}_0,\ \mathrm{y}_0,\ \alpha_0\text{为所求的零件位置坐标} \end{cases} \tag{6}$$

4）用模拟退火法求解模型。

流程见图 7.14，程序见附录二。

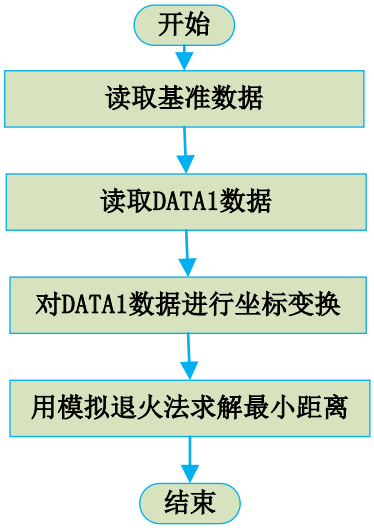


图 7.14 问题一的另一种建模过程

搬运结果如图 7.15 所示，零件位置信息如表 7.2 所示。

表 7.2 问题一零件位置信息

模型	X	Y	α （180°/π）
零件位置	295.4292	203.5155	-1.0865
最终摆放位置	200	250	**-π/2**

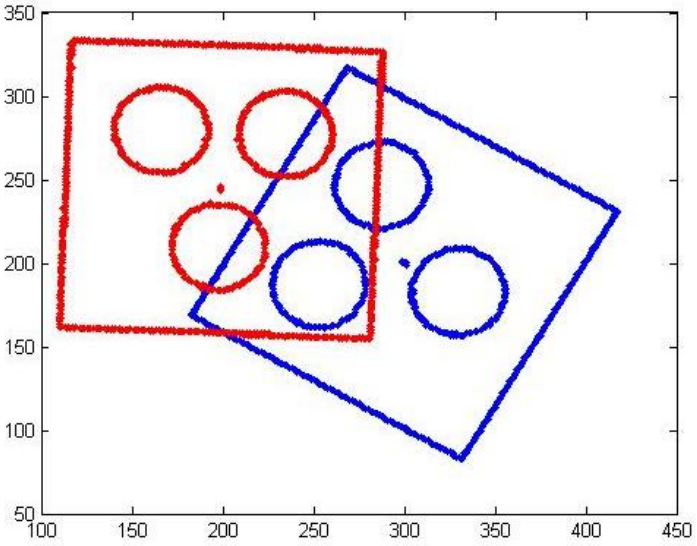


图 7.15 零件搬运前后效果对比图
（红色为摆放位置，蓝色为原始位置）

比较另种方法，位移上基本相等，但是 X 和 Y 坐标轴相反，转角 α 上正好差 360°，这与两种方法建立的坐标系不同是相关的。

7.1.3 算法评价

算法执行时间需通过一句该算法编制的程序在计算机上运行时所消耗的时间来衡量，而一般度量一个程序在计算机上运行所消耗的时间采用事后统计的方法，即计算程序在某一计算机上开始计算到计算结束所采用的时间。这种方法是度量程序效率的直观方法，但是由于计算机硬件不同、环境不同等因素，其所用时间有较大差异，因此容易掩盖算法本身的优势。因此，我们采用事前分析估计方法，通过评估算法的控制结构及原操作，形成一个预计的评估结果。其主要指标有 4 项：时间复杂度、空间复杂度、最坏时间复杂度、最坏空间复杂度。由于在本次的算法中，数值的迭代等为有限数集，因此最坏时间复杂度及最坏空间复杂度影响较小，因此我们选取时间复杂度及空间复杂度两项作为评判标准。

（1）时间复杂度

一个算法执行所耗费的时间，从理论上是不能算出来的，必须上机运行测试才能

知道。但我们不可能也没有必要对每个算法都上机测试，只需知道哪个算法花费的时间多，哪个算法花费的时间少就可以了。并且一个算法花费的时间与算法中语句的执行次数成正比例，哪个算法中语句执行次数多，它花费时间就多。一个算法中的语句执行次数称为语句频度或时间频度。记为$T(\mathrm{n})$。在刚才提到的时间频度中，n 称为问题的规模，当 n 不断变化时，时间频度$T(\mathrm{n})$也会不断变化。但有时我们想知道它变化时呈现什么规律。为此，我们引入时间复杂度概念。 一般情况下，算法中基本操作重复执行的次数$T(\mathrm{n})$ 是问题规模 n 的某个函数，用$T(\mathrm{n})$表示，若有某个辅助函数 $f(n)$，使得当 n 趋近于无穷大时，$T(\mathrm{n})/f(n)$的极限值为不等于零的常数，则称 $f(n)$是$T(\mathrm{n})$的同数量级函数。记作$T(\mathrm{n})=O(f(n))$,称$O(f(n))$为算法的渐进时间复杂度，简称时间复杂度。

在各种不同算法中，若算法中语句执行次数为一个常数，则时间复杂度为$O(1)$，另外，在时间频度不相同时，时间复杂度有可能相同，如$T(\mathrm{n})=n^2+3n+4$与$T(\mathrm{n})=4n^2+2n+1$它们的频度不同，但时间复杂度相同，都为$O(\mathrm{n}^2)$。 按数量级递增排列，常见的时间复杂度有：常数阶$O(1)$,对数阶$O(\log_2 n)$,线性阶$O(\mathrm{n})$,线性对数阶$O(\mathrm{n}\log_2 n)$,平方阶$O(\mathrm{n}^2)$，立方阶$O(\mathrm{n}^3)$,...，k 次方阶$O(\mathrm{n}^k)$,指数阶$O(2\mathrm{n})$。随着问题规模 n 的不断增大，上述时间复杂度不断增大，算法的执行效率越低。

常见的算法时间复杂度由小到大依次为：$O(1)<O(\log_2 n)<O(\mathrm{n})<O(\mathrm{n}\log_2 n)<O(\mathrm{n}^2)<O(\mathrm{n}^3)<\cdots<O(2\mathrm{n})<O(\mathrm{n}!)$。

求解算法的时间复杂度的具体步骤是：

1）找出算法中的基本语句；

算法中执行次数最多的那条语句就是基本语句，通常是最内层循环的循环体。

2）计算基本语句的执行次数的数量级；

只需计算基本语句执行次数的数量级，这就意味着只要保证基本语句执行次数的函数中的最高次幂正确即可，可以忽略所有低次幂和最高次幂的系数。这样能够简化算法分析，并且使注意力集中在最重要的一点上：增长率。

3）用大 *O* 记号表示算法的时间性能。

将基本语句执行次数的数量级放入大 *O* 记号中。

因此我们规定程序的分析法则如下：

1）对于一些简单的输入输出语句或赋值语句,近似认为需要$O(1)$时间；

2）对于顺序结构,需要依次执行一系列语句所用的时间可采用大 O 下"求和法则"；

求和法则:是指若算法的 2 个部分时间复杂度分别为$T_1(\mathrm{n})=O(f(n))$和$T_2(\mathrm{n})=O(f(n))$,则 $T_1(n)+T_2(n)=O(max(f(n),g(n)))$.特别地,若$T_1(m)=O(f(m))$,$T_2(n)=O(g(n))$,则 $T_1(m)+T_2(n)=O(f(m)+g(n))$。

3）对于选择结构,如 if 语句,它的主要时间耗费是在执行 then 字句或 else 字句所用的时间,需注意的是检验条件也需要$O(1)$时间；

4）对于循环结构,循环语句的运行时间主要体现在多次迭代中执行循环体以及检验循环条件的时间耗费,一般可用大 O 下"乘法法则"；

乘法法则：是指若算法的 2 个部分时间复杂度分别为$T_1(\mathrm{n})=O(f(n))$和$T_2(\mathrm{n})=O(f(n))$,则 $T_1*T_2=O(f(n)*g(n))$。

5）对于复杂的算法,可以将它分成几个容易估算的部分,然后利用求和法则和乘法法则技术整个算法的时间复杂度。

另外还有以下 2 个运算法则:

6）若$\mathrm{g}(\mathrm{n})=O(f(n))$,则$O(f(n))+O(g(n))=O(f(n))$;

7）$O(Cf(n))=O(f(n))$,其中 C 是一个正常数。

下面举例对时间复杂度进行说明：

表 7.3 时间复杂度评估示例

复杂度	命令示例	解释

$O(1)$	Temp=i;
	i=j;
	j=temp;
		以上三条单个语句的频度均为 1，该程序段的执行时间是一个与问题规模 n 无关的常数。算法的时间复杂度为常数阶，记作$T(n)=O(1)$

$O(n)$	**For** (i=1;i<n;i++)
	{
	y=y+1;
	for (j=0;j<=(2*n);j++)
	x++;
	}
		语句 1 的频度为 2,语句 2 的频度为 n,语句 3 的频度为,语句 4 的频度为 n-1,语句 5 的频度为 n-1,
		所以$T(\mathrm{n})=2+n+3(n-1)=4n-1=O(n)$

$O(\mathrm{n}^2)$	sum=0; （1 次）
	for(i=1;i<=n;i++) （n+1 次）
	for(j=1;j<=n;j++) （n^2 次）
	sum++; （n^2 次）
		因为$\theta(2\mathrm{n}^2+n+1)=n^2$（$\theta$即：去低阶项，去掉常数项，去掉高阶项的常参得到），所以$T(\mathrm{n})=O(n^2)$;

$O(n^3)$	**for**(i=0;i<n;i++)
	{
	For(j=0;j<I;j++)
	{
	For(k=0;k<j;k++)
	X=x+2;
	}
	} for(k=0;k<j;k++)
		当 i=m,j=k 的时候,内层循环的次数为 k 当 i=m 时,j 可以取 0,1,...,m-1 ，所以这里最内循环共进行了 0+1+...+m-1=(m-1)m/2 次所以,i 从 0 取到 n, 则循环共进行了:0+(1-1)*1/2+...+(n-1)n/2=n(n+1)(n-1)/6 所以时间复杂度为$O(n^3)$。

（2）算法的空间复杂度

类似于时间复杂度的讨论，一个算法的空间复杂度(Space Complexity)S(n)定义为该算法所耗费的存储空间，它也是问题规模 n 的函数。渐近空间复杂度也常常简称为空间复杂度。

空间复杂度(Space Complexity)是对一个算法在运行过程中临时占用存储空间大小的量度。一个算法在计算机存储器上所占用的存储空间，包括存储算法本身所占用的存储空间，算法的输入输出数据所占用的存储空间和算法在运行过程中临时占用的存储空间这三个方面。算法的输入输出数据所占用的存储空间是由要解决的问题决定的，是通过参数表由调用函数传递而来的，它不随本算法的不同而改变。存储算法本身所占用的存储空间与算法书写的长短成正比，要压缩这方面的存储空间，就必须编写出较短的算法。算法在运行过程中临时占用的存储空间随算法的不同而异，有的算法只需要占用少量的临时工作单元，而且不随问题规模的大小而改变，我们称这种算法是“就地\"进行的，是节省存储的算法，如这一节介绍过的几个算法都是如此；有的算法需要占用的临时工作单元数与解决问题的规模 n 有关,它随着 n 的增大而增大，当 n 较大时，将占用较多的存储单元，例如将在第九章介绍的快速排序和归并排序算法就属于这种情况。

如当一个算法的空间复杂度为一个常量,即不随被处理数据量 n 的大小而改变时，可表示为$O(1)$；当一个算法的空间复杂度与以 2 为底的 n 的对数成正比时，可表示为$O(10\log_2 n)$；当一个算法的空 I 司复杂度与 n 成线性比例关系时，可表示为$O(\mathrm{n})$.若形参为数组，则只需要为它分配一个存储由实参传送来的一个地址指针的空间，即一个机器字长空间；若形参为引用方式，则也只需要为其分配存储一个地址的空间，用它来存储对应实参变量的地址，以便由系统自动引用实参变量。

（3）评判结果

由于文件程序较多，列举问题一的部分程序进行时间复杂度的计算。其计算过程如下：

表 7.4 问题一的部分程序时间复杂度计算

序号	程序命令	时间复杂度
1	clear	1 次
2	clc	1 次
3	tic	1 次
4	load('DATA1.mat')	1 次
5	x1=D1; %数据读取	1 次
6	figure	1 次
7	subplot(1,2,1)	1 次
8	imshow(x1)	1 次
9	title('搬运前零件位置')	1 次
10	[m,n]=size(x1);	1 次
11	x0=x1;	1 次
12	x1=imdilate(x1,ones(5)); %图像膨胀	1 次
13	%%%正方形提取	1 次
	img=maxLianTongYu(x1); %求图像最大连通域	1 次

14	k=1;　%提取所有位置点	1 次
15	for i=1:m	m 次
16	for j=1:n	M×n

17	if img(i,j)+x0(i,j)==2
	a0(1,k)=i;
	a0(2,k)=j;
	k=k+1;
		m×n×k

18	end	1 次
19	end	1 次
20	end	1 次
21	Square_center=mean(a0,2); %正方形形心点坐标	1 次
22	%%%第一圆提取	1 次
23	x1=x1-img; %剩余图像	1 次
24	img1=maxLianTongYu(x1); %求图像最大连通域	1 次
25	img11=zeros(m,n); %图像还原	1 次
26	k=1;	1 次
27	for i=1:m	m 次
28	for j=1:n	m×n 次

29	if img1(i,j)+x0(i,j)==2
	img11(i,j)=1;
	a1(1,k)=i;
	a1(2,k)=j;
	k=k+1;
		m×n×k 次

30	end	1 次
31	end	1 次
32	end	1 次
33	circular_center1=mean(a1,2); %圆 1 形心点坐标	1 次
总计		2m×n×k+2m×n+2m+16

因此此段程序的时间复杂度为$O(n)=2m\times n\times k+2m\times n+2$m+16，其中 m 与 n 的取值为 420、560，k 值取 2，则$O(n)$的值为 1412056，计算剩余程序的时间复杂度，则整个程序的时间复杂度为:5035854。

计算整个程序的空间复杂度为程序中所需储存的变量为基准，则整个程序的空间复杂度为 285687。

（4）用程序评价

另外，可用 matlab 中的 tic—toc 命令对程序运行总时间进行计时或者用 profile 命令察看单个程序的运行时间和调用次数，对两种方法的执行结果分别显示如下表。

表 7.5 Tic-toc 命令测试的运行时长

模型	时长（s）
方法一	0.604257
方法二	30.935356

表 7.6 Profile 命令评估程序（方法一）

Function	Calls	Total Time	Self Time
imshow	2	0.201s	0.000s
newplot	4	0.155s	0.000s
graphics\private\clo	4	0.155s	0.000s
cla	4	0.155s	0.000s
newplot>ObserveAxesNexplot	4	0.155s	0.000s
setdiff	8	0.123s	0.016s
setdiff>setdifflegacy	8	0.107s	0.032s
maxLianTongYu	4	0.093s	0.032s
ismember	18	0.079s	0.016s

表 7.7 Profile 命令评估程序（方法二）

Function	Calls	Total Time	Self Time
Opt_Simu	1	31.734s	1.828s
@(x)myfun2(x,d,D)	50002	29.547s	0.860s
Myfun2	50002	28.687s	15.373s
repmat	200008	13.314s	13.315s
Mu_lnv	50001	0.358s	0.358s

综上可见，在同样的设备上，方法二的计算时间为 0.604 秒，方法一的计算时间为 31.935 秒。所以方法二比方法一更快速高效。

7.2 问题二

7.2.1 方法一（考虑零件间位置特征的连通域分割法）

（1）基本思想

求解这一问题时，关键考虑两零件的分割，以二值矩阵为研究对象，则关键考虑怎么对各坐标数值分类。分割时着重处理边界间隙。只要有效促成图像分割，即可采用问题一的方法分别对问题二中的两零件进行位置识别。方法中，充分利用了两零件不互相接触这一**位置特征**。

（2）模型及处理效果

1）原图填充孔洞

为了区别同一图像中的多个零件而不至于丢失个别零件的特征，将对原图进行填充孔洞处理[36]，使零件外轮廓内部形成一个整体。

图像填充的原理如下所述，将源图（二值图）反色，遍历。遇到第一个非 0 值是停下来，从这个点开始，进行种子算法，将所有与之相连的非 0 点设置成一个块，然后将这个区域内所有的值删除（设定为 0），进行反色。加入到源图中，其计算公式如下：

$$F(\mathrm{x},y)=\begin{cases}1-I(x,y) & (x,y)在I的边界上\\ 0 & 其他\end{cases} \tag{7}$$

其效果如下图 7.16 所示。

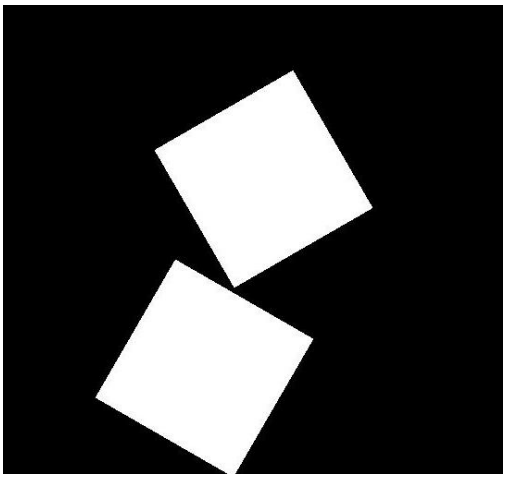


图 7.16　图像填充效果

2）图像腐蚀

考虑到零件可能存在接触的情况，直接用最大连通域法已经不能把零件分割开来，所以首先对填充的图像进行腐蚀处理，使其边缘分割开。图像腐蚀的原理如下：

图像腐蚀算法与图像膨胀算法理念相同，使用结构元素对二值图像进行遍历，然后做“与”操作，如果都为 1，结果图像的该元素变为 1,否则为 0，其公式如下所示：

$$\mathrm{X\Theta S}=\left\{(\mathrm{x},\mathrm{y})\middle|\mathrm{S_{xy}}\subseteq\mathrm{X}\right\} \tag{8}$$

其效果见图 7.17 所示。

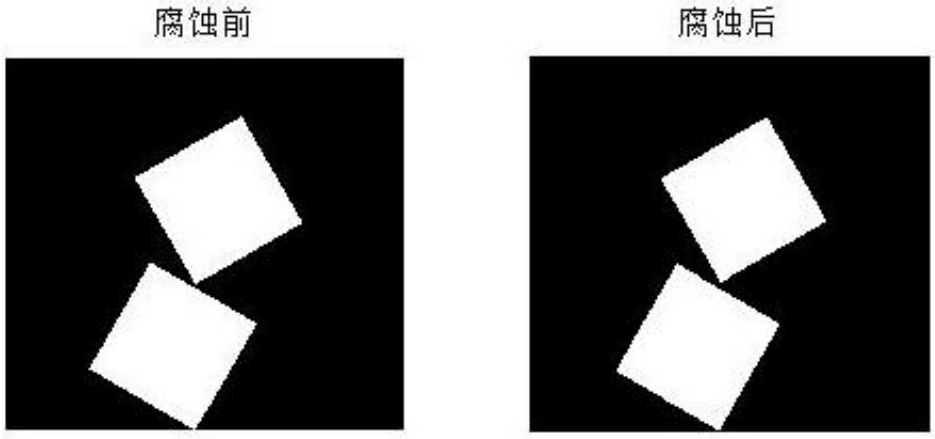


图 7.17 腐蚀前后效果对比

3）对腐蚀后的图像求最大连通域

经上述腐蚀处理后，零件边缘已分割，先对腐蚀后的图像求取最大连通域可拾取其中一个零件，再在腐蚀后的图像中剔除拾取的零件得到新的图像，之后对新的图像求取最大连通域可拾取第二个零件，重复上述步骤可拾取第三、第四等个零件。设定阈值可终止以上循环步骤。

其效果如下图 7.18 所示。

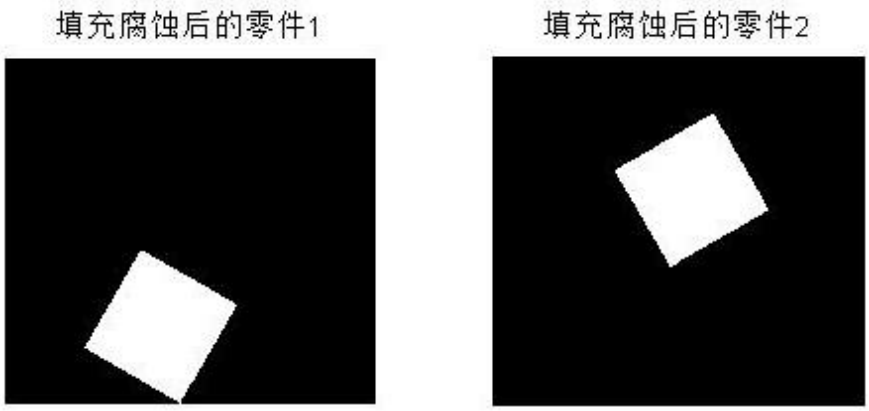


图 7.18 零件图分割效果

4）图像膨胀

上述区分开的填充腐蚀后的零件整体上比原图中的小，为提取原图中的零件，需将上述填充腐蚀后的零件图进行膨胀处理。

其效果如下图 7.19 所示。

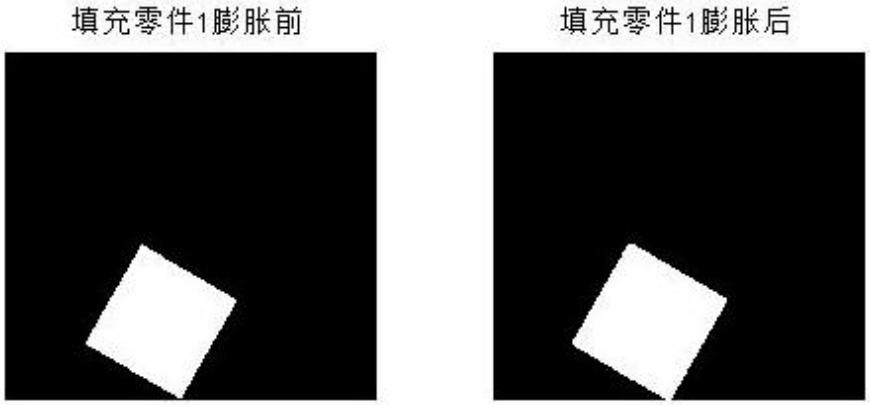


图 7.19 膨胀前后效果对比

5）提取单个零件

将上述膨胀后的零件图像与原始图像对比，提取出单个零件，对单个零件的图像保存。

6）零件位置求解

运用第一问中的模型对上述保存的单个零件的图像分析，求解各个零件的位置。其求解结果如下图 7.20 和表 7.8 所示。

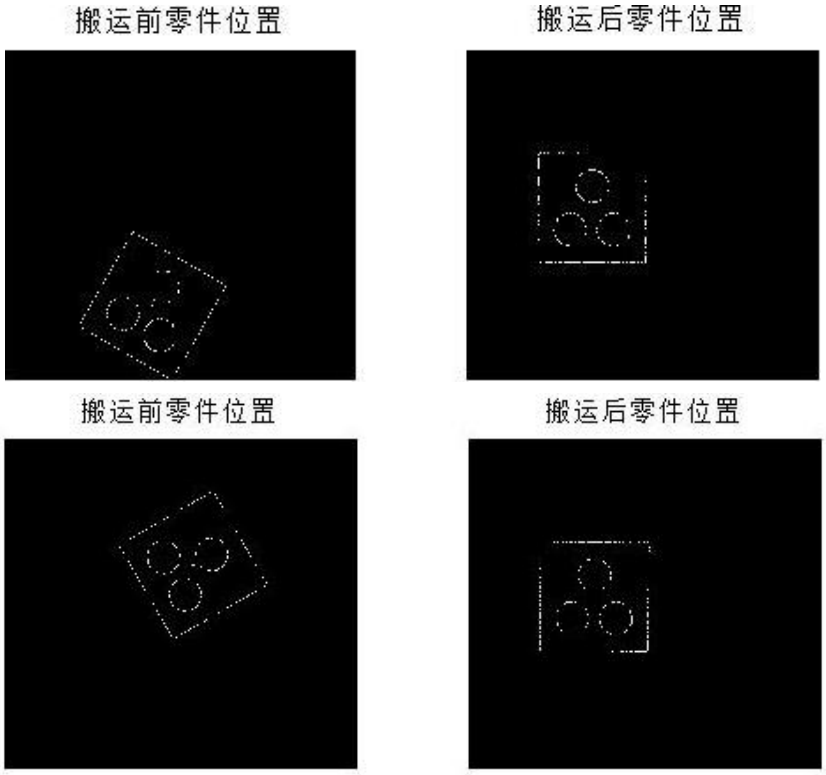


图 7.20 求解结果

表 7.8　求解结果

模型	X	Y	α （180°/π）
零件 1 位置	405	236	2.6224
零件 2 位置	200	300	2.1032
最终摆放位置	250	200	-180

整个建模过程的流程图详见下图 7.21 所示。程序源代码参见附录三。

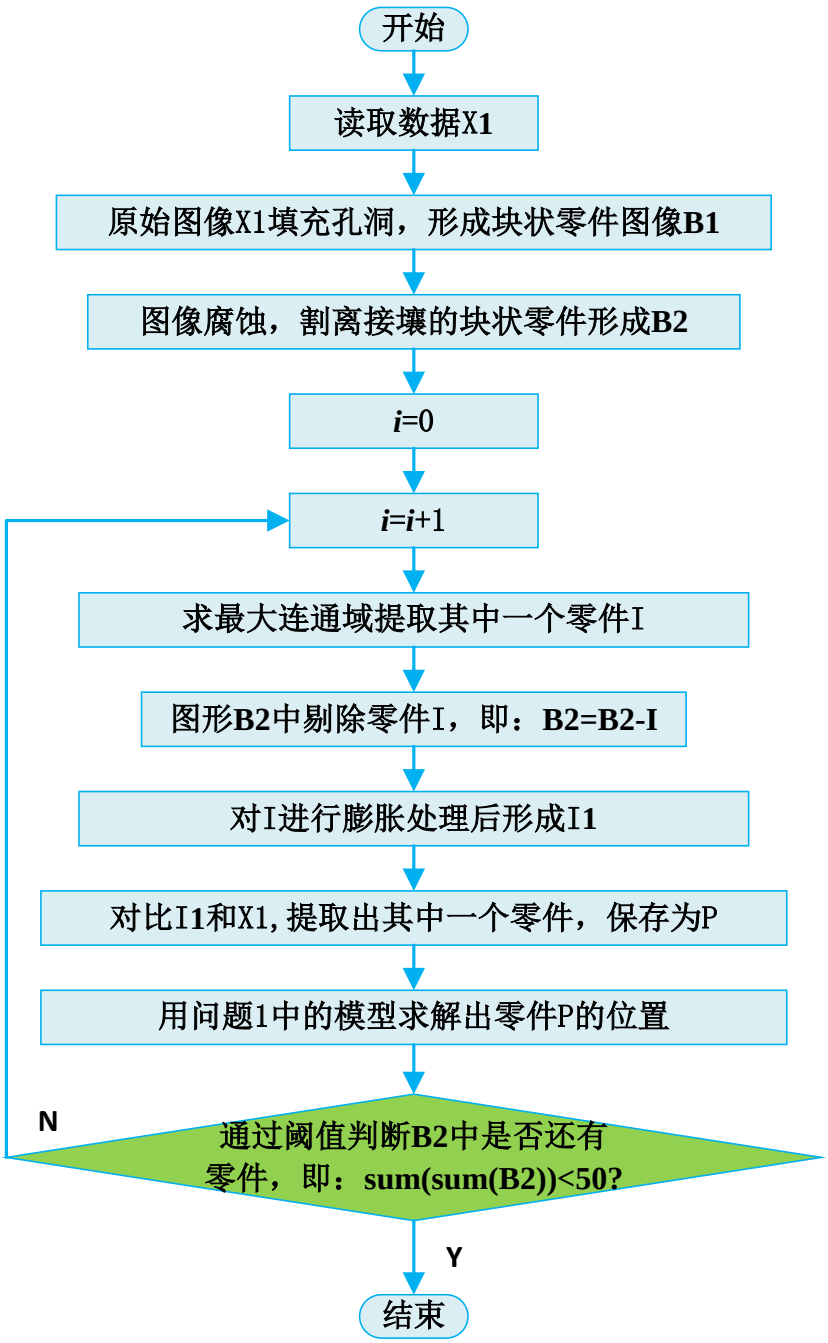


图 7.21　问题二建模流程

7.2.2 方法二

基本思想：

step1:通过圆检测（hough 检测）获取到 6 个圆中心点，将 6 个中心点坐标分两类，通过比对零件模板信息找到每个零件的近似中心点坐标，以中心点坐标为中心，以零件模板正方形对角线的一半为半径，将该区域数据信息分割出来；

step2:重新依次利用第一问中第二步的方法求解零件位置信息。

7.2.3 方法三

基本思想：

step1:在求出零件正方形边长后,改进利用 hough 检查原理，直接识别出两个正方形的中心点位置及旋转角。根据该信息提取出每个零件轮廓数据；

step2:重新依次利用第一问中第二步的方法求解零件位置信息。

但受限于比赛时间，本文不对方法二与方法三进行程序编程与实际操作。

8 模型评价与改进

8.1 模型的评价

8.1.1 模型的优点

针对问题一所提出的第一种模型在能实现零件位置的识别，计算量小，计算速度快，速度完全适应生产需求。

针对问题一所提出的第二种模型在零件位置识别中能取得较好的识别效果，定位准确，计算轮廓清晰，而且无需考虑轮廓形状，所以适用范围广。

针对问题二所提出的模型建立在问题一模型的基础上，只要分割出不同的零件，即可用问题一的两种模型中的任意一种求解位置信息，求解速度快，程序简单，方便实现。单从解题这个角度讲，这一方法是高效快速实用的。

8.1.2 模型的缺点

针对问题一所提出的第一种模型需要依赖于零件的形状特征,零件形状改变之后，本模型即需要做相应修改和调整，所以适用于特定零件的位置识别，适用范围较窄，有待提升。

针对问题一所提出的第二种模型需要进行多次比较之后才能得出最优解，所以计算量较大，计算速度慢，程序复杂，在生产线响应速度要求较高的场合下实用性较低。

针对问题二所提出的模型比较依赖于两个零件之间的位置特征，如果零件位置重叠，那么辨识率会降低。所以这一模型适用于同一平台上，零件不重叠的情况，并不适用于多零件重叠的情况。

综上，模型优缺点列表如下：

表 8.1 模型优缺点对比

模型	优点	缺点
问题一模型一	计算量小，计算速度快	依赖形状特征，应用范围窄
问题一模型二	应用范围宽	计算量大，计算速度慢
问题二模型	计算量小，计算速度快	依赖位置特征，应用范围窄

8.2 模型的改进方向

针对问题一所提出的第一种模型，扩展其适用范围是必须的，如特征提取不再建立在假设的前提下，而是能自动提取不同零件的形状特征，那么适用范围必然拓宽，这要求在模型中的前部分加入能自动唯一标识零件形状特征的模型。方可不再依靠人为判断零件形状来制定算法。

针对问题一所提出的第二种模型，简化计算模型，提升计算速度为必然趋势，其中，寻优算法可找改为其他计算步骤更少的算法，另外，如果能在寻优之前寻找特征点，如点、线能标识零件的几何集合，然后对几何集合进行寻优，计算步数则会减小。

针对问题二所提出的模型依赖于零件间的相对位置特征，假设零件更近点但没有重叠，则需要在图像分割算法更为精确，所以阈值确定是动态的，远，则阈值大点，提高计算速度，近，则阈值小点，提高分割精度。再假设，零件近到有重叠，则图像分割不再是最有效的算法，需要通过特征提取或者聚类等算法在图中寻找与问题一相同的轮廓形状。

9 应用前景与展望

9.1 模型应用前景

综上内容，本文模型具有一定的应用前景。而且，参赛队员本人认为问题一的第一种模型更有推广意义。

首先，随着智能化生产的快速发展，高效率高速度对图像处理技术的速率也提出了要求。从满足生产速度上讲，这种模型是有效的。

其次，在实际生产中的图像处理中，也是没必要考虑零件形状的，因为在一般的

生产条件下，单一的生产线加工或者处理的是单一类型的零件，而且零件信息在生产线搭建之前就是已知的、给定的，而且会针对零件形状搭建生产线，并依据零件形状制定位置识别算法。所以，图像处理模块在生产线上工作时，没必要再处理形状信息，而是直接依据进行定位运算。

再次，具体的形状特征可以直观察觉，比如本题中，观察到外围是矩形，内部有三个圆形，建立模型时，直接以矩形和圆形为假设前提建立模型并进行处理，得出结果了则证明假设是正确的，如果假设错误，并不能得出计算结果。所以，没必要事先证明和运算形状特征，运算反而是浪费计算时长。

综上，针对问题一提出的第一种位置识别框架还是有应用前景的，只是需要针对具体形状的零件建立具体的模型。

9.2 展望

由于本题中问题一和问题二给出的都是较为简单环境的位置识别的情况，所以，针对更为复杂环境下的位置识别，本文展开了深度展望，同时，也是对第二阶段比赛的提前预测与准备。

在实际生产工作中，还需要考虑如下几种情况：

（1）对单一零件的位置识别而言，需要考虑：

1）零件轮廓复杂的情况；
2）对定位精度较高的情况；
3）零件需要在空间内而非平面内找正的情况。

（2）对多个零件的位置识别而言，需要考虑：

1）多个零件有重叠的情况；
2）一条生产线上需要抓取多种不同零件的情况

在这些复杂情况下编制相应的位置定位模型，更具有应用和实用价值。

9.2.1 零件轮廓复杂的情况

（1）问题提出

本题中给出的零件是极其规则的，正方形、圆形的组合，所以，模型建立时，可通过圆心计算、直线计算等使计算过程简单，实际中，零件的种类多种多样、五花八门。

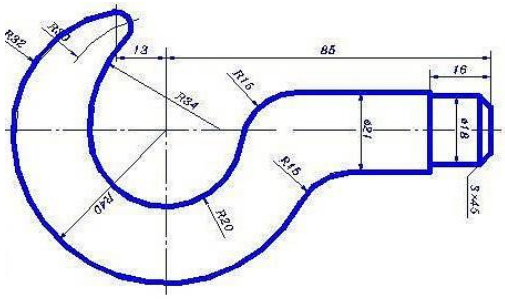


图 9.1 复杂轮廓零件

（2）解决方案及模型

如果生产线上需要抓取的是这样的零件，在建模时，需要充分了解零件的形状特征，然后进行特征提取，使得提取的特征能唯一标识零件，通常用角点检测实现，因为角点和点的组合通常能对图像进行唯一标识。

同样的，求解的是零件的位置，所以需要制定位置基准，相对于位置基准，错位的零件需要两个移动和一个转动来实现位置矫正。据此，可建立数学模型。

以特征点个数和坐标值为约束，建立求解零件各个特征点经过两个方向的位移和一个方向的转动之后与基准位置特征点做差球平方和以求解相对距离的最小值的优化模型，最终可求解出两个位移值和一个角度值，即为零件的位置信息。

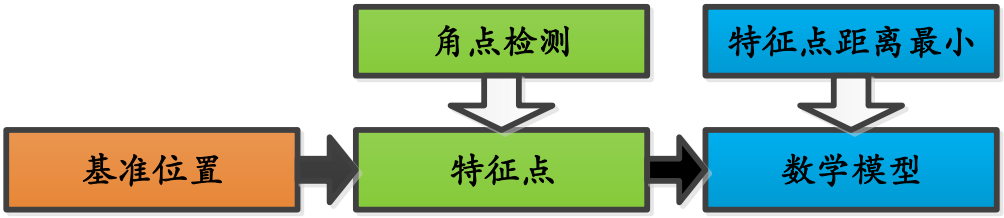


图 9.2 解决方案

（3）预期效果及评价

求解效果一定程度上决定于检测到的特征点的个数。对于不规则图像，由于特征点计算比全点计算计算速度快，应该效果不错，但是对于对称的复杂轮廓零件，比如齿轮等，会由于特征点的相似或者重合而导致计算错误或者多解。

9.2.2 对定位精度较高的情况

（1）问题提出

对零件位置识别的精度有时会影响到加工的精度，图像的轮廓精度不高时，往往影响识别精度，特别是对外表面粗加工后的毛坯件以及一些微小件，这些均对定位精

度提出要求。

（2）解决方案及模型

二值图像位置识别时，是通过有限的边界点计算定位精度，如果要提高精度，可以分别在基准和零件的各边界点之间进行插值，然后以求解所有点经过两个方向的位移和一个方向的转动之后与基准的所有点做差平方的距离的最小值为目标建立模型求解。

（3）预期效果及评价

插值会一定程度的提高定位精度，但是不可否认，插值越多，计算速度会越慢，可能难以满足工业上的响应速度要求。

9.2.3 零件需要在空间内而非平面内找正的情况

（1）问题提出

本题给定的零件是在平面上，而往往有的零件在空间上有了位置变动，针对这种零件的位置定位，需要对本文的模型进行进一步扩展。

图 9.3 空间位置错落

（2）解决方案及模型

三维重构建立基准，依然需要拿零件的特征点与基准的特征点做差，区别在于：空间变换中，错落零件相对于基准是 6 个自由度的变换，包括 3 个移动方向和 3 个转动方向，所以，建立的数学模型是零件进行 6 个自由度的变换之后，建立求解零件特征点与基准特征点距离最小值的优化模型，求解出的 6 个自由度即为零件的空间位置信息。

图 9.4 解决方案

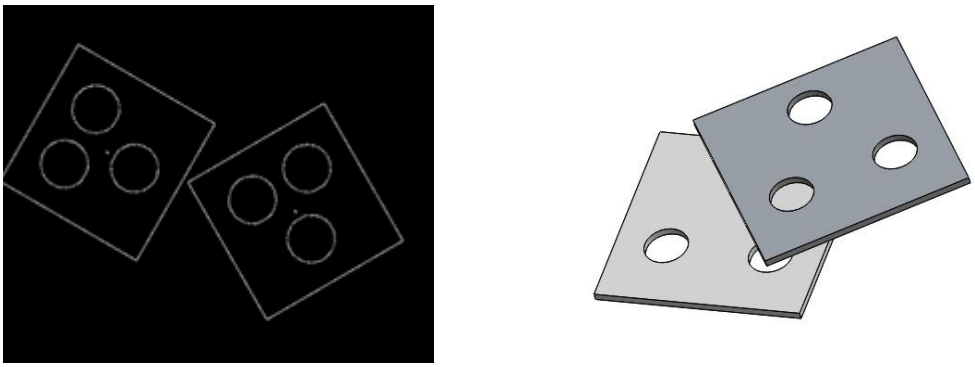
（3）预期效果及评价

其中，三维零件的基准较难得到，需要采用其他算法和模型。

9.2.4 多个零件有重叠的情况

（1）问题提出

本题中问题二的零件位置关系为图 9.5（a）所示，两个零件互不接触，降低了识别难度，如果两个零件位置关系如图 9.5（b），则本文中的模型不再适用。



（a） （b）

图 9.5 零件重叠

（2）解决方案及模型

首先，把 a 和 b 看成一个整体，运用问题一中的方法，以 a 和 b 为整体在问题一的同一位置建立对照基准，建立基准时，尽量让基准与 a、b 本体分离，实在做不到则修改问题一的基准，使其更远；再次，然后用问题一的方法求解 a 和 b 整体相对于其基准的位置（两个移动量和一个转动量）；其次，用图像分割算法把基准 2 和 a、b 本体分离，用基准 2 与基准 1 轮廓做对比，重合赋值为 0，则得到基准 2（b）的轮廓；再次，用基准 2（b）反向移动先前的两个移动量和一个转动量，则与 b 的位置重合，重合则赋值为 0，则图中只剩 a 零件，a 零件相对基准的位置可用问题一的方法求解；最后，同样的方法去除 a 零件，则 b 件相对基准的位置也可用问题一的方法求解。

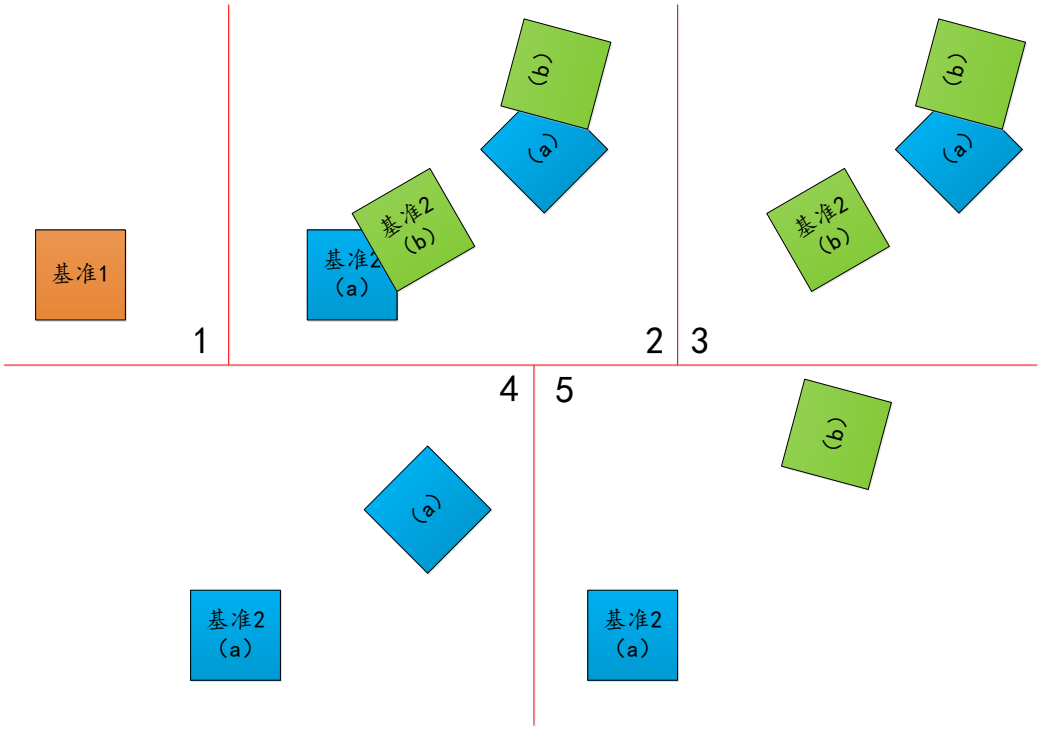


图 9.6 解决方案

9.2.5 一条生产线上需要抓取多个不同零件的情况

（1）问题提出

为了提高生产线的利用率，有的生产线加工的是一类型都是各不相同的零件，有的甚至加工完全不同的零件，此时的位置识别具有个体性。

（2）解决方案及模型

此时，本文中针对问题一提出的第二种方法是在这种场合下是适用的，它不需要考虑轮廓形状，无论任何形状都做全盘比较。当然，这要求事先将在生产线上加工的各种零件都做一基准置入比较状态。拾取前，用零件轮廓与各个基准用上述方法进行比对。

（3）预期效果及评价

该方法能实现预期功能，但是问题一所用的第二种方法在单一零件比较下尚且无速度优势，在多个零件时，要求执行更多的次数，所以，速度会更慢。

综上所述，本题及模型均具有较大的继续研究的潜力。

10 参考文献

[1] 杨晓峰,基于仿射变换模型的图像目定位跟踪方法,武汉: 华中科技大学研究生院，2005。
[2] 屠礼芬，彭祺，仲思东，一种适应相机抖动的运动目标检测方法，电子与信息学报，35（8）:1914-1920，2013。
[3] 秦筱槭,基于仿射变换重建的目标识别方法,武汉: 华中科技大学研究生院,2004。
[4] 中国电力网，我国核电安全高效发展特点逐渐彰显 智能化核电成“高精尖”技术密集地，http://news.bjx.com.cn/html/20171031/858583.shtml（2017 年 10 月 31 日）。
[5] 前瞻产业研究院，未来已来 2022 年全球智能制造市场规模将达 1.5 万亿美元，https://www.qianzhan.com/analyst/detail/220/171127-6c1301fc.html（2017 年 11 月 28 日）。
[6] 智能工厂，华为，OPPO，VIVO 三大智能手机自动化生产线车间大揭秘，http://www.sohu.com/a/144979355_177747（2017 年 05 月 31 日）。
[7] 中国经营报，董明珠：格力将推动智能制造工业化，http://tech.163.com/16/0313/09/BI1DGCG2000915BD.html（2016 年 03 月 13 日）。
[8] 前瞻产业研究院,未来已来 2022 年全球智能制造市场规模将达 1.5 万亿美元,https://www.qianzhan.com/analyst/detail/220/171127-6c1301fc.html（2017 年 11 月 28 日）。
[9] 前瞻产业研究院,2017 年中国智能制造行业发展现状与市场规模分析,https://www.qianzhan.com/analyst/detail/220/171208-fd6b824b.html（2017 年 12 月 08 日）。
[10] 新浪博客，图的连通性计算（求图的最大连通域），https://blog.csdn.net/jzwong/article/details/51720767（2014 年 05 月 09 日）。
[11] 姚敏，数字图像处理，北京：机械工业出版社，2008。
[12] 田浩，葛秀慧，王顶，数字图像处理，北京：清华大学出版社，2007。
[13] 周琳娜，杨义先，郭云彪等，基于二值图像的信息隐藏研究综述，中山大学学报，11（43）：70-75，2004。
[14] 刘九芬，付磊，张卫明，基于二值图像的信息隐藏算法，计算机工程，18（37）：120-124，2011。
[15] 杨晓东，吴玲达，谢毓湘等，二值图像轮廓局部描述和检索方法，计算机应用，1（30）：65-67，2010。
[16] 张琪，结合边缘检测的图像二值化算法，吉林:吉林大学，2011。
[17] Fu Chang，Chun-Jen Chen，Chi-Jen Lu，A Linear-Time Component-Labeling Algorithm Using Contour Tracing Technique，Computer Vision and Image Understanding，93：206-220，2004。
[18] J. Martin-Herrero，Hybrid object labeling in digital images，Machine Vision Application，18（1）：1-15，2007。
[19] T. Goto，Y. Ohta，M. Yoshida，et al，High speed algorithm for component labeling，IEICE，J72-D-II（2）：247－255，1989。
[20] K. Wu, E. Otoo and K. Suzuki，Optimizing two-pass connected-component abeling algorithms，Pattern Anal Applic，12：117-135，2009。
[21] Lifeng He，Yuyan Chao，Kenji Suzuki，et al。Fast connected-component labeling，Pattern Recognition，42：1977-1987，2009。

[22] Cai-Xia Li，Yong-ping Huang，Xiao-Xia Han，et al，Algorithm of Binary Image Labeling and Parameter Extracing Based on FPGA，Computer Engineering and Technology (ICCET)，3：542-545，2010。
[23] 覃方涛，房斌，GPU 加速的二值图连通域标记并行算法，计算机应用，30（10）：2774-2776，2010。
[24] 王静，二值图像连通域的分段标记算法及实现，红外与激光工程，39（4）：761-765，2010。
[25] 徐正光，鲍东来，张利欣，基于递归的二值图像连通域像素标记算法，计算机工程，32（24）：186-189，2006。
[26] 张云哲，赵海，宋纯贺等，一种新的连通区域标记算法，计算机应用研究，27（11）：4335-4337，2010。
[27] 刘奇琦，龚晓峰，一种二值图像连通区域标记的新方法，计算机工程与应用，48（11）：178-180，2012。
[28] 周跃，闫丰，章明朝等，基于标号回传的二值图像连通体标记算法，计算机工程与应用，45（33）：153-155，2009。
[29] 陈柏生，一种二值图像连通区域标记的新方法，计算机工程与应用，25（12）：46-47，2006。
[30] 刘贤喜，李邦明，苏庆堂等，一种新的二值图像连通区域准确标记算法，计算机工程与应用，43（22）：76-78，2007。
[31] 刘关松，吕嘉雯，徐建国等，一种新的二值图像标记的快速算法，计算机工程与应用，4：57-59，2002。
[32] 董星，一种基于距离的二值图像标记方法，沈阳：辽宁师范大学，2016。
[33] 穆天红，基于 CUDA 的二值图像连通域快速标记算法改进研究，西安：陕西科技大学，2014。
[34] sumofe，matlab 图像的膨胀 indilate 和腐蚀 imerode，https://blog.csdn.net/u013228046/article/details/40781249（2014 年 11 月 04 日）。
[35] sz-lcw，Matlab 得到二值图像中最大连通区域，https://blog.csdn.net/szlcw1/article/details/44227571（2015 年 03 月 12 日）。
[36] 阳光雨露，matlab 二值图像外轮廓提取，http://blog.sina.com.cn/s/blog_6f2d29af0101065p.html（2011 年 12 月 27 日）。

11 附录

附录一 question1_method1

```
%问题一求解过程
clear
clc
tic
load('DATA1.mat')
x1=D1; %数据读取
```

```
figure
subplot(1,2,1)
imshow(x1)
title('搬运前零件位置')
[m,n]=size(x1);
x0=x1;

x1=imdilate(x1,ones(5)); %图像膨胀
%%%正方形提取
img=maxLianTongYu(x1);   %求图像最大连通域
k=1;   %提取所有位置点
for i=1:m
        for j=1:n
            if img(i,j)+x0(i,j)==2
                a0(1,k)=i;
                a0(2,k)=j;
                k=k+1;
            end
        end
end
Square_center=mean(a0,2); %正方形形心点坐标

%%%第一圆提取
x1=x1-img; %剩余图像
img1=maxLianTongYu(x1); %求图像最大连通域
img11=zeros(m,n); %图像还原
k=1;
for i=1:m
        for j=1:n
            if img1(i,j)+x0(i,j)==2
                img11(i,j)=1;
                a1(1,k)=i;
                a1(2,k)=j;
                k=k+1;
            end
        end
end
circular_center1=mean(a1,2); %圆 1 形心点坐标

%%%第二圆提取
x1=x1-img1; %剩余图像
img2=maxLianTongYu(x1); %求图像最大连通域
```

```
img22=zeros(m,n); %图像还原
k=1;
for i=1:m
        for j=1:n
            if img2(i,j)+x0(i,j)==2
                img22(i,j)=1;
                a2(1,k)=i;
                a2(2,k)=j;
                k=k+1;
            end
        end
end
circular_center2=mean(a2,2); %圆 2 中心点坐标

%%%第三圆提取
x1=x1-img2; %剩余图像
img3=maxLianTongYu(x1); %求图像最大连通域
img33=zeros(m,n);%图像还原
k=1;
for i=1:m
        for j=1:n
            if img3(i,j)+x0(i,j)==2
                img33(i,j)=1;
                a3(1,k)=i;
                a3(2,k)=j;
                k=k+1;
            end
        end
end
circular_center3=mean(a3,2); %圆 3 形心点坐标

%位置判别特征点拾取
postion_n=find(a0(1,:)==max(a0(1,:)));   %X 最大时点的位置
postion(1:2,1)=a0(1:2,postion_n(1));      %X 最大时的点
postion_n=find(a0(2,:)==max(a0(2,:)));   %Y 最大时点的位置
if abs(a0(2,postion_n)-postion(2,1))>10       %以上两个位置点重复性判别
    postion(1:2,2)=a0(1:2,postion_n(1));   %Y 最大时的点
else
    postion_n=find(a0(2,:)==min(a0(2,:)));   %Y 最小时点的位置
    postion(1:2,2)=a0(1:2,postion_n(1));   %Y 最小时的点
end
postion_point=mean(postion,2)-Square_center; %位置判别特征点
```

```
%图像平移回原点并极坐标化
a_all=[a0,a1,a2,a3]; %图像合成
for i=1:length(a_all(1,:))
    a_zero(1:2,i)=a_all(1:2,i)-Square_center;   %图像平移
    [theta(i),r(i)] = cart2pol(a_zero(1,i),a_zero(2,i));   %极坐标化
end
circular_center=[circular_center1-Square_center,circular_center2-Square_center,circular_center3-Square_center]; %所有特征圆心坐标
[theta_point,r_point] = cart2pol(postion_point(1),postion_point(2)); %位置判别特征点极坐标化
for i=1:3   %位置判别特征点旋转
    [theta0(i),r0(i)] = cart2pol(circular_center(1,i),circular_center(2,i)); %圆心位置极坐标化
    f1(i)=theta_point-theta0(i);   %特征点顺时针旋转圆心角度后的角度位置
    k=floor(2*f1(i)/pi);
    f(i)=f1(i)-k*pi/2; %特征点旋转后与各坐标轴的最小夹角
    if f(i)>pi/4
        f(i)=pi/2-f(i); %特征点旋转后与各坐标轴的最小夹角
    end
end
p=find(f==min(f)); %旋正角度位置

%图像旋正
%定义中心点位置为（100,300）
x0_y0=[250;200];
x_img=zeros(m,n);
 for i=1:length(a_all(1,:))
    theta(i)=theta(i)-theta0(p)+pi; %图像旋转
    [x_new(1,i),x_new(2,i)]=pol2cart(theta(i),r(i));   %笛卡尔坐标化
    x_new(:,i)=round(x_new(:,i))+x0_y0;   %平移到指定位置
    x_img(x_new(1,i),x_new(2,i))=1; %新位置生成
 end

subplot(1,2,2)
imshow(x_img)
title('搬运后零件位置')

[Square_center;theta0(p)]   %零件位置

toc
```

```
%function [img]=maxLianTongYu(I)：求图像中最大的连通域
%输入：I       输入图像
%输出：img   仅包含最大连通域的图像
function [img]=maxLianTongYu(I)
if length(size(I))>2
    I = rgb2gray(I);
end
if ~islogical(I)
    imBw = im2bw(I);                                %转换为二值化图像
else
    imBw = I;
end
imBw = im2bw(I);                                %转换为二值化图像
imLabel = bwlabel(imBw);                        %对各连通域进行标记
stats = regionprops(imLabel,'Area');       %求各连通域的大小
area = cat(1,stats.Area);
index = find(area == max(area));             %求最大连通域的索引
img = ismember(imLabel,index(1));                %获取最大连通域图像若有多个面积相等的只取一个
end
```

附录二　question1_method2

```
%获取模板（基准位置）
tic
clear
clc
load('DATA2.mat')
x1=D2;
figure
imshow(x1)
[m,n]=size(x1);
%将原图填充孔洞
BW1= imfill(x1,'holes');
%图像腐蚀
SE=strel('arbitrary',eye(5));
BW2=imerode(BW1,SE);
%零件分离
for k=1:10 %取远大于零件个数的数即可
    if sum(sum(BW2))>50
```

```
        %%%正方形提取
        %求图像最大连通域
        img=maxLianTongYu(BW2);
        BW2=BW2-img;
        %图像膨胀
        SE=ones(5);
        img1=imdilate(img,SE);

        %单个零件提取
        part1=zeros(m,n);
        for i=1:m
            for j=1:n
                if img1(i,j)+x1(i,j)==2
                    part1(i,j)=1;
                end
            end
        end
        part{k}=part1;
        figure
        imshow(part1)
        else
            p=k-1;
            break;
        end
end
save part
toc

%function [img]=maxLianTongYu(I)：求图像中最大的连通域
%输入：I       输入图像
%输出：img   仅包含最大连通域的图像
function [img]=maxLianTongYu(I)
if length(size(I))>2
    I = rgb2gray(I);
end
if ~islogical(I)
    imBw = im2bw(I);                                %转换为二值化图像
else
    imBw = I;
end
imBw = im2bw(I);                                %转换为二值化图像
imLabel = bwlabel(imBw);                        %对各连通域进行标记
```

```
    stats = regionprops(imLabel,'Area');      %求各连通域的大小
    area = cat(1,stats.Area);
    index = find(area == max(area));          %求最大连通域的索引
    img = ismember(imLabel,index(1));         %获取最大连通域图像若有多个面积相等的只取一个
    end

    %主程序
    clc
    clear
    profile on
    tic
    load DATA1
    [Y X]=find(D1==1); %模板中的离散点
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%零件模板信息
    load muban
    I = muban;%零件模板
    [y x]=find(I==1); %模板中的离散点
    %已标定模板中心点
    tx=200;
    ty=250;
    %  对比模板
    figure
    plot(X,Y,'b.')
    hold on
    %plot(x,y,'r.')
    %%%%%%%%%%%%%%%%%%%%%%
    d=[x-tx y-ty];
    r=randperm(length(X));
    D=[X Y];
    D=D(r(1:100),:);    %偏移后图像
    d=d(r(1:500),:);    %标定图像
    l = [-400 -400 -pi]; %下限
    u = [400    400    pi]; %上限
    x0 = [100    100    0];
    TolFun = 1e-9;
    TolX=1e-5;
    kmax =50000;

    %%%%用模拟退火法求
      q =0.8;
    [xo_sa,fo_sa] =Opt_Simu(@(x)myfun2(x,d,D),x0,l,u,kmax,q,TolFun)
```

```
%%%%%%%%%%
t=xo_sa(3); %在图像中是顺时针旋转
a=xo_sa(1);
b=xo_sa(2);
T=[cos(t) sin(t)
        -sin(t) cos(t)];
temp=[X+a Y+b]*T;     %平移加旋转
X=round(temp(:,1)); %逆平移
Y=round(temp(:,2));
plot(X+tx,Y+ty,'r.')

%还原图像
[m,n]=size(D1);
pic=zeros(m,n);
for i=1:length(X)
    pic(Y(i)+ty,X(i)+tx)=1;
end
figure
imshow(pic)

%目标函数
%axis off
toc
profile viewer

function f=myfun2(s,d,D)
% s(1) X 平移量
% s(2) Y 平移量
% s(3)  旋转角

a=s(1);
b=s(2);
t=s(3); %在图像中是顺时针旋转
T=[cos(t) sin(t)
        -sin(t) cos(t)];
X=D(:,1);
Y=D(:,2);
D=[X+a Y+b]*T;            %平移加旋转变换
x=d(:,1)';
y=d(:,2)';
X=D(:,1);%更新
Y=D(:,2);
```

```
n=length(d);
N=length(D);
X=repmat(X,1,n);
Y=repmat(Y,1,n);
x=repmat(x,N,1);
y=repmat(y,N,1);
DS=(X-x).^2+(Y-y).^2; %所有 D 到所有 d 的距离矩阵
f=sum(min(DS'));

function [xo,fo] = Opt_Simu(f,x0,l,u,kmax,q,TolFun)
% 模拟退火算法求函数 f(x)的最小值点， 且 l <= x <= u
% f 为待求函数，x0 为初值点，l，u 分别为搜索区间的上下限，kmax 为最大迭代次数
% q 为退火因子，TolFun 为函数容许误差
%%%%算法第一步根据输入变量数，将某些量设为缺省值
if nargin < 7
    TolFun = 1e-8;
end
if nargin < 6
    q = 1;
end
if nargin < 5
    kmax = 100;
end
%%%%算法第二步，求解一些基本变量
N = length(x0); %自变量维数
x = x0;
fx = feval(f,x); %函数在初始点 x0 处的函数值
xo = x;
fo = fx;
%%%%%算法第三步，进行迭代计算，找出近似全局最小点
for k =0:kmax
    k;
    Ti = (k/kmax)^q;
    mu = 10^(Ti*100); % 计算 mu
    dx = Mu_Inv(2*rand(size(x))-1,mu).*(u - l);%步长 dx
    x1 = x + dx; %下一个估计点
    x1 = (x1 < l).*l +(l <= x1).*(x1 <= u).*x1 +(u < x1).*u; %将 x1 限定在区间[l,u]上
    fx1 = feval(f,x1);
    df = fx1- fx;
```

```
        if df < 0|rand < exp(-Ti*df/(abs(fx) + eps)/TolFun) %如果 fx1<fx 或者概率大于随机数 z
            x = x1;
            fx = fx1;
        end
        if fx < fo
            xo = x;
            fo = fx1;
        end
    end

    %模拟退火法中的 mu^（-1）定理
    function x = Mu_Inv(y,mu)
    x = (((1+mu).^abs(y)- 1)/mu).*sign(y);
```

附录三 question2

```
    %问题二求解过程
    tic
    clear
    clc
    load('DATA2.mat')
    x1=D2;
    %figure
    %imshow(x1)
    [m,n]=size(x1);
    %将原图填充孔洞
    BW1= imfill(x1,'holes');
    %图像腐蚀
    SE=strel('arbitrary',eye(5));
    BW2=imerode(BW1,SE);

    for numb=1:10     %取远大于零件个数的数即可
        if sum(sum(BW2))>50
            %%%正方形提取
            %求图像最大连通域
            img=maxLianTongYu(BW2);
            BW2=BW2-img;
            %图像膨胀
            SE=ones(5);
            img1=imdilate(img,SE);
```

```
%单个零件提取
part=zeros(m,n);
for i=1:m
    for j=1:n
        if img1(i,j)+x1(i,j)==2
            part(i,j)=1;
        end
    end
end
%figure
%imshow(part)


%%用问题一提出的方法求解
numb
x1_1=part;   %数据读取
figure
subplot(1,2,1)
imshow(x1_1)
title('搬运前零件位置')
[m,n]=size(x1_1);
x0=x1_1;

x1_1=imdilate(x1_1,ones(5)); %图像膨胀
%%%正方形提取
img_1=maxLianTongYu(x1_1);   %求图像最大连通域
a0=[];
k=1;   %提取所有位置点
for i=1:m
    for j=1:n
        if img_1(i,j)+x0(i,j)==2
            a0(1,k)=i;
            a0(2,k)=j;
            k=k+1;
        end
    end
end
Square_center=mean(a0,2); %正方形形心点坐标

%%%第一圆提取
x1_1=x1_1-img_1; %剩余图像
img1_1=maxLianTongYu(x1_1); %求图像最大连通域
```

```
img11=zeros(m,n); %图像还原
a1=[];
k=1;
for i=1:m
    for j=1:n
        if img1_1(i,j)+x0(i,j)==2
            img11(i,j)=1;
            a1(1,k)=i;
            a1(2,k)=j;
            k=k+1;
        end
    end
end
circular_center1=mean(a1,2); %圆 1 形心点坐标

%%%第二圆提取
x1_1=x1_1-img1_1; %剩余图像
img2=maxLianTongYu(x1_1); %求图像最大连通域
img22=zeros(m,n); %图像还原
a2=[];
k=1;
for i=1:m
    for j=1:n
        if img2(i,j)+x0(i,j)==2
            img22(i,j)=1;
            a2(1,k)=i;
            a2(2,k)=j;
            k=k+1;
        end
    end
end
circular_center2=mean(a2,2); %圆 2 中心点坐标

%%%第三圆提取
x1_1=x1_1-img2; %剩余图像
img3=maxLianTongYu(x1_1); %求图像最大连通域
img33=zeros(m,n);%图像还原
a3=[];
k=1;
for i=1:m
    for j=1:n
        if img3(i,j)+x0(i,j)==2
```

```
                    img33(i,j)=1;
                    a3(1,k)=i;
                    a3(2,k)=j;
                    k=k+1;
                end
            end
        end
        circular_center3=mean(a3,2); %圆 3 形心点坐标

        %位置判别特征点拾取
        postion_n=find(a0(1,:)==max(a0(1,:)));   %X 最大时点的位置
        postion(1:2,1)=a0(1:2,postion_n(1));      %X 最大时的点
        postion_n=find(a0(2,:)==max(a0(2,:)));   %Y 最大时点的位置
        if abs(a0(2,postion_n)-postion(2,1))>10       %以上两个位置点重复性判
别
            postion(1:2,2)=a0(1:2,postion_n(1));   %Y 最大时的点
        else
            postion_n=find(a0(2,:)==min(a0(2,:)));   %Y 最小时点的位置
            postion(1:2,2)=a0(1:2,postion_n(1));   %Y 最小时的点
        end
        postion_point=mean(postion,2)-Square_center; %位置判别特征点

        %图像平移回原点并极坐标化
        a_all=[a0,a1,a2,a3]; %图像合成
        for i=1:length(a_all(1,:))
            a_zero(1:2,i)=a_all(1:2,i)-Square_center;   %图像平移
            [theta(i),r(i)] = cart2pol(a_zero(1,i),a_zero(2,i));   %极坐标化
        end

circular_center=[circular_center1-Square_center,circular_center2-Square_center,circular_c
enter3-Square_center]; %所有特征圆心坐标
        [theta_point,r_point] = cart2pol(postion_point(1),postion_point(2)); %位置
判别特征点极坐标化
        for i=1:3   %位置判别特征点旋转
            [theta0(i),r0(i)] = cart2pol(circular_center(1,i),circular_center(2,i)); %
圆心位置极坐标化
            f1(i)=theta_point-theta0(i);   %特征点顺时针旋转圆心角度后的角度
位置
            k=floor(2*f1(i)/pi);
            f(i)=f1(i)-k*pi/2; %特征点旋转后与各坐标轴的最小夹角
            if f(i)>pi/4
                f(i)=pi/2-f(i); %特征点旋转后与各坐标轴的最小夹角
```

```
            end
        end
        p=find(f==min(f)); %旋正角度位置

        %图像旋正
        %定义中心点位置为（100,300）
        x0_y0=[250;200];
        x_img=zeros(m,n);
        for i=1:length(a_all(1,:))
            theta(i)=theta(i)-theta0(p)+pi; %图像旋转
            [x_new(1,i),x_new(2,i)]=pol2cart(theta(i),r(i));   %笛卡尔坐标化
            x_new(:,i)=round(x_new(:,i))+x0_y0;   %平移到指定位置
            x_img(x_new(1,i),x_new(2,i))=1; %新位置生成
        end

        subplot(1,2,2)
        imshow(x_img)
        title('搬运后零件位置')

        [Square_center;theta0(p)]   %零件位置
    else
        p=k-1;
        break;
    end
end
toc


%function [img]=maxLianTongYu(I)：求图像中最大的连通域
%输入：I      输入图像
%输出：img   仅包含最大连通域的图像
function [img]=maxLianTongYu(I)
if length(size(I))>2
    I = rgb2gray(I);
end
if ~islogical(I)
    imBw = im2bw(I);                                %转换为二值化图像
else
    imBw = I;
end
imBw = im2bw(I);                                %转换为二值化图像
imLabel = bwlabel(imBw);                        %对各连通域进行标记
```

```
    stats = regionprops(imLabel,'Area');      %求各连通域的大小
    area = cat(1,stats.Area);
    index = find(area == max(area));          %求最大连通域的索引
    img = ismember(imLabel,index(1));         %获取最大连通域图像若有多个面
积相等的只取一个
    end
```