基于耳切法的复杂结构衍射研究

摘　要：光的干涉与衍射现象是光的基本现象之一，其揭示了光的波动性，对光的研究有着重要意义；光的干涉与衍射虽有着概念上的区别，但其物理本质并没有显著区别，故在本次研究中，我们专注于复杂结构的衍射成像。由于一个十分复杂且对称性较弱的复杂图形，衍射理论是难以给出解析解的，但我们开创性地使用三角形的分解方式，能够精确且迅速地导出复杂结构的解析解。在本文中，我们首先提出了分解复杂图形再利用光场叠加的原理得出解析解的方法，再以直角三角形为分解单元利用 Python 编程实现边缘检测以及耳切法分割，并对标量衍射理论进行了推导，夫琅禾费衍射可以得出较为直观的解析解，菲涅尔衍射的理论结果中包含积分公式，故我们专注于研究夫琅禾费衍射；其次，我们进行了夫琅禾费衍射实验，通过实验、理论、仿真进行对比，实验图像基本一致，并且为保证实验的完整性，我们对比了菲涅尔衍射的实验和仿真结果，效果也基本一致；然后，进行了均匀光栅以及缺刻光栅的探究，并且利用反演验证了理论的准确性；最后，我们利用具有较高对称性的复杂结构进行了数值分析，对正方形的水平轴光强分布和正多边形的角分布进行了不确定度分析以及误差分析，以及探究了弧形近似的精度，在实验光强归一化的条件下，正方形的水平轴光强分布与理论差值的结果为 0.092±0.007，角分布扩展不确定度均在 1°-3°左右，不确定度较小，弧形近似的光强零点拟合优度为 R-Square=0.99979，皮尔逊相关系数为 0.9999，实验效果十分直观，弧形近似方法有效。且我们的研究具有创新性、完整性、精确性、合理性和应用性，研究十分成功。

关键词：复杂结构；夫琅禾费衍射；菲涅尔衍射；耳切法；标量衍射理论

目录

1. 题目调研

1.1 题意理解

不同于常规的单孔、双孔、单缝、双缝、周期性多缝，复杂结构指的是常规结构之外的结构，其能产生许多新奇的干涉衍射现象。对于常规结构，其干涉与衍射的解析解较为容易导出，而对于一个复杂孔，其较难直接给出理论干涉衍射结果，但对于一般的复杂结构，可以利用图形处理的方法将其分解为若干个直角三角形，简单地利用三角形干涉衍射叠加的效果直接导出复杂结构的解析解，更加简洁直观，并选取复杂结构衍射与干涉的特殊形状位置及光强分布等给出误差及不确定度分析。

1.2 研究背景及目标

1.2.1 研究背景

光是我们认知世界、理解环境的重要工具，也是物理学中的重要研究对象。光的干涉与衍射现象是光的基本现象之一，其揭示了光的波动性，对光的研究有着重要意义，为变换光学、全息成像等方向提供了重要的理论依据[1][2]。

研究光的干涉与衍射有许多理论方法，一般由惠更斯-菲涅耳原理导出[3]，同时我们也可以用量子力学中的态叠加原理去描述光的叠加[4]，以及傅里叶光学理论通过频域和空间域的转化，也能够很好地解释干涉与衍射现象[5]。

光在传播过程中遇到尺寸接近于光波长的障碍物时，会发生偏离直线传播的现象而进入按直线传播划定的阴影区内，这种现象称为光的衍射。光的衍射现象一般分为两类，分别是菲涅尔衍射和夫琅禾费衍射。

菲涅尔衍射原理如图 1(a)所示，指光源和观察屏或其中任意一个与障碍物的距离为有限远的情况。夫琅禾费衍射的原理如图 1(b)所示，指光源和观察屏两者与障碍物的距离均为无限远的情况，即入射光和衍射光都是平行光束，也称为平行光束的衍射。由于衍射后的平行光难以直接观测，可以近似地或利用光学成像系统将平行光聚焦后进行观测，所以夫琅禾费衍射的实际光路可以有多种变形。

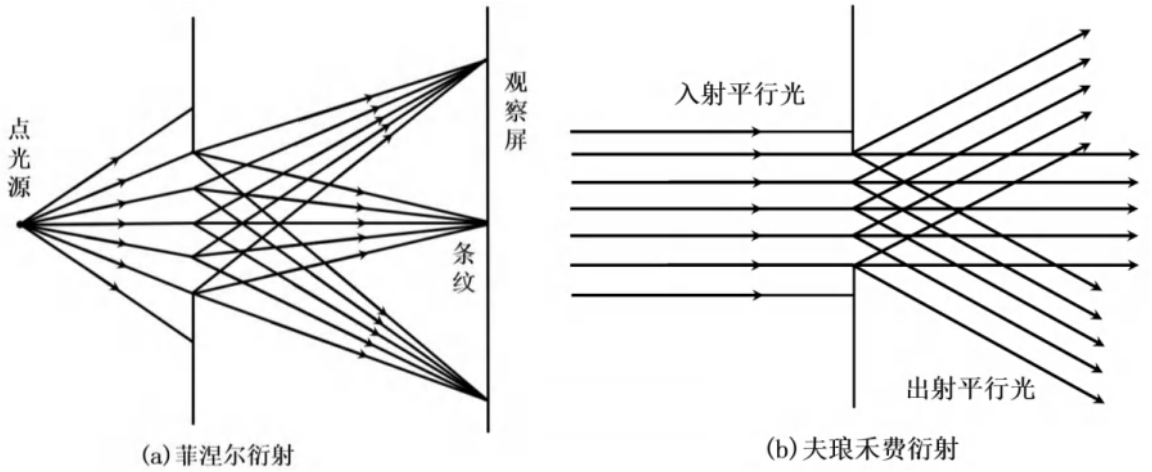


图 1 光的衍射原理图

光的干涉与衍射在生产生活中有着重要的应用，例如在光学望远镜中进行干涉成像[6]、生物医学中利用干涉获取组织的细节成像[7]、衍射光栅对光进行分束操作进行高精度测量等[8]。

在科学研究中，我们常常遇到的并不是常规的单孔、双孔、单缝、双缝或者周期

性多缝，更多的是复杂的结构，在这样的条件下，为得到简洁优美的解析解，我们需要对复杂结构进行分解，对分解后的结构进行分析，再将结果进行重构。

对于三角孔的干涉和衍射，能够产生一些不同于圆孔以及光栅的特殊图像[9]，并且由于三角孔具有三个角，可以轻易地拼凑出更加复杂的图形。利用三角孔对复杂图形的分解和重构可以实现更便捷的研究过程[10]。

为更加便捷地实现，结合机器视觉和图形处理是必然的。边缘检测算法的目标是寻找图像中灰度、颜色或者纹理发生明显变化的地方[11]。由于图像的噪声干扰、光照条件的变化等因素，实际的边缘检测工作并非易事，我们尝试采用轮廓追踪算法得到多边形的边缘。

计算机图形处理是计算机科学的一个重要分支[12]，包含计算技术创建和操作视觉内容。从 2D 像素到复杂的 3D 模型，计算机图形处理已经渗透到科研、工业、娱乐、医疗等众多领域[14][15][16]。一大重点是处理和理解形状，尤其是多边形，常用的多边形分割方法有耳切法(Ear Clipping Method)、描线法(scan-line algorithm)和高斯-赛德尔迭代法(Seidel's Algorithm)等，我们采用耳切法的方法对多边形分割，进而叠加得到衍射图样。

1.2.2 研究目标

本实验旨在通过三角分解的方法研究复杂结构的干涉衍射图像，结合计算机更加简洁、精确地导出干涉衍射的理论结果，并对其反演验证理论结果的准确性。

对一个复杂结构的干涉衍射计算通常用到常规的傅里叶变换方法，需要用到计算机进行大量计算且难以得出理论结果，一般只有数值结果，结果虽然精确但计算量较大。故我们考虑用更加便捷并且计算量较小且得到较为精确结果的方法，我们现利用边缘检测以及图形处理的方法快速且最优地将复杂孔分解为若干三角孔，再将对应的三角孔衍射理论结果进行叠加，即可简单导出复杂结构的干涉衍射结果。

1.3 目标定位

光的干涉与衍射在概念上不同，但事实上，在物理的理论层面，其并没有本质上的区别，故我们之后不再对其进行区分，我们专注于研究复杂结构孔的衍射效果。

基于题意理解、研究背景以及研究目标，我们在研究复杂结构的干涉衍射中将依次解决以下问题：

- 研究三角孔的干涉衍射性质，给出理论解。
- 设计复杂结构，探究弧形的近似方式，利用机器视觉和图形分割程序探究最优的分解方式，验证分解方案的可行性以及便捷性。
- 优化复杂结构，设计多种结构利用反演变换充分证明方法可行性以及优越性。
- 利用干涉衍射特殊位置进行误差分析以及不确定度分析。
- 寻找应用目标，初步探究应用场景。

2. 实验原理和设计方案

2.1 三角孔衍射实验原理

对于分解，我们采用直角三角形，其具体原因在设计方案中阐述，在此我们仅仅进行直角三角形的衍射结果推导。

惠更斯原理指出，波动所到达的面上每一点将作为次级球面波的点源，那么随后任一时刻的波，可以由作出次级子波的包络而得到。菲涅尔将该理论扩展，认为衍射

现象是由于次波之间的叠加干涉而形成衍射条纹。并由基尔霍夫将光作为标量来处理，将其置于一个坚实的数学基础之上，得到基尔霍夫衍射理论公式[12]。

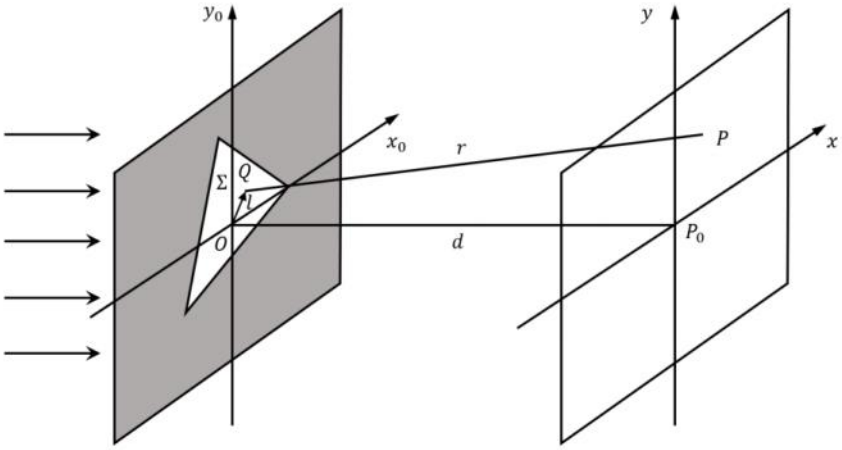


图 2 标量衍射理论示意图

其中，$Q(x_0,y_0)$为小孔上一点，$P(x,y)$为观察屏上一点，d 为两平面之间距离，r 为 P、Q 点之间的距离，$\vec{l}$为原点 O 到小孔上源点 P 的向量。

这个公式只要满足以下两个条件，就可以得到精确的结果：

①衍射孔径比波长大得多。②观察屏较为远离衍射小孔。

$$E(P)=\frac{A}{i\lambda}\iint_{\Sigma}E_{\Sigma}(x_0,y_0)\frac{e^{ikr}}{r}\left[\frac{\cos(\vec{n}\cdot\vec{r})-\cos(\vec{n}\cdot\vec{l})}{2}\right]\mathrm{d}\sigma \tag{1}$$

事实上如果直接利用此公式进行计算，计算过程十分复杂，因此我们根据实验条件以及一些特殊情况对上面的公式做出近似。首先考虑傍轴近似，即传播方向与光轴的夹角不大的情况：

此时$r\approx d$,$\cos(\vec{n}\cdot\vec{r})\approx 1$,$\cos(\vec{n}\cdot\vec{l})\approx -1$，上式可以近似为：

$$E(P)=\frac{A}{i\lambda d}\iint_{\Sigma}E_{\Sigma}(x_0,y_0)\,e^{ikr}\mathrm{d}\sigma \tag{2}$$

2.1.1 夫琅禾费衍射

首先我们考虑夫琅禾费衍射，即成像平面位于无穷远处，其满足条件：$\frac{k}{2z}(x_0^2+y_0^2)_{\max}\ll 1$时，可以将源点与场点的距离 r 近似为[17]：

$$r\approx d\left(1+\frac{1}{2}\frac{(x-x_0)^2+(y-y_0)^2}{d^2}\right)\approx d+\frac{x_0^2+y_0^2}{2d}-\frac{xx_0+yy_0}{d} \tag{3}$$

代入基尔霍夫衍射公式可得：

$$\begin{aligned}E(P)&=\frac{A}{i\lambda d}\iint_{\Sigma}E_{\Sigma}(x_0,y_0)e^{ikr}\,\mathrm{d}\sigma\\&=A\frac{e^{ikd}}{i\lambda d}e^{ik\frac{x^2+y^2}{2d}}\iint_{\Sigma}E_{\Sigma}(x_0,y_0)e^{-ik\frac{xx_0+yy_0}{d}}\,\mathrm{d}x_0\mathrm{d}y_0\end{aligned} \tag{4}$$

上式即为观察屏上的光场分布，若要得到光强分布，只需将上式取模方即可。

在此基础上，我们接着具体考虑边长分别为 a、b 的直角三角形，其两条直角边都位于坐标轴上，直角顶点位于坐标原点 O 处，如图 3 所示。

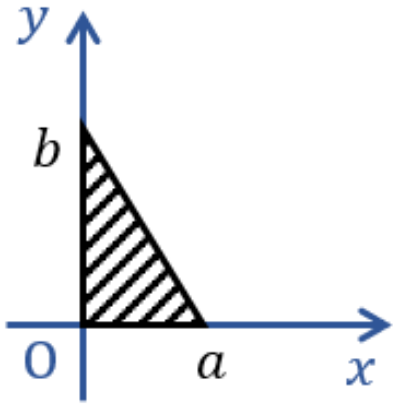


图 3 基本直角三角形示意图

与先前推导相同，设光的角波数为 k，由式(4)代入计算，可以得到观察屏上光场的分布为：

$$\begin{aligned} E(P) &= \iint_{\Sigma} A e^{-\frac{ik(xx_0+yy_0)}{d}} \mathrm{d}x_0 \mathrm{d}y_0 \\ &= A\int_0^a \mathrm{d}x_0 \int_0^{b-\frac{b}{a}x_0} e^{-\frac{ik(xx_0+yy_0)}{d}} \mathrm{d}y_0 \end{aligned} \tag{5}$$

可得结果光场为：

$$E(P) = -A\frac{d^2\left[ax\left(1-e^{-\frac{ibky}{d}}\right)+by\left(-1+e^{-\frac{iakx}{d}}\right)\right]}{k^2xy(ax-by)} \tag{6}$$

将上式取模方可以得到观察屏上的光强分布，并用 MATLAB 作图如图 4 所示：

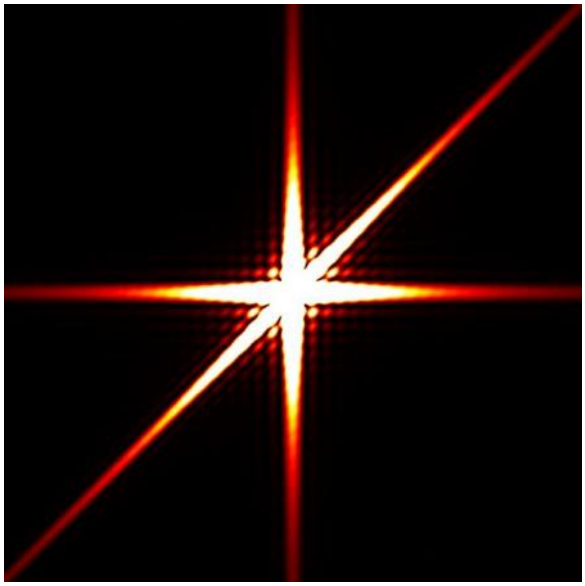


图 4 基本直角三角形夫琅禾费衍射图样

为了使得结果更具普适性，接着我们考虑任意位置直角三角形的衍射图样。

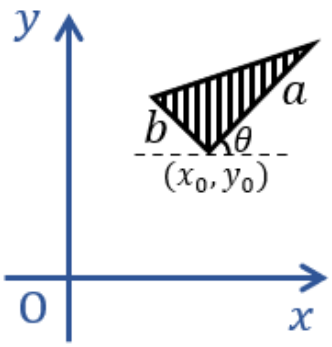


图 5 任意位置的三角形示意图

相比于我们前面给出的特殊直角三角形，该任意位置的直角三角形可以视为特殊位置直角三角形的平移与旋转操作得到的，因此我们只需要对特殊直角三角形的结果进行适当的变换，即可实现任意位置直角三角形的衍射光场分布理论求解，我们将该操作分解为平移与旋转两部分来进行对应的理论推导。

首先，我们先考虑旋转操作，即考虑一个绕直角顶点旋转了θ的直角三角形衍射图样，如图 6 所示。

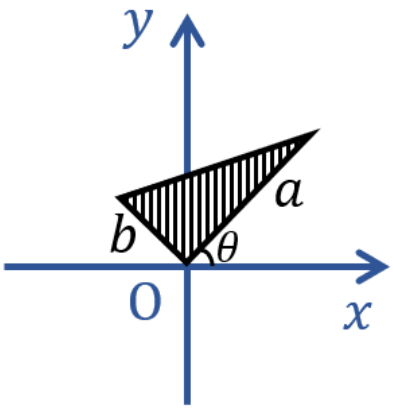


图 6 旋转角度后的直角三角形示意图

显然，对于旋转操作可以发现，与一般的坐标旋转变换类似，倘若我们对该坐标系进行以下的旋转变换[18]：

$$\begin{cases} x = u\cos\theta + v\sin\theta \\ y = -u\sin\theta + v\cos\theta \end{cases} \tag{7}$$

即可以实现原本特殊直角三角形计算结果的光场旋转，从而得到绕直角顶点旋转了角度为θ的直角三角形衍射光场分布：

$$\begin{aligned} E(P) = {} & \frac{-Ad^2}{k^2(x\cos\theta + y\sin\theta)(y\cos\theta - x\cos\theta)} \\ & \cdot \left[a(x\cos\theta + y\sin\theta)\left(1 - e^{-\frac{ibk(-x\cos\theta + y\cos\theta)}{d}}\right)\right. \\ & \left. + b(-x\cos\theta + y\cos\theta)\left(-1 + e^{-\frac{iak(x\cos\theta + y\sin\theta)}{d}}\right)\right] \\ & \cdot \frac{1}{[\,a(x\cos\theta + y\sin\theta) - b(y\cos\theta - x\cos\theta)]} \end{aligned} \tag{8}$$

接着我们再考虑经过平移三角形的光场分布，即该直角三角形的直角顶点位于点(u_0, v_0)处，如图 7 所示：

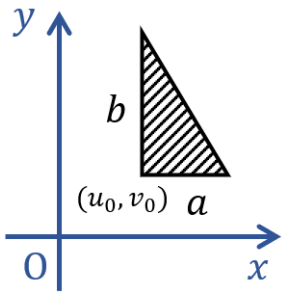


图 7 平移的直角三角形示意图

与前面推导相同，我们带入基尔霍夫衍射公式，有：

$$E(P) = \iint_{\Sigma} A e^{-\frac{ik(xx_0+yy_0)}{d}} \mathrm{d}x_0 \mathrm{d}y_0 \tag{9}$$

然后我们对上面的二重积分进行坐标变换，将该平移过的直角三角形，通过雅可比行列式移回坐标原点，即可得到平移过的直角三角形的衍射光场。于是我们先做以下的变换：

$$\begin{cases} u = x_0 - u_0 \\ v = y_0 - v_0 \end{cases} \tag{10}$$

该变换的雅可比行列式为：

$$J(u,v) = \frac{\partial(x_0, y_0)}{\partial(u,v)} = \begin{vmatrix} \frac{\partial x_0}{\partial u} & \frac{\partial x_0}{\partial v} \\ \frac{\partial y}{\partial u} & \frac{\partial y_0}{\partial v} \end{vmatrix} = 1 \tag{11}$$

因此可以得到，上积分变为：

$$\begin{aligned} E(P) &= A \iint_{\Sigma} e^{-\frac{ik(xx_0+yy_0)}{d}} \mathrm{d}x_0 \mathrm{d}y_0 \\ &= A \iint_{\Sigma} e^{-\frac{ik[x(u+u_0)+y(v+v_0)]}{d}} \cdot \mathrm{J}(u,v) \mathrm{d}u\mathrm{d}v \\ &= A \iint_{\Sigma_0} e^{-\frac{ik(xu+yv)}{d}} \mathrm{d}u\mathrm{d}v \cdot \exp\left[-\frac{ik}{d}(xu_0 + yv_0)\right] \end{aligned} \tag{12}$$

对比最初的式子可以看到，平移变换即对原本的积分整体乘上一个相位因子。

综合前面的结果，我们就可以得到经过旋转与平移变换后，任意位置的直角三角形衍射光场的分布：

$$\begin{aligned} E(P) = & \frac{-Ad^2}{k^2(x\cos\theta + y\sin\theta)(y\cos\theta - x\cos\theta)} \\ & \cdot \left[a(x\cos\theta + y\sin\theta)\left(1 - e^{-\frac{ibk(-x\cos\theta + y\cos\theta)}{d}}\right) \right. \\ & \left. + b(-x\cos\theta + y\cos\theta)\left(-1 + e^{-\frac{iak(x\cos\theta + y\sin\theta)}{d}}\right) \right] \\ & \cdot \frac{\exp\left[-\frac{ik}{d}(xu_0 + yv_0)\right]}{[\, a(x\cos\theta + y\sin\theta) - b(y\cos\theta - x\cos\theta)]} \end{aligned} \tag{13}$$

将上式取模可以得到光强分布，如图 8 所示[19]：

图 8 旋转平移后的直角三角形光强分布

2.1.2 菲涅尔衍射

在菲涅尔近似的条件下，虽然式(2)中被积函数的距离r与夫琅禾费衍射中不同，不可以取作d，但对于我们所探究的具体情况还可以做一些更精确的近似[20]。

与夫琅禾费衍射相同，我们将场点与源点距离r作 Taylor 展开，可以得到：

$$r \approx d\left(1+\frac{1}{2}\frac{(x-x_0)^2+(y-y_0)^2}{d^2}-\frac{1}{8}\left[\frac{(x-x_0)^2+(y-y_0)^2}{d^2}\right]^2\right) \tag{14}$$

但与夫琅禾费衍射不同的是，根据实验中的条件，此时我们取近似条件为$\frac{k}{8d^3}(x_0^2+y_0^2)_{\max}^2 \ll 1$，此时$r$可以近似为

$$r \approx d\left(1+\frac{1}{2}\frac{(x-x_0)^2+(y-y_0)^2}{d^2}\right) \tag{15}$$

则衍射的积分公式变为：

$$\begin{aligned} E(P) &= \frac{A}{i\lambda d}\iint_{\Sigma} E_{\Sigma}(x_0,y_0)e^{ikr}\,\mathrm{d}\sigma \\ &= A\frac{e^{ikd}}{i\lambda d}e^{ik\frac{x^2+y^2}{2d}}\iint_{\Sigma} E_{\Sigma}(x_0,y_0)e^{-ik\frac{xx_0+yy_0}{d}}\cdot\exp\left[\frac{ik}{2d}(x_0^2+y_0^2)\right]\mathrm{d}x_0\mathrm{d}y_0 \end{aligned} \tag{16}$$

则我们继续具体考虑边长分别为 a，b 的直角三角形，其两条直角边都位于坐标轴上，直角顶点位于坐标原点 O 处，如图 3 所示，则观察屏上光场的分布为：

$$\begin{aligned} E(P) &= \iint_{\Sigma_0} Ae^{-ik\frac{xx_0+yy_0}{d}}\cdot\exp\left[\frac{ik}{2d}(x_0^2+y_0^2)\right]\mathrm{d}x_0\mathrm{d}y_0 \\ &= A\int_0^a \mathrm{d}x_0\int_0^{b-\frac{b}{a}x_0} e^{-\frac{ik(xx_0+yy_0)}{d}}\cdot\exp\left[\frac{ik}{2d}(x_0^2+y_0^2)\right]\mathrm{d}y_0 \end{aligned} \tag{17}$$

计算可得结果光场为：

$$\begin{aligned} E(P) = \int_0^a \frac{\left(\frac{1}{2}-\frac{i}{2}\right)\sqrt{\pi}\sqrt{d}e^{-\frac{ik\left(2xx_0-{x_0}^2+y^2\right)}{2d}}}{a\sqrt{k}}\Bigg[a\cdot\mathrm{erfi}\left(\frac{\left(\frac{1}{2}+\frac{i}{2}\right)\sqrt{k}y}{\sqrt{d}}\right) \\ -|a|\cdot\mathrm{erfi}\left(\frac{\left(\frac{1}{2}+\frac{i}{2}\right)\sqrt{k}(a(y-b)+bx_0)}{\sqrt{ada^*}}\right)\Bigg]\mathrm{d}x_0 \end{aligned} \tag{18}$$

由于菲涅尔衍射近似下我们无法得到一个简洁优美的结果，式中结果光场是以一个积分形式表出的，其中 erfi 为虚误差函数，其表达式为

$$erfi(x) = -ierf(ix) = \frac{1}{\sqrt{\pi}}\int_{-x}^{x} e^{t^2} dt \tag{19}$$

由此可见，对于菲涅尔衍射，像夫琅禾费衍射一样得到一个通用的表达式是十分困难的，为给出简洁明了的解析解，我们下面的理论对比部分都将基于夫琅禾费衍射进行，菲涅尔衍射仅仅进行仿真结果的对比。

2.2 叠加原理

受到电路中叠加原理（戴维南定理）的启发，我们这里提出一个叠加原理来求解若干个简单图形叠加的衍射结果图样。如图 9 所示，对于一个任意的区域Σ，设其可以被分为若干区域$\Sigma_1, \Sigma_2, \Sigma_3$ ...，则该总区域Σ的衍射图样可以由区域$\Sigma_1, \Sigma_2, \Sigma_3$ ...的衍射图样叠加得到，以下我们将论证这个结论并且给出一个实例。

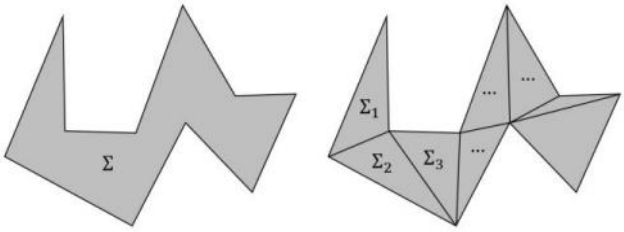


图 9 叠加原理示意图

2.2.1 叠加原理的论证

标量衍射理论及其计算公式已在前文给出，我们这里直接引用前文的结果。

以图 9 为例，它的衍射光场可以由式(2)给出，而由于二重积分的可加性，我们就可以对积分区域Σ进行拆分，体现在物理上就是对衍射区域进行分割，从而将若干个区域的衍射光场叠加。

$$\begin{aligned} E(P) &= \frac{A}{i\lambda d}\iint_{\Sigma} E_1(x_0, y_0)\, e^{ikr}\mathrm{d}\sigma \\ &= \frac{A}{i\lambda d}\left[\iint_{\Sigma_1} E_1(x_0, y_0)\, e^{ikr}\mathrm{d}\sigma + \iint_{\Sigma_2} E_1(x_0, y_0)\, e^{ikr}\mathrm{d}\sigma + \iint_{\Sigma_3} E_1(x_0, y_0)\, e^{ikr}\mathrm{d}\sigma + \cdots\right] \end{aligned} \tag{20}$$

上式中第一行即为总区域的衍射光场表达式，但由于区域Σ并不规则，即没有很高的对称性，其积分求解十分复杂，难以给出积分边界条件，因此我们在数学上对积分区域进行分割，分割为若干个易于计算的三角形区域，将其光场分布叠加后即可得到与复杂区域相同的光场结果。

2.2.2 叠加原理的实例

在前文讨论的基础之上，由于正方形的分解十分简易，故我们考虑一个最简单的正方形分割例子。以图 10 为例，我们将一个正方形按照对角线分割为两个斜边重合的等腰直角三角形，分别标记它们的区域为Σ_1、Σ_2。

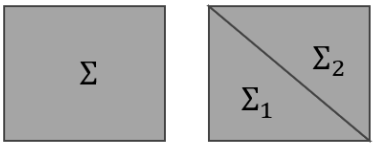


图 10 正方形分割实例

两者的坐标原点均选取于正方形的中心，则根据标量衍射理论，倘若我们直接对正方形区域Σ进行积分，则有：

$$\begin{aligned} E(P) &= \iint_{\Sigma} A e^{-\frac{ik(xx_0+yy_0)}{d}} \mathrm{d}x_0 \mathrm{d}y_0 \\ &= A\int_0^a \mathrm{d}x_0 \int_0^{b-\frac{b}{a}x_0} e^{-\frac{ik(xx_0+yy_0)}{d}} \mathrm{d}y_0 \\ &= A\frac{4d^2}{kxy}\cdot \sin\frac{akx}{2d}\cdot \sin\frac{aky}{2d} \end{aligned} \tag{21}$$

然后根据叠加原理，我们可以将两区域Σ_1, Σ_2的光场结果叠加从而得到相同的解析解。其中，由式(13)可知，两区域的夫琅禾费衍射光场分别为：

$$\begin{cases} E_1(P) = \dfrac{d^2 e^{-\frac{ik(ax+by)}{2d}}\left[ax\left(-1+e^{\frac{ibky}{d}}\right)-by\left(-1+e^{\frac{iakx}{d}}\right)\right]}{k^2xy(ax-by)} \\ E_2(P) = -\dfrac{d^2 e^{\frac{ik(ax+by)}{2d}}\left[a\left(x-xe^{-\frac{ibky}{d}}\right)+by\left(-1+e^{-\frac{iakx}{d}}\right)\right]}{k^2xy(ax-by)} \end{cases} \tag{22}$$

将二者相加，即可以得到：

$$E_1(P)+E_2(P) = -\frac{d^2\left(-1+e^{\frac{iakx}{d}}\right)\left(-1+e^{\frac{iaky}{d}}\right)e^{-\frac{ik(ax+ay)}{2d}}}{k^2xy} \tag{23}$$

再进一步化简，即可得到$E(P)=E_1(P)+E_2(P)$。因此，正方形的实例验证了叠加原理的正确性，并且后面的更多解析结果的准确性也进一步将其验证。

2.3 仿真原理

之前我们已经推导了理论公式，由于一般的实验一般包含三部分，实验、理论、仿真，故我们考虑用数值的方法对所设计的复杂结构进行数值计算衍射图像，仿真原理我们在此部分进行介绍。

2.3.1 夫琅禾费衍射

设e_{in}为孔径分布，e_{out}为衍射成像,由衍射卷积公式得：

$$e_{out}(x',y',z) = e_{in}(x,y) * \left(\frac{e^{i2\pi\frac{z}{\lambda}}}{i\lambda z} exp\left(i\pi\frac{x^2+y^2}{\lambda z}\right)\right) \tag{24}$$

展开得到：

$$e_{out}(x',y',z) = \frac{1}{i\lambda z}\exp\left\{i2\pi\frac{z}{\lambda}\right\}\times\iint e_{in}(x,y)\exp\left\{i\pi\frac{(x'-x)^2+(y'-y)^2}{\lambda z}\right\}dxdy \tag{25}$$

有：

$$\frac{(x'-x)^2+(y'-y)^2}{\lambda z} \approx \frac{x'^2-2xx'+y'^2-2yy'}{\lambda z} \tag{26}$$

则我们令：

$$u=\frac{x'}{\lambda z}, v=\frac{y'}{\lambda z} \tag{27}$$

可得：

$$e_{out}(x',y',z)=\exp\left\{i2\pi\frac{z}{\lambda}+i\pi\frac{x'^2+y'^2}{\lambda z}\right\}\times\iint e_{in}(x,y)\exp\{-i2\pi(ux+vy)\}dxdy \quad (28)$$

而对于夫琅禾费衍射，有$z\to\infty$，则有：

$$\exp\left\{i2\pi\frac{z}{\lambda}+i\pi\frac{x'^2+y'^2}{\lambda z}\right\}\to\exp\left\{i2\pi\frac{z}{\lambda}\right\} \quad (29)$$

参考二维傅里叶变换定义式：

$$E(u,v)=\iint_{-\infty}^{\infty}e(x,y)\exp\big(-i2\pi(ux+vy)\big)dxdy \quad (30)$$

即可得：

$$e_{out}(x',y',z)\propto E(u,v) \quad (31)$$

故对于夫琅禾费衍射，光源经孔径衍射成像可以视为孔径在衍射平面上的二维傅里叶变换图像。

2.3.2 菲涅尔衍射

设e_{in}为孔径分布，e_{out}为衍射成像，由衍射卷积公式展开得到：

$$e_{out}(x',y',z)=\frac{1}{\mathrm{i}\lambda z}\exp\left\{i2\pi\frac{z}{\lambda}\right\}\times\iint e_{in}(x,y)\exp\left\{i\pi\frac{(x'-x)^2+(y'-y)^2}{\lambda z}\right\}dxdy \quad (32)$$

在菲涅耳近似的基础上，我们将其中的相因子展开得：

$$e_{out}(x',y')=\frac{\exp(ikz)}{i\lambda z}\exp\left[\frac{ik}{2z}(x^2+y^2)\right]\cdot\iint e_{in}(x,y)\exp\left[\frac{ik}{2z}(x'^2+y'^2)\right]\exp\left[-i\frac{2\pi}{\lambda z}(xx'+yy')\right]dxdy \quad (33)$$

为方便计算，我们引入二次相位因子函数

$$Q_{\alpha,\beta}(2x,2y)=\exp\left[i\pi\left(\frac{x^2}{\alpha}+\frac{y^2}{\beta}\right)\right] \quad (34)$$

则可得

$$e_{out}(x',y',z)=\frac{exp(\mathrm{i}kz)}{i\lambda z}Q_{\lambda z}(x',y')\mathbb{F}\{e_{in}(x,y)Q_{\lambda z}(x,y)\} \quad (35)$$

其中$\mathbb{F}$表示二维傅里叶变换，故光源经孔径衍射成像的菲涅尔积分可以通过傅里叶变换求得。

2.3.3 反演变换

对于正向过程，设e_{in}为孔径分布，e_{out}为衍射成像,由衍射卷积公式得：

$$e_{out}(x',y',z)=\frac{1}{\mathrm{i}\lambda z}\exp\left\{i2\pi\frac{z}{\lambda}\right\}\times\iint e_{in}(x,y)\exp\left\{i\pi\frac{(x'-x)^2+(y'-y)^2}{\lambda z}\right\}dxdy \quad (36)$$

由此，对于反演过程已知衍射成像$e'_{out}(x',y',z)$，为得到孔径分布$e_{in}'(x,y)$，需经过以下运算：

$$e_{in}^{\prime(x,y)}=\iint(e'_{out}(x',y',z)/const)\exp\left\{i\pi\frac{(x-x')^2+(y-y')^2}{\lambda z}\right\}dx'dy' \quad (37)$$

其中$const$为正向过程中增加的振幅调制因子。

值得注意的是，在反演过程中进行的二维傅里叶逆变换需要已衍射成像$e'_{out}(x',y',z)$的相位信息，而在实际实验中 CCD 捕捉到的衍射图像为光强信息，实际衍射相位已丢失。故我们采用理论计算得到的带有相位项的衍射结果通过仿真反演得

到孔径分布，再与理想孔径分布对比。而对于实验部分，我们可以在未来通过全息成像得到带有相位的孔径分布数据进行反演对比。

2.4 三维物体衍射理论初步

同样，我们可以将上述三角形分割求解思路与方法应用于三维物体的衍射。对于一个具有复杂表面的三维物体，我们可以利用有限元法将其分割为三角形块，对于每个有限元所在平面定义正交坐标(x,y)。

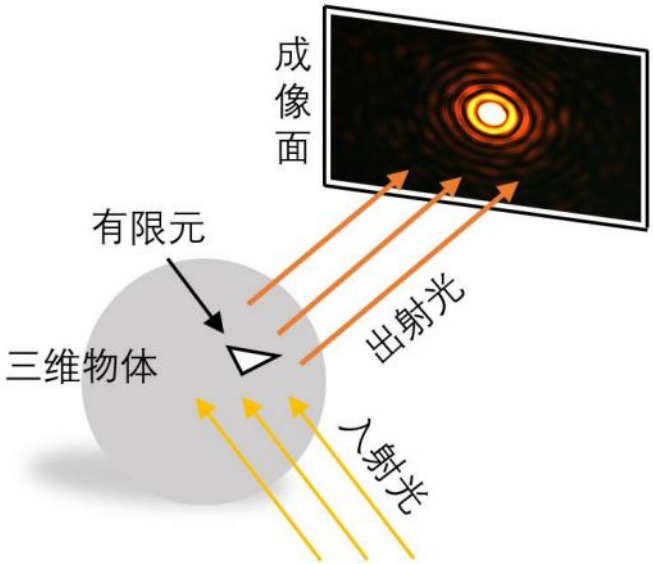


图 11 三维物体衍射成像示意图

考虑特定角度的入射光，可将波矢作为法线定义入射平面，在入射平面内选取正交坐标(ρ,θ)，故对应有限元在入射平面的坐标微分变换为：

$$\mathrm{d}x\mathrm{d}y=\frac{\partial(x,y)}{\partial(\theta,\rho)}\mathrm{d}\theta\mathrm{d}\rho \tag{38}$$

其中，雅可比行列式$\frac{\partial(x,y)}{\partial(\theta,\rho)}$中正交坐标的夹角可由波矢于有限元平面法线夹角得到。值得注意的是，部分有限元存在前后位置的遮挡关系，对于入射平面投影重叠部分避免进行重复运算。由此，对于分割完毕的入射面有限元，可以运用上文提出的计算方法高效地得出运算结果，这也是我们实验区别于传统衍射理论的优越性。

同理，对于相位的光场信息，我们也可以通过上文的反演方法得到空间点的相对位置分布。上述实现将在我们未来的研究中进一步探索。

2.5 图形处理

2.5.1 机器视觉边缘检测

耳切法分割需要多边形顶点，故我们考虑利用机器视觉的方法读取顶点加快效率，具体方法为：

首先将一张包多边形的图片导入，对其进行灰度处理，然后利用自动二值化的方法得正多边形的图案。

然后利用 OpenCV 中的 findContours 函数，它基于一套叫做“边缘追踪”的算法，从图像的某个像素点开始，顺时针或者逆时针方向检测，来搜索下一个与当前点相邻的边缘点，这个过程会一直持续到所有的边缘点都被访问到，从而形成一个闭环，也

就是一个轮廓。经过运行后能够得到一系列轮廓以及层次树(表示轮廓之间的关系)，通过选取最外部的轮廓，进行下一步操作。

接着为删除冗杂的顶点，我们利用多边形的总边长进行近似，通过控制点间隔为多边形总边长的几分之一，对复杂图形以及简单图形进行分别近似。对复杂图形，我们应当使得其点间隔较小(总边长的 0.1%)，而简单图形则应控制点间隔较大(总边长的 1%)，从而得到一个十分精确的边缘以及顶点。

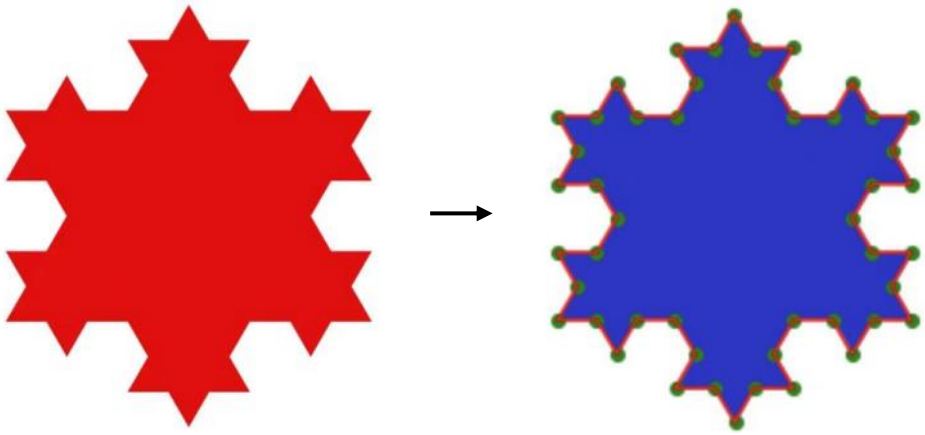


图 12 边缘检测示意图

最后将顶点传递给耳切法模块，利用多边形顶点进行分割处理。

2.5.2 耳切法分割多边形

在计算几何和图形学领域，多边形的三角剖分是一项基础而关键的问题。为了实现高效准确的三角剖分，一种常见的方法是采用所谓的“耳切法(ear clipping)”[21]。

对于一个简单二维多边形，至少存在两个凸顶点，也就是我们所谓的“耳朵”。通过剪除这些“耳朵”，可以将多边形剖分为一系列三角形。重复这一流程直到多边形被分解为一系列的三角形。对于一个顶点而言，如果它与相邻的两个顶点可以组成一个三角形，并且这个三角形不包含其他顶点，且其内角小于或等于 180 度，那么这个顶点就被称为“耳朵”。需要注意的是，这个过程中多边形不会出现自相交情况。其分解步骤如图 13 所示。

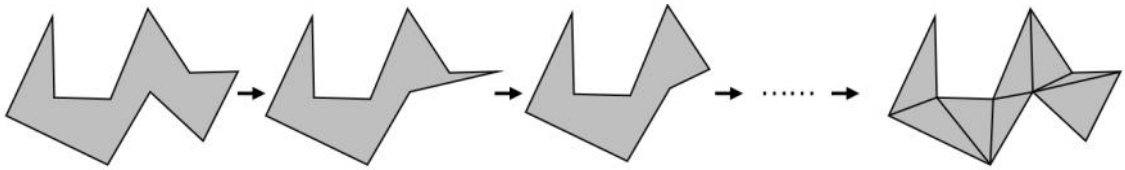


图 13 耳切法分解步骤

需要注意的是，耳切法一般只适用于单连通的多边形(图 14 左侧)，也即简单多边形，对于多边形之间只通过点连接的情况，我们可以将其拆分为两个多边形进行操作，即将某点所连接的两个多边形分别用耳切法操作，此时只需增加人为操作的一步即可。

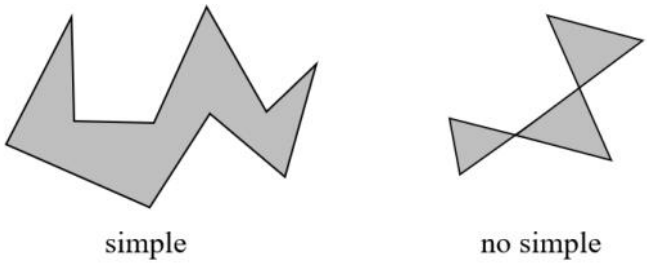


图 14 多边形分类

我们按照以下步骤实现了这一算法：首先，初始化一个空的结果数组和一个包含多边形所有顶点的数组。然后，遍历顶点数组：对于每个顶点，首先确定它是凹顶点还是凸顶点。我们在这里只处理凸顶点，判断方法是观察顶点与其两个邻居点形成的向量的方向。对于每个凸顶点，接下来判断它是否为一个“耳朵”。我们创建一个由该顶点及其相邻的两个顶点组成的三角形，并检查该三角形是否包含任何其他顶点。如果不包含其他顶点，则确认该顶点为一个“耳朵”。当找到一个“耳朵”后，将其剪切掉，即将其添加到结果数组中，并从原始数组中移除。重复以上步骤，直到顶点数组为空，多边形被完全剖分为三角形。代码对于多边形顶点的处理严格按照逆时针或顺时针方向，这一点是很有必要的，因为处理方向的不同可能会影响到顶点凸凹性质的判断以及最终的三角形分解结果。

由于 “尖锐”的三角形在某些情况下可能会引起计算错误或不稳定性，我们选取让每个三角形的最小角尽可能大的分解结果。若两个三角形有公共部分，我们会尝试将它们合并为一个四边形，然后使用尝试将这个四边形划分为两个新的三角形并计算两种方式的最小角的弧度值，然后选择元最小角最大的一种方式来进行切割。在优化过程中，需要确保新生成的两个三角形不会与优化列表中的其他三角形有重叠。这是通过遍历剩余的其他三角形，检查新生成的两个三角形的面积的大小来验证的。只有当所有其他三角形都不与新生成的两个三角形有重叠时，才会用这两个新的三角形分割方式替换原来的三角形分割方式。

最后，整个优化过程结束后，最终得到每个四边形的最优划分方式，从而使得整个剖分系统的最小角最大。

为了将多边形分解成直角三角形，我们可以在耳切法的结果上进行操作。首先，我们需要判断分解出来的每个三角形是否为直角三角形。如果不是直角三角形，我们可以选择其中最长的边（对于特殊图形，选择底边），然后构造该边所对应的高，通过这个高，将三角形分解成两个直角三角形。

而对于圆弧，我们可以先去圆弧起点、中点和末点构建一个三角形，再取圆弧起点、起点与中点之间的中点和中点构建三角形，以此类推，将圆弧近似分解为多个三角形。

2.5.3 弧形近似

对于圆弧，我们可以先取圆弧起点、二等分点和末点构建一个三角形，再取圆弧起点、第一个四等分点和二等分点以及圆弧末点、第三个四等分点和二等分点构建三角形，再以此类推，将圆弧近似分解为多个三角形，通过控制迭代次数即可控制分割的三角形个数，即控制精度。其效果如图 15 所示。

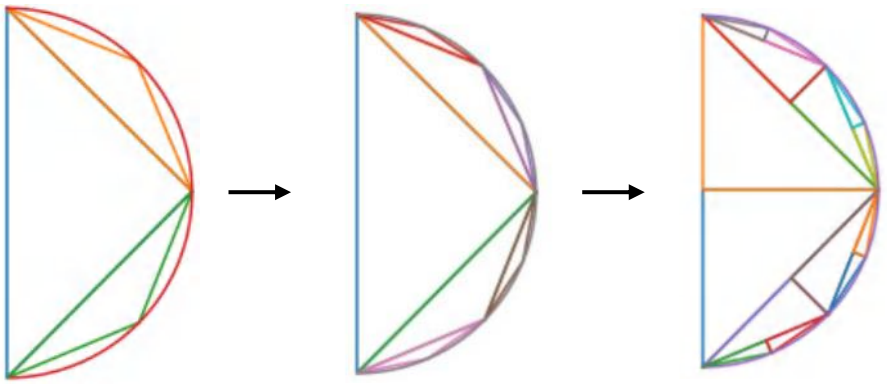


图 15 圆弧近似的过程

可见，其在三次迭代时丢失精度几乎可以忽略了，我们在后面也会研究其丢失的精度大小。

同之前的耳切法，为了将圆弧近似分解成直角三角形，我们可以在上述的结果上进行操作。先判断分解出来的每个三角形是否为直角三角形，如果不是直角三角形，构造最长边所对应的高，通过这个高，将三角形分解成两个直角三角形。

通过此方法，我们可以得到将圆弧近似分解成直角三角形的结果，以便于扩展理论适用范围。之后，我们也会继续探究更复杂的非圆弧的近似方法，进而获得更加一般化的近似方式。

2.6 复杂结构设计方案

2.6.1 初步设计

对于三角孔，为统一分解模式，我们选定了一种基础三角孔形状，即为直角三角形，因为通过直角三角形的拼接、拉长，利用做垂线的方法我们可以轻易地拼出其余类别的三角形，即钝角三角形、锐角三角形等等，其示意图如图 16 所示。

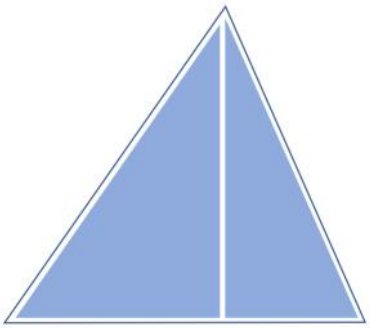


图 16 直角三角形拼接可行性示意图

其基本思路即为选取其中一角(钝角三角形应选择最大的角，为满足分割线在三角形内)对其对边作垂线，此时三角形即被分解为两个直角三角形。可以见得直角三角形作为基础三角孔进行分解是完全可行的，故作为我们的基础三角孔。

2.6.2 进阶设计

对于进一步的复杂结构，我们首先想到了正多边形结构，首先设计了偶数边形，其均较易分解为直角三角形，并且设计了圆形用于讨论弧线近似带来的误差，其示意图如图 17 所示。

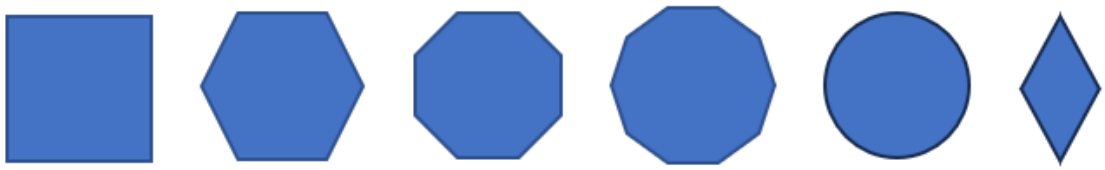


图 17 基础图形示意图

并且我们也设计了一部分奇异结构，其结构包括多角星和直角三角形直接组合的结果，以及一些环形结构，效果如图 18 所示。

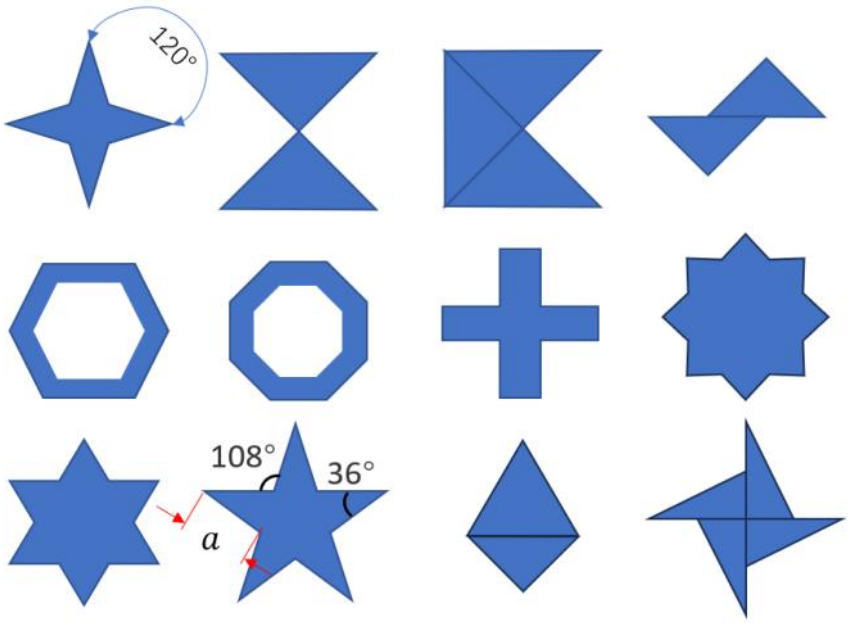


图 18 奇异结构示意图

2.6.3 弧形及无规则设计

最后我们设计了完全无规则的图形，完全用三角形进行了拼接，以便于进行理论验证，其结构如图 19 所示

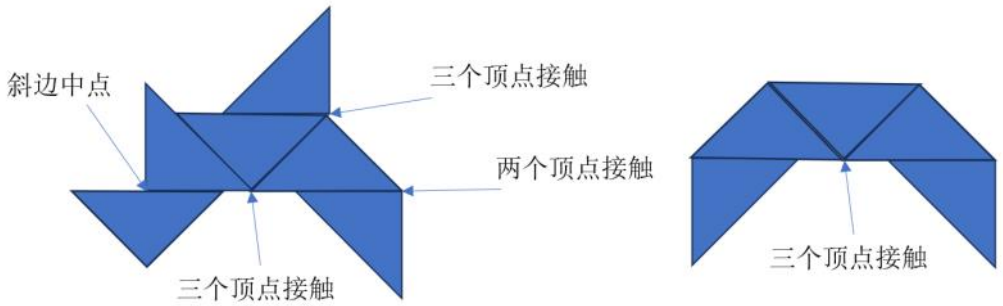


图 19 无规则图形示意图

并且我们设计了弧形结构，用于探索弧形近似，由于弧形近似还在初步阶段，故我们仅仅将半圆形与基础图形拼接，得到的图形如图 20 所示

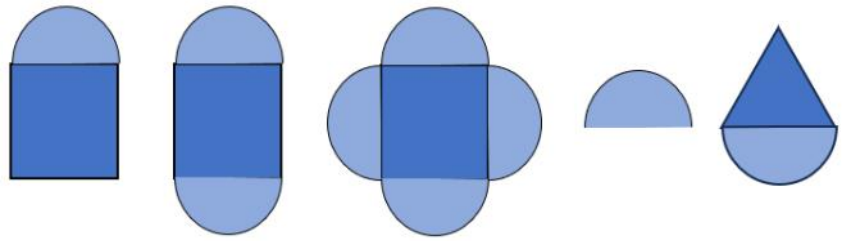


图 20 弧形拼接图形示意图

3. 实验装置设计

3.1 研究流程

本次研究中，我们的结果主要分为理论部分和实验部分，理论部分负责图形处理

以及解析的导出，实验部分负责给出真实实验的结果，从而与理论、仿真进行对比，验证三角形分割的正确性。我们本次研究的流程图如所示

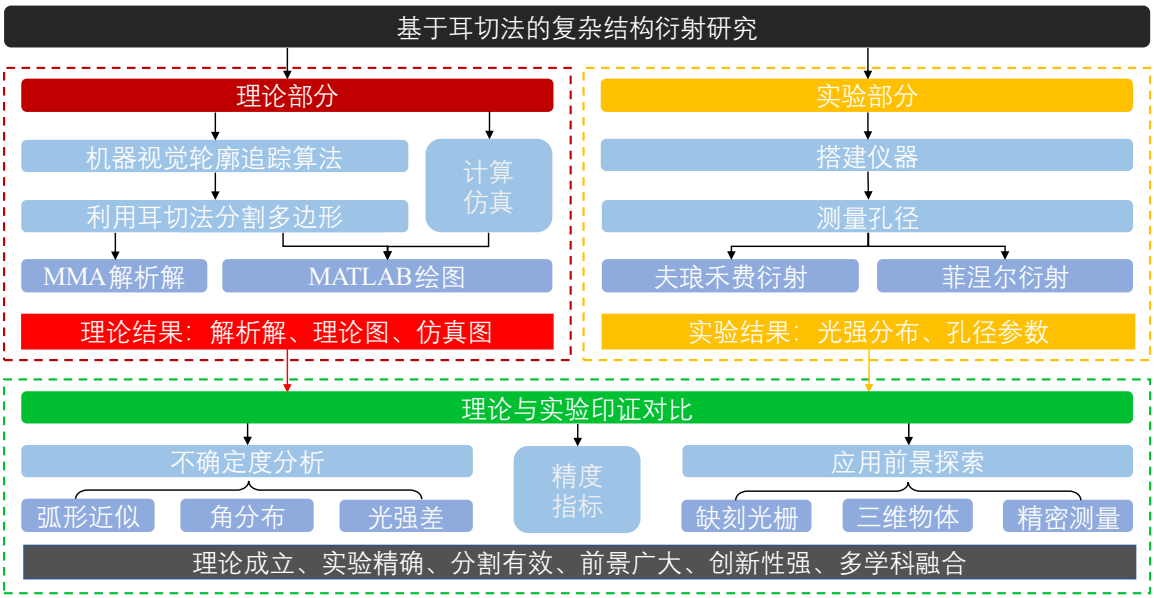


图 21 实验流程图

理论与实验并进，相互交叉验证。

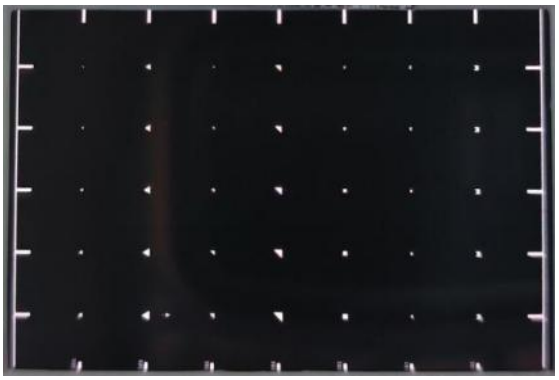
3.2 实验仪器

在本次实验中，我们所用到的实验仪器如表 1 所示。

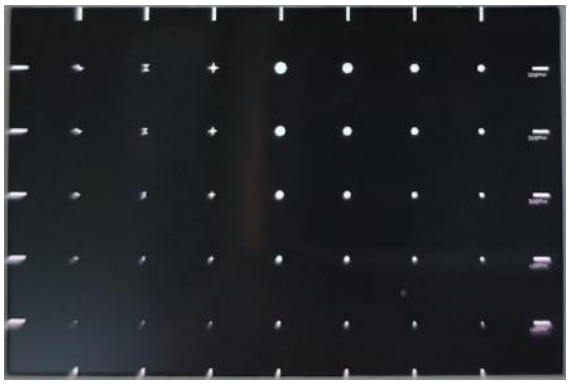
表 1 实验用仪器表

序号	名称	型号	参数	用途
1	激光光源	氦氖激光器	0.8mW/1.5mW	发射成像激光
2	CCD 相机	MER-310-12UC /MER-630-60U3C-L	2048*1536 3.20μm*3.20μm	拍摄衍射图像
3	实验用测控计算机	小新 pro14	ADM 锐龙 6800HS	捕捉衍射图像
4	光学底座	GCM	7 个	稳定光学元件
5	凸透镜	/	焦距f=15.00cm	搭建夫琅禾费衍射光路
6	夹板	SZ-12	1 个	稳定孔薄片
7	扩束镜	GCO-2503	5*5	扩大激光光源覆盖面
8	吸收型中性密度滤光片	HANF-50*50	T=10%、25%、50%	削减光强防止过曝
9	智能生物一体成像系统	SmartMi500B	/	显微观察小孔
10	孔薄片	/	4 片	印制复杂结构
11	小孔光阑	GCM-4301M	SZ-15	调节激光准直

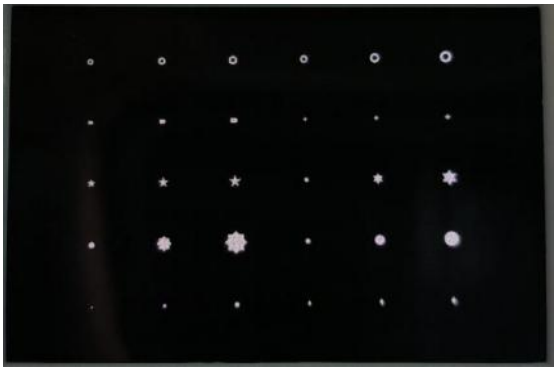
我们根据之前的孔设计进行了孔订制，整体效果图如图 22 所示。



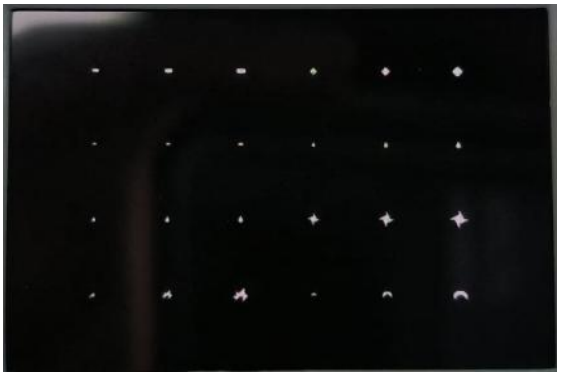
(a)孔板 1



(b)孔板 2



(c)孔板 3



(d)孔板 4

图 22 孔薄板整体示意图(具体用到的孔实际大小见附录 D.实际孔尺寸)

3.3 实验步骤及装置设计

我们对各种复杂结构进行两种衍射的实验，即夫琅禾费衍射和菲涅尔衍射。

3.3.1 夫琅禾费衍射

(1)搭建光路

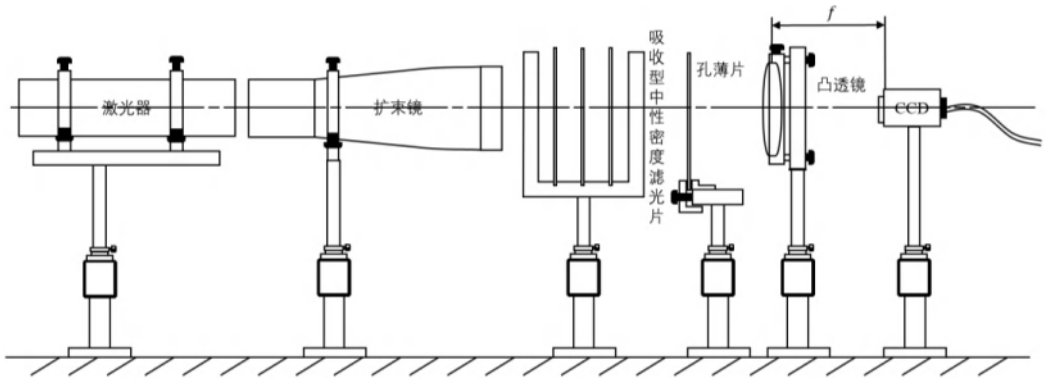


图 23 夫琅禾费衍射实验光路

如图 23 所示搭建仪器，先利用小孔光阑前后移动，调节激光准直，再将孔薄片固定在夹板上，保证透光方向水平，然后将激光器以及扩束镜放置于孔洞的正前方，调节激光器强度并适当在激光器与孔薄片之间增加滤光片，保证光束垂直入射到孔上。将透镜放置在孔洞的另一侧，并且保证透镜与 CCD 之间的距离为透镜焦距，从而使得激光通过孔洞后，平行光束能够汇聚到 CCD 上，实现出射光是平行光。

(2)用电脑捕捉夫琅禾费衍射的图样，测量并记录衍射图案的光强分布。

(3)换不同焦距的透镜进行夫琅禾费衍射成像，获得不同大小的衍射图案，从而捕捉更多细节。

在本次实验中，我们选取焦距参数为 150mm 的透镜，并利用自准法测量了透镜焦距为 150.0mm，搭建光路实现夫琅禾费衍射。

3.3.2 菲涅尔衍射

(1)搭建光路

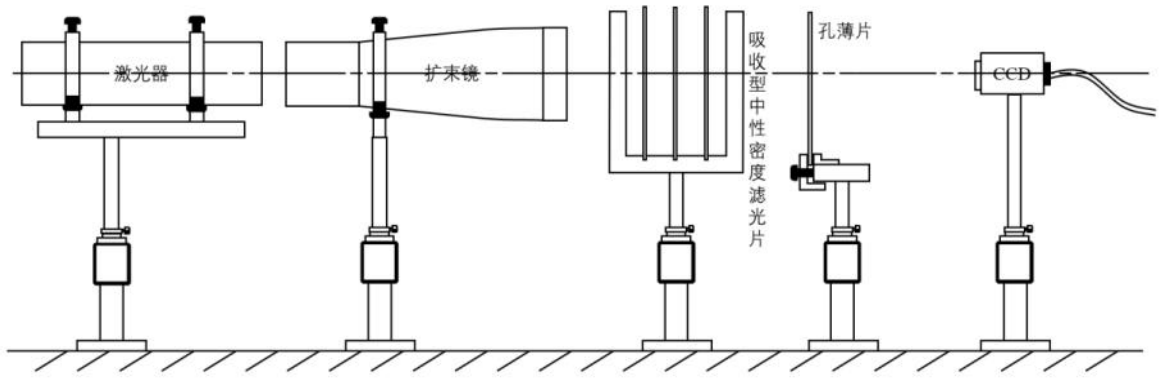


图 24 菲涅尔衍射实验光路

如图 24 所示搭建仪器，先利用小孔光阑前后移动，调节激光准直，再将孔薄片固定在夹板上，保证透光方向水平，然后将激光器以及扩束镜放置于孔洞的正前方，调节激光器强度并适当在激光器与孔薄片之间增加滤光片，保证光束垂直入射到孔上。将 CCD 放置在孔洞的另一侧，出射光会聚到 CCD 上。

(2)用电脑捕捉菲涅尔衍射的图样，测量并记录衍射图案的各种尺寸。

(3)调节 CCD 与孔薄片的距离，从而实现各种情况的菲涅尔衍射。

在本次实验中，我们选取 CCD 与孔薄片的距离为 150mm 成像，搭建光路实现菲涅尔衍射。

3.4 实验装置系统误差

在我们设计的孔以及实验装置中，存在以下不可避免的系统误差：

• 孔加工过程中，由于其制作精度有限，故无法避免产生误差。

• 光束通过扩束镜、衰减片、CCD 镜头等光滑表面发生反射，导致光场改变，从而对实验结果产生误差，但我们可以明显地分辨出反射光点，从而可以忽略。

• 激光出射光强分布不均匀。由于我们使用的是氦氖激光器，出射的激光发散角较小，利用扩束镜扩束时难免会造成光强分布不均匀的问题，从而对实验结果产生误差。

• 光学仪器表面可能存有灰尘，从而会出现一些细小结构的衍射干涉，影响高级衍射级次，但我们知道，由于灰尘十分细小，其衍射图像的强度会远小于孔的衍射强度，故其存在误差但可以忽略。

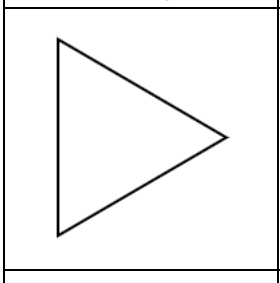
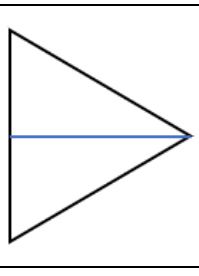
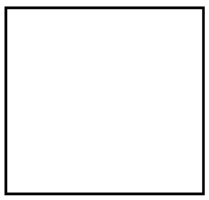
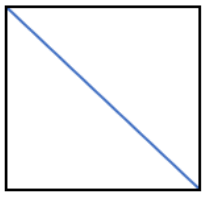
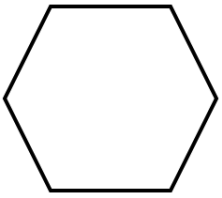
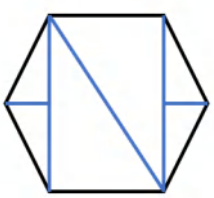
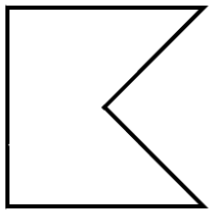
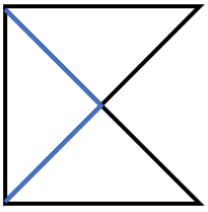
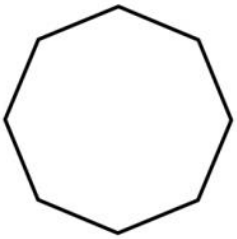
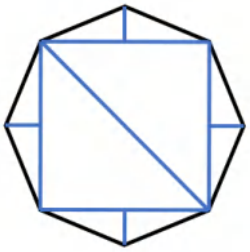
• 仪器分辨率有限。对于各种仪器，由于海森堡极限，其分辨率均有限，我们将其考虑为 B 类不确定度。

4. 实验结果及分析

4.1 分解模式

首先我们研究所用到的结构的分解模式，某些结构是我们直接利用三角形拼出的结果，故我们可以直接给出其分解模式，其余结构我们利用 Python 编程实现的耳切法进行分割，其部分孔的分解模式如表 2 所示。

表 2 孔结构分解模式

孔形状	分解模式	孔形状	分解模式
			
			
			

其余的孔我们在附录中的 B.耳切法分割程序运行结果中进行展示。

4.2 实验结果

4.2.1 夫琅禾费衍射

对于夫琅禾费衍射，我们按照实验步骤中所述的实验方法进行实验，选取透镜焦距为 150mm，采用不同形状、不同孔径大小的孔进行实验，并且我们利用所求理论结果与其对比，首先我们利用 Mathematica 给出各种图形的理论解析结果，然后在图中对其画出以证明其正确性，其图形如表 3 中孔模式所示，我们在下面对其简单描述并给出其光场的解析解，观察屏光强$I \propto$光场分布$|E|^2$：

①边长为 a 的等边三角形(原点位于内心)：

$$E(x,y)=\frac{d^2 e^{-\frac{iak(2\sqrt{3}x+3y)}{6d}}\left(\left(\sqrt{3}y-3x\right)e^{\frac{iak(\sqrt{3}x+2y)}{2d}}+\left(3x+\sqrt{3}y\right)e^{\frac{i\sqrt{3}akx}{2d}}-2\sqrt{3}ye^{\frac{iaky}{2d}}\right)}{k^2(3x^2y-y^3)} \quad (39)$$

②边长分别为 a、b 的直角三角形(原点位于直角顶点)：

$$E(x,y)=-\frac{d^2\left(a\left(x-xe^{-\frac{ibky}{d}}\right)+by\left(-1+e^{-\frac{iakx}{d}}\right)\right)}{k^2xy(ax-by)} \tag{40}$$

③边长为 a 的正方形(原点位于中心)：

$$E(x,y)=-\frac{d^2e^{-\frac{iak(x+y)}{2d}}\left(-1+e^{\frac{iakx}{d}}\right)\left(-1+e^{\frac{iaky}{d}}\right)}{k^2xy} \tag{41}$$

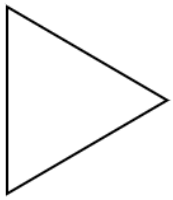
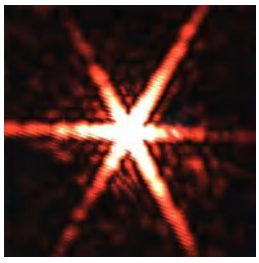
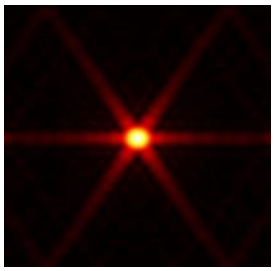
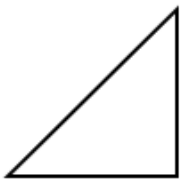
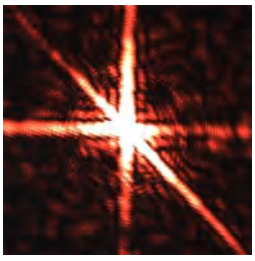
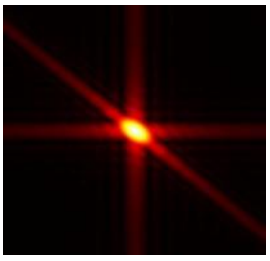
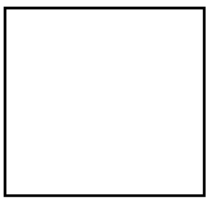
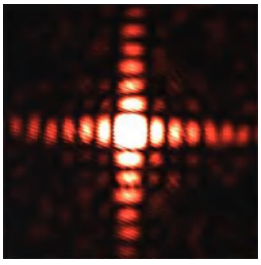
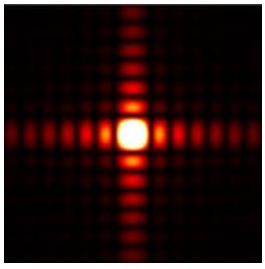
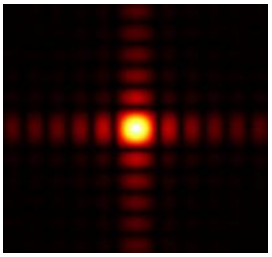
④边长为 a 的六边形(原点位于中心)：

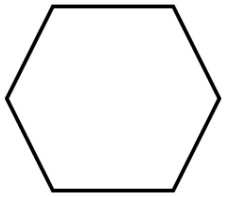
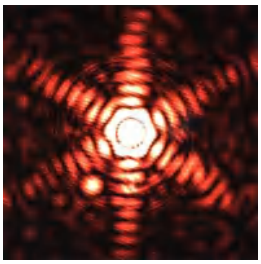
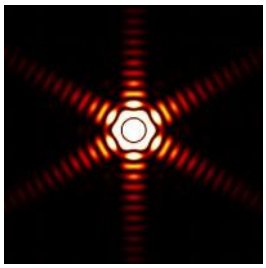
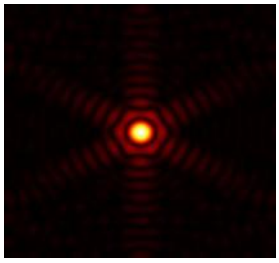
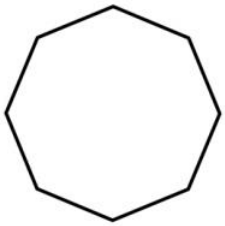
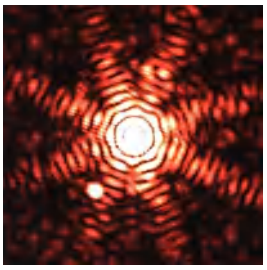
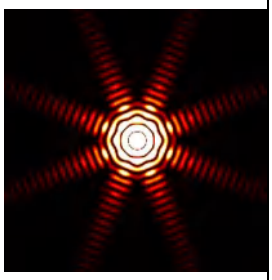
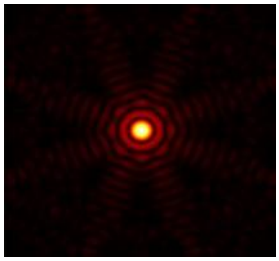
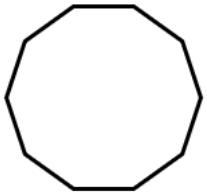
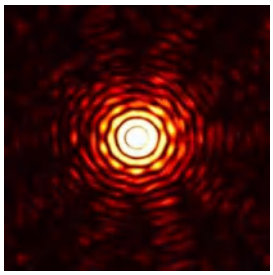
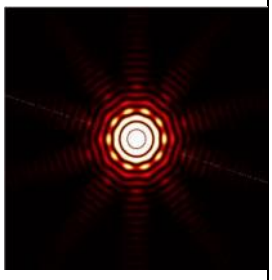
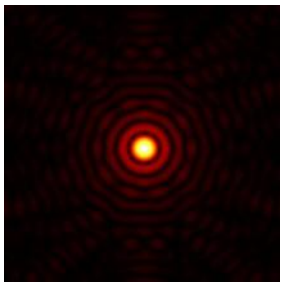
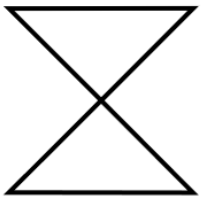
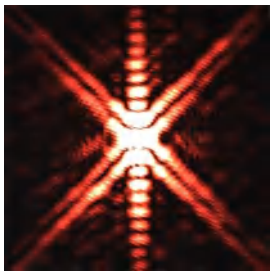
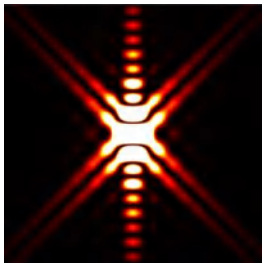
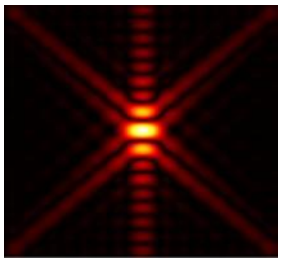
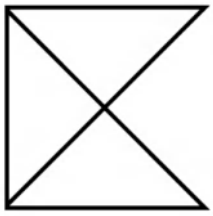
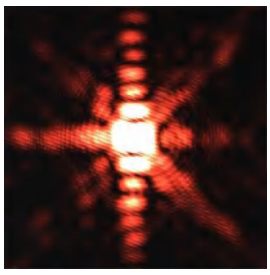
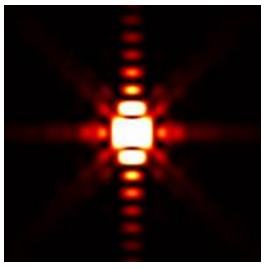
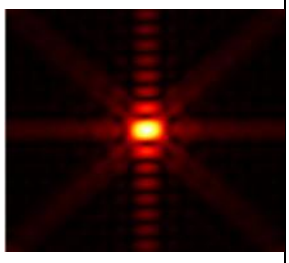
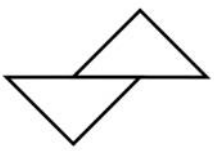
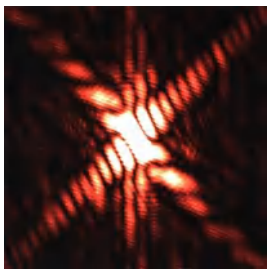
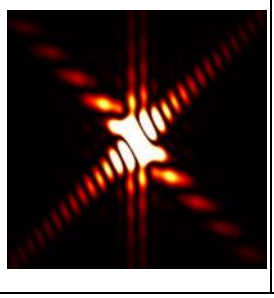
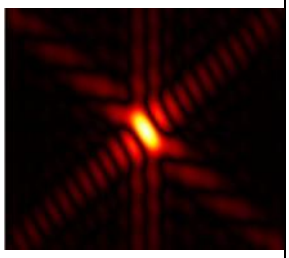
$$E(x,y)=-\frac{4d^2\left(3y\sin\left(\frac{akx}{2d}\right)\sin\left(\frac{\sqrt{3}aky}{2d}\right)+\sqrt{3}x\left(\cos\left(\frac{akx}{d}\right)-\cos\left(\frac{akx}{2d}\right)\cos\left(\frac{\sqrt{3}aky}{2d}\right)\right)\right)}{k^2(x^3-3xy^2)} \tag{42}$$

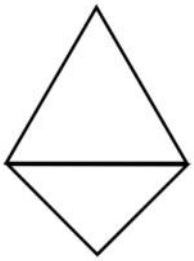
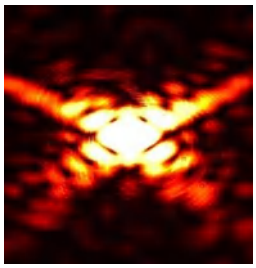
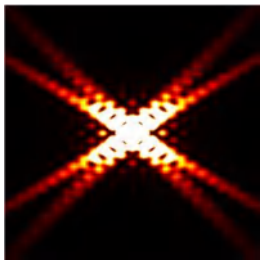
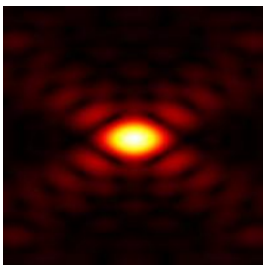
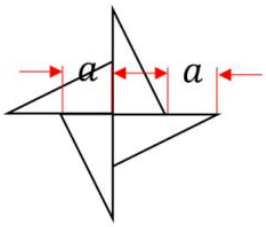
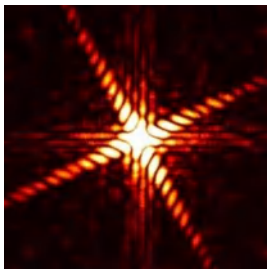
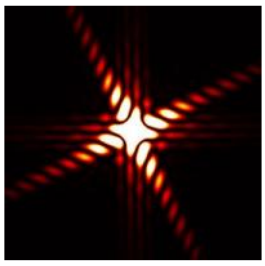
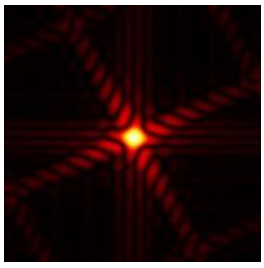
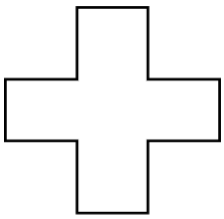
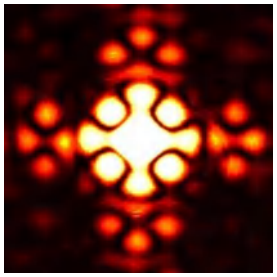
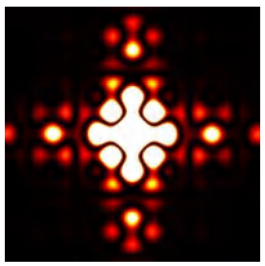
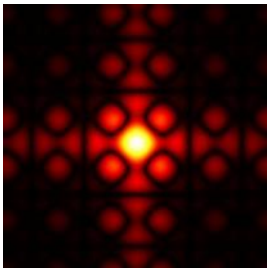
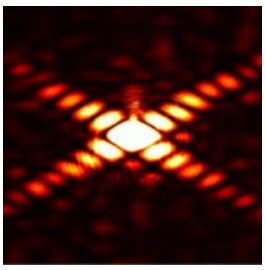
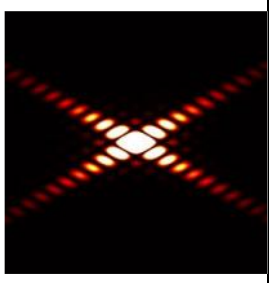
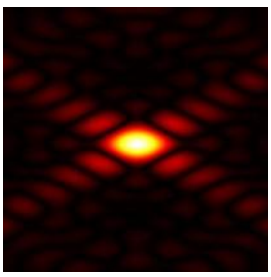
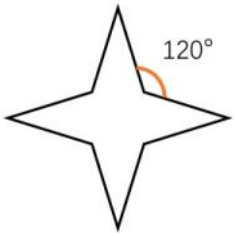
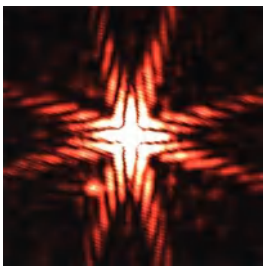
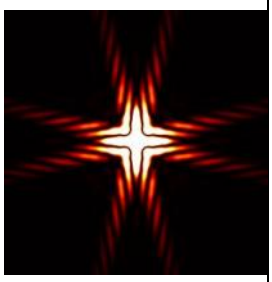
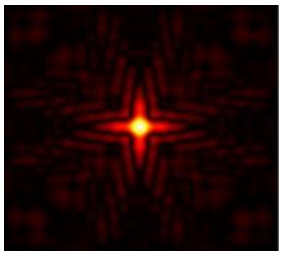
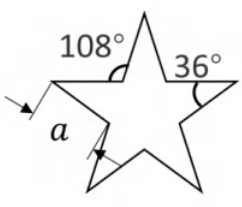
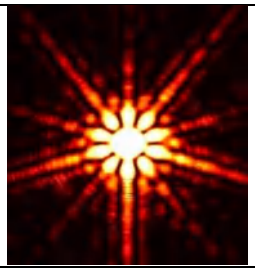
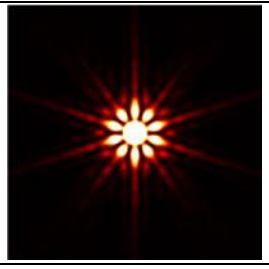
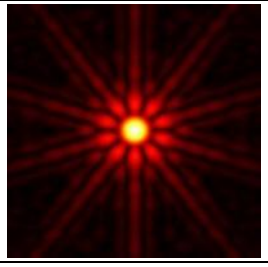
剩余部分的孔我们在附录中进行展示。

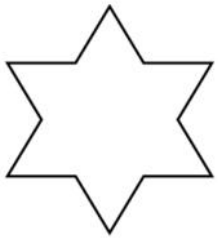
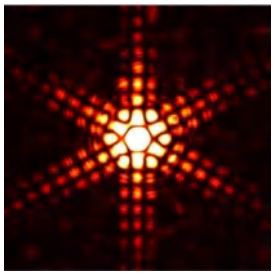
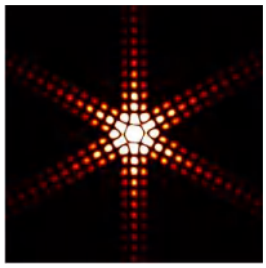
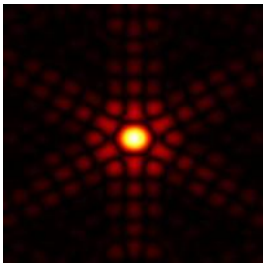
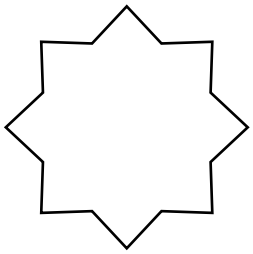
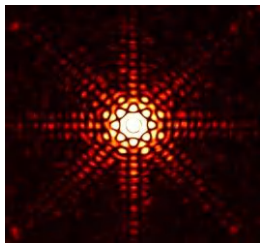
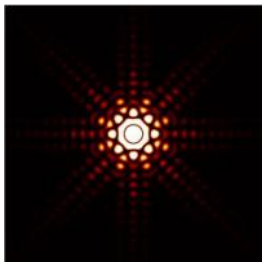
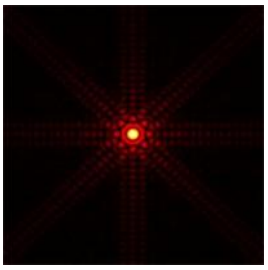
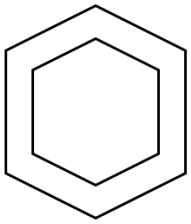
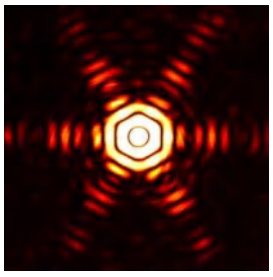
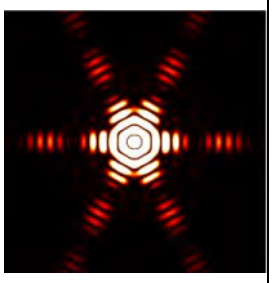
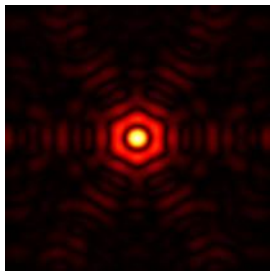
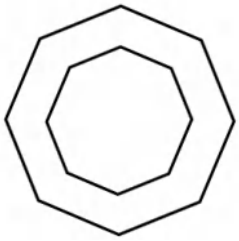
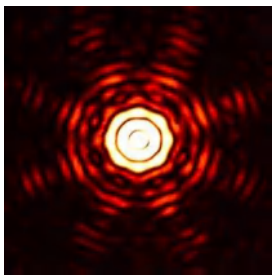
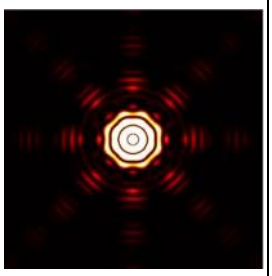
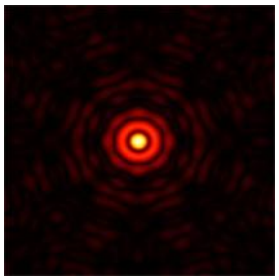
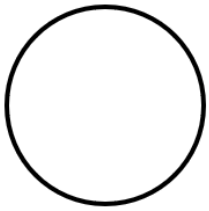
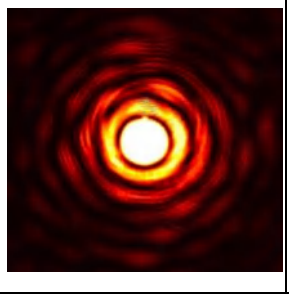
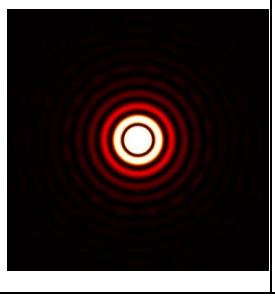
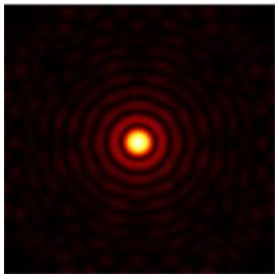
其实验、理论、仿真结果如表 3 所示。

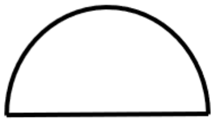
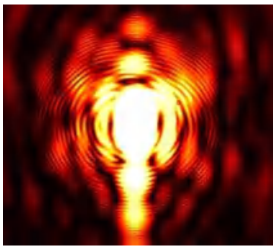
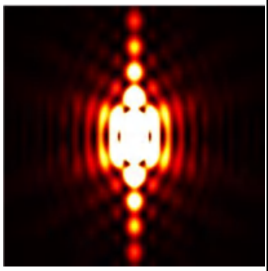
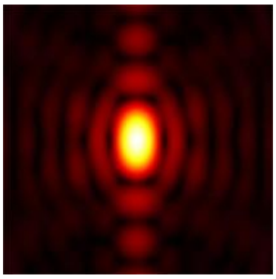
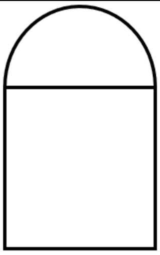
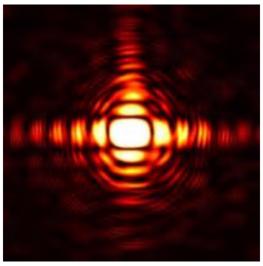
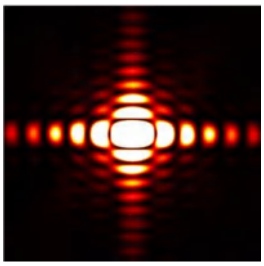
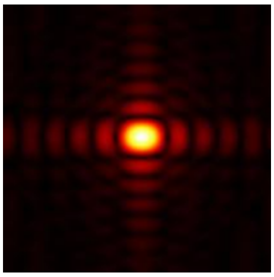
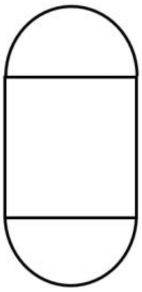
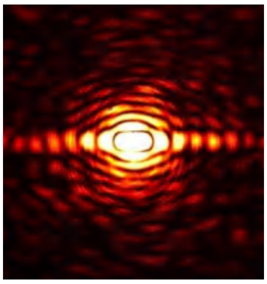
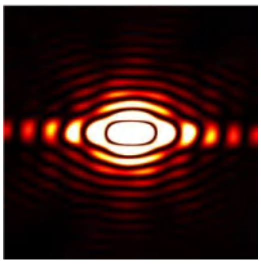
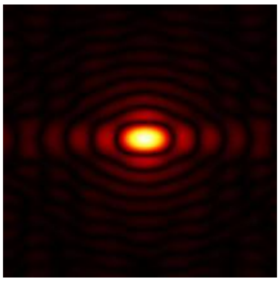
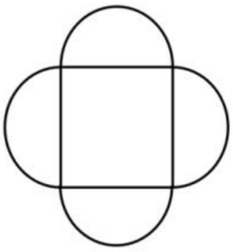
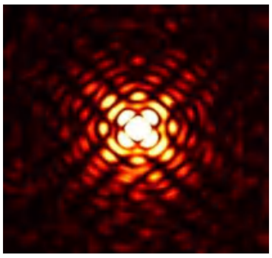
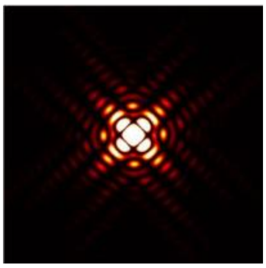
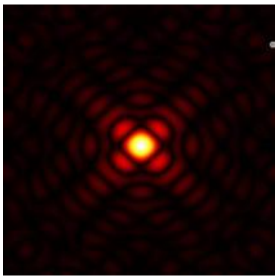
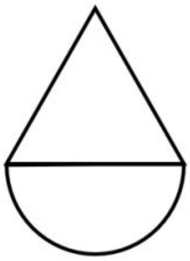
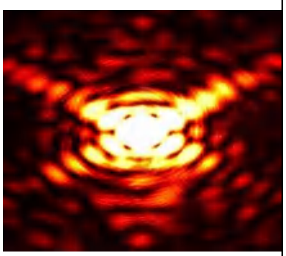
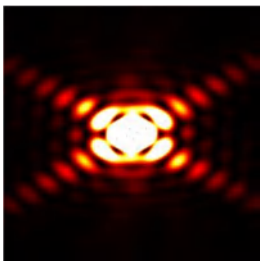
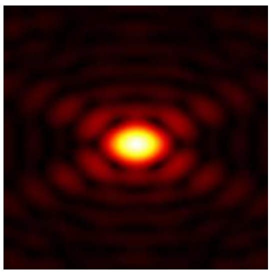
表 3 夫琅禾费衍射结果

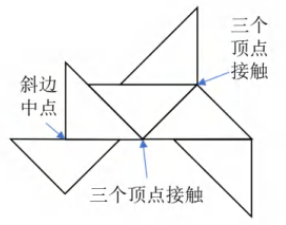
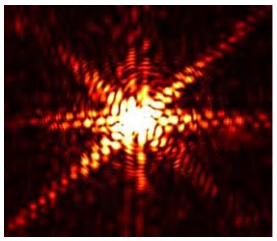
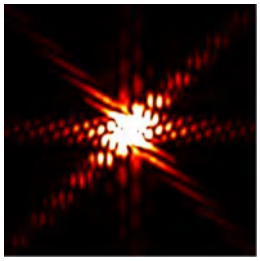
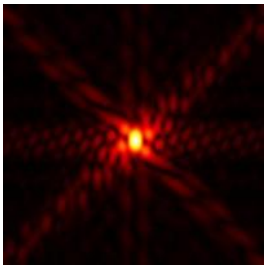
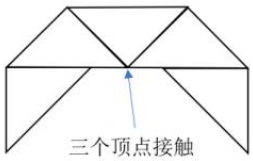
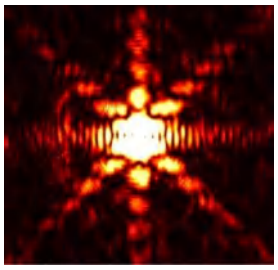
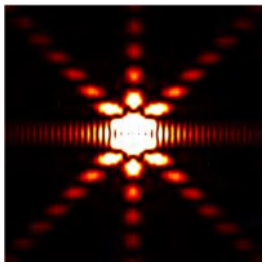
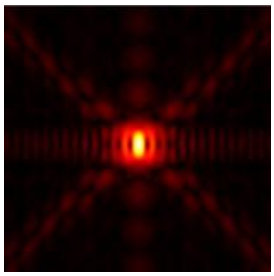
孔模式	实验结果	理论结果	仿真结果
 边长 500μm			
 直角边长 550μm			
 边长 300μm			

 边长 250μm			
 边长 250μm			
 边长 200μm			
 直角边长 300μm			
 直角边长 300μm			
 直角边长 300μm			

 等边边长 300μm			
 a =250μm			
 边长 100μm			
 边长 300μm			
 边长 300μm			
			

边长 300μm			
 边长 200μm			
 边长 300μm			
 外边长 300μm			
 外边长 250μm			
 半径 150μm			

 半径 150μm			
 边长 300μm			
 边长 300μm			
 边长 300μm			
 边长 300μm			

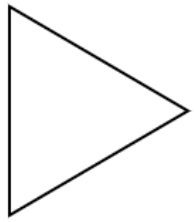
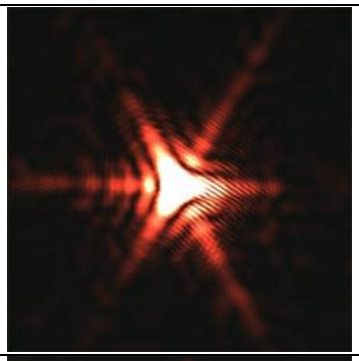
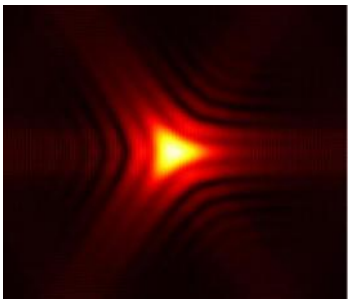
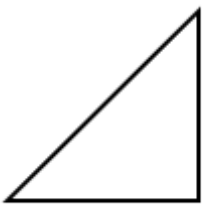
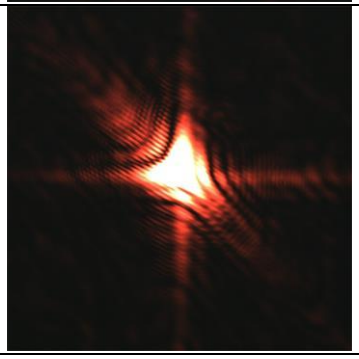
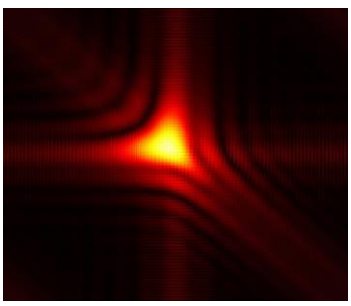
 直角边长 300μm			
 直角边长 300μm			

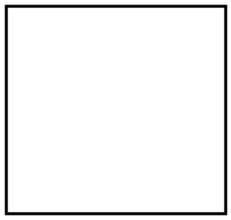
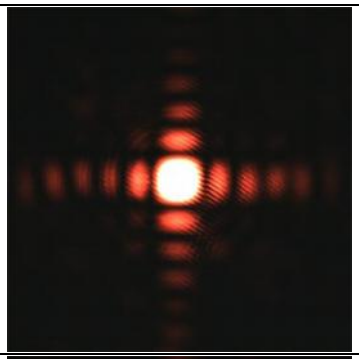
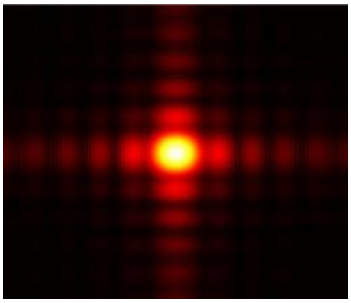
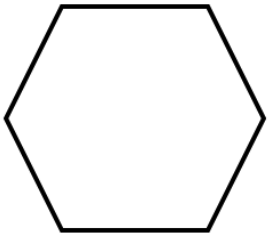
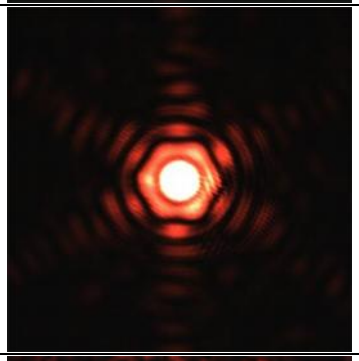
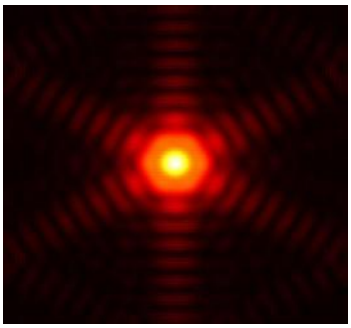
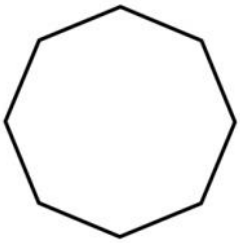
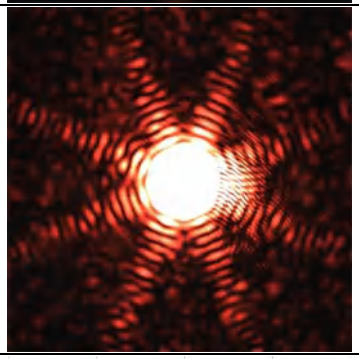
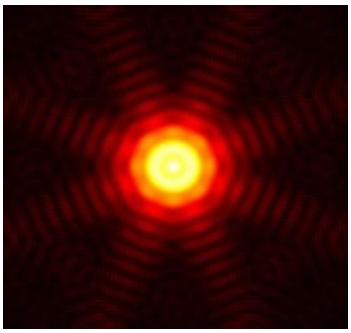
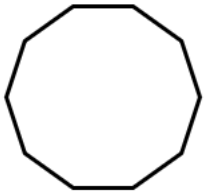
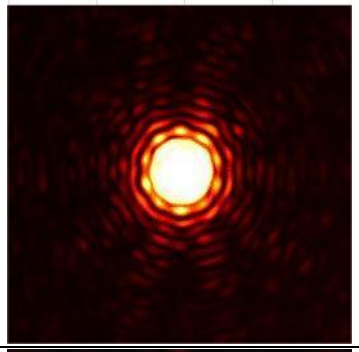
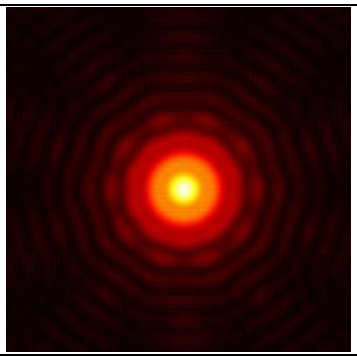
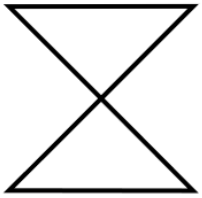
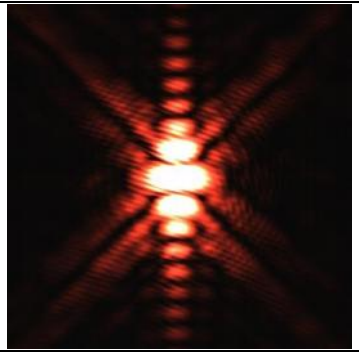
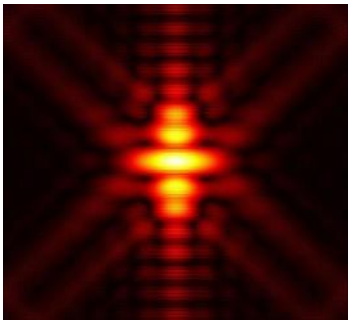
根据图像对比可见，其实验、理论、仿真的结果均一致，复杂结构的细节特征十分明显，且光强也基本一致，实验十分成功，拼凑的理论结果十分正确。

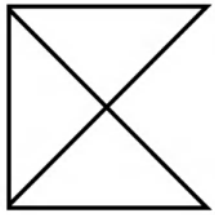
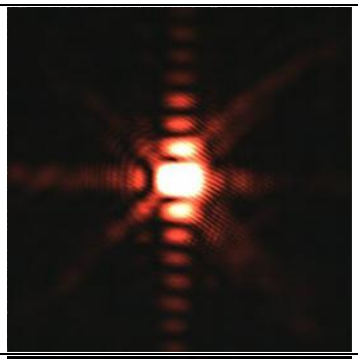
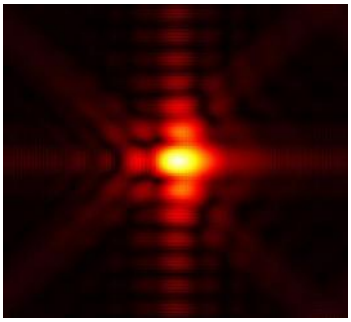
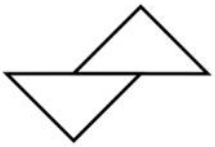
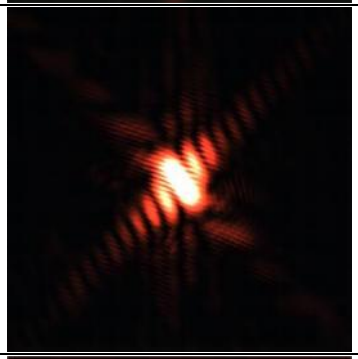
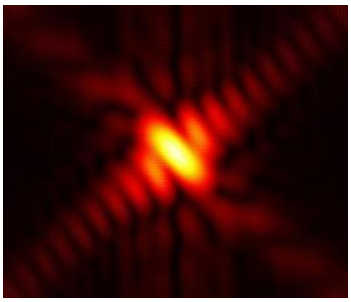
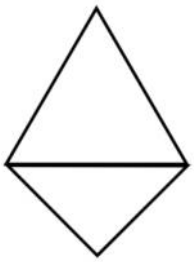
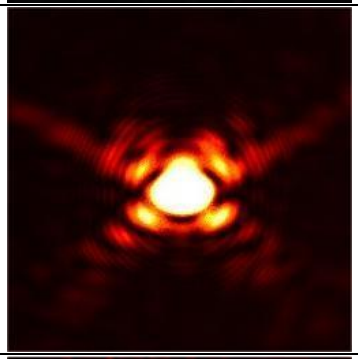
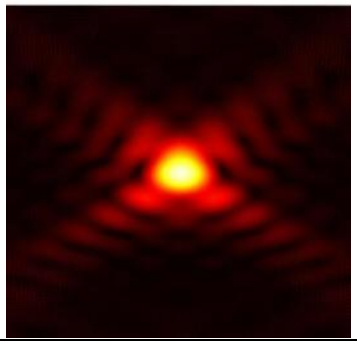
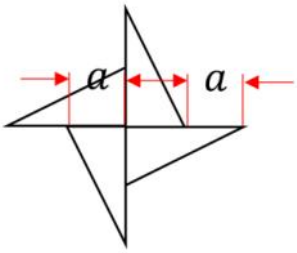
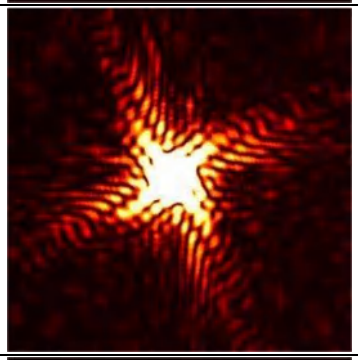
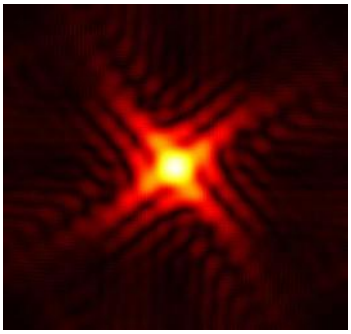
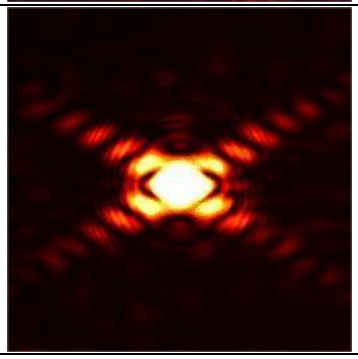
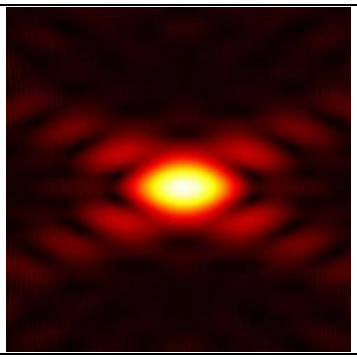
4.2.2 菲涅尔衍射

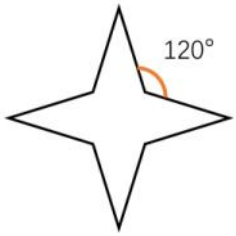
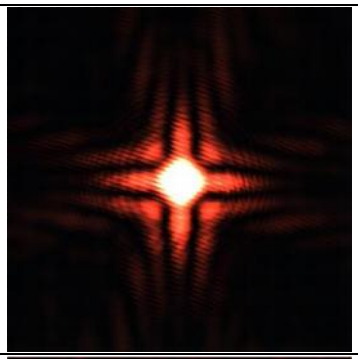
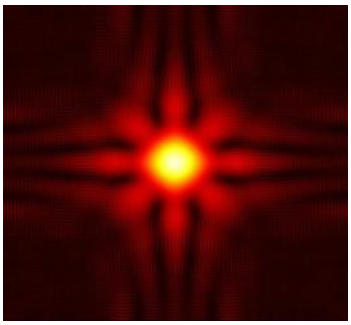
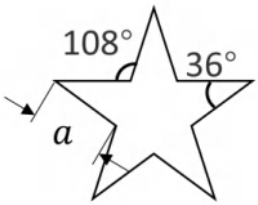
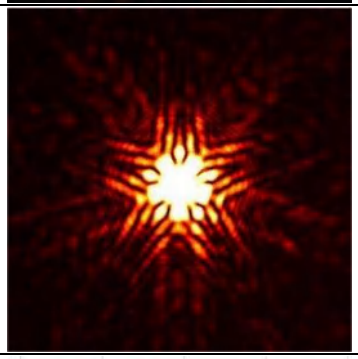
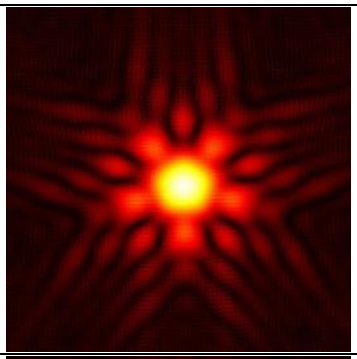
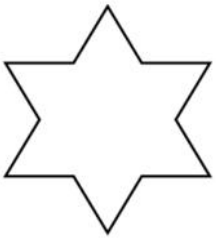
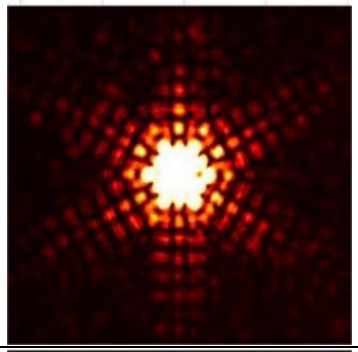
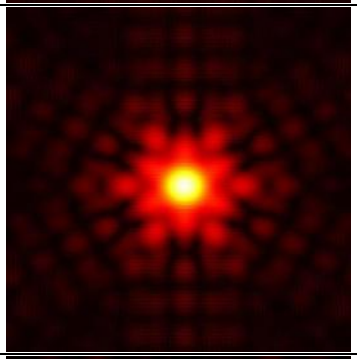
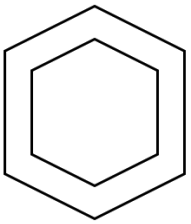
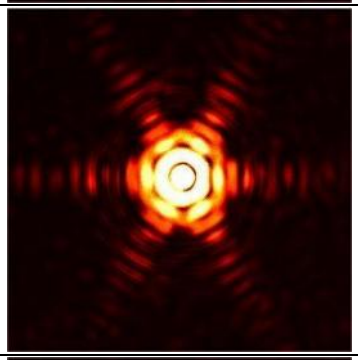
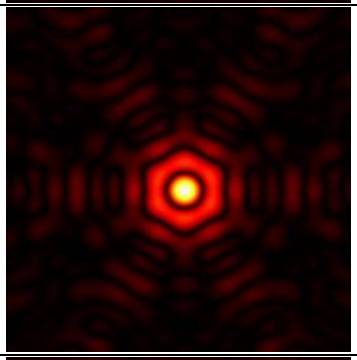
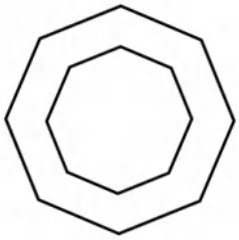
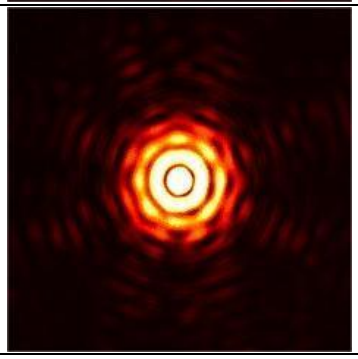
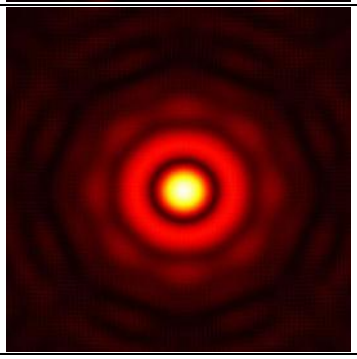
对于菲涅尔衍射，其较难给出理论结果，故我们仅仅探究这些结构的实验与仿真的对比，选取成像距离为 150mm，其结果如表 4 所示。

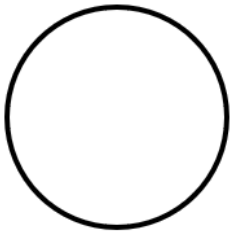
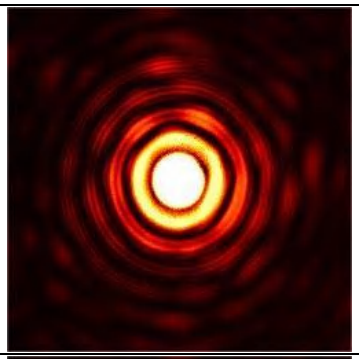
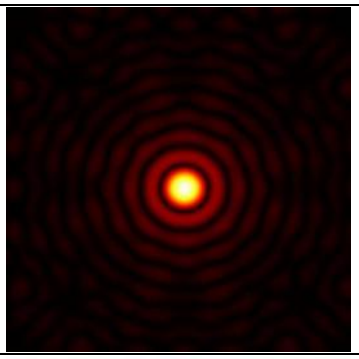
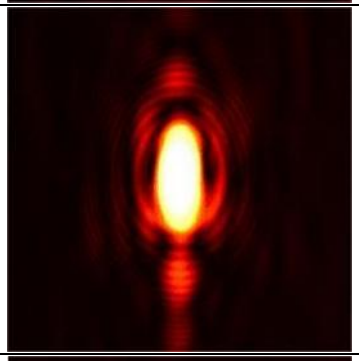
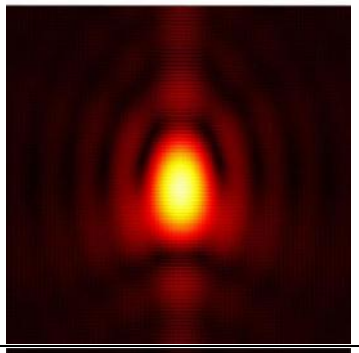
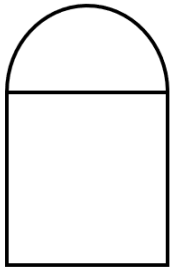
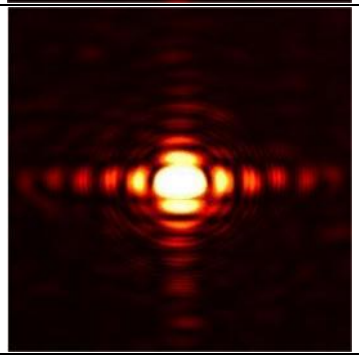
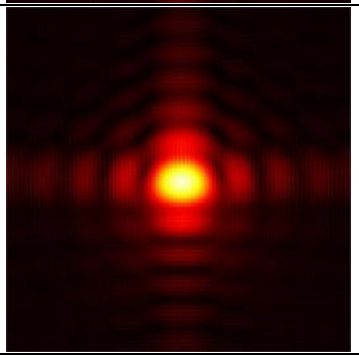
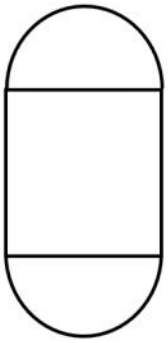
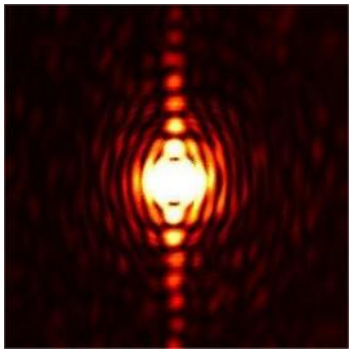
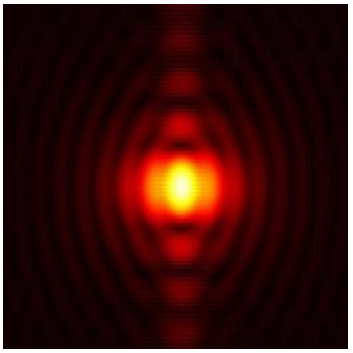
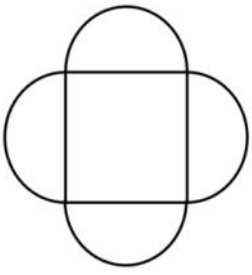
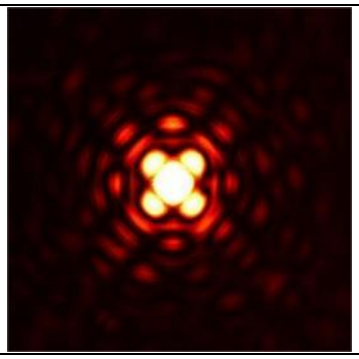
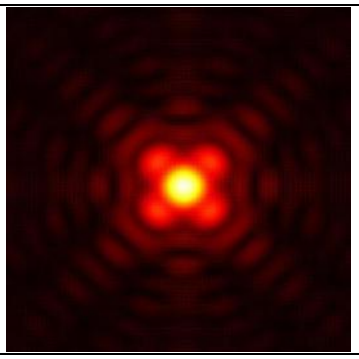
表 4 菲涅尔衍射结果

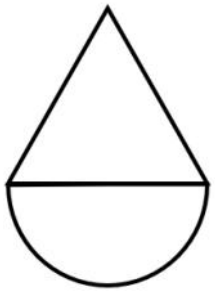
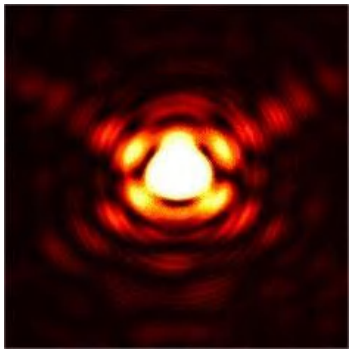
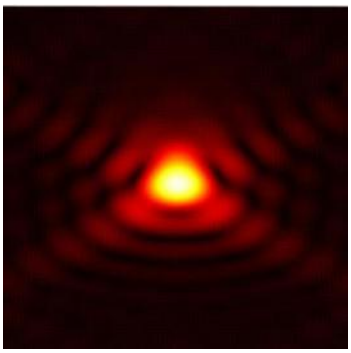
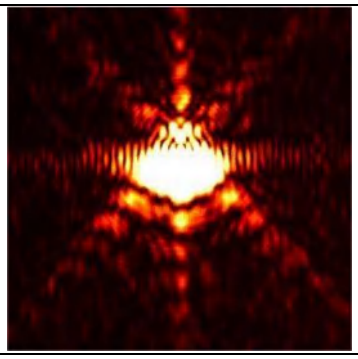
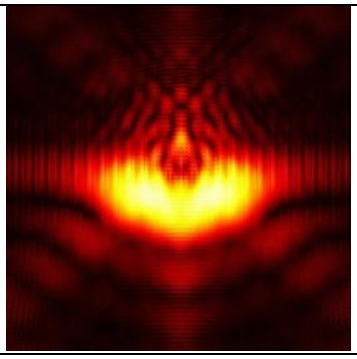
孔模式	实验结果	仿真结果
 边长 500μm		
 直角边长 500μm		

 边长 250μm		
 边长 200μm		
 边长 300μm		
 边长 200μm		
 直角边长 300μm		

 直角边长 300μm		
 直角边长 300μm		
 等边边长 300μm		
 a =250μm		
 边长 300μm		

 边长 300μm		
 边长 300μm		
 边长 200μm		
 外边长 200μm		
 外边长 200μm		

 半径 150μm		
 半径 150μm		
 边长 200μm		
 边长 300μm		
 边长 300μm		

<table>
<tr><td> 
边长 300μm</td><td>  </td><td>  </td></tr>
<tr><td> 
直角边长 300μm</td><td>  </td><td>  </td></tr>
</table>

对比可见，在 150mm 距离的菲涅尔衍射成像也十分正确，其与仿真基本一致，进一步验证了实验结果的准确性，间接验证了夫琅禾费衍射结果的准确性，即验证了三角形分解理论的正确性。

4.3 光栅探究

4.3.1 光栅验证

对于一般的衍射光栅，是在二维平面上由多个夫琅禾费单缝在衍射基础上，引入缝间相位差而产生的干涉调制形成的，其总光强分布不仅仅为每个单缝衍射光强的简单叠加，而是由每个单缝的复光场叠加而成。

设各单缝的宽度为a，缝间距为d，缝数为 N。在经典方法的讨论下，则对于单条缝，平行光正入射于 N 缝表面时，其振幅应为：

$$A_{1\theta} = \frac{A_{10}\sin\dfrac{\pi a\sin\theta}{\lambda}}{\dfrac{\pi a\sin\theta}{\lambda}} \tag{43}$$

则 N 个狭缝沿衍射角θ方向发出 N 束衍射线，在像平面上的P_0点叠加，其复振幅总和$A_{N\theta}$为

$$A_{N\theta} = A_{1\theta}e^{i0} + A_{1\theta}e^{i\delta} + A_{1\theta}e^{i2\delta} + \cdots + A_{1\theta}e^{i(N-1)\delta} \tag{44}$$

其中，δ为相邻缝间对应点的衍射相位差$\delta = \frac{2\pi d}{\lambda}\sin\theta$.

化简整理可得

$$A_{N\theta} = \frac{A_{10}\sin\dfrac{\pi a\sin\theta}{\lambda}}{\dfrac{\pi a\sin\theta}{\lambda}} \cdot \frac{\sin N\beta}{\sin\beta} \tag{45}$$

其中，$\beta = \frac{\pi d}{\lambda}\sin\theta$.而对于夫琅禾费衍射，由于成像屏距离小孔足够远，可以将衍射角θ进行近似，有$\sin\theta \approx \theta \approx \frac{x}{d}$.

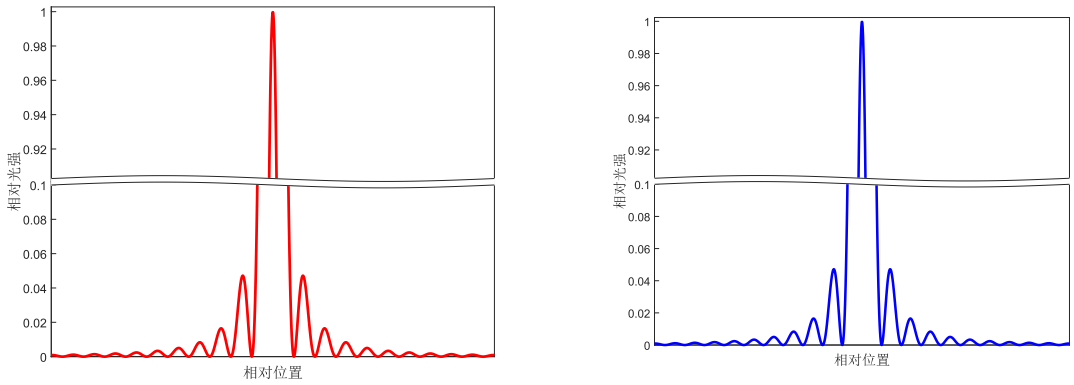
由此，便可以得到衍射屏上光强分布的表达式：

$$I_{N\theta} = I_0 \operatorname{sinc}^2 \frac{\pi a x}{\lambda d} \cdot \frac{\sin^2 \frac{N\pi x}{\lambda}}{\sin^2 \frac{\pi x}{\lambda}} \tag{46}$$

另一方面，考虑用上文耳切法的思路对光栅进行分割，虽其不为单连通，但仍可以分解为多个直角三角形所拼接而成的矩形阵列，从而运用耳切法可以轻而易举地得到观察屏上衍射光场的分布为：

$$E_{\text{total}} = \Sigma_{n=1}^{N}\Sigma_{i=1}^{n_0} E_{ni} \tag{47}$$

其中 n 表示光栅中的第 n 条狭缝，i 表示第 n 条狭缝中分割出的第 i 个衍射三角形元。则由以上两种方法求出的光强分布如图 25 所示：



(a)理论结果　　(b)叠加结果

图 25 经典光栅光强分布

可见，理论与我们的拼凑结果基本一致，验证了三角形拼凑的适用性以及准确性，进而我们在下面讨论缺刻光栅的部分，给与其基础理论的支撑。

4.3.2 缺刻光栅

所谓缺刻光栅，实际上指的是在原本均匀的光栅基础之上，将某些非对称的条纹遮住的光栅，其能够产生不均匀的光场分布以及改变主极强的位置，其一般可以应用于光栅或类光栅物质的形态结构研究，或者用于特殊光谱的鉴别等[22]。

我们在前文多缝夫琅禾费衍射的基础上，再考虑去掉某一或某些狭缝对衍射图样的影响。相比于经典的方法，用三角形叠加来讨论光栅的缺刻是非常简单且直观的，下面将以$N = 21$为例展示部分计算结果，以中间的一根为$n = 0$，向左为负，向右为正。

(1)当缺n =-5 时，其衍射图和水平方向光强分布图为

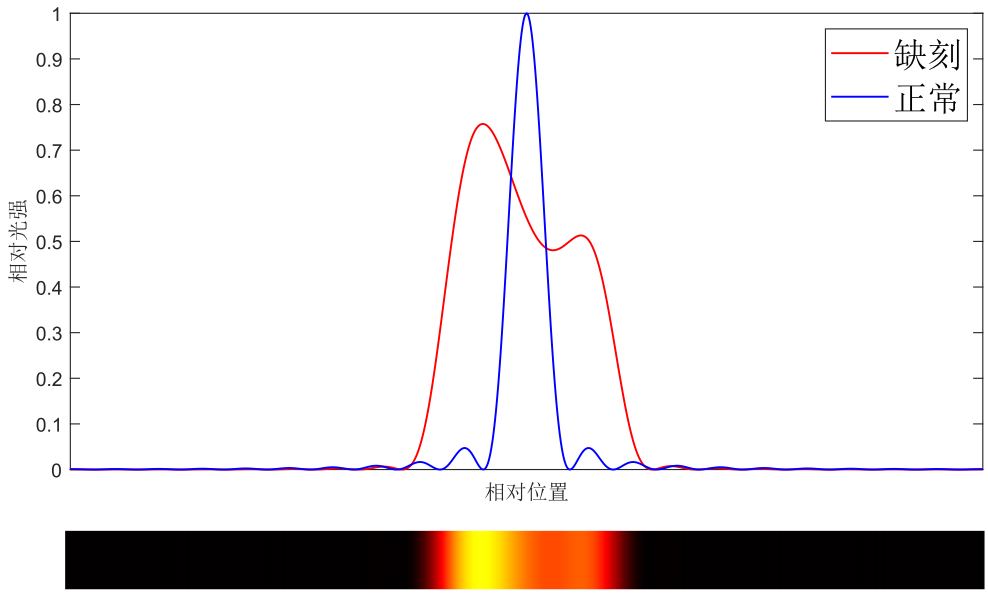


图 26 缺刻 n=-5 时的缺刻光栅光强分布

(2)当缺n =3、4、5 时，其衍射图和水平方向光强分布图为

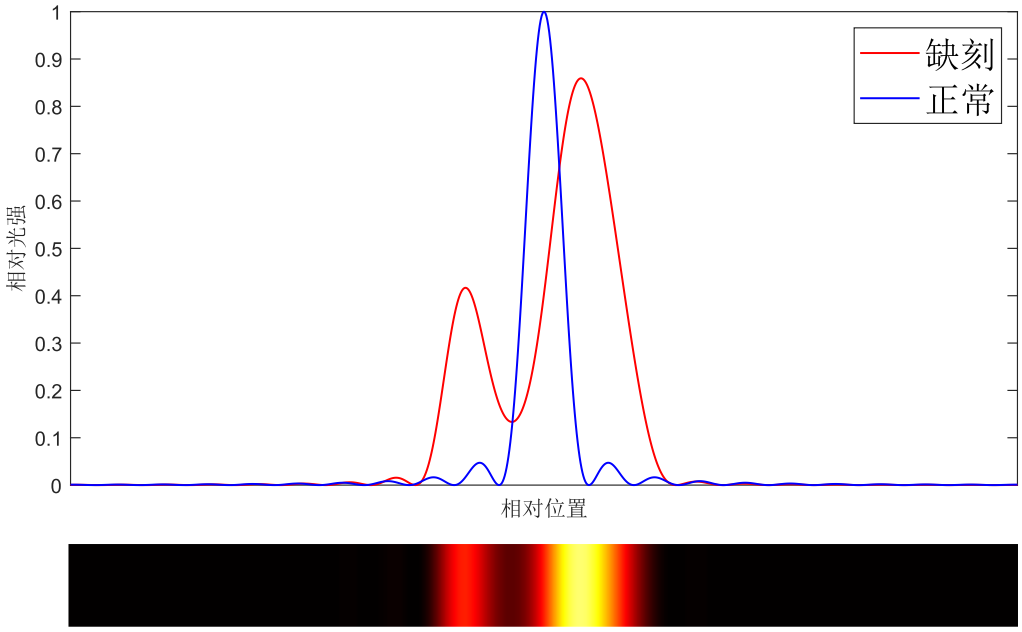


图 27 缺刻 n=3、4、5 时的缺刻光栅光强分布

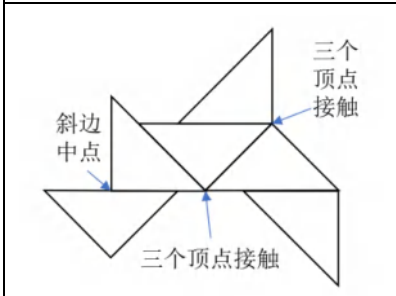
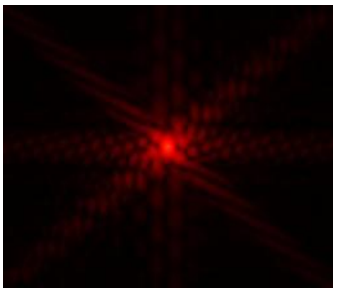
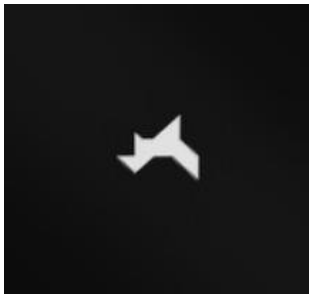
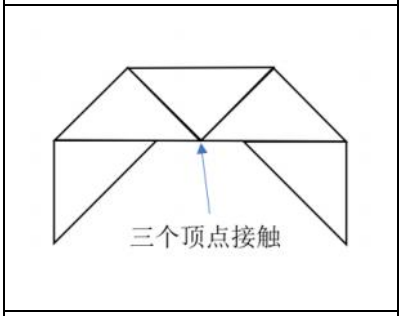
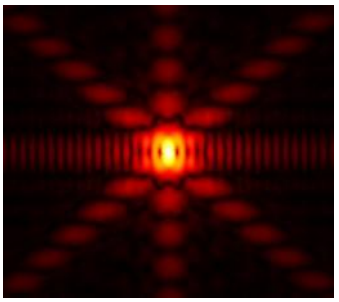
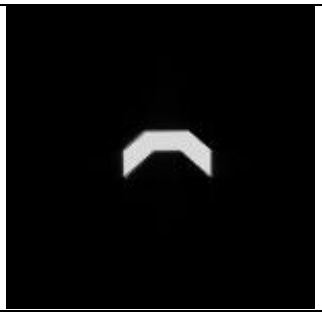
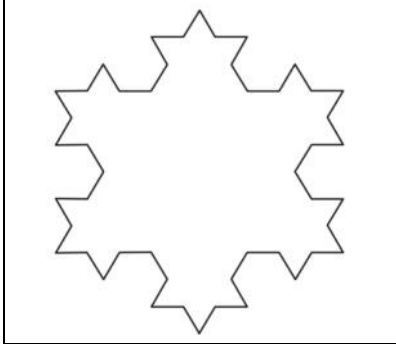
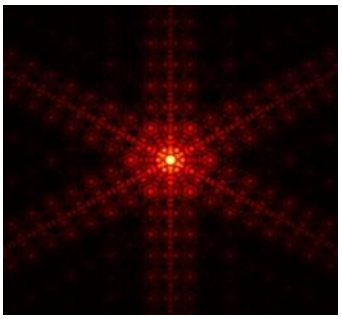
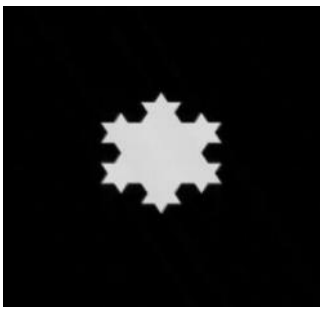
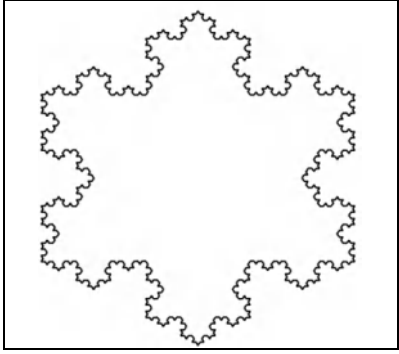
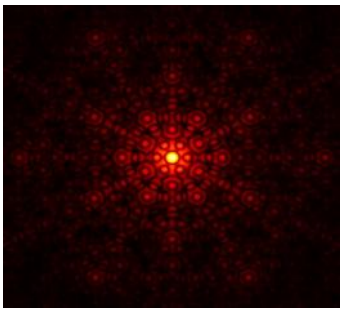
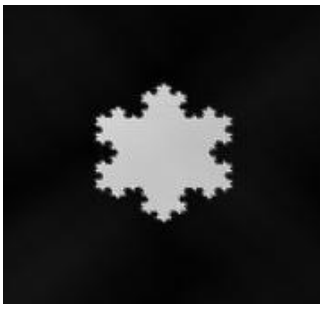
一般地可以发现，衍射条纹移动的方向总是与缺刻的光栅所在位置相反，并且由于缺刻，其中央极强的线(角)宽度也增大了，从而在相同光强入射的情况下，其峰值光强即会降低，从图中即可直观地看出。

由于篇幅关系，我们在此仅仅研究了几种情形，实际上可以研究更多的缺刻类型从而总结出或推导出更具有一般化的光强分布规律。

4. 4 反演验证

为验证解析解的正确性，我们选取复杂孔与分型孔的理论计算数值进行反演运算，运算结果如下：

表 5 反演结果

孔模式	理论衍射结果	仿真反演结果
		
		
		
		

可以看到，不论是三角形拼接的复杂孔模式，还是具有无限可分性质的分型孔模式，仿真反演得到的孔径结果与理想孔模式高度重合，反演验证较为成功，分割方式有效且精确。

5. 数据分析及不确定度分析

我们的不确定度分析分析均基于 JJF1059.1-2012《测量不确定度评定与表示》，且忽略自由度的影响，即认为自由度为无穷。

5.1 光强分布曲线

5.1.1 分布曲线结果

对于正方形衍射图案，由于其具有较高的对称性以及衍射级次，我们对边长为

300μm的正方形的夫琅禾费衍射图样横轴(水平轴)读取光强分布，其原始图如图 28 所示。

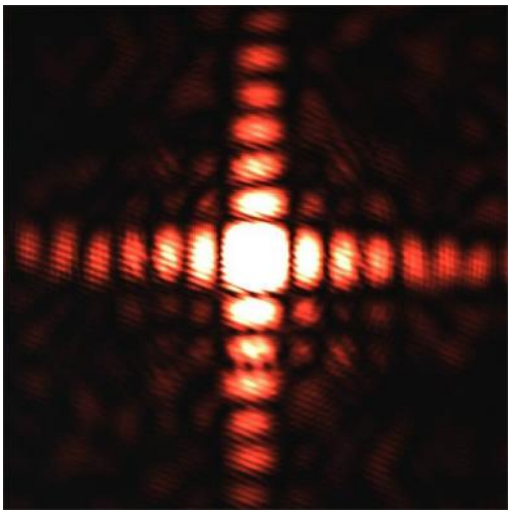


图 28 边长为 300μm的正方形夫琅禾费衍射图样

然后我们将理论值与其作在同一张图内，将实验值归一化，并调节理论值倍数直至其边缘级次衍射强度与实验基本一致，其效果如图 29 所示。

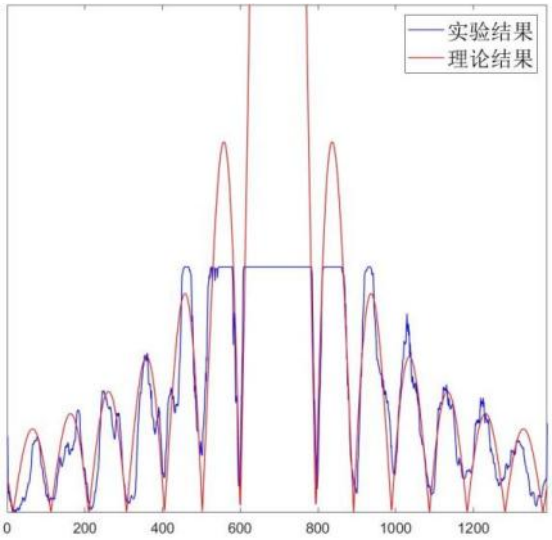


图 29 正方形夫琅禾费衍射水平轴光强分布图(边长为 300μm)

我们利用实验光强未过曝的部分与理论值相减，并利用实验的最大值对其进行归一化处理，得到相对光强差值大小的平均值，其结果为

$$e_{ave} = 0.092 \tag{45}$$

5.1.2 不确定度分析

①A 类不确定度：

对于 A 类，我们算差值大小的平均值的标准差，根据贝塞尔函数法，利用贝塞尔公式得到残差大小的标准差，然后除以其自由度的算数平方根即可得到其平均值的标准差，其公式及结果为

$$u_A = S_{\bar{e}} = \sqrt{\frac{\sum_{i=1}^{n}(e_i - e_{ave})^2}{n(n-1)}} = 0.0021 \tag{46}$$

我们认为其为光强差值大小的 A 类不确定度。

②B 类不确定度：

对于 B 类不确定度，我们认为误差满足均匀分布，由于 CCD 的位深度为 8，其可以储存 256 种颜色，因而 CCD 的光强分辨率为

$$\Delta = \frac{1}{255} \tag{47}$$

故 B 类不确定度为

$$u_B = \frac{\Delta}{\sqrt{3}} = 0.002264 \tag{48}$$

③合成不确定度：

根据国标公式，实验光强与理论光强差值的合成不确定度为

$$u_c = \sqrt{u_A^2 + u_B^2} = 0.0031 \tag{49}$$

④扩展不确定度：

我们认为不确定度的包含概率$p = 95\%$，在不考虑自由度时，认为包含因子$k = 2$，则扩展不确定度为

$$U = ku_c = 0.007 \tag{50}$$

不确定度分析结果：

则正方形夫琅禾费衍射水平轴的实验光强分布与理论的差值不确定度分析结果为

$$e = 0.092 \pm 0.007 \tag{51}$$

注：此光强已经用实验最大值进行归一化。

5.2 正多边形角分布

对于正多边形，其具有十分优良的对称性，故我们考虑利用其周边环绕的星芒(高强度分布)的间隔角度为数值，对其进行不确定度分析：

首先我们利用 MATLAB 读取光强后，读取中心点，并对各个星芒利用直线标定，得到其各线之间的夹角，其结果如表 6、表 7 所示。

表 6 正多边形夫琅禾费衍射角分布情况

孔形状	边长/um	θ_1/°	θ_2/°	θ_3/°	θ_4/°	θ_5/°	θ_6/°	θ_7/°	θ_8/°
八边形	250	40.42	46.49	46.71	46.49	40.52	45.97	46.79	46.61
	300	44.61	43.38	45.16	46.34	43.67	45.49	45.51	45.84
六边形	200	61.64	52.83	58.32	62.81	58.36	66.04		
	250	55.94	60.65	61.88	57.31	61.16	63.06		
	300	53.57	63.92	62.54	54.68	63.08	62.21		
三角形	500	56.01	68.83	56.23	55.14	69.04	54.75		
	550	57.50	65.80	55.86	58.31	65.65	56.88		
正方形	250	90.23	90.23	89.27	90.27				
	300	89.56	90.00	89.55	90.89				

表 7 正多边形菲涅尔衍射角分布情况

孔形状	边长/um	θ_1/°	θ_2/°	θ_3/°	θ_4/°	θ_5/°	θ_6/°	θ_7/°	θ_8/°
八边形	250	40.98	45.66	49.92	45.04	40.28	48.84	44.07	45.21
	300	42.73	44.95	48.71	44.73	41.91	46.76	45.76	44.45
六边形	200	55.45	63.85	62.41	55.31	61.21	61.77		
	250	54.71	63.80	62.63	55.30	61.95	61.61		
	300	56.15	62.05	62.19	56.49	61.44	61.68		
三角形	500	59.49	66.83	56.53	56.57	63.54	57.04		
	550	58.41	66.4	57.05	55.94	65.18	57.02		
正方形	250	90.48	89.41	90.00	90.11				
	300	91.17	90.1	89.52	89.21				

我们对其各个角求平均值，由于其相当于将 360 度等分，故其平均值均为

$$\theta_{ave} = \frac{360°}{n} \tag{52}$$

其中 n 为分割个数，然后我们对其进行不确定度分析。

① A 类不确定度分析：

对于 A 类不确定度，我们同样利用贝塞尔公式，取其平均值的标准差作为 A 类不确定度，公式与之前相同。

② B 类不确定度分析：

对于 B 类不确定度，我们要考虑的是仪器的最小角分辨率，对于一个图，由于其是像素点，其效果图如图 30 所示。

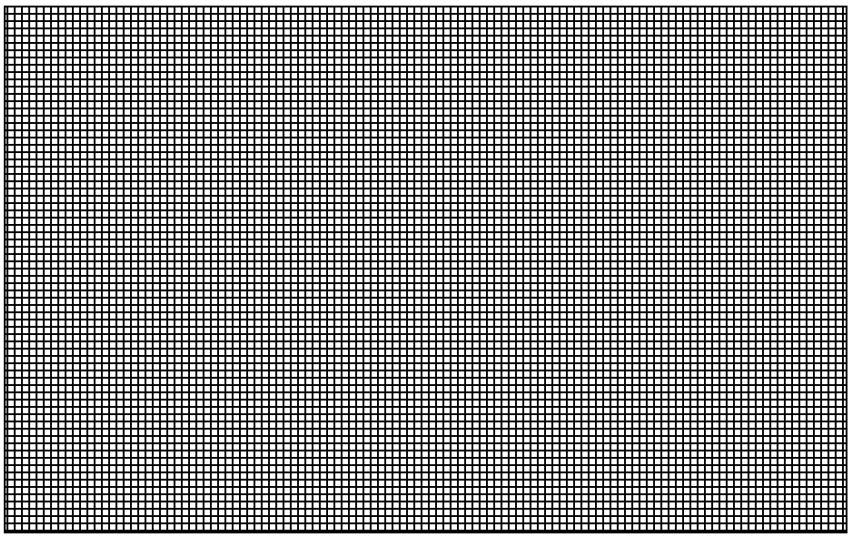


图 30 图片像素点示意图

对于此图，当我们作一条线去重合极强位置时，其最小分辨角的两条边线的定位方法即为边缘两点与图片中心连线之间的夹角，故此时两线夹角即为可分辨角，而对于边缘位置，利用三角形的关系我们可知，在假设两像素点的间隔相同的情况下(图片像素点十分密集)，当某方向线段长度最小时，其分辨角最大，故我们取其作为可分辨角的最大大小，并且认为误差满足均匀分布，其较为直观的图如图 31 所示。

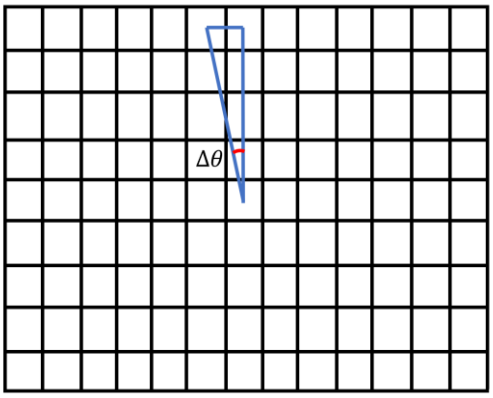


图 31 最小分辨角度示意图

首先，由于 CCD 读取后的图片为矩形，则最短线段为竖直方向像素点数的一半，而 CCD 的分辨率为 2048×1536，而我们知道，图像像素点个数远大于 1，故我们可以近似将图 31 中的三角形近似为等腰三角形，则利用余弦公式，我们可以轻易得到

$$\cos\Delta\theta = \frac{768^2 + 768^2 - 1}{2 \times 786 \times 786} \tag{53}$$

则有

$$\Delta\theta = 0.073° \tag{54}$$

即为其最小分辨角，则其 B 类不确定度为

$$u_B = \frac{\Delta\theta}{\sqrt{3}} = 0.42° \tag{55}$$

③ 合成不确定度

根据国标公式，其分辨角的合成不确定度为

$$u_c = \sqrt{u_A^2 + u_B^2} \tag{56}$$

④ 扩展不确定度：

我们认为不确定度的包含概率$p = 95\%$，在不考虑自由度时，认为包含因子$k = 2$，则扩展不确定度为

$$U = ku_c \tag{57}$$

则各个正多边形的角分布结果及各种不确定度为

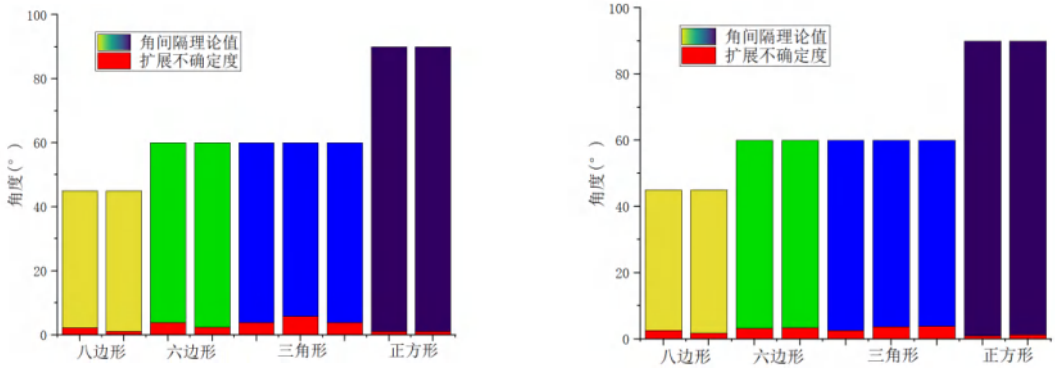
表 8 正多边形夫琅禾费衍射角分布不确定度分析结果

孔形状	边长/um	θ_{ave}/°	A 类不确定度/°	B 类不确定度/°	合成不确定度/°	扩展不确定度/°
八边形	250	45.00	0.99	0.42	1.08	2.2
	300	45.00	0.37	0.42	0.56	1.1
六边形	200	60.00	1.86	0.42	1.91	3.8
	250	60.00	1.13	0.42	1.21	2.4
	300	60.00	1.88	0.42	1.92	3.8
三角形	500	60.00	2.83	0.42	2.87	5.7
	550	60.00	1.84	0.42	1.89	3.8
正方形	250	90.00	0.24	0.42	0.49	1.0
	300	90.00	0.31	0.42	0.52	1.0

表 9 正多边形菲涅尔衍射角分布不确定度分析结果

孔形状	边长/um	θ_{ave}/°	A 类不确定度/°	B 类不确定度/°	合成不确定度/°	扩展不确定度/°
八边形	250	45.00	0.99	0.42	1.26	2.5
	300	45.00	0.37	0.42	0.87	1.7
六边形	200	60.00	1.86	0.42	1.56	3.1
	250	60.00	1.13	0.42	1.66	3.3
	300	60.00	1.88	0.42	1.24	2.5
三角形	500	60.00	2.83	0.42	1.80	3.6
	550	60.00	1.84	0.42	1.91	3.8
正方形	250	90.00	0.24	0.42	0.47	0.9
	300	90.00	0.31	0.42	0.60	1.2

可见，其不确定度较小，实验效果较好。我们将其理论值与扩展不确定度合并画图，以展示出其相对大小，实验结果如图 32 所示



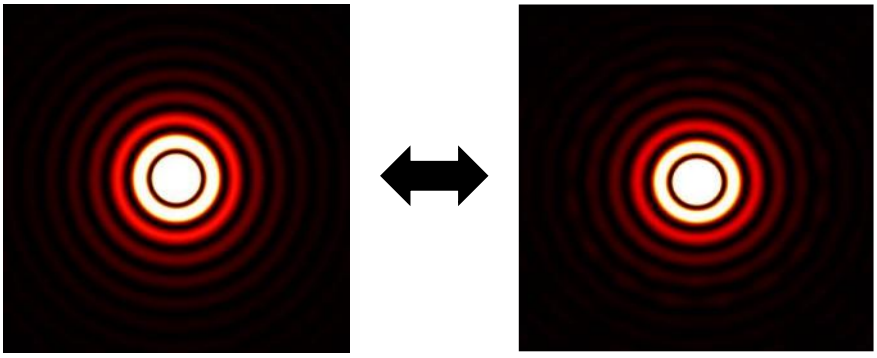
(a)夫琅禾费衍射　　(b)菲涅尔衍射

图 32 星芒角分布不确定度

5.3 圆弧的近似精度

对于我们的近似方法，需给出验证的方案，在此考虑对圆孔进行操作，我们知道，对于圆孔衍射，其具有十分明显的环状结构，故考虑对圆弧的环形光强极小值点进行读取，并将理论结果与三角形叠加结果进行比较。

首先给出两者的作图结果，其如图 33 所示



(a)理论结果　　(b)叠加结果

图 33 圆孔夫琅禾费衍射对比图

经过比较发现，我们的近似在低级级次的结果是十分可观的，圆环状十分精确直观，但在高级级次时，其出现了非圆弧的衍射斑环，即为若干细小三角形衍射级次叠加的效果。我们看到，第六级衍射斑环出现了块状，其一环的块状个数即为 16 个，而我们利用圆弧近似的方法，通过迭代三次，得到的三角形分割方式如图 34 所示

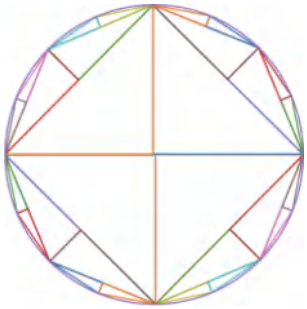


图 34 圆孔分割模式示意图

可见，其外部即为 16 个直角三角形，对于之前的理论，其即斜边会产生 12 道星芒结构，从而即产生第六级次衍射的 16 个小块状结构，低级级次之所以并未产生的原因是因为大三角形等将其掩盖了，故我们的近似在五级之内能够满足近似条件，从而我们分别选取理论结果与拼凑结果的前五级极小值点，对其进行拟合操作，从而判断其精度。分别对理论结果和拼凑结果任意选取一个大小的孔读取的前五级近似结果为

表 10 圆孔零点位置分布结果

级次	理论零点	拼凑零点
1	3.81	5.91
2	7.04	10.70
3	10.22	15.53
4	13.34	20.34
5	16.47	25.35

我们知道，对不同大小的孔其最多成比例缩放，利用 Origin 进行拟合分析，其结果如图 35 所示。

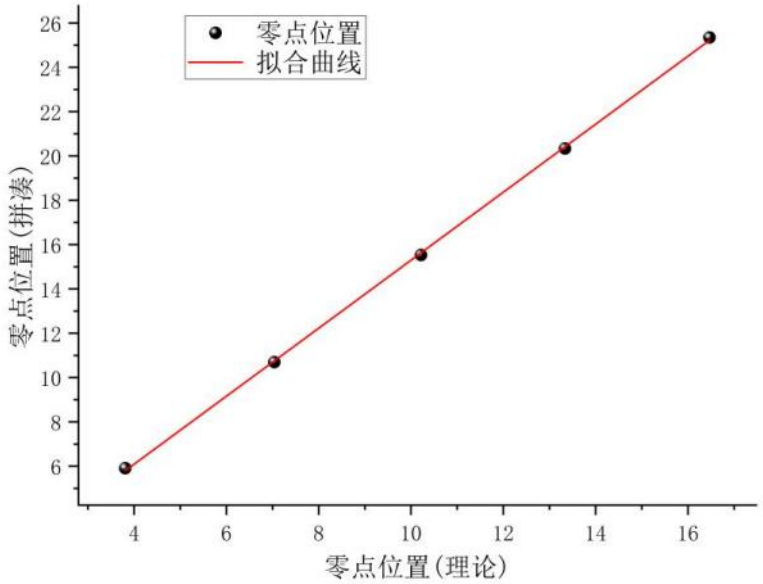


图 35 圆孔零点拟合结果

其拟合评分 R-Square=0.99979，皮尔逊相关系数γ=0.9999，可见我们的近似方法十分有效，且这只是三次迭代的结果，倘若我们取更高级次的迭代，其几乎能达到与理论一样的效果。

5.4 误差来源分析

本次实验中可能的实验误差来源有(系统误差见 3.4)：

• 人为读数的误差。在实验中，由于我们许多步骤需要用到人为操作读取数据，故难免存在实验误差，但这是无法避免的，实验过程中已经保证操作正确，减小人为误差。

• 夫琅禾费衍射中，由于人为调节以及透镜焦距的测量存在误差，透镜和孔的距离难以精确保证为焦距，故入射光不一定完全平行，但实验中已经细致调整，误差较小。

6. 实验创新点

本次研究中，在多方面具备创新性，其具体阐述如下：

(1)研究思路创新

一般的复杂结构难以得到干涉衍射解析解，在以往的研究中，对于复杂结构的衍射理论研究十分局限，且对复杂结构的对称性要求极高，而本实验开创性地使用直角三角孔分解叠加的模式，能够简易、迅速地给出复杂结构的干涉衍射解析解，对单连通、多连通的多边形以及圆弧均适用，大大扩展了适用范围，对光学复杂结构的衍射解析解导出具有重要意义。并且在三维立体空间中，传统的 FFT 是无法直接得到三维物体的衍射结果的，而利用三角形有限元分割以及坐标变换即可十分便捷的得到衍射光场分布。

(2)研究方法创新

耳切法在多边形图像分割处理中有着许多应用，我们开创性地将图形处理的方法应用至光学衍射方向，利用 Python 编程实现了复杂结构的机器视觉边缘检测以及耳切法自动化分解，得到最优化、模式化的分解模式，并且分解后通过一般的直角三角形衍射光场叠加的方法，即可快速地利用 Mathematica 编程求解出解析解，再用 MATLAB 进行理论作图，可以得到与实验近乎一致的效果，将物理与图形处理的方法结合，研究方法十分新颖。

(3)数据处理方法创新

数据处理是实验中十分重要的部分，利用正多边形衍射的星芒结构，我们提取图形特征，研究了星芒之间的夹角角分布，开拓了数据分析方向。并且在角分布的 B 类不确定度分析中，为得出图片的角分辨能力，我们开创性地使用像素点的分析模式，利用从中点发散直线的方式，得出了图片的分辨角，得到了与 A 类不确定度基本相当的 B 类不确定度，可见方法合理且具备创新性。以及研究了圆弧近似所丢失的精度，证实了圆弧近似的有效性。

(4)多编程语言结合

在研究中，我们多重化编程语言，利用各个语言的优势，如 Python 的易编程性，Mathematica 对理论求解的精确性，MATLAB 对矩阵处理的优异性，将各种语言相互结合，即使实现研究要在多个软件中交互运行，但我们已经做好模块对接，能够直接进行交互，并且这对获取精确的理论和模拟结果是十分重要的，具有一定的创新意义。

7. 实验评估与展望

7.1 研究方法优势

对于我们的研究，其具有多方面优势，具体如下：

• 开创了图形处理与光学衍射结合的研究方式。利用耳切法分解的方式通过三角形叠加即可十分便捷地得出直观的复杂结构解析解，对于具有较小对称性以及较难给出解析解的多边形复杂结构我们都可以分解求解。

• 提升空间大，能够在三维中体现出不同于 FFT 变换的优势，实现更快捷、精确的光场导出。

• 可适用范围广，在缺刻光栅、三维立体图形衍射等多方面能得出精确的结果，研究推广性强。

• 多语言结合。研究中使用多模块化、多语言化的方式，利用各种编程语言的特色，配合实现更加精确的研究结果。

• 孔设计及加工费用较少，为我们后期的大范围研究打下了坚实的经济基础，且加工周期短，利于研究。

• 实验装置经典，易于复现。

7.2 研究限制

7.2.1 机器视觉的限制

对于机器视觉的边缘检测算法，由于我们需要寻找边缘的是一个图片，其分辨率有限，故此时无法得到一个十分精确的点，其顶点一定是整数，故对于一些无理数以及分数，其均会近似到整数上面，但我们图形的分辨率较高，基本不会对实验结果造成影响，这也是在提升研究速率时无法避免的。

边缘检测算法对参数敏感性很强，同一组参数不能应用于所有的复杂结构，必须通过调节自动二值化参数以及边缘近似参数等才能更好地适应，后面我们也会继续探究自适应的算法，以得到更加优异的结果。

7.2.2 耳切法的限制

耳切法作为一种重要的多边形分割方法，尽管在许多情况下表现出色，但也有其局限性。首先，它无法处理多连通多边形，无法准确确定多连通多边形中的"耳朵"位置。其次，耳切法的计算复杂度较高，对于大多数实现而言，其时间复杂度为$O(n^2)$，这意味着对于非常庞大的多边形而言，可能需要较长的计算时间。另外，耳切法可能会产生过多的三角形，对一些应用可能不利。此外，耳切法的三角剖分结果可能并非最优，某些应用可能希望最小化生成的三角形数量或最大化最小角度等其他方面的优化。因此，耳切法无法保证此类最优性。

对于复杂多边形或对精细度要求较高的应用，我们可能需要考虑其他更高级的算法。这些算法通过考虑更多的几何特性、优化目标或利用先进的数据结构，能够产生更优的分割结果。在实际应用中，我们应根据具体情况选择适合的算法，以获得最佳的效果。

总体而言，耳切法虽然是一种有效且直观的算法，但并非适用于所有情况。在一些复杂或高性能的应用中，可能需要使用更复杂、更精细的算法来解决问题。

7.2.3 理论与仿真计算的限制

由于理论与仿真计算中一些不可避免的数值问题，我们的计算有一定的限制与近

似，但它们并不会对我们的结果有很大影响。

首先是关于夫琅禾费衍射的近似。实验中我们是使用了透镜来制造小孔与成像平面之间的无穷远间距，但是在实验与仿真的计算中，我们并无法取一个无穷大的数字来进行数值模拟，如式(17)所示，解析式中总会出现距离因子d。因此我们在计算中选取了一个较大的d来保证夫琅禾费近似的正确性。经过与实验比对之后，事实证明我们的近似是合理的，计算所得的结果能与实验较好地吻合。

其次是理论计算中的过曝问题。在实验中，由于 CCD 机器本身的限制，光斑中心的位置光强过大，在其拍摄的图中，中心亮斑附近亮度都是一致的，导致我们无法读取到中心亮斑附近的更多信息，它们在 MATLAB 中读取的灰度数值也都是一致的，这一点从前面的图 29 也可以看出来。因此为了实现对实验结果的还原，并且体现出更多衍射图样的细节部分，我们在理论计算作图的过程中也对光强进行了一定的放大，使得中心部分图样也有一定的过曝，显得亮度一致。这样可以让我们理论的计算图样与实验更相似，并且细节也更清楚，并且这并不会影响我们后续的数值计算。

7.3 精度指标

本实验精度指标的限制主要为仪器测量限制。

• CCD 的分辨率为 2048*1536，采集范围大小为 3.20μm*3.20μm，由于仪器设计极限，其无法捕捉到完全平滑的实验光强，光强分布是一个矩阵形式，最小可分辨大小为 1.56nm*2.08nm，即为我们分辨图像的精度指标。

• CCD 具有八位的像素深度(BPP)，故光强分辨率为 1/255，即为我们采集光强的精度指标，我们能够分辨成像最亮的光强的 1/255，但由于过曝会导致光强为 1，故分辨率即为过曝部分光强的 1/255。

7.4 实验展望

对于我们的实验，在有限的时间内我们完成了基础研究，其研究有许多可待提升的部分，研究前景广，具体如下：

(1) 加入三维物体衍射。对于三维物体，用有限元分割后的图形可以变为一个个的三角形，利用我们所推导的理论再加上一定的坐标变换的方法，即可快速地导出光场分布，而这是 FFT 无法做到的，更能体现出我们的研究价值与优势，研究推广性强。

(2) 加入非圆弧的弧线图形。我们对于孔的分割部分，由于我们使用直角三角形分割，对于一般的棱角型区域，其不包含弧线，必定能分割成单连通区域后利用耳切法进行分割，但此方法不能处理弧形图案；我们目前已经能够实现圆弧的近似，之后我们将考虑加入非圆弧，探究更加一般的近似的方法，在失去一定精度的情况下将其利用直角三角形分割，可以扩大系统的适用范围。

(3) 提升机器视觉的自动寻参。边缘检测算法对参数依赖性高，我们需要根据不同的图形选取不同的参数，从而获得正确的图形顶点，之后我们将探索参数的自动化调节，便于更好地展开探究。

(4) 编程语言进一步统一。在本次研究中，我们利用 Python 语言实现耳切法分解，Mathematica 实现理论值求解，MATLAB 用于作图以及仿真，语言较多，需要多软件配合，虽然各语言的交互模块已经基本完善，但仍然可以考虑进一步统一语言，提升理论作图效率。但由于不同语言有其各自的特性，故我们在切换语言时也应当考虑切换方法以利用某种语言的优势，得到更好的结果。

7.5 应用前景

对于我们的研究，其有许多扩展的应用前景，具体如下：

• 对于复杂结构的干涉与衍射问题，利用现有的仿真软件难以得到解析解，且结果不具有通用性。我们利用直角三角形分割复杂结构，再利用直角三角形干涉衍射叠加的效果可以直接导出复杂结构的解析解。在本次研究中，我们验证了这种方法的可行性、通用性与正确性，且这种方法更加便捷且快速，使用起来更加方便快捷。因此，我们可以利用这种思路制作仿真软件，从而实现理论解的一般性导出。

• 三维物体衍射全息成像领域，本研究在三维结构中相比于传统的 FFT 具备更好的优势，可以快速地导出解析解，并记录相位，从而可以光学再现，实现全息成像，其结果精确、导出迅速，有着较高的应用价值。

• 调节缺刻光栅，可以将我们的三角形分割应用至光栅领域，通过快速调节拼凑结果，能够便捷地实现缺刻，从而调节其光强分布，获得不同位置的主极强。不仅仅在缺刻光栅领域，我们也能在其余的复杂结构衍射领域快速地适用。

• 在光学理论求解领域，本次研究给出了复杂结构衍射的解析解求解方法，其一般性结果有利于一些光学问题的结果分析。

• 精密测量领域中，可以通过衍射光场反推孔的形状，测量复杂结构的边长、角度等参数，实现低成本的精密测量功能。基于此精细化的数据测量可以对复杂结构零件表面进行观察，以此保障零件的质量和精细要求。

8. 总结

在本实验中，我们的总体思路是利用三角形分割的方式给出复杂结构的夫琅禾费衍射理论结果。

首先，我们设计了一系列的图形，其中有基础三角形、正多边形(偶数部分)、星形、环状、弧形以及复杂的拼接结构等的一些复杂结构图形，部分具有较好的对称性和美观性，可以给出十分优异的干涉衍射图案，且有些图形不具有很强的对称性，可以用于验证我们理论的准确性以及可行性，并且我们利用 Python 编程，利用机器视觉边缘检测算法结合耳切法实现了复杂多边形的分割，并结合弧形近似方法，将复杂结构其分割成若干个直角三角形。

然后，我们给出了夫琅禾费衍射的理论推导，以及验证了叠加原理，证明了三角形分割的可行性，进行了夫琅禾费衍射实验，同时利用理论所得的解析解和仿真与其对比，实验效果较明显，再次证明了理论结果的准确性。同时作了各种孔的菲涅尔衍射图案，由于解析解不够直观，故我们作了菲涅尔衍射部分的仿真与其对比，其效果也基本一致，可见实验条件较好，进一步证明了夫琅禾费衍射的实验准确性，间接验证了夫琅禾费分解理论的正确性。

接着，我们进行了多种探究及验证。首先探究了缺刻光栅，其能够实现主极强位置的调控，验证了方法的适用性；接着作了反演变换，利用数值的 iFFT 方法反演变换理论值，从而得到孔结构，其与孔完全一致，从而直观地验证了理论的精确性。

并且我们讨论了研究的优越性，其在三维物体的衍射部分能够体现出优越于 FFT 的优势，从而能够实现三维物体光场的全息再现，在全息成像方面有着较高的应用价值，之后我们也将考虑将其加入并进行一些初步的探究。

最后，我们作了实验的误差分析部分，专注于夫琅禾费衍射，首先做了正方形的光强分布曲线，利用实验和理论的差值做了不确定度分析，在实验光强归一化的情况

下，得到实验与理论光强差大小的平均值为 0.092，在包含概率为 95%的条件下，其扩展不确定度为 0.007；然后做了正多边形的角分布不确定度分析，也同样取包含概率为 95%，扩展不确定度结果也十分可观，扩展不确定度的角度均在 1°-3°之间，实验较为精确；最后我们探究了弧形近似所丢失的精度，拟合理论以及拼凑结果的光强极小值点，R-Square=0.99979，线性度极高，在三次迭代近似下即可得到十分精确的结果。

总的来说，我们开创性的利用 Python 编程实现的边缘检测算法以及耳切法使用直角三角形分割复杂图形，从而利用 Mathematica 编程计算给出复杂结构的夫琅禾费衍射解析解，对解析解反演变换验证了准确性，并且我们也同时研究了这些复杂结构的衍射实验、仿真结果，给出了误差分析和不确定度分析，探究了缺刻光栅，讨论了处理三维物体的优越性，实验具备合理性、准确性、创新性和完整性，研究较为彻底。

并且我们的研究不但具备开创性，还具备较好的应用性，首先可以利用此思路进行三维物体衍射光场分布的导出，从而在全息成像方向进行研究；然后是仿真软件的开发，能够快速导出一个复杂结构的衍射解析解，对衍射的数学形式研究具有很强的应用性；并且利用复杂结构衍射图样的特征长度等我们可以对某些尺寸在 um 量级的孔径进行精密测量，在已知结构轮廓的情况下可以用于精密测量方向，兼具性价比；最后是在许多衍射结构研究比如缺刻光栅的研究上有着较强的应用性，用三角形分割可以快速以及精确地导出光场分布及特性，加快研究进度，对解析解规律性研究有着较大的优势性。

参考文献

[1]谢子康.光的干涉与衍射的理解与解析[J].科学技术创新,2019(07):47-48.
[2]陈子阳,陈丽,范伟如等.基于相关全息原理的散射成像技术及其进展[J].激光与光电子学进展,2021,58(02):9-22.
[3]孙庆莉.惠更斯—菲涅尔原理在克希霍夫积分方法中的应用[J].西部探矿工程,2007(06):52-54+59.
[4]邓小玖,李国祥.电子衍射、干涉的量子理论[J].量子电子学报,1999(02):190-192.
[5]曾阳素. 分数傅里叶光学的应用研究[D].四川大学,2003.
[6]李杨. 光学合成孔径成像系统的共相探测技术研究[D].中国科学院大学(中国科学院光电技术研究所),2017.
[7]谷怀民,杨思华,向良忠.光声成像及其在生物医学中的应用[J].生物化学与生物物理进展,2006(05):431-437.
[8]齐晓庆,高春清,刘义东.利用相位型衍射光栅生成能量按比例分布的多个螺旋光束的研究[J].物理学报,2010,59(01):264-270.
[9]辛秀,咸夫正,孙尚倩等.基于夫琅禾费衍射的星芒现象探究[J].大学物理实验,2021,34(01):39-43.
[10]宋易知.任意正多边形小孔夫琅禾费衍射成像探讨[J].物理实验,2017,37(11):48-51.
[11]段瑞玲,李庆祥,李玉和.图像边缘检测方法研究综述[J].光学技术,2005(03):415-419.
[12]羊国光,宋菲君.高等物理光学 第 2 版[M].中国科学技术大学出版社,2008.
[13]陈敏雅,金旭东.浅谈计算机图形学与图形图像处理技术[J].长春理工大学学报,2011,6(01):138-139+146.
[14]王改莲,吴翠微,董建新等.计算机图形处理软件在 SEM 图像定量测定中的应用[J].电子显微学报,2001(04):279-282.
[15]方芳.新媒体时代计算机图形图像处理技术在传媒中的应用[J].电子技术与软件工程,2018(24):127-128.
[16]徐天乐,郭华章,张景波等.微机交互式图形处理系统及其在胚胎学 CAI 中的应用[J].解剖学杂志,1991(02):179-181.
[17]奥坎 K.埃尔索伊,埃尔索伊,蒋晓瑜,等.衍射、傅里叶光学及成像[M].机械工业出版社,2016.
[18]杨坤,成泰民.不同三角形孔和矩形孔构成的双孔 Fraunhofer 衍射图样模拟[J].曲阜师范大学学报：自然科学版, 2012, 38(2):5.DOI:10.3969/j.issn.1001-5337.2012.02.013.
[19]张韵华,王新茂.Mathematica 7 实用教程.第 2 版[M].中国科学技术大学出版社,2014.
[20]古德曼,J. W.).傅里叶光学导论[M].科学出版社,1976.
[21]Eberly D. Triangulation by ear clipping[J]. Geometric Tools, 2008: 2002-2005.
[22]刘云啸,田瑞琳,王立豪.缺刻光栅:一种特殊不均匀光栅的研究[J].科技创新导报,2013(20):2-5.DOI:10.16660/j.cnki.1674-098x.2013.20.005.

附录

A. 算法部分

功能：边缘检测及耳切法多边形分割

语言：Python

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
import math
from shapely.geometry import LinearRing
import random
from shapely.geometry import Polygon, LineString ,Point
import openpyxl as xl
import xlwings as xw
import os

os.chdir("E:\\大学学科竞赛\\物理实验竞赛")# 选取文件路径
# 打开需要处理的图片
img = cv2.imread('image2.jpg')
# 把彩色图片转换为灰度图片
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
# 使用自适应阈值法进行二值化
binary = cv2.adaptiveThreshold(gray, 255, cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY_INV, 201, 15)
# 在二值化图片上找出所有轮廓
contours, _ = cv2.findContours(binary, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
# 选取第一个轮廓，使用 cv2.arcLength 和 cv2.approxPolyDP 来简化该轮廓
cnt = contours[0]
epsilon = 0.01 * cv2.arcLength(cnt, True)#简化参数需要根据图形调节
approx = cv2.approxPolyDP(cnt, epsilon, True)
# approx 变量中存储了简化轮廓的所有顶点坐标
vertices = approx.squeeze()
# 输出所有的顶点坐标
print(vertices)
# 使用 matplotlib 将图像，轮廓的边缘，和顶点一起绘制出来
plt.imshow(img)
# 使用绿色的点来标记所有的顶点
plt.scatter([vertex[0] for vertex in vertices], [vertex[1] for vertex in vertices], color='g')
# 使用红色的线来连线顶点，绘制出轮廓的边缘
plt.plot([vertex[0] for vertex in vertices]+[vertices[0,0]], [vertex[1] for vertex in vertices]+[vertices[0,1]], 'r-')
```

```
# 因为图像的坐标系统和 matplotlib 默认的坐标系统 y 轴方向是相反的，这里需要翻转 y 轴
plt.gca().invert_yaxis()
def Check_Simple_Poly(poly):
    """
    判断多边形是否为单连通多边形
    若不为单连通多边形，无法使用切耳法
    poly:shapely.geometry.Polygon，多边形对象
    """
    boundary_var = poly.boundary
    if boundary_var.is_ring and list(poly.interiors) == []:
        return True
    else:
        return False
def Get_Vex_Idx(polygon):
    """
    得到 poly 中的可划分顶点的下标序列
    polygon:shapely.geometry.Polygon，多边形对象
    verteices_array:list，多边形的顶点的数组
    index_of_divisable_vertex:list，可划分的顶点的 index
    angle_list:list，每个角点对应的角度
    """
    index_of_divisable_vertex = []
    angle_list = []
    verteices_array = np.asarray(polygon.exterior.coords)
    verteices_array = verteices_array[:-1,:]
    count_of_verteices = verteices_array.shape[0]
    if count_of_verteices <= 3:
        return verteices_array, index_of_divisable_vertex, angle_list
    def Judge_order(poly):
        return 1 if LinearRing(poly.exterior).is_ccw else -1
    polygon_order = Judge_order(polygon)
    for i in range(count_of_verteices):
        v = verteices_array[i,:]
        l = verteices_array[i-1,:]
        r = verteices_array[(i+1)%count_of_verteices,:]
        vec1 = v - l
        vec2 = r - v
        A = []
        A = np.array([vec1,vec2])
        if polygon_order*np.linalg.det(A) > 0:
            remainvex_arr                                                    =
```

```
np.concatenate([verteices_array[:i,:],verteices_array[i+1:,:]],axis=0)
                remain_poly = Polygon(remainvex_arr)
                triangular_polygon = Polygon([l,v,r])
                if (remain_poly.is_valid
                    and remain_poly.intersection(triangular_polygon).area < 1e-8
                    and polygon.equals(remain_poly.union(triangular_polygon))):
                    index_of_divisable_vertex.append(i)
                    arc = np.arccos(-
np.dot(vec1,vec2)/np.linalg.norm(vec1)/np.linalg.norm(vec2))
                    angle_list.append(arc)
    return verteices_array, index_of_divisable_vertex, angle_list
def Div_Tris(poly, triangle_list = []):
    """
    递归的将多边形，进行三角剖分，每次都以角度最小的可划分顶点为依据
    poly：shapely.geometry.Polygon，待划分的多边形
    triangle_list：list
    triangle_list：list，poly 划分后的三角形列表
    """
    vertex_array, division_index_list, division_arc_list = Get_Vex_Idx(poly)
    vertex_count = vertex_array.shape[0]
    if vertex_count <= 3:
        triangle_list.append(poly)
        return triangle_list
    select_index = division_index_list[np.argmin(np.array(division_arc_list))]
    vertex = vertex_array[select_index, :]
    left_vertex = vertex_array[select_index - 1, :]
    right_vertex = vertex_array[(select_index + 1) % vertex_count, :]
    triangle = Polygon([left_vertex, vertex, right_vertex])
    triangle_list.append(triangle)

    remain_vertex_array =
np.concatenate([vertex_array[:select_index,:],vertex_array[select_index+1:,:]],axis=0)
    remain_poly = Polygon(remain_vertex_array)
    Div_Tris(remain_poly, triangle_list)
    return triangle_list
def PolyNormalization(poly_arr):
    """
    用于对 poly_arr 进行归一化处理
    """
    arr_temp = poly_arr - np.min(poly_arr,axis=0)
    return arr_temp / np.max(arr_temp)
def MinAngle(tri):
```

```
    """
    计算一个三角形的最小角的弧度
    tri:shapely.geometry.Polygon，三角形
    min(arc_li):float，三角形的最小角度。
    """
    point = np.asarray(tri.exterior.coords)
    arc_li = []
    for i in range(3):
        j = (i+1)%3; k=(i+2)%3
        a = np.linalg.norm(point[i,:] - point[j,:])
        b = np.linalg.norm(point[j,:] - point[k,:])
        c = np.linalg.norm(point[k,:] - point[i,:])
        arc = np.arccos((a**2 + b**2 - c**2)/(2*a*b))
        arc_li.append(arc)
    return min(arc_li)
def Opt_Div(poly4_vex_arr):
    """
    对四边形进行优化划分，返回其最优化的两个三角形
    poly4_vex_arr:numpy.ndarray, 一个四边形的顶点集数组，
    tri1,tri2:shapely.geometry.Polygon，划分优化后的两个三角形。
    """
    tri1 = Polygon(poly4_vex_arr[[0,1,2]])
    tri2 = Polygon(poly4_vex_arr[[0,2,3]])
    arc1 = min([MinAngle(tri1),MinAngle(tri2)])
    tri3 = Polygon(poly4_vex_arr[[0,1,3]])
    tri4 = Polygon(poly4_vex_arr[[1,2,3]])
    arc2 = min([MinAngle(tri3),MinAngle(tri4)])
    if arc1 >= arc2:
        return tri1,tri2
    else:
        return tri3,tri4
def Opt_Alltris(triangle_list):
    """
    对已经给出的三角剖分进行进一步的优化，使得最小角最大
    triangle_list:list，划分后的三角形列表
    triangle_list:list，细分和优化后的三角形列表。
    """
    random.shuffle(triangle_list)
    count_of_tris = len(triangle_list)
    for idx1 in range(count_of_tris):
        tris_1 = triangle_list[idx1]
        for idx2 in range(idx1+1, count_of_tris):
```

```
                tris_2 = triangle_list[idx2]
                if tris_1.intersection(tris_2).length > 1e-10:
                    union_set = tris_1.union(tris_2)
                    vertex_array, _ , _ = Get_Vex_Idx(union_set)
                    if len(_) == 4:
                        tris_A, tris_B = Opt_Div(vertex_array)
                        validation_flag = True
                        for idx in set(range(count_of_tris)) - {idx1, idx2}:
                            if tris_A.intersection(triangle_list[idx]).area > 0. or
tris_B.intersection(triangle_list[idx]).area > 0.:
                                validation_flag = False
                        if validation_flag:
                            triangle_list[idx1], triangle_list[idx2] = tris_A, tris_B
    return triangle_list
def Check_Right(triangle):
    """
    判断三角形是否为直角三角形
    triangle:shapely.geometry.Polygon，三角形
    """
    points = np.array(triangle.exterior.coords)[:-1]
    a = np.linalg.norm(points[0]-points[1])
    b = np.linalg.norm(points[1]-points[2])
    c = np.linalg.norm(points[2]-points[0])
    a, b, c = sorted([a, b, c])
    if abs(a*a + b*b - c*c) < 1e-10:
        return True
    else:
        return False
def longest_side(triangle):
    """
    计算三角形的最长边并返回其中点和对应的另一点坐标
    triangle:shapely.geometry.Polygon，三角形
    longest:tuple，其中包含两个点的坐标，这两个点形成的线段是三角形的最长边
    other_point:tuple，包含另一点坐标的元组，此点是三角形的第三个点，不在最
长边上
    """
    points = list(triangle.exterior.coords)[:-1]
    sides = [(points[i], points[(i+1)%3]) for i in range(3)]
    longest = max(sides, key=lambda side: np.linalg.norm(np.array(side[0]) -
np.array(side[1])))
    other_point = next(point for point in points if point not in longest)
    return longest, other_point
```

```
def perpendicular_line_Div_Tri(longest, otherpoint):
    """
    画出从其他点到最长边的垂线，将其分割为两个直角三角形
    longest:tuple，其中包含两个点的坐标，这两个点形成的线段是三角形的最长边
    other_point:tuple，包含另一点坐标的元组，此点是三角形的第三个点，不在最长边上
    """
    line = LineString([(longest[0][0],longest[0][1]), (longest[1][0],longest[1][1])])
    u = line.project(otherpoint)
    footpoint = line.interpolate(u)

    tri1 = Polygon(((longest[0][0],longest[0][1]),footpoint,otherpoint))
    tri2 = Polygon(((longest[1][0],longest[1][1]),footpoint,otherpoint))
    nntris.append(tri1)
    nntris.append(tri2)
def Find_Gradient(point1,point2,point3):
    '''
    point1,point2,point3:numpy.ndarray，二维数组，代表在平面上的点，数组中的两个元素是点的 x 和 y 坐标
    '''
    a = np.linalg.norm(point1-point2)
    b = np.linalg.norm(point1-point3)
    if point1[0]-point2[0] != 0:
        theta1 = math.atan2((point2[1]-point1[1]),(point2[0]-point1[0]))
        if point2[1]-point1[1] < 0:
            theta1 += 2*math.pi
    elif point1[1]<point2[1]:
        theta1 = math.pi/2
    else:
        theta1 = 3*math.pi/2
    if point1[0]-point3[0] != 0:
        theta2 = math.atan2((point3[1]-point1[1]),(point3[0]-point1[0]))
        if point3[1]-point1[1] < 0:
            theta2 += 2*math.pi
    elif point1[1]<point3[1]:
        theta2 = math.pi/2
    else:
        theta2 = 3*math.pi/2
    if theta2 > math.pi:
        if theta1 < theta2 - math.pi:
            theta1 = theta1 + 2*math.pi
    if theta1 > math.pi:
```

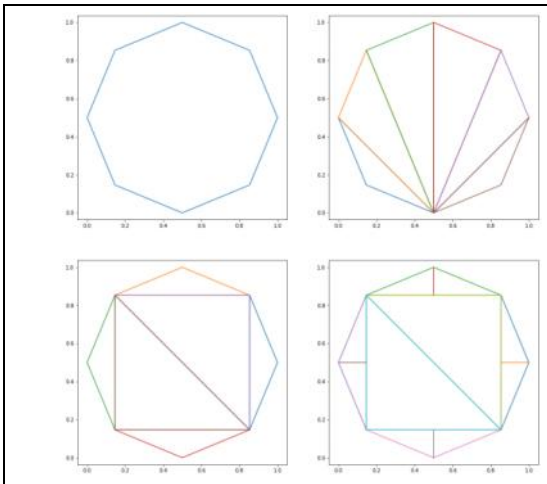
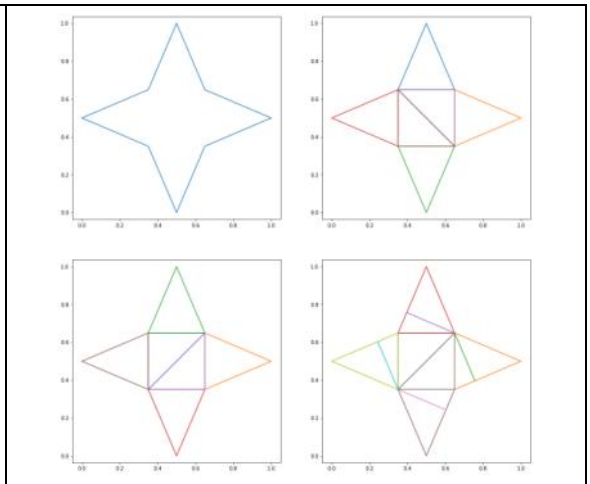
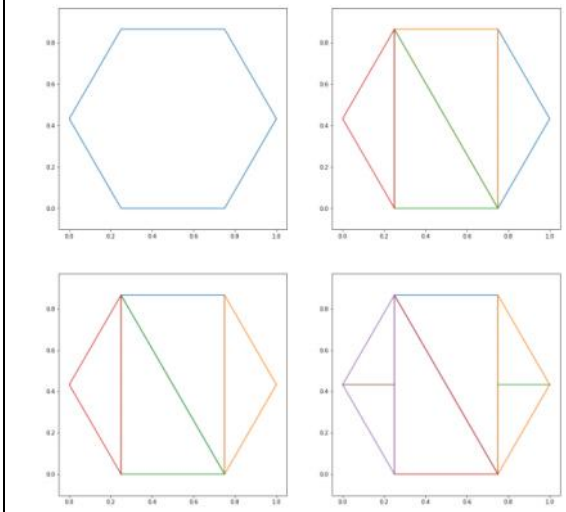
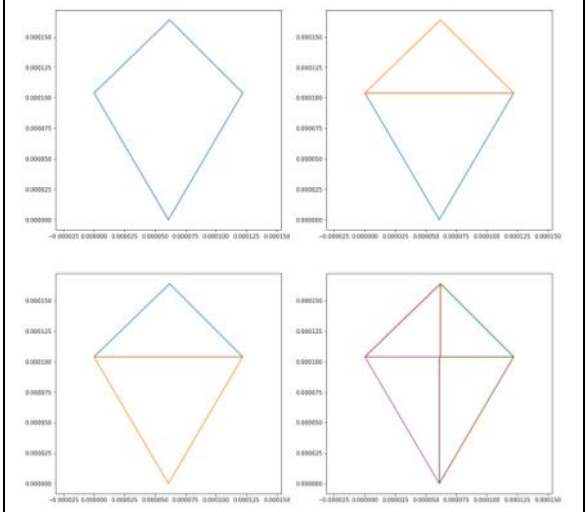
```
            if theta2 < theta1 - math.pi:
                theta2 = theta2 + 2*math.pi
        if theta1 < theta2:
            return theta1,a,b
        else:
            return theta2,b,a
def Find_Right_Detail(triangle):
        '''
        triangle:shapely.geometry.Polygon，三角形
        '''
        points = np.array(triangle.exterior.coords)[:-1]
        a = np.linalg.norm(points[0]-points[1])
        b = np.linalg.norm(points[1]-points[2])
        c = np.linalg.norm(points[2]-points[0])
        if abs(a*a + b*b - c*c) < 1e-10:
            theta,a,b=Find_Gradient(points[1],points[2],points[0])
            return points[1],theta,a,b
        elif abs(b*b + c*c - a*a) < 1e-10:
            theta,a,b=Find_Gradient(points[2],points[0],points[1])
            return points[2],theta,a,b
        elif abs(c*c + a*a - b*b) < 1e-10:
            theta,a,b=Find_Gradient(points[0],points[1],points[2])
            return points[0],theta,a,b
        else:
            print("不是直角三角形")
a = math.sin(math.pi/12)/math.sin(math.pi/1.5)
#顶点序列
#poly = Polygon((poly_arr)) #构造多边形
poly = Polygon((vertices*0.000001))
#运算,绘图脚本
if Check_Simple_Poly(poly):
        plt.figure(figsize=(16,16))
        tris = []
        tris =    Div_Tris(poly,triangle_list = tris)
        plt.subplot(2,2,1)
        plt.plot(*poly.exterior.xy)#原来图形的线框
        plt.axis("equal")
        plt.subplot(2,2,2)
        for tri in tris:
            plt.plot(*tri.exterior.xy)#用线框画出剖分
        plt.axis("equal")
        plt.subplot(2,2,3)
```

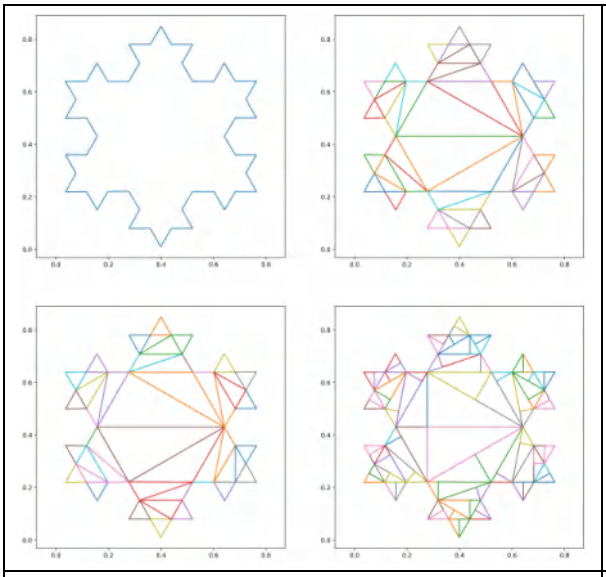
```
    newtris = tris.copy()
    newtris = Opt_Alltris(newtris)
    newtris = Opt_Alltris(newtris)
    for newtri in newtris:
        plt.plot(*newtri.exterior.xy)#进行优化，并用线框画出新的剖分
    plt.axis("equal")
    #添加垂线分割三角形
    plt.subplot(2,2,4)
    nntris = []
    for newtri in newtris:
        plt.plot(*newtri.exterior.xy)
        if not Check_Right(newtri):
            # 获取最长边和另一点的坐标
            longest, op = longest_side(newtri)
            # 从另一点到最长边的垂线分割后的三角形
            perpendicular_line_Div_Tri(longest, Point(op))
        else:
            nntris.append(newtri)
    for nntri in nntris:
        plt.plot(*nntri.exterior.xy)
    plt.axis("equal")
    plt.show()
    #导出数据
    detail = []
    for nntri in nntris:
        rp,theta,a,b=Find_Right_Detail(nntri)
        detail.append([rp[0],rp[1],theta,a,b])
#folder_path = 'E:/'
#name_path = input("文件名(需加上'.xlsx')： ")
#result_path = os.path.join(folder_path, name_path)
result_path = os.path.join('E:/1.xlsx')
if os.path.exists(result_path):
    workbook = xl.load_workbook(result_path)
else:
    workbook = xl.Workbook()
    workbook.save(result_path)
sheet = workbook.active
#写第一行
headers = ["x0", "y0", "theta", "a", "b"]
sheet.append(headers)
#写内容
for i in range(len(detail)):
```

```
        sheet['A'+str(int(i+2))] = detail[i][0]
        sheet['B'+str(int(i+2))] = detail[i][1]
        sheet['C'+str(int(i+2))] = detail[i][2]
        sheet['D'+str(int(i+2))] = detail[i][3]
        sheet['E'+str(int(i+2))] = detail[i][4]
workbook.save(result_path)
workbook.close()
app = xw.App(visible=False,add_book=False) #启动 Excel 程序
workbook = app.books.open(result_path) #  打开要调整行高和列宽的工作簿
worksheet = workbook.sheets #获取工作簿中的所有工作表
for i in worksheet: #遍历工作簿中的工作表
        i.autofit() #  自动调整工作表的行高额列宽
workbook.save() #保存工作簿
workbook.close() #  关闭工工作簿
app.quit() #退出 Excel  程序
```

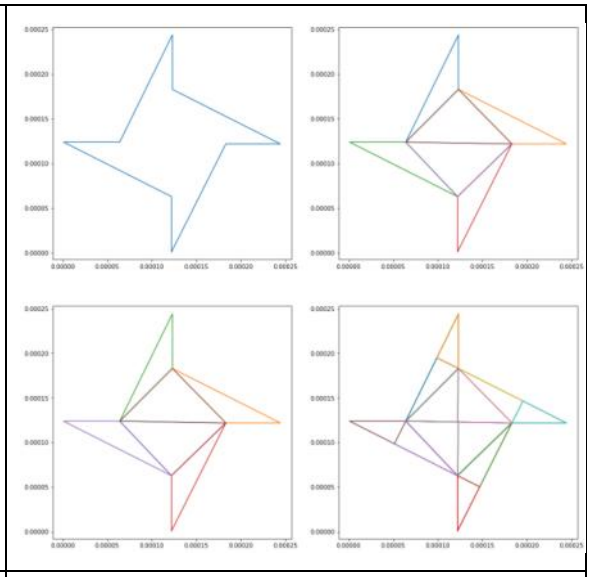
B. 耳切法分割程序运行结果

表 1 耳切法分割示意图

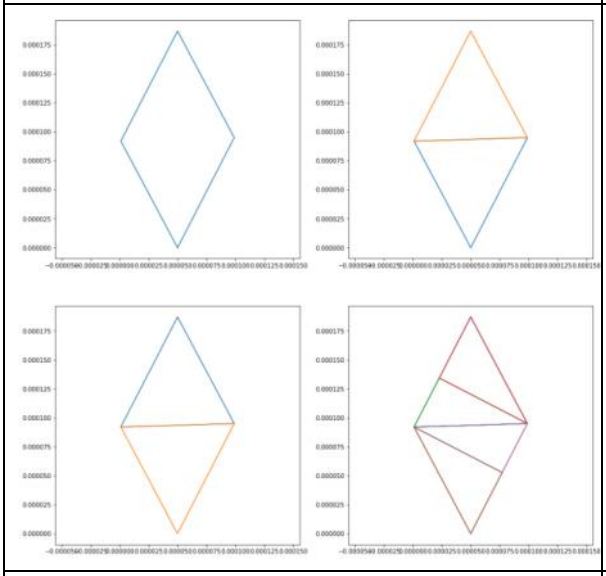
	
正八边形	四角星
	
正六边形	等边拼等腰



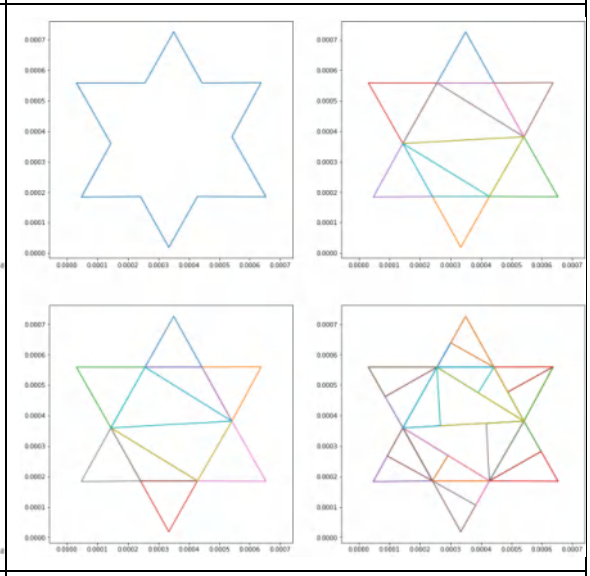
简单分型



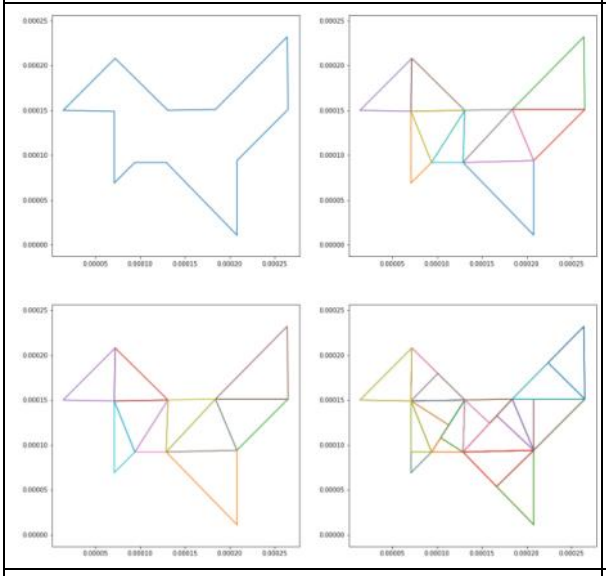
风车



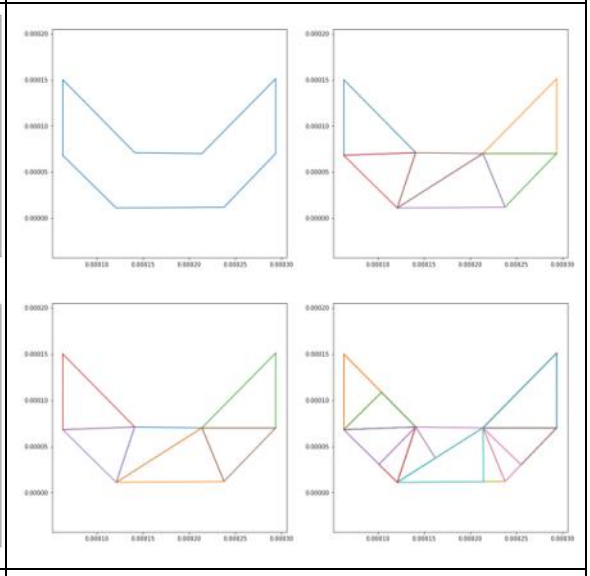
菱形



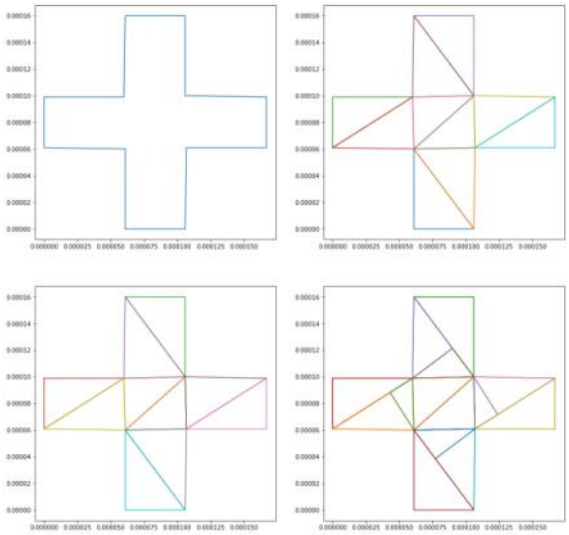
六角



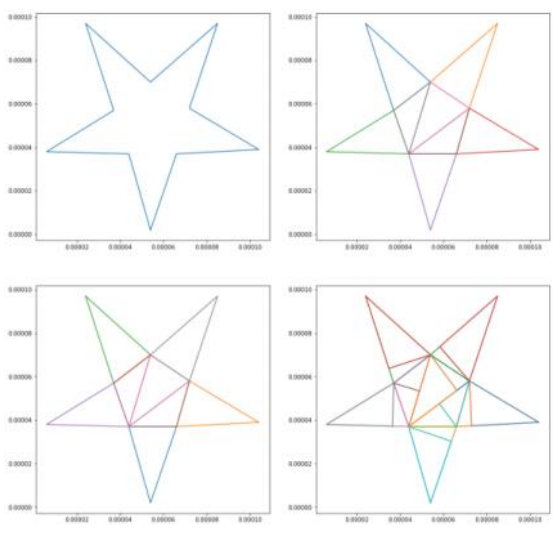
复杂拼凑 1



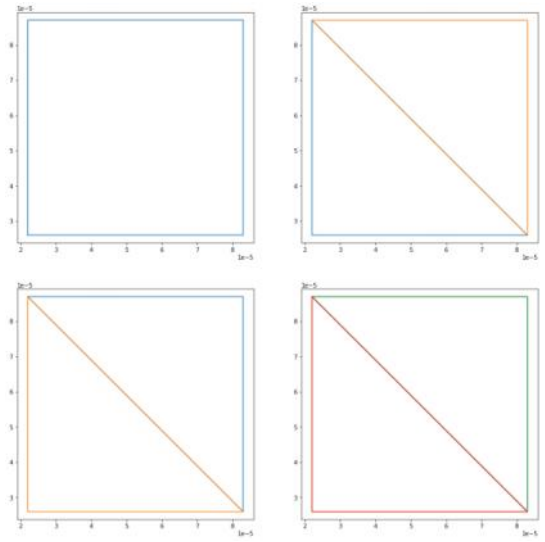
复杂拼凑 2



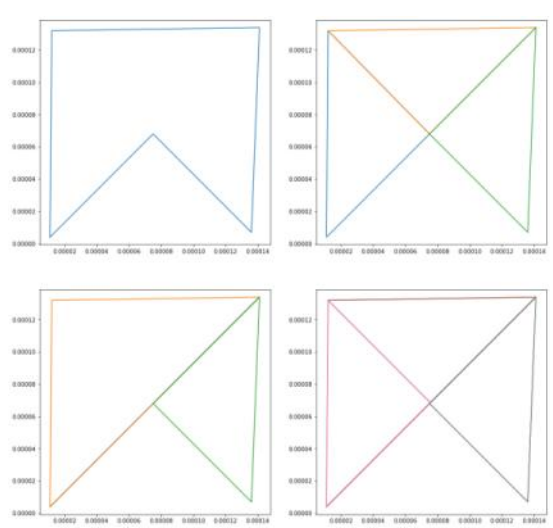
十字



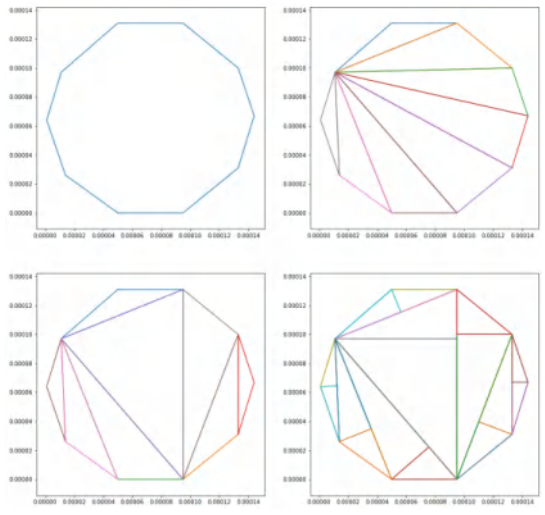
五角星



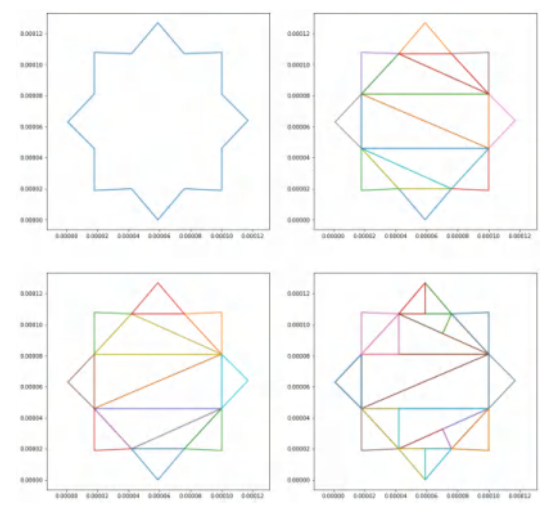
正方形



正方形缺一个



正十边形



八角星

C. 剩余解析解

①边长为 a 的八边形(原点位于中心)：

$$E(x,y)=\frac{4d^2}{k^2(x^4-6x^2y^2+y^4)}\left[\left(x^2-2\sqrt{2}xy+y^2\right)\cos\left(\frac{ak\cos\left(\frac{\pi}{8}\right)(x-y)}{d}\right)\right.$$
$$+\left(x^2+2\sqrt{2}xy+y^2\right)\cos\left(\frac{ak\cos\left(\frac{\pi}{8}\right)(x+y)}{d}\right)$$
$$-\left(\left(\sqrt{2}+1\right)x^2-\left(\sqrt{2}-1\right)y^2\right)\cos\left(\frac{akx\left(\sin\left(\frac{\pi}{8}\right)+\cos\left(\frac{\pi}{8}\right)\right)}{d}\right)$$
$$\left.+\left(\left(\sqrt{2}-1\right)x^2-\left(\sqrt{2}+1\right)y^2\right)\cos\left(\frac{aky\left(\sin\left(\frac{\pi}{8}\right)+\cos\left(\frac{\pi}{8}\right)\right)}{d}\right)\right] \quad (1)$$

②边长为 a 的星形(原点位于中心)：

$$E(x,y)=\frac{\left(1+\sqrt{3}\right)d^2e^{-\frac{iak\sqrt{3}x}{\sqrt{2}d}}}{k^2(x^4-14x^2y^2+y^4)}\left[\left(5-3\sqrt{3}+\left(5-3\sqrt{3}\right)e^{\frac{iak\sqrt{6}x}{d}}+\left(1+\sqrt{3}\right)e^{-\frac{iak\sqrt{3}(x-y)}{\sqrt{2}d}}\right.\right.$$
$$+\left(1+\sqrt{3}\right)e^{\frac{iak\sqrt{3}(x+y)}{\sqrt{2}d}}+\left(-3+\sqrt{3}\right)e^{\frac{iak\left(x+\sqrt{3}x+y-\sqrt{3}y\right)}{2\sqrt{2}d}}+\left(-3+\sqrt{3}\right)e^{\frac{iak\left(-x+3\sqrt{3}x+y-\sqrt{3}y\right)}{2\sqrt{2}d}}$$
$$\left.+\left(-3+\sqrt{3}\right)e^{\frac{iak\left(x+\sqrt{3}x-y+\sqrt{3}y\right)}{2\sqrt{2}d}}+\left(-3+\sqrt{3}\right)e^{\frac{iak\left((-1+3\sqrt{3})x+(-1+\sqrt{3})y\right)}{2\sqrt{2}d}}\right)x^2$$
$$+4\left(-3+\sqrt{3}\right)e^{\frac{iak\left((-1+\sqrt{3})x-(1+\sqrt{3})y\right)}{2\sqrt{2}d}}\left(e^{\frac{iak\sqrt{3}x}{\sqrt{2}d}}-e^{\frac{iakx}{\sqrt{2}d}}\right)\left(e^{-\frac{iak\sqrt{3}y}{\sqrt{2}d}}-e^{-\frac{iaky}{\sqrt{2}d}}\right)xy$$
$$+\left(1+\sqrt{3}+\left(1+\sqrt{3}\right)e^{\frac{iak\sqrt{6}x}{d}}+\left(5-3\sqrt{3}\right)e^{-\frac{iak\sqrt{3}(x-y)}{\sqrt{2}d}}+\left(5-3\sqrt{3}\right)e^{\frac{iak\sqrt{3}(x+y)}{\sqrt{2}d}}\right.$$
$$+\left(-3+\sqrt{3}\right)e^{\frac{iak\left(x+\sqrt{3}x+y-\sqrt{3}y\right)}{2\sqrt{2}d}}+\left(-3+\sqrt{3}\right)e^{\frac{iak\left(-x+3\sqrt{3}x+y-\sqrt{3}y\right)}{2\sqrt{2}d}}$$
$$\left.\left.+\left(-3+\sqrt{3}\right)e^{\frac{iak\left(x+\sqrt{3}x-y+\sqrt{3}y\right)}{2\sqrt{2}d}}+\left(-3+\sqrt{3}\right)e^{\frac{iak\left((-1+3\sqrt{3})x+(-1+\sqrt{3})y\right)}{2\sqrt{2}d}}\right)y^2\right] \quad (2)$$

③两个直角边长为 a 的直角三角形对顶拼接(原点位于直角顶点连接部分)：

$$E(x,y)=-\frac{d^2e^{\frac{i\sqrt{2}aky}{d}}}{k^2x(x-y)(x+y)}\left[-4xe^{-\frac{i\sqrt{2}aky}{d}}+(x-y)e^{\frac{iak(x-y)}{\sqrt{2}d}}\right.$$
$$\left.+(x-y)e^{-\frac{iak(x+3y)}{\sqrt{2}d}}+(x+y)e^{\frac{iak(x-3y)}{\sqrt{2}d}}+(x+y)e^{-\frac{iak(x+y)}{\sqrt{2}d}}\right] \quad (3)$$

④正方形顶缺一个(位于正方形中心，缺口朝右)：

$$E(x,y)=\frac{d^2e^{\frac{i\sqrt{2}aky}{d}}}{k^2x(x-y)y(x+y)}\left[2e^{-\frac{i\sqrt{2}aky}{d}}xy\right.$$
$$+e^{-\frac{iak(x+3y)}{\sqrt{2}d}}y(-x+y)+e^{\frac{iak(x-3y)}{\sqrt{2}d}}(x-y)(x+y)$$
$$\left.-e^{-\frac{iak(x+y)}{\sqrt{2}d}}y(x+y)+e^{\frac{iak(x-y)}{\sqrt{2}d}}(-x^2+y^2)\right] \quad (4)$$

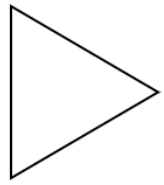
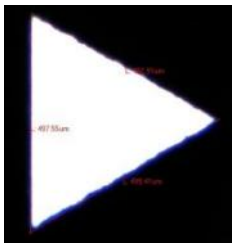
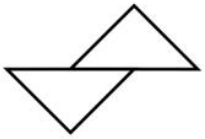
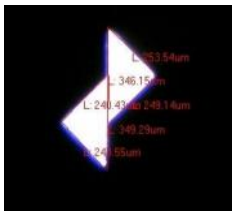
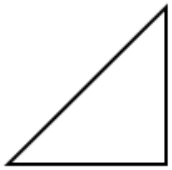
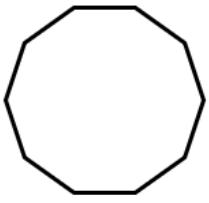
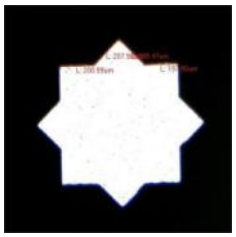
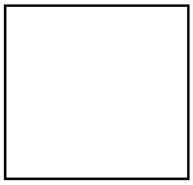
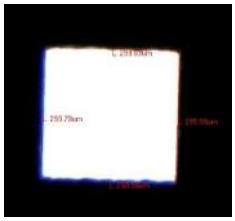
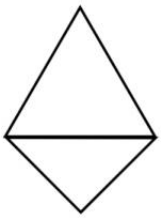
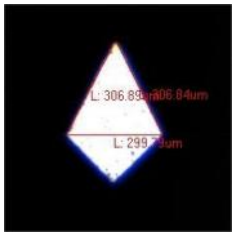
⑤两个边长为 a 的直角三角形斜边贴合，贴合部分为斜边的一半(原点位于贴合部分中点)：

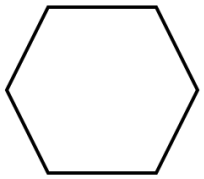
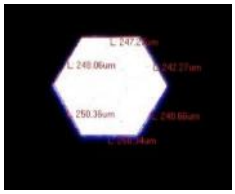
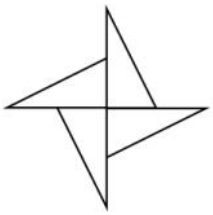
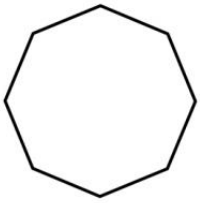
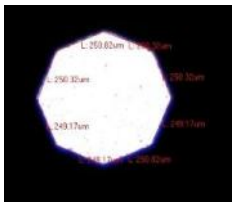
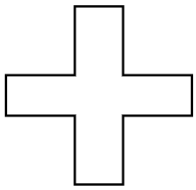
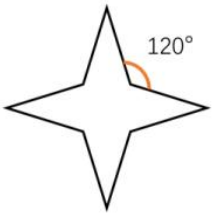
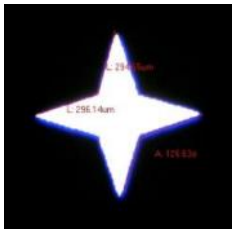
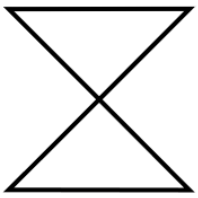
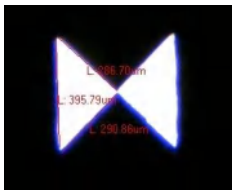
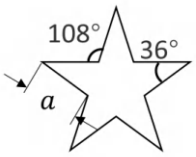
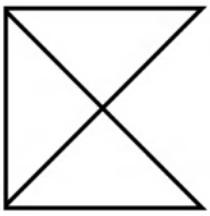
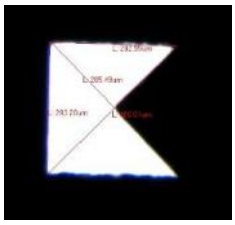
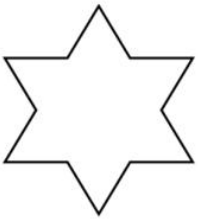
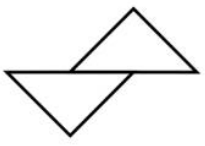
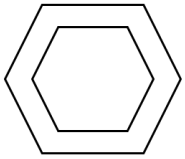
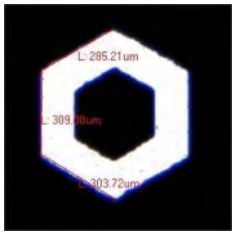
$$E(x,y)=-\frac{d^2e^{-\frac{iak(x-2y)}{2\sqrt{2}d}}}{k^2(x-y)x(x+y)}\left[-2e^{-\frac{i\sqrt{2}aky}{d}}x-2e^{-\frac{iakx}{\sqrt{2}d}}x\right.$$
$$+e^{\frac{iak(x-y)}{\sqrt{2}d}}(x-y)+e^{\frac{iak(2x-y)}{\sqrt{2}d}}(x+y)$$
$$\left.+e^{-\frac{iaky}{\sqrt{2}d}}(x-y)+e^{-\frac{iak(x+y)}{\sqrt{2}d}}(x+y)\right] \quad (5)$$

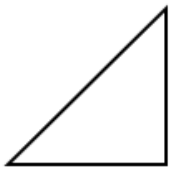
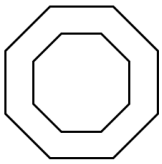
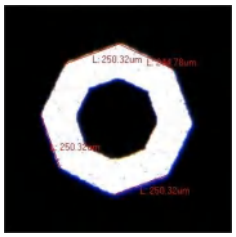
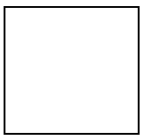
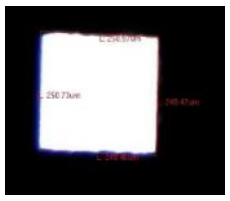
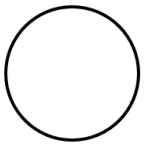
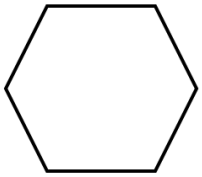
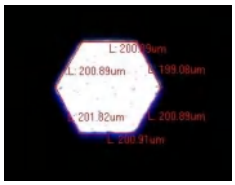
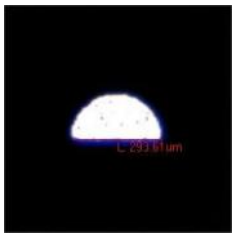
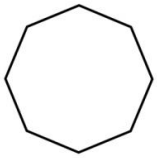
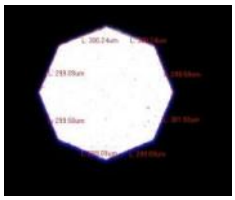
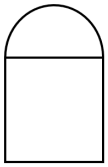
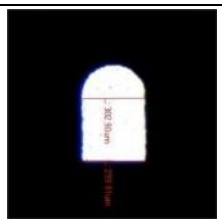
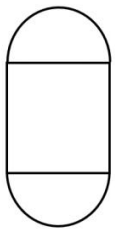
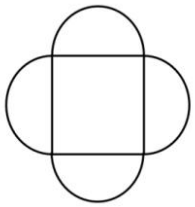
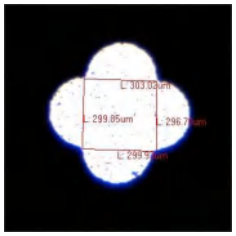
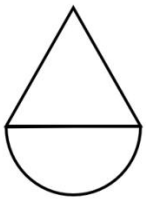
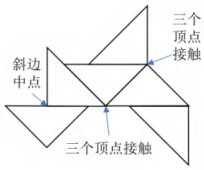
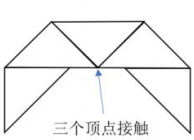
D. 实际孔尺寸

我们利用光学显微镜进行了孔尺寸的测量，其精度较高，误差范围约在 2%以内，以下即为实际孔的尺寸。

表 2 实际孔尺寸

孔模式	实际尺寸	孔模式	实际尺寸
 边长 500μm		 直角边长 250μm	
 直角边长 550μm		 边长 200μm	
 边长 300μm		 等边边长 300μm	

 边长 250μm		 短边长 250μm	
 边长 250μm		 短边长 100μm	
 边长 300μm		 边长 300μm	
 直角边长 300μm		 边长 300μm	
 直角边长 300μm		 边长 200μm	
 直角边长 300μm		 外边长 300μm	

 直角边长 500μm		 外边长 250μm	
 边长 250μm		 半径 150μm	
 边长 200μm		 半径 150μm	
 边长 300μm		 边长 300μm	
 边长 300μm		 边长 300μm	
 边长 300μm		 直角边长 300μm	
 直角边长 300μm		/	/