第九届“认证杯”数学中国

数学建模网络挑战赛

承　诺　书

我们仔细阅读了第九届“认证杯”数学中国数学建模网络挑战赛的竞赛规则。

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

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

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

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

我们的参赛队号为：4242

参赛队员　(签名)

队员 1：

队员 2：

队员 3：

参赛队教练员 (签名)：教师指导组

参赛队伍组别：大学本科组

2016 年第九届“认证杯”数学中国
数学建模网络挑战赛第二阶段论文

题　目： 低分辨率下看世界

关键词： 超分辨率重构图像　迭代反投影法　POCS 算法

摘要

图像处理应用广泛，超分辨率恢复技术是通过一幅或者多幅低分辨率图像，经由重建的方法来获得一幅高分辨率图像，还原图像中的细节最有效的方法是超分辨率图像重构。本文提出了三种有效的超分辨率图像重建方法：迭代反投影算法、凸集投影算法和 Paponlis-gerchberg 算法。

本文主要阐述迭代反投影算法。首先，该算法利用已知的多张低分辨率图像 *LR* 去反投影出模拟的高分辨率图片 *HR*；然后，由模拟的高分辨率图片 *HR* 去投影出模拟低分辨率图片 *LR*；最后，由模拟的低分辨率图片和实际的低分辨率图片的差值去对模拟的高分辨率图片 *HR* 进行更新，完成一次迭代。在迭代的过程中，模拟的高分辨率图片会不断接近我们想要的清晰度，差值也在不断接近阀值，当差值变得小于阀值 $e=10^{-4}$ 时，停止迭代，得到期望图片。

针对 *POCS* 算法，首先，我们对需要生成的超分辨率图像 HR 进行预估, 也就是建立参考帧 ；然后根据传感器得到的观察图像序列对参考帧进行修正, 直至定位到交集中的某一点 ；最后，不断循环对参考帧进行修正，最后得到可以接受的重构结果为止。

为了判别迭代反投影算法、 *POCS* 算法和 Papouli-Gerchberg 算法在算法性能上的优劣。本文采用了四个指标：图像视觉效果、算法运行时间、峰值信噪比（*PSNR*，其值越大说明处理效果越好，图片越清晰）和均方差。通过对比分析，得出结论：迭代反投影算法纹理特征明显，图像清晰，但算法运行时间较长；*POCS* 算法的 *PSNR* 值较大，图像放大质量良好，算法运行速度一般。

我们对模型的优缺点进行了分析，并对模型改进研究了推广方向。

参赛队号：4242

所选题目：B 题

参赛密码_____

（由组委会填写）

Abstract

Iterative back projection algorithm, projection onto convex sets (POCS algorithm and Paponlis-gerchberg algorithm super-resolution restoration technology is through a picture or multi frame low resolution image, through reconstruction method to obtain a high - resolution image. Reduction in the image details are the most effective method for super-resolution image reconstruction. This paper presents three effective super-resolution image reconstruction method.

This paper mainly expounds the iterative back projection algorithm. First of all, the algorithm using the known more than one low resolution image to projection simulation of high resolution images ; then, by the simulation of high resolution images projection simulated low resolution images ; finally, by the simulation of the low resolution difference rate picture and the actual low resolution images to simulated high resolution images update, one iteration is completed. In the iterative process, the simulation of high resolution images will be getting close to what we want clarity, the difference is also in constant close to threshold, the messenger value becomes smaller than a threshold stop iteration, to obtain the desired picture.

According to the algorithm . First of all, we to require the generation of super resolution image HR were estimated, is the establishment of the reference frame; then according to the sensor is obtained by the observed image sequence in the reference frame correction, until the locating to intersection of a point; finally, continuous cycle is used to modify the reference frame, the last to get acceptable reconstruction results so far.

In order to distinguish the iterative back projection algorithm, projection onto convex sets (POCS algorithm and Papouli-Gerchberg algorithm on the algorithm performance advantages and disadvantages. This paper uses the four indicators: image visual effect, algorithm running time, peak signal to noise ratio (abbreviated as PSNR, and mean square error. Through comparative analysis, draw the conclusion: iterative back projection algorithm for texture feature is obvious, clear image, but algorithm running time is too long; projection onto convex sets (POCS algorithm in PSNR value larger to enlarge the image quality is good, the running speed of the algorithm in general, visible this algorithm is practical.

The advantages and disadvantages of the model are analyzed, and the promotion of the model is improved.

Key words: super resolution reconstruction image iterative back projection algorithm POCS algorithm

一、问题的提出

1.1 问题重述

数码摄像技术被广泛使用于多种场合中。有时由于客观条件的限制，拍摄设备只能在较低的分辨率下成像。为简单起见，我们只考虑单色成像。假设成像的分辨率为32×64，成像方式是将整个矩形视野划分成 32×64 个相同大小的矩形格子，图像中每个像素的取值为对应格子的亮度平均值。每间隔一定时间拍摄一帧图像，运动的画面体现为图像的序列。

将一副静态的图像而言，每个像素对应于视野中的一个格子，每个格子内部的细节信息已经无法还原。但如果在视野移动的过程中拍摄系列图像，我们通过对多帧图像进行对比分析，仍然有可能还原出来一些单张照片中无法体现的细节。通过建立合理的数学模型和算法，对多帧图像分析，尽可能多地还原出被摄物的细节。

1.2 研究背景和目的

高分辨率重建技术是数字图像处理的一个重要的研究方向，利用序列图像进行超分辨率图像的复原成了人们的研究热点。因为它充分利用了不同相邻低分辨率帧图像之间类似而又不同的信息，也就是图像序列中的附加空域时域信息，所以其超分辨率复原能力好于利用单帧图像进行复原所获得的超分辨率能力。该技术被广泛应用于医学、遥感、计算机视觉、气象预报、军事等方面。本文建立合理的数学模型进行对图像的分析与处理。

1.3 超分辨率重构

图像超分辨率重构(super resolution,SR)是指利用计算机将多幅低分辨率图像(low resolution,LR)或图像序列进行处理，然后再通过一定的重建算法得到的高分辨率图像。恢复出高分辨率图像(high resolution,HR)的一种图像处理技术。HR 意味着图像具有高像素密度，可以提供更多的细节，这些细节往往在应用中起到关键作用。超分辨重构主要分为单幅图像的超分辨率重构和多幅图像的超分辨重构。

二、问题的分析

如果是对一幅静态的图像而言，没有其他图片和其做对比，再加上其视野内的物体不动，又由于每个像素对应于视野中的一个格子，其值是恒定的，所以我们没有特定的评价标准将其细化至清晰。但如果在视野移动的过程中像，然后我们再把这一系列多帧图像给提取出来，再通过一定的算法将其进行转换，进而可以将在原低分辨率图中未成体现的细节更清晰地表现出来。

三、模型的假设

1. 假设索取画面的各个像素点都是完好无损的，不存在像素点的缺失。
2. 假设图片中所有的像素点都是连续的。
3. 假设所采用的灰度图像和彩色图像在一定的范围内变化，系统误差小。

四、名词解释与符号说明

4.1 名词解释

1.阈值：阈值又叫临界值，是指一个效应能够产生的最低值或最高值。

2.帧数：就是在 1 秒钟时间里传输的图片的量，也可以理解为图形处理器每秒钟能够刷新几次，通常用 fps 表示。每一帧都是静止的图象，快速连续地显示帧便形成了运动的假象。高的帧率可以得到更流畅、更逼真的动画。帧数越高，所显示的动作就会越流畅。

3.峰值信噪比：峰值信噪比（经常缩写为 PSNR）是一个表示信号最大可能功率和影响它的表示精度的破坏性噪声功率的比值的工程术语。由于许多信号都有非常宽的动态范围，峰值信噪比常用对数分贝单位来表示。

4.凸集：若从原点出发，并且通过该集合中任意一点的所有射线以及连接这些射线的任意两点的所有线段仍然在该凸集内，则该凸集可以称为凸锥。

4.2 符号说明

符号	解释说明
$g_i(m_1,m_2)$	拍摄到的低分辨率图像
$f(n_1,n_2)$	待估计（模拟）的高分辨率图像
$\hat{g}_i^n(m_1,m_2)$	第 n 次迭代所得的第 i 幅模拟 LR 图像
$\hat{f}^{n+1}(n_1,n_2)$	模拟估计得到的 HR 图像
y_i	第 i 个传感器上输出的图像序列
$p_i[n_1,n_2]$	传感器的空间响应特性
C	凸集投影集合
pf	迭代投影使图像重构

x	高分辨率图像经模糊下采样后得到低分辨率图像
x^0	经过上采样处理得到的初始估计图像
x^n	重构所得到的高分辨率图像

五、模型的建立与求解

5.1 迭代反投影算法（IBP）

5.1.1 迭代反投影算法的建立

在文献[1]迭代反投影超分辨重构方法中，通过对模拟原始低分辨率（low resolution,LR）图像和观测低分辨率（LR）图像的误差进行迭代反投影，恢复出高分辨率（high resolution,HR）图像。

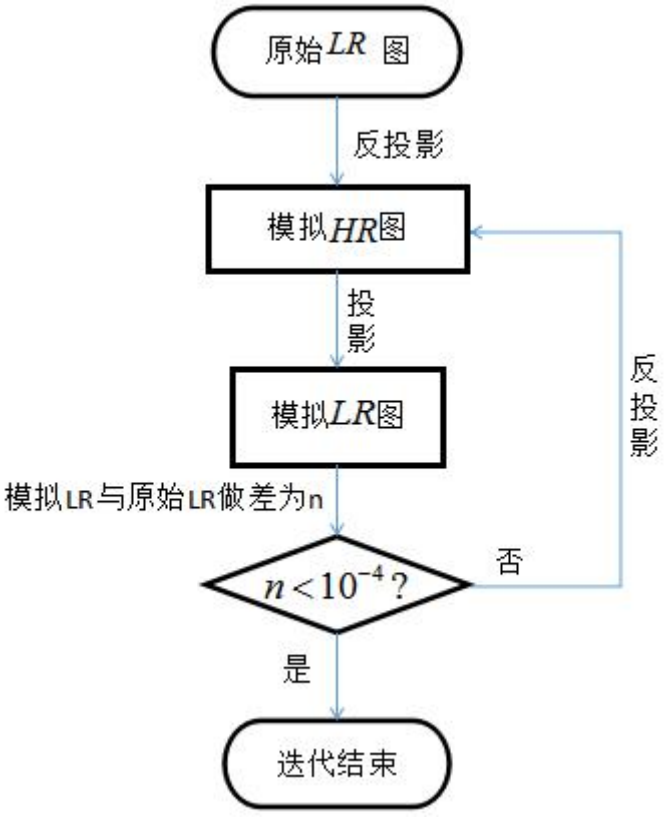


图 1 迭代反投影算法流程图

5.1.2 迭代反投影算法的求解

Step1: 令输入的 i 幅观测低分辨率图像为 $g_i(m_1,m_2)$，分辨率均为 $[M_1,M_2]$，待估计（模拟）的高分辨率图像 $f(n_1,n_2)$ 在 x,y 方向分辨率均扩大 1 倍，即：$[N_1,N_2]=(2M_1)\times(2M_2)$，通过反投影得到估计的 HR 图如下图所示：

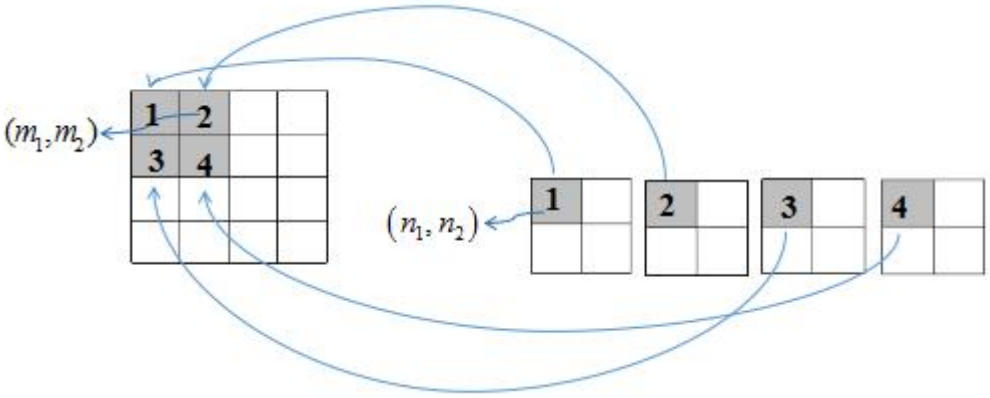


图 2 高分辨率（HR）图　　图 3 低分辨率（LR）图

Step2: 我们对得到的估计 *HR* 图进行加权平均，即把 *A* 区的部分向右移动很小的 Δx 和向下移动很小的 Δy，那么移动后的 *A* 区部分图像就融入了一部分的 *B*、*C*、*D* 区图像的信息。

我们假设新的 *A* 区有 70% 的原 *A* 区信息，15% 的 *B* 区信息，10% 的 *C* 区信息，5% 的 *D* 区信息。假设原 *A* 区的灰度值为100，原 *B* 区灰度值为110，原区灰度值为120，原 *D* 区灰度值为130，那么新的 *A* 区的灰度值 β 就为：

$$\beta = 70\%*100+15\%*110+10\%*120+5\%*130$$

由得到新的灰度值 β 信息的 A_1 替代原 A_1 区图像，完成对 *A* 区图像的更新。

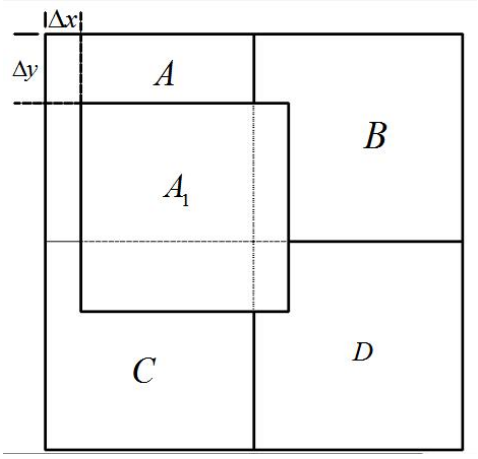


图 4　更新后的高分辨率图

然后再把更新后的 *HR* 图投影，得到 *i* 幅模拟 *LR* 图。

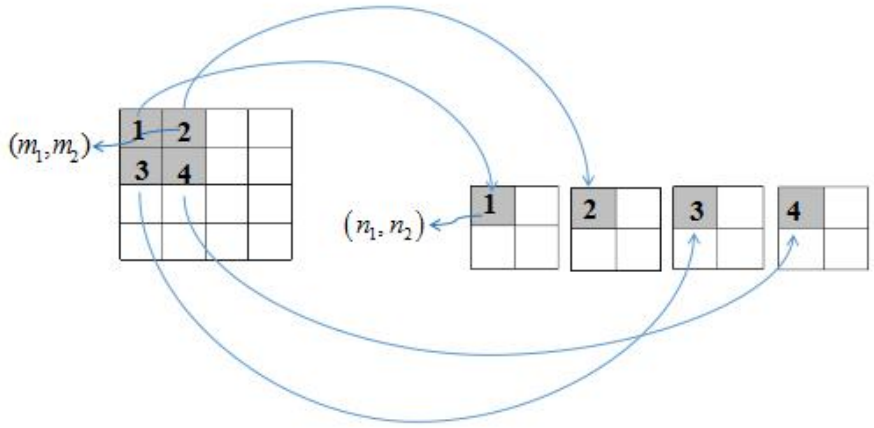


图 5　投影生成模拟 *LR* 图

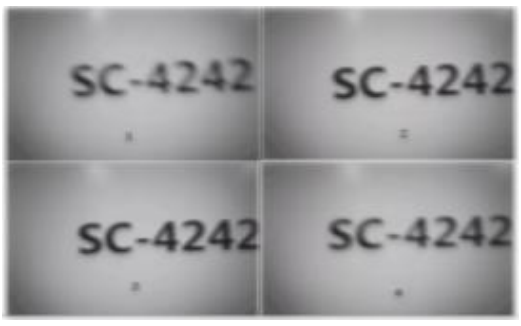
Step3: 本文把模拟 *LR* 图 $\hat{g}^n{}_i(m_1,m_2)$ 与原始 *LR* 图 $g_i(m_1,m_2)$ 作差，令其差值为 n，用 n 去对模拟 *HR* 图像进行反投影更新，完成一次迭代。并给定迭代过程一个阀值 e，经多次测试可知，当 $e=10^{-4}$ 时可得最佳精确度。在不断迭代中，n 的值不断缩小，当 $n<e$ 时，图像清晰度达到理想值，停止迭代，得到最终的 *HR* 图。

用迭代反投影 (*IBP*) 方法模拟估计得到 *HR* 图像的公式可表示为：

$$\hat{f}^{n+1}(n_1,n_2)=\hat{f}^{n}(n_1,n_2)+\lambda\sum_{m_1,m_2\in\Omega}\left(g_i(m_1,m_2)-\hat{g}_i^{n}(m_1,m_2)\right) \qquad (5\text{-}1\text{-}1)$$

式中，$\hat{g}_i^n$ 是第 n 次迭代所得的第 i 幅模拟 *LR* 图像，它是依据实际 *LR* 图像间的位移信息生成的。Ω 表示为模拟 $f(n_1,n_2)$ 所用到的 *LR* 图的位置集合。λ 为反投影系数，它决定误差 $\left(g_i(m_1,m_2)-\hat{g}_i^{n}(m_1,m_2)\right)$ 对模拟得到的 $\hat{f}^n(n_1,n_2)$ 的影响方式。

迭代反投影 (*IBP*) 方法[2] 处理效果图如下：



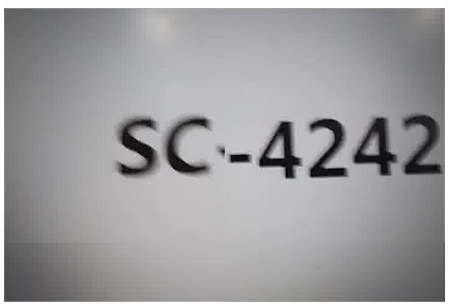


图 5 截取视屏的帧原图　　图 6 迭代反投影法的效果展示图

从上图可以看出通过迭代反投影法处理过后的图片成像较清晰，能够反映出细节。

5.2 凸集投影算法(*POCS*)

5.2.1 凸集投影算法模型的建立

凸集投影算法是一种集合理论的图像重建方法，所重建的图像的可行域是1组凸约束集合的交集，而这些凸约束集合由重建图像的各种先验知识组成。

POCS 算法[3]是一种迭代运算，相应的凸约束集合的投影算子将解空间中的点投影到距离表面最近的点上，经过有限次迭代，最终可以找到1个收敛于凸约束集合的交集的解。

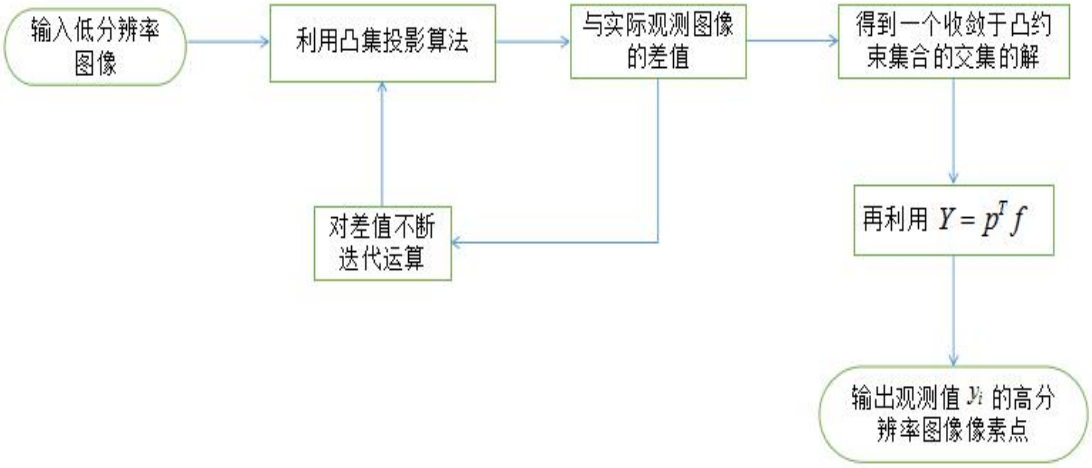


图 7　POCS 的算法流程图

5.2.2 凸集投影算法模型的求解

Step1：首先对需要生成的超分辨率图像 HR 进行预估, 也就是建立参考帧 , 由于建模第一问中已经得到估计运动，所以保证了观察图像中所有的点被投射到参考帧中正确的位置。

Step2：然后根据传感器得到的观察图像序列对参考帧进行修正, 直至定位到交集中的某一点 。

Step3：不断循环对参考帧进行修正，最后得到可以接受的重构结果。

首先我们的采样矩阵为 $M_1 \times M_2$，然后得到一系列低分辨率图像 $M_1 \times M_2$，由于图片成像受到多种条件的约束，故我们定义 p 为空间响应特征，表示高分辨率像素点 $[n_1, n_2]$ 落到传感器感应范围内的概率，则这个阵列中第 i 个传感器的输出可以表示为：

$$y_i = p \times \sum_{n_1} \sum_{n_1} f_i(n_1, n_2) \qquad (5\text{-}2\text{-}1)$$

其中，$1 \le n_1 \le N_1, 1 \le n_2 \le N_2, 1 < i < M_1 M_2$ 。

$$p_i[n_1,n_2]=\begin{cases}0 & ,\ [n_1,n_2]\text{在感应器感应范围外}\\ p_i\ (0\le p_i\le 1)\ , & [n_1,n_2]\text{部分位于感应器感应范围内}\\ 1 & ,\ [n_1,n_2]\text{全部位于感应器感应范围内}\end{cases} \quad (5\text{-}2\text{-}2)$$

写成的矩阵形式即为：

$$Y = p_i^T f \quad (5\text{-}2\text{-}3)$$

其中 Y 表示经过成像系统得到的观察图像序列 Y 。

其次，可以将其转化为集合表示为：

$$C = \left\{ f : Y = p_i^T f \right\} \quad (5\text{-}2\text{-}4)$$

上式集合表示包含了所有低分辨率图像序列的每个像素点所对应的输出观测值 y_i 的高分辨率图像的像素点。由于将 f 向集合 C 投影的方法不一，而通常采用与投影算子项城的办法，则定义投影算子 P 为

$$Pf = \begin{cases} f & ,Y = p_i^T f\text{(在范围内则还是原来像素所在位置)} \\ f + \dfrac{Y - p_i^T f}{p_i p_i^T} p_i & ,\text{（位置有偏差，通过不断迭代进行修正）} \end{cases} \quad (5\text{-}2\text{-}5)$$

POCS 算法将预估的结果 f_0 逐次投影到所有限制集 C 上，使得最终重构结果与观测图像一致：

$$f_{n+1} = P_1 P_2 P_3 \cdots P_k f_n \quad (5\text{-}2\text{-}6)$$

其中 k 表示限制集的个数，也就是观测点的总数。由此我们可以通过 *Matlab* 编程重构出较细节化的照片，关于 *POCS* 的详细源代码见附录。

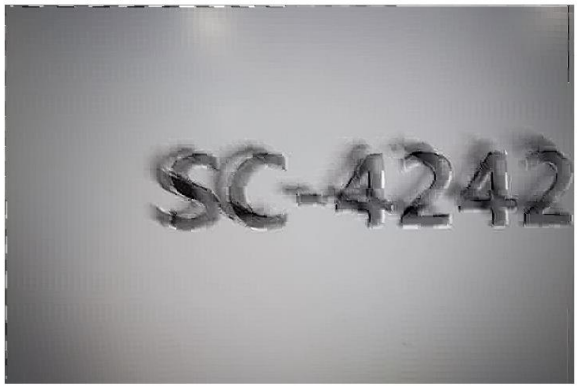


图 8 凸集投影算法的效果展示图

5.3 Papouli-Gerchberg 算法

由于真实场景景象经相机系统拍摄会得到明显低于真实景象分辨率的低分辨率图像。根据文献[1]超分辨成像模型理论可表示为：

$$g_i\left(m_1,m_2\right)=DHX+N \qquad (5\text{-}3\text{-}1)$$

其中 $g_i\left(m_1,m_2\right)$ 为所拍摄的低分辨率图像，N 为噪声模型，H 为镜头模糊矩阵；X 为真实景象（即为一幅高分辨率图像经模糊下采样后得到低分辨率图像，低分辨率图像中有不同的运动参数）

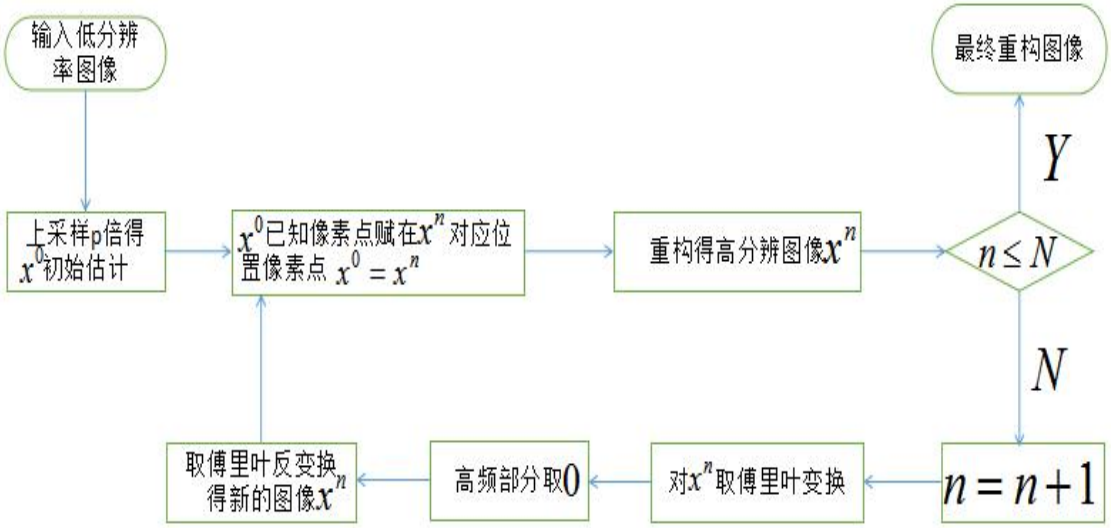


图 9　Papoulis-Gerchbeng 算法流程图

上图表示 Papoulis-Gerchbeng 算法的流程图，源代码详见附录。

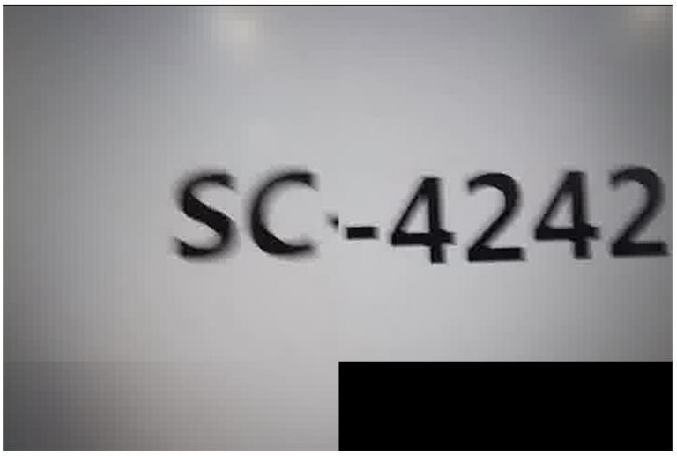


图 10　Papoulis-Gerchberg 算法的效果展示图

结论：$P-G$ 算法运行时间最短，清晰度适中，但是图片处理后右下角有黑

色区域。

5.4 算法的性能分析

为了比较常见的低分辨率图形处理方法和本文所用到的迭代反投影算法，对所拍摄的图形进行测试，并通过采用**视觉效果**和**峰值信噪比**(*PSNR*)及**算法运行时间**三个指标对算法进行分析和比较。

5.4.1 峰值性噪比

峰值信噪比(*PSNR*)是一个表示信号最大可能功率和影响它表示精度的破坏性噪声功率比值的工程术语，峰值信噪比经常用作图像压缩等领域中信号重建质量的测量方法，它常简单地通过均方差(*MSE*)进行定义。理论上讲性噪比是越高越好，两个 $m \times n$ 单色图像 I 和 K，如果一个为另外一个的噪声近似，那么它们的的均方差定义为：

$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}\| I(i,j) - K(i,j) \|^2 \qquad (5\text{-}4\text{-}1)$$

则峰值信噪比定义为：

$$PSNR = 10 \cdot \log_{10}(\frac{MAX_I^2}{MSE}) = 20 \cdot \log_{10}(\frac{MAX_I}{\sqrt{MSE}}) \qquad (5\text{-}4\text{-}2)$$

其中，MAX_I 是表示图像点颜色的最大数值，如果每个采样点用 8 位表示，那么就是 255。下面我们通过一个具体的例子来演示了峰值信噪比（PNSR）的计算方法：

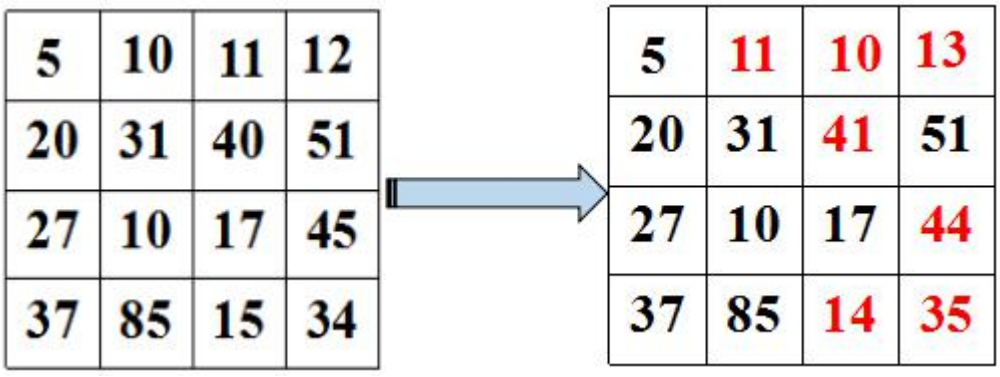


图 11　峰值性噪比的像素转换示意图

其中左侧是图像中的 16 个像素值，右侧是经过变动的图像的像素值。红色代表像素值有改变，那么这两幅图的 *MSE* 和 *PSNR* 的计算方式如下：

$$MSE = \frac{1}{16}\left[(5-5)^2 + (10-11)^2 + \ldots + (34-35)^2 \right] = 0.4375$$

$$PSNR = 10 \times \log_{10}\left[\frac{255^2}{MSE}\right] = 51.72$$

5.4.3 算法评价及数据分析

从以上三种重构实验结果可以得到如下表的数据：

表 1　各算法测试的比较

	迭代反投影算法（IBP）	凸集反投影算法（POCS）	Papouli-Gerchberg 算法
视觉效果	特别清晰	很模糊	有点模糊
峰值性噪比	40.3907	40.0688	40.0714
均方差（MSE）	5.9431	6.4003	6.3965
运行时间	44.52	39.41	17.31

实验数据的分析：通过 *MATLAB* 程序得出以上数据，可知 *IBP* 算法的峰值性噪比最大为 40.3907，均方差最小为 5.9431，但运行速度较慢； *POCS* 算法的峰值性噪比最大为 40.0688，均方差最小为 5.9431，但图像处理较差；Papouli-Gerchberg 算法的峰值性噪比最大为 40.3907，均方差最小为 5.9431，其运行时间最快。

结论：改进的效果算法相对于一般的超分辨率重构算法精确度较高，能很好的把在低分辨率下的未曾体现的细节经过重构后体现出来。

六、模型的稳定性检验

表 2　运行时间与所取帧数

	迭代反投影算法（IBP）	迭代反投影算法（IBP）	迭代反投影算法（IBP）	迭代反投影算法（IBP）	迭代反投影算法（IBP）
峰值性噪比	40.258	40.2356	40.3907	40.0688	40.0714
运行时间	48.37	46.78	44.52	39.41	17.31
帧数(n)	30	25	20	10	5

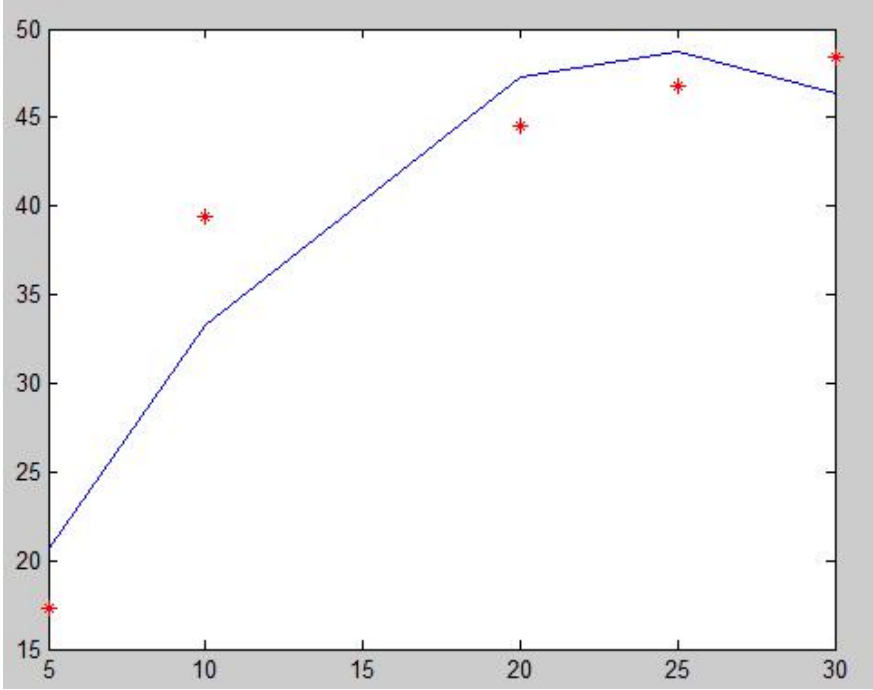


图 12　算法运行时间（t）随着帧数（n）线性拟合

结论：我们运用 *matlab* 对算法的运行时间以及所取不同的帧数进行线性拟合，可以知道时间帧数越大，运行的时间就越长，效果就越好，模型就越稳定。时间会随着帧数增加而逐渐趋于平稳。

七、模型的评价与改进

7.1 模型的评价

7.1.1 迭代反投影算法（*IBP*）的优缺点

优点：①迭代反投影算法投影效果良好，得到的图形纹理分明；

②*IBP* 在迭代过程中以所有观测图像为基准，取差值反投影，但运算

时间过长。

③峰值性噪比大，均方差较小。

缺点：由于迭代次数过多，算法运行时间较慢。

7.1.2 凸集反投影算法(*POCS*)的优缺点

优点：*POCS*（凸集投影）算法利用投影至凸集的原理进行图像重构，直观而有效，是有效的空间域算法之一。

缺点：均方差较大。

7.1.3Papouli-Gerchberg 算法的优缺点

优点：运行时间较快。

缺点：Papoulis-Gerchberg 模型中迭代次数太多耗费运算时间太多，而且限次的迭代又会造成结果误差较大。

7.2 模型的改进

1. *POCS* 算法在基于点扩散函数的图像修正的过程中，同时会产生边缘振荡效应，应该找到消除掉边缘振荡效应所产生的误差。

2. 对于边缘附近的区域，应该对边界进行光顺处理，以提高图像清晰度。

3. 对于边缘可以深度学习[4]，去获得区域的边界曲线。

4. 迭代反投影算法(IBP)还可以与遗传算法相结合的方法，设计出一种新的算法，使得图像边缘刻画清晰，较好地保持图像纹理特征，使图像更加细节化。

八、参考文献

[1] 郭伟伟，章品正. 基于迭代反投影的超分辨率图像重建. 南京：东南大学，2009,3.

[2] 李慧芳，杜明辉. 基于改进的 POCS 算法的超分辨率图像恢复. 广州：华南理工大学，2013，10.

[3] 李超峰，刘辉. 基于 POCS 算法的图像超分辨率重建. 昆明：昆明理工大学，2010,3.

[4] 刘琚，乔建草. 基于学习的超分辨率重建技术. 智能系统学报，2009,4（3）：199-207.

[5] 向海燕. 超分辨率图像恢复方法综述. 重庆理工大学学报，2014，9（9）：72-76.

[6] 穆绍硕，张叶. 一种改进 Papoulis-Gerchberg 的多幅超分辨率重构方法. 长春：中国科学院大学，2015，10:118-123.

[7] 赵海峰，周永飞，黄子强，图像放大算法比较研究 [J],现代电子技术，2010 年第 24 期 www.cnki.net

九、附录

①Papoulis-Gerchberg 算法程序

```
function rec = papoulisgerchberg(s,delta_est,factor)
       max_iter = 25;
```

```
temp = upsample(upsample(s{1}, factor)', factor)';
     y = zeros(size(temp));
coord = find(temp);
             y(coord) = temp(coord);
NHR = size(temp);
          NLR = floor(sqrt(length(s))*size(s{1})/2)*2;
     zero_rows = (1+NLR(1)/2+1)-1:(NHR(1)-NLR(1)/2-1)+1;
zero_cols = (1+NLR(2)/2+1)-1:(NHR(2)-NLR(2)/2-1)+1;
     for i = length(s):-1:1
    temp = upsample(upsample(s{i}, factor)', factor)';
    temp = shift(temp, round(delta_est(i, 2)*factor), round(delta_est(i,
1)*factor));
    coord = find(temp);
    y(coord) = temp(coord);
end
y_prev=y;
E=[];
iter=1;
wait_handle = waitbar(0, 'Reconstruction...', 'Name', 'SuperResolution
GUI');
while iter < max_iter
   waitbar(min(4*iter/max_iter, 1), wait_handle);
   Y=fft2(y);
   Y(zero_rows,:)=0;
   Y(:,zero_cols)=0;
   y=ifft2(Y);
   for i = length(s):-1:1
        temp = upsample(upsample(s{i}, factor)', factor)';
        temp = shift(temp, round(delta_est(i, 2)*factor),
round(delta_est(i, 1)*factor));
        coord = find(temp);
        y(coord) = temp(coord);
   end
   delta= norm(y-y_prev)/norm(y);
   E=[E; iter delta];
   iter= iter+1;
   if iter>3
     if abs(E(iter-3,2)-delta) <1e-8
       break
     end
   end
   y_prev=y;
end
close(wait_handle);
```

```
rec = real(y);
```

②迭代反投影算法程序

```
function [I Frames] = iteratedbackprojection(s, delta_est, phi_est,
factor)
if nargout > 1
    outputFrames = true;
else
    outputFrames = false;
end
movieCounter = 1;
imOrigBig = imresize(s{1}, factor, 'nearest');
if(outputFrames)
    figure;
end

lambda = 0.1;
max_iter = 100;
iter = 1;

X = imOrigBig;
X_prev = X;
E = [];

blur = [0 1 0;...
       1 2 1;...
       0 1 0];
blur = blur / sum(blur(:));

sharpen = [0 -0.25 0;...
         -0.25 2 -0.25;...
          0 -0.25 0];

wait_handle = waitbar(0, 'Reconstruction...', 'Name', 'SuperResolution
GUI');

while iter < max_iter
    waitbar(min(10*iter/max_iter, 1), wait_handle);

    if(outputFrames)
        imshow(X);
        Frames(movieCounter) = getframe;
        movieCounter = movieCounter + 1;
```

```
    end

    G = zeros(size(X));
    for i=1:length(s)
        temp = circshift(X, -[round(factor * delta_est(i,1)), round(factor
* delta_est(i,2))]);
        temp = imrotate(temp, phi_est(i), 'crop');

        temp = imfilter(temp, blur, 'symmetric');

        temp = temp(1:factor:end, 1:factor:end);
        temp = temp - s{i};
        temp = imresize(temp, factor, 'nearest');

        temp = imfilter(temp, sharpen, 'symmetric');

        temp = imrotate(temp, -phi_est(i), 'crop');
        G = G + circshift(temp, [round(factor * delta_est(i,1)),
round(factor * delta_est(i,2))]);
    end

    X = X - (lambda) * G;

    delta = norm(X-X_prev)/norm(X);
    E=[E; iter delta];
    if iter>3
      if abs(E(iter-3,2)-delta) <1e-4
         break
      end
    end
    X_prev = X;
    iter = iter+1;
end

disp(['Ended after ' num2str(iter) ' iterations.']);
disp(['Final error is ' num2str(abs(E(iter-3,2)-delta)) ' .']);

close(wait_handle);
I = X;
```

③凸集投影算法的程序图

```
function y = pocs(s,delta_est,factor)
```

```
     max_iter = 50;

temp = upsample(upsample(s{1}, factor)', factor)';
            y = zeros(size(temp));
coord = find(temp);
    y(coord) = temp(coord);


for i = 2:length(s)
    temp = upsample(upsample(s{i}, factor)', factor)';
    temp = shift(temp, round(delta_est(i, 2)*factor), round(delta_est(i,
1)*factor));
    coord = find(temp);
    y(coord) = temp(coord);
end

y_prev=y;

E=[];
iter=1;

blur =[.25 0 1 0 .25;...
       0  1 2 1  0;...
       1  2 4 2  1;...
       0  1 2 1  0;...
      .25 0 1 0 .25];

blur = blur / sum(blur(:));
wait_handle = waitbar(0, 'Reconstruction...', 'Name', 'SuperResolution
GUI');

while iter < max_iter
   waitbar(min(4*iter/max_iter, 1), wait_handle);
   y = imfilter(y, blur);
   for i = length(s):-1:1
       temp = upsample(upsample(s{i}, factor)', factor)';
       temp = shift(temp, round(delta_est(i, 2)*factor),
round(delta_est(i, 1)*factor));
       coord = find(temp);
       y(coord) = temp(coord);
   end

   delta= norm(y-y_prev)/norm(y);
   E=[E; iter delta];
```

```
   iter = iter+1;
   if iter>3
     if abs(E(iter-3,2)-delta) <1e-4
        break
     end
   end
   y_prev=y;
end

close(wait_handle);
```

⑤算法的性能评价

```
function [PSNR, MSE] = psnr(X, Y)
% 计算峰值信噪比 PSNR、均方根误差 MSE
% 如果输入 Y 为空，则视为 X 与其本身来计算 PSNR、MSE

if nargin<2
      D = X;
else
      if any(size(X)~=size(Y))
            error('The input size is not equal to each other!');
      end
      D = X-Y;
end
MSE = sum(D(:).*D(:))/prod(size(X));
PSNR = 10*log10(255^2/MSE);


clc; close all;
I = imread('rice.png');
I1 = imnoise(I, 'salt & pepper');
figure;
subplot(1, 2, 1); imshow(I); title('原图像');
subplot(1, 2, 2); imshow(I1); title('加噪声图像');
[PSNR, MSE] = psnr(I, I1)
% 图像峰值信噪比 PSNR、均方根误差 MSE
```