第九届数学中国数学建模网络挑战赛

地址：数学中国数学建模网络挑战赛组委会　　网址：www.tzmcm.cn
电话：0471-4969085　　邮编：010021　　Email：2016@tzmcm.cn

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：1430

参赛队员　(签名)　：

队员 1：田恒乐

队员 2：明瑞晨

队员 3：王林茜

参赛队教练员 (签名)：

参赛队伍组别（例如本科组）：本科组

第九届数学中国数学建模网络挑战赛

地址：数学中国数学建模网络挑战赛组委会　　网址：www.tzmcm.cn
电话：0471-4969085　　邮编：010021　　Email：2016@tzmcm.cn

第九届“认证杯”数学中国

数学建模网络挑战赛
编 号 专 用 页

参赛队伍的参赛队号：（请各个参赛队提前填写好）：1430

竞赛统一编号（由竞赛组委会送至评委团前编号）：

竞赛评阅编号（由竞赛评委团评阅前进行编号）：

第九届数学中国数学建模网络挑战赛

地址：数学中国数学建模网络挑战赛组委会　　网址：www.tzmcm.cn
电话：0471-4969085　　邮编：010021　　Email：2016@tzmcm.cn

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　低像素还原、时间序列模型、数值量化、迭代算法

摘　　要：

运动模糊图像复原技术是当今图像处理研究领域的一个重要分支。景物形成过程中可能出现畸变或模糊，使所成图像降质，因而快速高效的图像信息复原方法是亟待解决的问题。考虑到低像素的图像分辨率低，特征被弱化，并且在每次获取图像的时候特征可能发生偏动，所以本文中建立了基于时间序列的复原模型。

首先基于第一阶段的模型，本文确定相邻两个图像之间的关系以及图像运动的具体参数，进而得到原始像素点的低像素与复原之后高像素的关系模型。

模型一是对一维运动的模糊图像进行还原。首先使用数值量化的方式，得到扩展像素点与相邻时间相邻位置的像素点间的关系，列写递推关系的方程组，并采用循环迭代方法求得所有的像素点的数值，并且使用随机数建立的时间序列模型进行模拟并检验模型的合理性。同时我们采用纵向插值、中值滤波等方法对复原结果进行了改进。将此方法应用在像素为 32*64 的图片中，得到了很好的处理效果，文中对复原前和复原后做了对比。

模型二是对模型一的推广，用来对二维运动的模糊图像进行还原，首先使用数值量化的方式，得到扩展像素点与相邻时间相邻位置的像素点间的关系，列写递推关系的方程组，并采用循环迭代方法求得所有的像素点的数值，并用中值滤波对复原结果进行了改进。将此方法应用在像素为 32*64 的图片中，得到了很好的处理效果，文中对复原前和复原后做了对比。

本文建立的模型从单个像素点入手，无需提取图像特征，应用范围广，对原图像的质量要求低，在夜间监控等场合低像素图像的复原能充分发挥作用。因此该模型的推广性和移植性都很好。

参赛队号： 1430

所选题目： B　**题**

参赛密码 _____
（由组委会填写）

第九届数学中国数学建模网络挑战赛

地址：数学中国数学建模网络挑战赛组委会　　网址：www.tzmcm.cn
电话：0471-4969085　　邮编：010021　　Email：2016@tzmcm.cn

Summary

Motion blurred image restoration technology is an important branch of image processing research field today. During the formation of the scene, distortion, blurry, and noise may appear, so images retrograde. Therefore, fast and efficient methods of image restoration are requried urgently. We noticed that thelow-pixel images have low resolution and greatly weakened features. And each time wecapture the images' feature, their value may occur bias.

In this paper, we established an image restorationmodel based on time series to solve the problem about information recognition of low-pixel images.

Based on the model in first phase, we determine the relationship between two adjacent images and the specific parameters of the image motion , and thus we obtain the relationship between an original low pixel and high pixel after restoration.

Model one is designed for one-dimensional motion blurred image restoration. We use the way of numerical quantitative to establish the relationship between the time extension pixel and its adjacent pixels, then we can establish equation set. With the help of Iterative loop, we can caculate the values for all the pixels. We also use the random number to simulate and test the rationality of this model. Then we use longitudinal interpolation, median filtering and other methods to improve the results. Application of this method on 32 * 64 pixels in the picture can get a good treatment effect, we make a comparison between the original low pixel and high pixel after restoration.

Model two is designed for two-dimensional motion blurred image restoration, which is an promotion and extension of model one. After deviding the origin picture in both horizontal and vertical directions. We use the same way with model one. Application of this method on 32 * 64 pixels in the picture can get a good treatment effect, we also make a comparison between the original low pixel and high pixel after restoration.

Models established in this paper start from a single pixel, and they needn't to extract image features. They are suitable for a wide range of situations even when the original images have a low quality. These models all have a good promotion and portability, and can be used to monitoring and other occasions during the night.

Key Words:　Low pixel reduction,　Time Series Models,
Number quantization,　Iterative algorithm

参赛队号 # 1430

低分辨率下看世界

目录

参赛队号 # 1430

一、问题重述

1.1 问题背景

在高分辨率的照片中，我们通过肉眼可以很容易得到图片中反映的信息。但是由于一些客观条件的限制，拍摄设备只能在较低的分辨率下成像，这就为我们图像信息的获取带来很大难度。如何从一系列有序列的图像的灰度和颜色信息尽可能地还原出这个低分辨率图片中的细节信息，即基于图像序列的信息复原建模和计算，是图像序列分析的重要内容之一。

运动模糊图像复原技术是当今图像处理研究领域的一个重要分支。景物形成过程中可能出现畸变、模糊、失真或混入噪声，使所成图像降质，称之为图像的“退化”。其中由于曝光瞬间相机与被摄景物之间存在相对位移而造成的模糊失真称为运动模糊退化，也就是说感光介质与被摄物体存在着像移。运动模糊复原技术能够去除或减轻在获取数字图像过程中发生的退化问题，从而使图像尽可能地接近于真实的场景。

1.2 问题提出

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。

第二阶段问题：对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在单张照片中无法体现的细节。请建立合理的数学模型和算法，通过对多帧图像进行分析，尽可能多地还原出被摄物的细节。

二、问题分析

1.动态图像序列是由一系列具有相对次序的帧图像组成。除了具有与图像一样的空间特性外，动态图像序列还具有时序特性，即运动信息。而序列图像灰度（或颜色值）与运动参数之间的关系是运动估计算法以及图像细节识别的主要基础。

2.图像复原是试图利用退化过程的先验知识使已退化的图像恢复本来面目，即根据退化的原因，分析引起退化的环境因素，建立相应的数学模型，并沿着使图像降质的逆过程恢复图像。从图像质量评价的角度来看，图像复原就是提高图像的可理解性。而图像增强的目的是提高视感质量，图像增强的过程基本上是一个探索的过程，它利用人的心理状态和视觉系统去控制图像质量，直到人们的视觉系统满意为止[1]。

对于图像复原，一般采用两种方法。一种方法是对于图像缺乏先验知识的情况下的复原，此时可对退化过程如模糊和噪声建立数学模型，进行数学描述，并进而寻找一种去除或削弱其影响的过程；另一种方法是对原始图像已经知道是那些退化因素引起的图像质量下降过程，来建立数学模型，并依据它对图像退化的影响进行拟合的过程[2]。

3.序列图像的获取：首先我们对一个选定的视野进行视频拍摄。根据题目要求，在拍摄过程中，摄像机应缓慢地移动。为了模拟图像序列实时地传输到计算机中，我们对视频逐帧进行截图，最后调整每张图片的分辨率为32×64，从而得到题目要求中低分辨率的序列图像。为了尽可能多地还原出被摄物的细节，一维情况下分别从中按时间顺序选取了 10 张示意如下图 1（a）——图 1（j）所示，二维情况下如图 2（a）——图 2（j）所示。因而后续的模型求解与检验都分别用这 10 张图片进行。

- 一维运动的图像序列

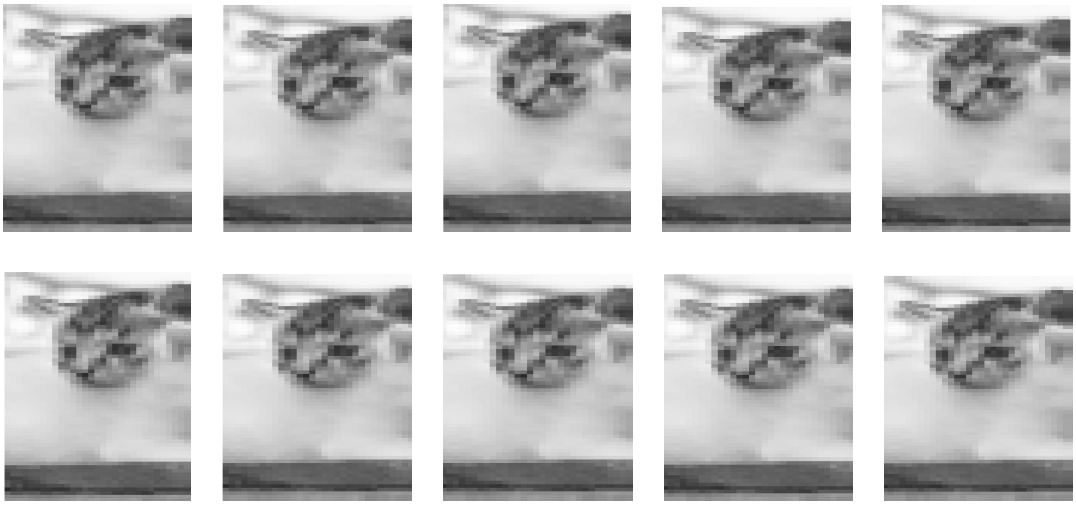

图 1（a）——图 1（j）:一维运动的图像序列

- 二维运动的图像序列

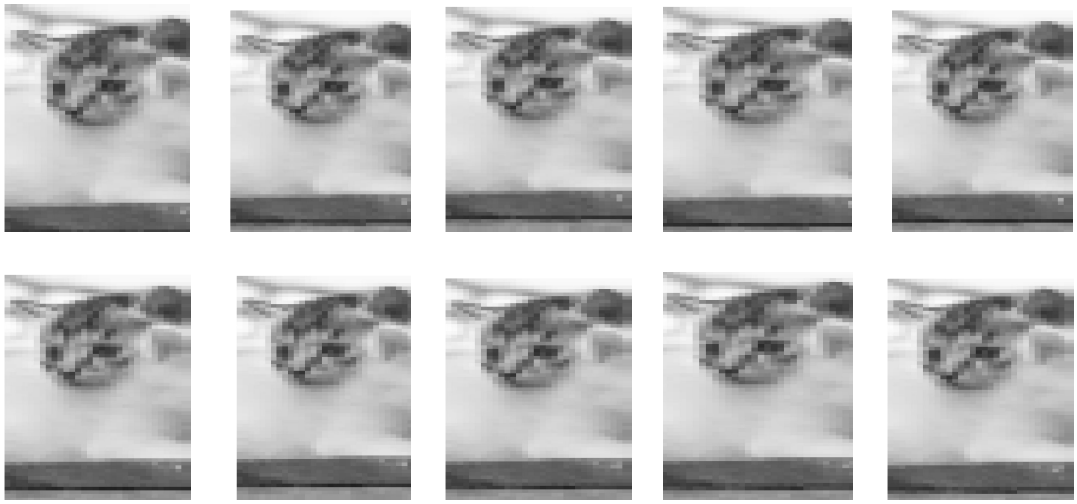

图 2（a）——图 2（j）:二维运动的图像序列

4.图片预处理：导致图像模糊的原因有很多种，对于某些模糊较为严重的图像，在识别前进行适当的预处理将会对识别效果有很大的提升。如果待识别的图像有明显的噪声干扰，对图像进行平滑处理后所获得的匹配效果会比没有平滑处理的效果好很多。同时，有明显噪声干扰的图像通常会有较大的方差，因此在预处理环节，我们会设定一个方差阈值，当待识别图像的方差大于此值时，算法会对待识别的图像进行平滑预处理[3]。

三、模型假设

1.假设只有摄像机运动（移动、变焦等）而场景保持静止。

2.假设运动轨迹上的图像值不变，即只是运动造成图像序列中像素值得改变，而场景光照不变。

3.假设图像上所有的像素点都以某种相同的方式运动。

4.假设每两个相邻帧之内，图像的移动不超过一个矩阵格子。

5.在一定条件下，非直线运动和变速运动都可以看做是由多段的匀速直线运动构成。另外，成像曝光时间一般都非常短，在这期间，可以将拍摄物体与相机间的运动看做为匀速直线运动。

四、符号说明

符号	意义	单位
i	摄像视野中32×64矩形格子的第i行（$1<i<64$）	无
j	摄像视野中32×64矩形格子的第j列（$1<j<32$）	无
k	摄像视野在水平方向移动的速度	像素点/帧
m	摄像视野在竖直方向移动的速度	像素点/帧
n	时间节点（即第n帧）	帧
a_j^n	一维情况下第n帧时，第j列个像素点的灰度值	无
a_{ij}^n	二维情况下第n帧时，第i行j列个像素点的灰度值	无
K	移动时，a_{ij}^n中水平方向上第一个细分块所占的比例	无
M	移动时，a_{ij}^n中竖直方向上第一个细分块所占的比例	无
r	移动过程中，a_{ij}^n水平方向第一个细分块所处的位置	无
s	移动过程中，a_{ij}^n竖直方向第一个细分块所处的位置	无
Q	每个原始像素点进一步细分时的倍数	无

注：未列出符号及重复的符号以出现处为准。

五、模型建立与求解

5.1 模型一

5.1.1 模型一的准备

为了简化模型，我们首先考虑一维情况，即假设摄像视野仅做水平方向上的移动。在此，我们以摄像视野水平向右移动为例进行分析讨论，移动的示意图如下图3（a)和图3（b）所示。

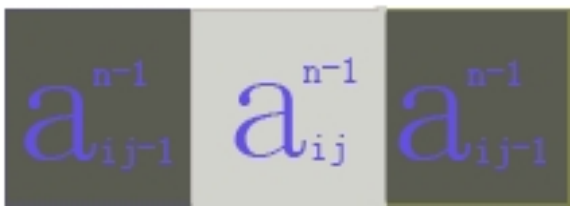

图3（a)：原图像各矩阵格子灰度值

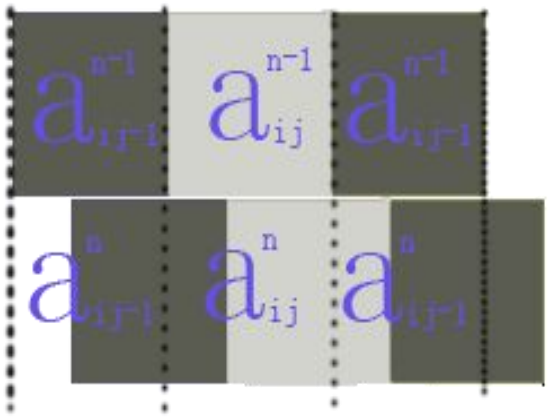

图3（b)：摄像视野水平向右移动时的灰度值影响

为了尽可能地还原模糊图像的信息，我们对原来的视野区域进行进一步的细分，将原来的单个像素点划分为更小的格子，如下图 4 所示。

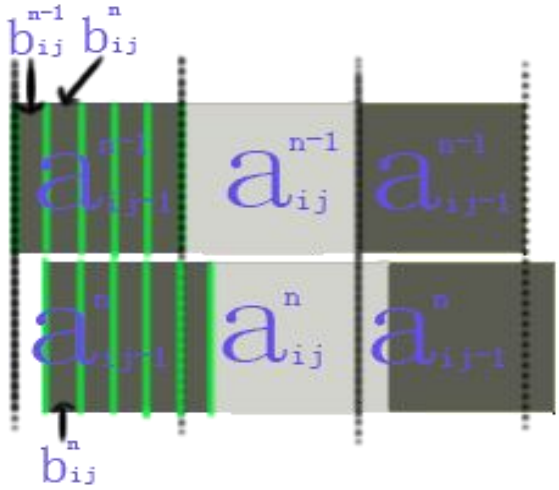

图 4：进一步细分后摄像视野水平向右移动时的灰度值影响

因为图像中每个像素的取值为对应格子的亮度平均值，假设取 n 帧图像，研究区域内原始大小的像素点共有 W 个。根据上图易知，第 n 帧时第一个像素点的灰度值 a_{11}^{n} 可按照如下公式（1）计算。

$$a_1^n = Kb_r + b_{r+1} + b_{r+2} \cdots + b_{r+Q-1} + (1-K)b_{r+Q} \tag{1}$$

其中，b_r 是指细分后第 r 个小像素点的灰度值，K 指 a_j^n 大像素点中水平方向上第一个小像素点所占的比例。因为摄像视野在水平方向移动的速度 k 值不确定，所以移动过程中，a_j^n 的第一个细分块所处的位置 r 计算公式如下所示：

$$r = \mathrm{mod}\left[\frac{floor(k\times(\mathrm{n}-1)\times Q)+1}{Q}\right] \tag{2}$$

再根据移动时，r 与 a_j^n 中水平方向上第一个细分块所占的比例 K 以及每个原始像素点进一步细分时的倍数 Q 之间的关系，K 值可按如下公式（3）所示。

$$K = \frac{r\times\frac{1}{Q} - k(n-1)}{\frac{1}{Q}} \tag{3}$$

根据以上分析，可推导出该行第 j 个像素点的灰度值计算公式为：

$$a_j^n = Kb_{r+(j-1)Q} + b_{r+(j-1)Q+1} + b_{r+(j-1)Q+2} \cdots + (1-K)b_{r+jQ} \tag{4}$$

由题意知，摄像视野缓慢移动，此时的速率不便于衡量。为了计算方便，我们取 $K=1$，即 a_j^n 中水平方向上第一个细分块所占的比例为 1，这表明图像移动总是以整数帧为单位步长，这样公式（4）就可以简化如下：

$$a_j^n = b_{r+(j-1)Q} + b_{r+(j-1)Q+1} + b_{r+(j-1)Q+2} \cdots + b_{r+jQ\text{-}1} \tag{5}$$

通过选取多帧图像进行分析，列些方程组编程求解，即可得到进一步细分之后小像素点的灰度值，因而该模型可对原来模糊图像进行一定的细节复原[4]。

5.1.2 模型一的建立

我们将每一个原始的大像素点划分为 5 个小像素点，根据上述模型一的分析及我们获取的 10 张图片列写方程如下：

$$\begin{cases} a_1^{(2)} = a_1^{(1)} - b_1^{(1)} + b_2^{(1)} \\ a_1^{(3)} = a_1^{(2)} - b_1^{(2)} + b_2^{(2)} \\ a_1^{(4)} = a_1^{(3)} - b_1^{(3)} + b_2^{(3)} \\ a_1^{(5)} = a_1^{(4)} - b_1^{(4)} + b_2^{(4)} \\ a_1^{(6)} = a_1^{(5)} - b_1^{(5)} + b_2^{(5)} \end{cases}$$

$$\begin{cases} a_2^{(2)} = a_2^{(1)} - b_2^{(1)} + b_3^{(1)} \\ a_2^{(3)} = a_2^{(2)} - b_2^{(2)} + b_3^{(2)} \\ a_2^{(4)} = a_2^{(3)} - b_2^{(3)} + b_3^{(3)} \\ a_2^{(5)} = a_2^{(4)} - b_2^{(4)} + b_3^{(4)} \\ a_2^{(6)} = a_2^{(5)} - b_2^{(5)} + b_3^{(5)} \end{cases} \qquad (6)$$

$$\vdots$$

$$\begin{cases} a_{32}^{(2)} = a_{32}^{(1)} - b_{32}^{(1)} + b_{33}^{(1)} \\ a_{32}^{(3)} = a_{32}^{(2)} - b_{32}^{(2)} + b_{33}^{(2)} \\ a_{32}^{(4)} = a_{32}^{(3)} - b_{32}^{(3)} + b_{33}^{(3)} \\ a_{32}^{(5)} = a_{32}^{(4)} - b_{32}^{(4)} + b_{33}^{(4)} \\ a_{32}^{(6)} = a_{32}^{(5)} - b_{32}^{(5)} + b_{33}^{(5)} \end{cases}$$

解得进一步细分之后小像素点的灰度值如下：

$$\begin{bmatrix} b_n^{(1)} \\ b_n^{(2)} \\ b_n^{(3)} \\ b_n^{(4)} \\ b_n^{(5)} \end{bmatrix} = \begin{bmatrix} a_{n-1}^2 - a_{n-1}^1 \\ a_{n-1}^3 - a_{n-1}^2 \\ a_{n-1}^4 - a_{n-1}^3 \\ a_{n-1}^5 - a_{n-1}^4 \\ a_{n-1}^6 - a_{n-1}^5 \end{bmatrix} + \begin{bmatrix} b_{n-1}^{(1)} \\ b_{n-1}^{(2)} \\ b_{n-1}^{(3)} \\ b_{n-1}^{(4)} \\ b_{n-1}^{(5)} \end{bmatrix} \qquad (7)$$

由公式（7）容易看出，只要知道第一帧（第一帧可以在图像序列中任意确定）的五个小像素点的灰度值，再经过逐步迭代求解，即可得到所有小像素点的灰度值。因此第一帧各小像素点灰度值的确定是整个图片信息复原的关键，本文分析、讨论了以下两种方法。

- **特殊像素点选取法**

我们初步设想，寻找图像中灰度值特殊的像素点（纯黑或者纯白），用该点作为第一帧参与计算，但实际图像中我们可能找不到这样的特殊点，所以该方法的移植性和推广性不强。

- **平均值替代法**

根据题意，图像中每个像素的取值为对应格子的亮度平均值。因此对于第一帧的图像，我们可以用大像素点的灰度值（即平均值）来替代每一个小像素点的平均值。为了论证这样取值的可靠性，我们用一个随机数组模拟整张图片的灰度值对该算法进行测试。首先生成一个1×200的随机数组，将其真实值代入公式（7）中求解，以方程组的解（即算法求解值）为纵轴，对比真实值与实测值如下图 5 所示：

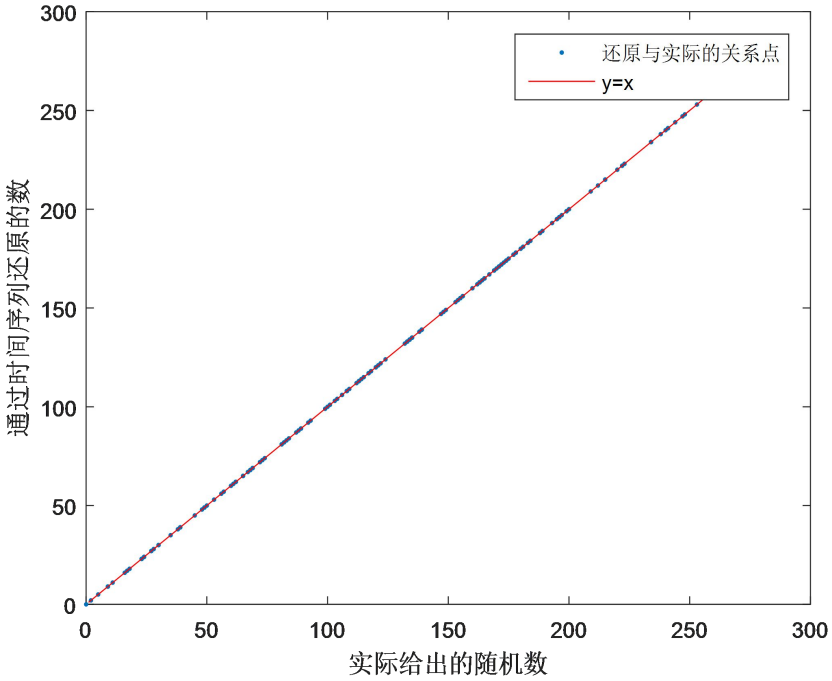

图 5：用随机数的真实值测试该算法的结果

由上图 5 可以看出，真实值与求解值的关系可近似为 $y=x$。这表明：如果代入第一帧各小像素点的真实灰度值作为方程求解真实值，则其余点的求解值与真实值近似相等，也就是说其余点的复原效果会非常好。

为了说明平均值替代法的可行性，我们将前五个数的平均值作为方程求解的初值，以原始随机数组（即真实值）为横轴，以方程组的解（即算法求解值）为纵轴，对比真实值与实测值如下图 6 所示：

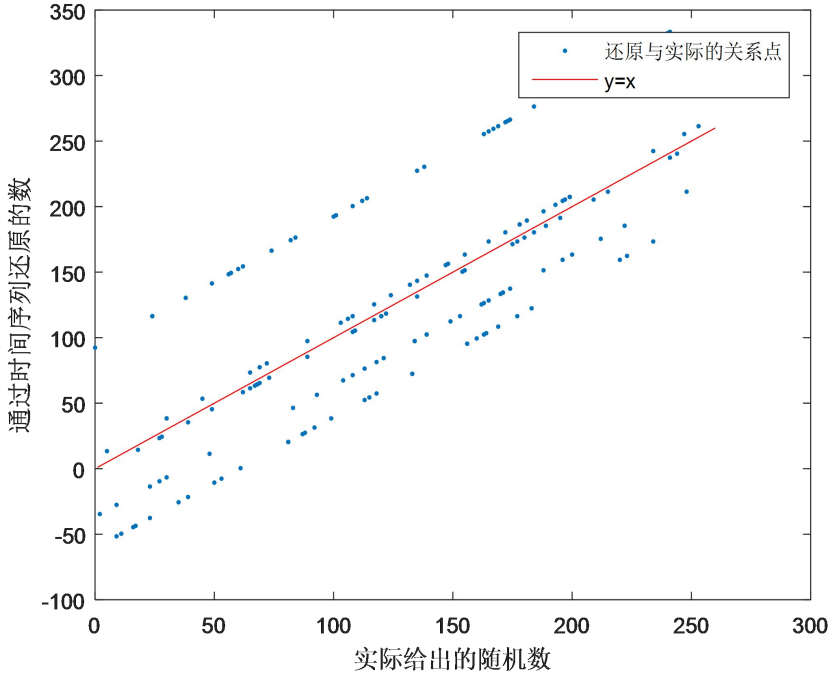

图 6：检验平均值替代法的原始结果

根据图 6 可以看出，真实值与求解值的关系可近似为与 $y=x$ 平行的几条直线，并且方程组的解中出现了不在[0,255]区间范围内的灰度值，这与实际情况是不相符的，因

此对小于 0 和大于 255 的结果需要进行人工调整。调整方法如下：

①对于最小值小于 0 的平行线整体上调，上调幅度由最小值恰好落在 0 值处确定；

②对于最大值大于 255 的平行线整体下调，下调幅度由最大值恰好落在 255 值处确定；

按照上述方法对图 5 表示的求解结果进行调整，调整结果如下图 7 所示。

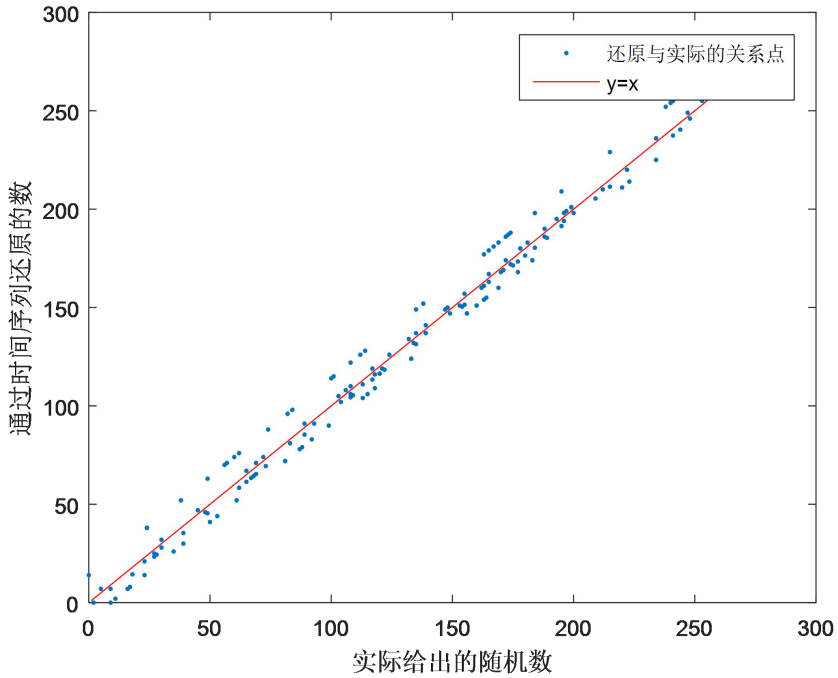

图 7：检验平均值替代法的调整结果

由图 7 知，经过调整之后的测试结果与真实结果近似相等，连线基本分布在 $y=x$ 附近，求解结果较好。这表明：如果将第一帧大像素点的平均值作为方程组求解的初始值，得到的其他点灰度值与真实值非常接近，复原效果很好。

5.1.3 模型一的求解

经过上述论证，我们得到用平均值替代法里获取方程初值这种解法是合理的。因此对于第一帧的图像，我们可以用大像素点的灰度值（即平均值）来替代每一个小像素点的平均值。用模型一对得到的 10 张序列图像进行对比分析，得到原来 32×64 的模糊图像与经过像素点进一步细分后的复原图如下图 8 和图 9 所示。

图 8：原 32×64 的模糊图像

图 9：像素点细分后的复原图

通过对比图 8 和图 9 可明显的看出通过进一步细分像素点的方法，原来低像素的模糊图像可以得到一定程度的复原，效果较好，我们可以观察到原图像更多的细节信息。

5.1.4 模型一的改进

- **Step1：噪声处理**

如果待识别的图像有明显的噪声干扰，对图像进行平滑处理后所获得的匹配效果会比没有平滑处理的效果好很多。为了得到更好效果的复原图像，我们对细分后的复原图做了必要的滤波处理。滤波处理后的复原图如下图 10 所示。

图 10：噪声处理后的复原图

由上图 10 可知，经过噪声处理，复原效果得到了进一步的提高，被拍摄物体的细节信息也得到了更大程度的体现。

- **Step2：纵向插值处理**

因为模型一只是针对水平方向运动的模糊图像进行复原处理，而在竖直方向并没有采取措施进行更多细节信息的还原。为了进一步提高图像的处理效果，我们对得到的水平复原图进行纵向插值。根据每个小像素点上一行与下一行对应位置的灰度值差值计算

得到该点的灰度值，改进后的复原效果如下图 11 所示[5]。

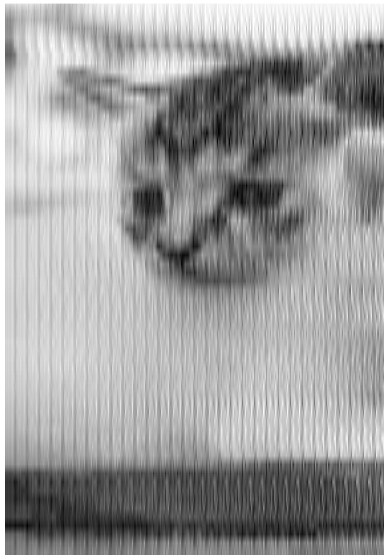

图 11：纵向插值处理后的复原图

由上图 11 可知，再滤波处理之后，进行纵向插值处理可以使复原效果提高，图像竖直方向上的信息体现的更为清晰、明确。

- **Step3：最终滤波处理**

在我们以上各步的水平方向复原步骤以及改进处理中会引入许多不可避免的人为干扰或者算法干扰，采用迭代的方法进行图像复原，在有噪声干扰的情况下，此算法的迭代对图像噪声有放大的功能。所以在图像处理的最后，我们对这些干扰做了最终的中值滤波处理，以求得最大程度上的信息还原，最终中值滤波后的复原效果如下图 12 所示。

图 12：最终中值滤波处理后的复原图

由上图 12 可知，经过最终的中值滤波处理，图像变得更为平滑，相比于原来分辨率为32×64 的模糊原图，经过这一系列的复原措施后，我们可以观察到关于被拍摄物体的更多细节信息，复原效果更好。

5.1.5 模型一的总结

在模型一的建立、求解和改进过程中，我们对原来分辨率为32×64 的模糊原图逐步进行复原，最终得到了很好的处理效果。其中滤波处理和纵向插值处理使效果得到了显著的提高。原模糊图像、模型一复原图像以及改进复原图可对比如下图 13（a）、图 13（b）和图 13（c）所示。

图 13（a）：原 32×64 的模糊图像

图 13（b）：像素点细分后的复原图

图 13（c）:最终滤波处理后的复原图

因此，对于一维情况下的运动模糊图像，我们可以采用该方法进行处理。该模型适用范围广，对原图像的质量要求低，推广性和移植性都很好。

5.2 模型二

5.2.1 模型二的准备

将简单的一维模型推广到二维，我们考虑拍摄视野同时在水平方向和竖直方向均发生了移动，即移动方向存在倾角。我们以摄像视野向右下方向移动为例进行分析讨论，移动的示意图如下图 14（a)和图 14（b）所示，易得图形运动过程中，各像素点之间的关系。

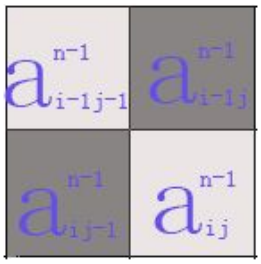

图 14（a)：原图像各矩阵格子灰度值

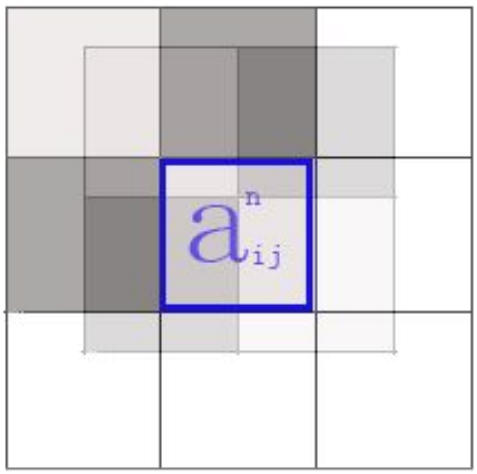

图 14（b)：摄像机向左上方向运动时的灰度值影响

为了尽可能地还原模糊图像的信息，我们对原来的视野区域进行进一步的细分，将原来

的单个像素点划分为更小的格子，如下图 15 所示。

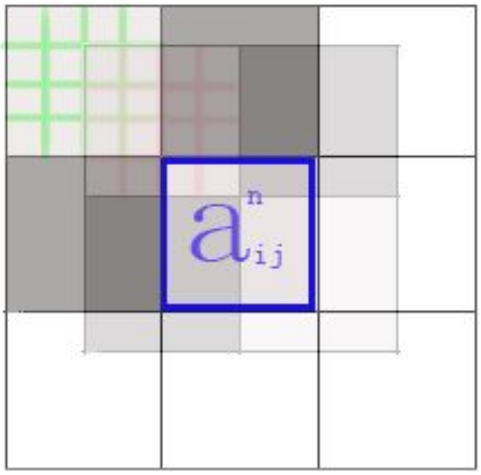

图 15:进一步细分后摄像视野水平向右移动时的灰度值影响

因为图像中每个像素的取值为对应格子的亮度平均值，假设取 n 帧图像，研究区域内原始大小的像素点共有 $W\times H$ 个（横向为 W 块，纵向 H 块，）。根据上图易知，第 n 帧时第一个像素点的灰度值 a_{11}^{n} 可按照如下公式（8）计算。

$$\begin{aligned} a_{11}^{n} = {} & KMb_{r,s} + Mb_{r+1,s} + Mb_{r+2,s}\cdots + Mb_{r+Q-1,s} + (1-K)Mb_{r+Q,s} \\ & + Kb_{r,s+1} + b_{r+1,s+1} + b_{r+2,s+1}\cdots + b_{r+Q-1,s+1} + (1-K)b_{r+Q,s+1} \\ & + Kb_{r,s+2} + b_{r+1,s+2} + b_{r+2,s+2}\cdots + b_{r+Q-1,s+2} + (1-K)b_{r+Q,s+2} \\ & \vdots \\ & + K(1-M)b_{r,s+Q} + (1-M)b_{r+1,s+Q} + (1-M)b_{r+2,s+Q}\cdots + \\ & (1-M)b_{r+Q-1,s+Q} + (1-K)(1-M)b_{r+Q,s+Q} \end{aligned} \tag{8}$$

其中，$b_{r,s}$ 是指细分后第 r 行第 s 列个小像素点的灰度值，K 指 a_{ij}^{n} 大像素点中水平方向上第一个小像素点所占的比例，M 指 a_{ij}^{n} 大像素点中竖直方向上第一个小像素点所占的比例。因为摄像视野在水平方向移动的速度 k 值和竖直方向移动的速度 m 值不确定，所以移动过程中，a_{ij}^{n} 的第一个细分块所处的位置 r 计算公式如下所示：

$$r = \mathrm{mod}\left[\frac{floor\left(k\times(\mathrm{n}-1)\times Q\right)+1}{Q}\right] \tag{9}$$

s 的计算公式如下所示：

$$s = \mathrm{mod}\left[\frac{floor\left(m\times(\mathrm{n}-1)\times Q\right)+1}{Q}\right] \tag{10}$$

再根据移动时，r 与 a_{ij}^{n} 中水平方向上第一个细分块所占的比例 K 以及每个原始像素点进一步细分时的倍数 Q 之间的关系，K 值可按如下公式（11）计算：

$$K = \frac{r\times\frac{1}{Q} - k(n-1)}{\frac{1}{Q}} \tag{11}$$

同理可得，M 值可按如下公式（12）计算：

$$M=\frac{s\times\frac{1}{Q}-m(n-1)}{\frac{1}{Q}} \tag{12}$$

根据以上分析，可推导出该行第 i 行第 j 列像素点的灰度值计算公式为：

$$\begin{aligned}a_{ij}^{n}=&KMb_{r+(j-1)Q,s+(i-1)Q}+Mb_{r+(j-1)Q+1,s+(i-1)Q}+Mb_{r+(j-1)Q+2,s+(i-1)Q}\cdots+Mb_{r+jQ-1,s+(i-1)Q}+(1-K)Mb_{r+jQ,s+(i-1)Q}\\&+Kb_{r+(j-1)Q,s+(i-1)Q+1}+b_{r+(j-1)Q+1,s+(i-1)Q+1}+b_{r+(j-1)Q+2,s+(i-1)Q+1}\cdots+b_{r+jQ-1,s+(i-1)Q+1}+(1-K)b_{r+jQ,s+(i-1)Q+1}\\&+Kb_{r+(j-1)Q,s+(i-1)Q+2}+b_{r+(j-1)Q+1,s+(i-1)Q+2}+b_{r+(j-1)Q+2,s+(i-1)Q+2}\cdots+b_{r+jQ-1,s+(i-1)Q+2}+(1-K)b_{r+jQ,s+(i-1)Q+2}\\&\vdots\\&+K(1-M)b_{r+(j-1)Q,s+iQ}+(1-M)b_{r+(j-1)Q+1,s+iQ}+(1-M)b_{r+(j-1)Q+2,s+iQ}\cdots+(1-M)b_{r+jQ-1,s+iQ}\\&+(1-K)(1-M)b_{r+jQ,s+iQ}\end{aligned} \tag{13}$$

由题意知，摄像视野缓慢移动，此时的速率不便于衡量。为了计算方便，我们取 $K=1$，$M=1$。即 a_{ij}^{n} 中水平方向上第一个细分块所占的比例为 1，竖直方向上第一个细分块所占的比例也为 1，这表明图像移动总是以整数帧为单位步长，这样公式（13）就可以简化如下：

$$\begin{aligned}a_{ij}^{n}=&KMb_{r+(j-1)Q,s+(i-1)Q}+Mb_{r+(j-1)Q+1,s+(i-1)Q}+Mb_{r+(j-1)Q+2,s+(i-1)Q}\cdots+Mb_{r+jQ-1,s+(i-1)Q}\\&+Kb_{r+(j-1)Q,s+(i-1)Q+1}+b_{r+(j-1)Q+1,s+(i-1)Q+1}+b_{r+(j-1)Q+2,s+(i-1)Q+1}\cdots+b_{r+jQ-1,s+(i-1)Q+1}\\&+Kb_{r+(j-1)Q,s+(i-1)Q+2}+b_{r+(j-1)Q+1,s+(i-1)Q+2}+b_{r+(j-1)Q+2,s+(i-1)Q+2}\cdots+b_{r+jQ-1,s+(i-1)Q+2}\\&\vdots\\&+Kb_{r+(j-1)Q,s+iQ-1}+b_{r+(j-1)Q+1,s+iQ-1}+b_{r+(j-1)Q+2,s+iQ-1}\cdots+b_{r+jQ-1,s+iQ-1}+b_{r+jQ,s+iQ-1}\end{aligned} \tag{14}$$

通过选取多帧图像进行分析，列些方程组编程求解，即可得到进一步细分之后小像素点的灰度值，因而该模型可对原来模糊图像进行一定的细节复原[6][7]。

5.2.2 模型二的建立

我们将每一个原始的大像素点划分为9×9个小像素点，根据上述模型二的分析及我们获取的 10 张图片列写迭代方程如下：

$$81\times a_{\frac{\lceil i+1\rceil}{9}\frac{\lceil j+1\rceil}{9}}^{ii}-\sum_{r=j+1}^{j+8}b_{i+9,r}-\sum_{r=i+1}^{i+9}b_{r,j+9}=81\times a_{\frac{\lceil i+1\rceil}{9}\frac{\lceil j+1\rceil}{9}}^{ii-1}-\sum_{r=j}^{j+8}b_{i-1,r}-\sum_{r=i+1}^{i+8}b_{r,j-1}\quad(\text{其中}ii=i+1) \tag{15}$$

解得进一步细分之后小像素点的灰度值如下：

$$b_{i+9,j+9}=81\times a_{\frac{\lceil i+1\rceil}{9}\frac{\lceil j+1\rceil}{9}}^{ii}-\sum_{r=j+1}^{j+8}b_{i+9,r}-\sum_{r=i+1}^{i+8}b_{r,j+9}-81\times a_{\frac{\lceil i+1\rceil}{9}\frac{\lceil j+1\rceil}{9}}^{ii-1}+\sum_{r=j}^{j+8}b_{i-1,r}+\sum_{r=i+1}^{i+8}b_{r,j-1} \tag{16}$$

如果 $ii>10$，令 $ii=ii-9$ 再计算。由公式（16）容易看出，只要知道第一帧（第一帧可以在图像序列中任意确定）的9×9个小像素点的灰度值，再经过逐步迭代求解，即可得到所有小像素点的灰度值。根据模型一的分析和讨论，我们同样将第一帧大像素点的平均值作为方程组求解的初始值，从而得到的其他点灰度值，来复原模糊图像的信息。

5.2.3 模型二的求解

根据上述模型二的准备与建立，我们采用大像素点的灰度值（即平均值）来替代每一个小像素点的平均值。用该算法对得到的 10 张序列图像进行对比分析，得到原来

32×64 的模糊图像与经过像素点进一步细分后的复原图如下图 16 和图 17 所示。

图 16：原 32×64 的模糊图像

图 17：像素点细分后的复原图

通过对比图 16 和图 17 可明显的看出通过进一步细分像素点的方法，原来低像素的模糊图像可以得到一定程度的复原，效果较好，我们可以观察到原图像更多的细节信息。

5.2.4 模型二的改进

在我们以上各步的水平方向复原步骤以及改进处理中会引入许多不可避免的人为干扰或者算法干扰，采用迭代的方法进行图像复原，在有噪声干扰的情况下，此算法的迭代对图像噪声有放大的功能。所以在图像处理的最后，我们对这些干扰做了最终的中值滤波处理，以求得最大程度上的信息还原，最终中值滤波后的复原效果如下图 18 所示。

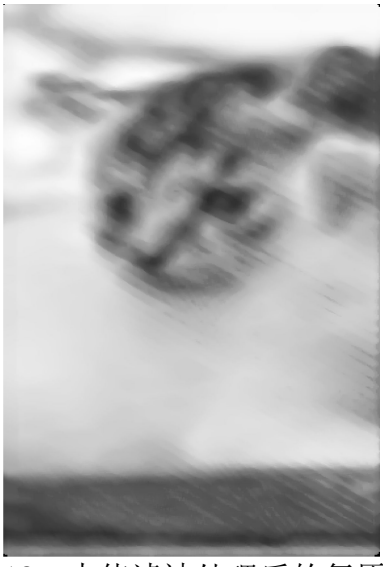

图 18：中值滤波处理后的复原图

由上图 18 可知，经过最终的中值滤波处理，图像变得更为平滑，相比于原来分辨率为32×64 的模糊原图，经过这一系列的复原措施后，我们可以观察到关于被拍摄物体的更多细节信息，复原效果更好。

5.2.5 模型二的总结

在模型二的建立、求解和改进过程中，我们对原来分辨率为32×64的模糊原图逐步进行复原，最终得到了很好的处理效果。其中最终的中值滤波处理使复原效果得到了显著的提高。

因此，对于二维情况下的运动模糊图像，我们可以采用模型二的方法进行处理。该模型适用范围广，对原图像的质量要求低，推广性和移植性都很好。

六、模型评价与展望

6.1 模型的优点

- 该模型适合低分辨率识别，可以应用于夜晚或光线较暗的地方。
- 用此种方法既可以提升系统的灵敏度，而且还能降低边缘模糊误差。
- 我们的模型适用范围广，可移植性好。现实生活中的模糊图像的模糊成因往往非常复杂，而当前有关模糊图像处理的算法大都只能够处理某一类因素所导致的模糊图像，当这些算法处理其他图像时，很可能会导致图像更进一步的模糊。以“去雾算法”为例，对于那些有雾的图像，“去雾算法”能够取得很好的处理效果，但是当这些算法作用于正常图像的时候，反而会使得图像效果下降。

6.2 模型的缺点

- 因为在实际拍摄视频过程中不能避免摄影机的抖动问题，这样会有许多不可控的干扰噪声。
- 算法参数的复杂性。在我们的模型中，参数的选取直接决定了最终的处理效果，而且这些参数的选择取决于实际的模糊图像，当前的算法还无法智能地选择相关的最优参数，很大程度上要依赖于人的经验。

6.3 模型的展望

- **探索相关学科的交叉融合**

将传统的光学理论与正在发展的数字图像处理方法相结合，利用计算机对运动模糊图像进行复原，进一步提高运动模糊图像的复原精度，从而降低在拍摄过程中对光学设备精度和拍摄人员的要求。

- **完善特征提取过程**

图像的特征提取是所有相关视频和图像处理的核心问题，然而如何鲁棒提取图像特征仍有待研究。近年来，研究人员从生物角度出发，使用计算机模拟人脑处理视觉信息的过程,从而有效提取视觉特征，获得了很好的效果，这也是特征提取的发展趋势。将合理的生物学上的特征引入到动态图像序列描述模型中，可提高准确性和鲁棒性。

- **使用新兴的图像复原算法**

（1）神经网络图像复原算法：伴随着神经网络研究的发展，神经网络在图像复原处理中也得到了应用。基于神经网络的图像复原方法大体可以分为两类：

①基于 Hopfield 神经网络的图像复原， 将图像复原问题转化为极小值的问题来处理，再映射为 Hopfield 的能量函数，从而利用 Hopfield 网络求解最优问题。

②运用大量的原图与模糊图像进行学习训练，再利用训练后的网络进行图像复原。神经网络算法能够通过输入数据自适应找到隐含在样本中的内在规律，使其具有很高的推广能力，但是算法的实施关键在于怎么处理数据与网络函数的映射。

（2）图像超分辨率复原技术：这种复原技术是指利用多帧低分辨率图像，求解成像的逆过程，重建原图的高分辨率图像。这种技术能够在不改变成像设备硬件的前提下实现优于系统分辨率的观测。超分辨率重建过程可分为三步：

①预处理，即去噪；

②配准，即对低分辨率序列间的矢量进行估计；

③重建，即把多帧低分辨率信息融合在一起。

- **图像复原问题发展方向**

图像复原问题的未来研究方向将主要围绕**参数识别**和**复原滤波**两方面展开，参数识别问题以后会向着增加先验知识的方向发展，而复原滤波算法的研究未来会以去除噪声与图像恢复相结合做为研究的重点，做到在图像复原的过程中不引入任何的噪声。这是对传统复原算法改进的方向。还有一种发展方向就是开发新的复原算法，特别是神经算法的出现为图像复原提供了新的研究思路，也为以后的研究打开了新的方向。另外传统的图像复原技术能够分析模糊的原因，但是只能将频率复原到衍射极限相应的截止频率处，而截止频率外的信息将丢失。现在的超分辨率复原方法却能恢复丢失的信息，这将是未来图像复原研究的另一重要方向[8]。

七、参考文献

[1] 赵帅.基于 SIFT 算子的模糊图像识别算法[D].苏州:苏州大学，2014 年 05 月.

[2] 赵鹏，曹军，韦兴竹.匀速直线运动模糊图像的模糊参数鲁棒识别[J].光学 精密工程，2013 年 09 月，21(9):2430—2438.

[3] 赵艳.带噪声离焦模糊图像复原[D].西安:西安电子科技大学，2014 年 12 月.

[4] 罗来接.运动模糊图像恢复理论分析与实现[D].南昌:南昌大学机电工程学院，2014 年 05 月.

[5] 朱婷婷.图像插值超分辨率重建算法研究[D].成都:西南交通大学，2013 年 05 月.

[6] 孙艳丽.运动模糊图像的恢复与处理[D].大连:大连海事大学，2008 年 03 月.

[7] 程姝,赵志刚,吕慧显,潘振宽,郝鑫鑫.顺序结构的运动模糊图像复原技术综述[J].计算机应用，2013，06，33(SI):161 — 165，185.

[8] 王玉全，隋宗宾.运动模糊图像复原算法综述[J].微型机与应用，2014，33(19):54—57.

八、附录

8.1 模型一求解源代码

```
 %%
%一维复原图像
%
%%
%得到水平移动的图
clear all
pic = imread('model2.jpg');
gpic=rgb2gray(pic);
%imshow(gpic)
[hight width]=size(gpic);
lb=int16(hight/64);
t=1;
prik=1/double(lb);
```

参赛队号 # 1430

```
while t<(width-double(lb)*32)/prik/double(lb)
    for j=1:64
        for i=1:32
            block =
gpic((j-1)*lb+1:j*lb,(i-1)*lb+1+(t-1)*prik*lb:i*lb+(t-1)*prik*lb);
            avr = sum(sum(block));
            pic32(j,i) = avr/double(lb)/double(lb);
        end
    end
    pic32 = uint8(pic32);
    %imshow(pic32);
    p{t}=pic32;
    t=t+1;
end

%------show picture
t=140;
%gca=figure(1)
while t<140+lb+1
    imshow(p{t});
    t=t+1;
    pause(0.01);
end
%-----------------
%开始复原图像
t=140;
tt=140;
xmax=zeros(1,lb);
xmin=255*ones(1,lb);
for j=1:64
    %初始化 x
    for r=1:lb
        for s=1:lb
            x{j,1}(s,r)=double(p{t}(j,1));
        end
    end
%     for r=1:lb
%         for s=1:lb
%             x{j,1}(s,r)=gpic((j-1)*lb+s,140+r);
%         end
%     end
    for i=2:32
        for r=1:lb
            for s=1:lb
```

```
x{j,i}(s,r)=x{j,i-1}(s,r)+(double(p{tt+r}(j,i-1))-double(p{tt+r-1}(j,i-1)))
/prik;
                if x{j,i}(s,r)>xmax(r)
                    xmax(r)=x{j,i}(s,r);
                end
                if x{j,i}(s,r)<xmin(r)
                    xmin(r)=x{j,i}(s,r);
                end
            end
        end
    end
    %--------------------滤波----------------
    for r=1:lb
        for s=1:lb
            if xmax(r)>255
                for i=1:32
                    x{j,i}(s,r)=x{j,i}(s,r)-xmax(r)+255;
                end
            end
            if xmin(r)<0
                for i=1:32
                    x{j,i}(s,r)=x{j,i}(s,r)-xmin(r);
                end
            end
        end
    end
    %----------------------------------------
end
%----------------------插值-----------------
for j=1:63
    for i=1:32
        for r=1:lb
            buff=interp1([1 double(lb)+1],[x{j,i}(1,r)
x{j+1,i}(1,r)],1:double(lb));
            x{j,i}(:,r)=buff';
        end
    end
end

recpic=cell2mat(x);
recpic=uint8(recpic);
imshow(recpic);
figure
```

```
imshow(p{140});
```

8.2 模型二求解源代码

```
%%
%二维复原图像
%
%%
%得到斜着移动的图
clear all
pic = imread('model2.jpg');
%转化为灰度图
gpic=rgb2gray(pic);
%imshow(gpic)
[hight width]=size(gpic);
% lb=int16(hight/64/2);
%定义精度
lb=int16(9);
%定义时间序列的图像移动的方向的速度 k 和 m
prik=1/double(lb);
prim=prik;
t=1;
%模拟得到 20 张时间序列的图像
while t<20
    for j=1:64
        for i=1:32
            block =
gpic((j-1)*lb+1+(t-1)*lb*prim:j*lb+(t-1)*lb*prim,(i-1)*lb+161+(t-1)*lb*prik:
i*lb+160+(t-1)*lb*prik);
            avr = sum(sum(block));
            pic32(j,i) = avr/double(lb)/double(lb);
        end
    end
    pic32 = uint8(pic32);
    %imshow(pic32);
    p{t}=pic32;
    t=t+1;
end
t=1;
%gca=figure(1)
while t<20
    imshow(p{t});
    t=t+1;
    pause(0.01);
end
```

参赛队号 # 1430

```
% for i=1:10
%     figure
%     imshow(p{i})
% end
%%
%开始复原图像

%已有的时间序列图像为
%   p{1}.....p{lb+1}
%   对 p{1}进行还原
x=zeros(64*lb,32*lb);
%初始化 x
for j=1:64*double(lb)
    for i=1:32*double(lb)
        if j<=lb || i<=lb
            %利用平均值代替原先的初始值
            x(j,i)=p{1}(ceil(j/double(lb)),ceil(i/double(lb)));
%             x(j,i)=gpic(j,i+160);
        end
    end
end
%复原
prix=imresize(p{1},9);
for j=2:63*double(lb)
    for i=2:31*double(lb)
        jj=j;
        if jj>10;
            jj=2;
        end
        %核心公式，由 lb^2-1 个量，得到第 lb^2 个变量的值
        %并且经过循环迭代，得到扩展后的所有的像素点

x(j+lb-1,i+lb-1)=double(p{jj}(ceil(j/double(lb)),ceil(i/double(lb))))*doubl
e(lb)*double(lb)...

-double(p{jj-1}(ceil(j/double(lb)),ceil(i/double(lb))))*double(lb)*double(l
b)...

-sum(x(j+lb-1,i:i+lb-2))-sum(x(j:j+lb-2,i+lb-1))+sum(x(j-1,i-1:i+lb-2))+sum
(x(j:j+lb-2,i-1));
        %限幅滤波
        x(j+lb-1,i+lb-1)=uint8(x(j+lb-1,i+lb-1));
        x(j+lb-1,i+lb-1)=double(x(j+lb-1,i+lb-1));
    end
```

```
    %求得复原的趋势点与模糊图的点的差
    deltax=double(x(j,:))-double(prix(j,:));
    %对原模糊图像进行补偿
    x(j,:)=double(prix(j,:))+deltax./255.*30;
end
x=uint8(x);
%由于未对最后一行进行清晰化，因此截掉最后一行
x=x(1:63*double(lb),1:32*double(lb));
%展示清晰化的图像
imshow(x)
%对清晰化后的图像进行中值滤波
recpic=medfilt2(x,[9 9]);
figure
%展示中值滤波后的图像
imshow(recpic);
figure
%显示原图
imshow(p{1});
```

8.3 用随机数对模型进行测试的源代码

```
%%
%测试
%使用一行随机数进行测试
%复原的也只有一行

%生成 200 个随机数进行测试（实际上只用到了 192 个）
a=uint8(rand(1,200)*255);
%定义时间序列移动的速度大小，为 0.2 张/帧
k=0.2;
%求得测试数据的随机数列的数组

%1 时刻
for i=1:32
    aa1(i)=sum(a((i-1)*5+1:5*i));
end
%2 时刻
for i=1:32
    aa2(i)=sum(a((i-1)*5+2:5*i+1));
end
%3 时刻
for i=1:32
    aa3(i)=sum(a((i-1)*5+3:5*i+2));
end
%4 时刻
```

参赛队号 # 1430

```
for i=1:32
    aa4(i)=sum(a((i-1)*5+4:5*i+3));
end
%5 时刻
for i=1:32
    aa5(i)=sum(a((i-1)*5+5:5*i+4));
end
%6 时刻
for i=1:32
    aa6(i)=sum(a((i-1)*5+6:5*i+5));
end
%定义复原图像的元胞类型，并初始化为[0 0 0 0 0]
for i=1:32
    x{i}=zeros(1,5);
end

%给定初始值 x{1},使用第一个时间序列的第一个元素的平均值
x{1}(1)=k*aa1(1);
x{1}(2)=k*aa1(1);
x{1}(3)=k*aa1(1);
x{1}(4)=k*aa1(1);
x{1}(5)=k*aa1(1);

%给定初始值 x{1}，使用实际准确的 5 个值进行验证
%（此部分用于验证，因为我们在应用过程中是不知道实际值的，因此此部分使用实际
值为了验证模型是否合理可行）
% x{1}(1)=a(1);
% x{1}(2)=a(2);
% x{1}(3)=a(3);
% x{1}(4)=a(4);
% x{1}(5)=a(5);
% 由初始值 x{1}求得所有的值
for i=2:32
    %算法核心，用过初始值得到下一时刻的值，并循环得到全部的值
    x{i}(1)=x{i-1}(1)+aa2(i-1)-aa1(i-1);
    x{i}(2)=x{i-1}(2)+aa3(i-1)-aa2(i-1);
    x{i}(3)=x{i-1}(3)+aa4(i-1)-aa3(i-1);
    x{i}(4)=x{i-1}(4)+aa5(i-1)-aa4(i-1);
    x{i}(5)=x{i-1}(5)+aa6(i-1)-aa5(i-1);
end

%-----------------------------------------------
%滤波，考虑到实际值是不知道的，但是在值出现小于 0 或大于 255 的时候是不合理的，
%    可以将这不合理的点经过平移，得到更接近实际的点
```

参赛队号 # 1430

```
%定义最小最大值
xmax=zeros(1,5);
xmin=zeros(1,5);
%求得每个对应时间的最小最大值
for i=1:32
    for j=1:5
        if xmax(j)<x{i}(j)
            xmax(j)=x{i}(j);
        end
        if xmin(j)>x{i}(j)
            xmin(j)=x{i}(j);
        end
    end
end

%利用最小最大值进行相关补偿，
for i=1:5
    if xmax(i)>255
        for j=1:32
            x{j}(i)=x{j}(i)-(xmax(i)-255);
        end
    end
    if xmin(i)<0
        for j=1:32
            x{j}(i)=x{j}(i)-xmin(i);
        end
    end
end

%-----------------------------------------

%得到恢复后的点的序列
reca=[];
for i=1:32
    reca=[reca x{i}];
end
%画出原来的点
plot(a,'r')
hold on
%画出恢复后的点
plot(reca,'g')
hold off
```

参赛队号 # 1430

```
%----------------------由于上面的图像展示不直观
%----------------------使用原图点作为 x,得到的点作为 y，与 y=x 进行比对
plot(a(1:160),reca,'.')
hold on
plot(1:260,1:260,'r')
legend('还原与实际的关系点','y=x')
hold off
xlabel('实际给出的随机数')
ylabel('通过时间序列还原的数')
```