

运动模糊建模

假设物体 $f(x, y)$ 进行平面运动， $x_0(t)$ 和 $y_0(t)$ 分别是在 x 和 y 方向上随时间变化的运动参数，相机曝光时间为 Δt 。运动模糊得到的像 $g(x, y)$ 满足

$$\begin{aligned} g(x, y) &= \int_0^{\Delta t} f[x - x_0(t), y - y_0(t)] dt \\ &= \int_0^{\Delta t} f(x, y) \otimes \delta[x - x_0(t), y - y_0(t)] dt \\ &= f(x, y) \otimes \int_0^{\Delta t} \delta[x - x_0(t), y - y_0(t)] dt \end{aligned} \tag{1}$$

假设图像只在 x 方向运动，且速度均匀， $x_0(t) = a \frac{t}{\Delta t}$ ，则卷积核为

In[]:=

kerformula = Assuming[a > 0 && Δt > 0 && x ∈ Reals,

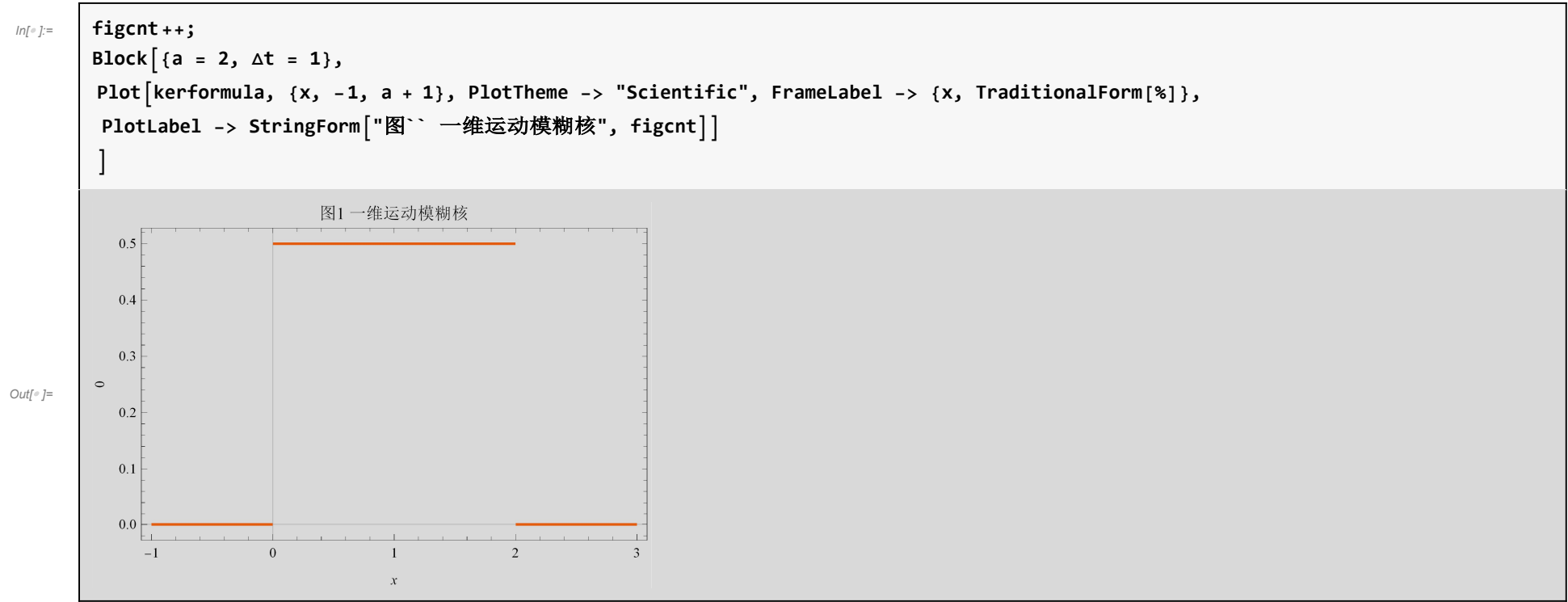
$\int_0^{\Delta t} \text{DiracDelta}\left[x - \frac{a t}{\Delta t}\right] dt$

]

Out[]:=

$$\frac{\Delta t \text{HeavisideTheta}[a - x] \times \text{HeavisideTheta}[x]}{a}$$

其图像大致为



即矩形脉冲。

相仿地，对于二维的，离散的情况，运动模糊的卷积核为对应平面上的运动轨迹

In[]:=

getWaypoints[path:{{_Integer,_}..}]/;AllTrue[path[[All,1]],GreaterThan[1]]:=
Reap[(Composition@Table
With[{len=step[[1]],θ=step[[2]]},
Nest[Sow[#+{-Sin[θ],Cos[θ]}]&,Sow[#,len-1]&
],
{step,Reverse@path}
])@{0,0}
]//Last//First//Round//DeleteDuplicates
getWaypoints[___]=\$Failed;

In[]:=

motionMatrix[onpath: {_Integer,_}]:=motionMatrix[{onpath}]
motionMatrix[path:{{_Integer,_}..}]:=
With[{waypoints=getWaypoints[path]},
With[{
halfw=Max@Abs@waypoints[[All,1]],
halfh=Max@Abs@waypoints[[All,2]],
ReplacePart
ConstantArray[0,2{halfw,halfh}+1],
(#+{halfw,halfh}+1&/@waypoints)->1/Length[waypoints]
]
]//FreeQ[waypoints,\$Failed]
]
motionMatrix[___]=\$Failed;

运动模糊复原

测试图片：

```
In[ ]:= img=
```



```
;
```

Mathematica的文档¹中给出了可用的反卷积方法：

- 频谱反卷积方法的可能设置为：

"DampedLS"	阻尼最小二乘，广义 Tikhonov 正则化
"Tikhonov"	Tikhonov 正则化方法
"TSVD"	截断奇异值分解
"Wiener"	Wiener 反卷积
- 基于迭代的反卷积方法的可能设置为：

"Hybrid"	Tikhonov–Golub–Kahan 双对角正则化
"RichardsonLucy"	Richardson–Lucy 迭代反卷积
"SteepestDescent"	改进的残差范数最速下降
"TotalVariation"	迭代全变差正则化

```
In[ ]:= methods={  
  "DampedLS","Tikhonov","TSVD","Wiener",  
  "Hybrid","RichardsonLucy","SteepestDescent","TotalVariation"};
```

A. ¹ <http://reference.wolfram.com/language/ref/ImageDeconvolve.html>

多次运动模糊

多次运动模糊的物理操作是对模糊的像再次拍摄结果又进一步模糊了，其对应的数学模型是对图像复合多个运动模糊核。
随机生成一组单一速度的运动模糊核

In[]:=

```
movePath = BlockRandom[{RandomInteger[{5, 10}, 3], RandomReal[2  $\pi$ , 3]}T, RandomSeeding  $\rightarrow$  1346]
kers = motionMatrix /@ movePath;
```

Out[]:=

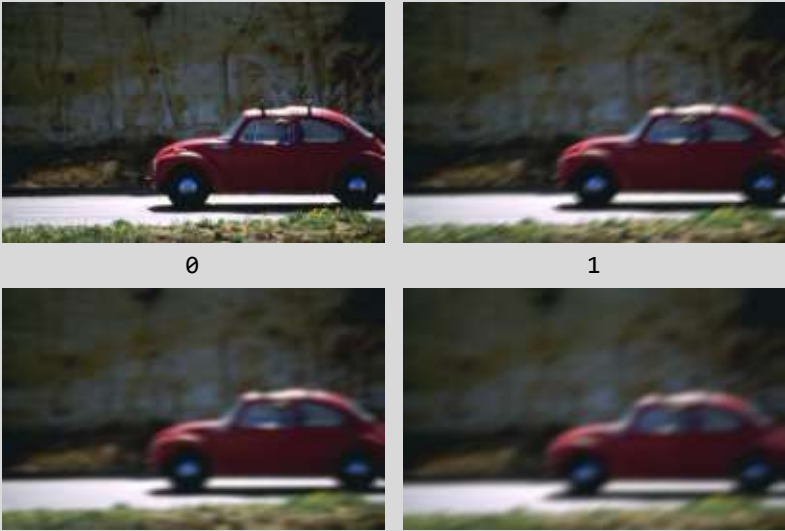
```
{ {5, 0.485519}, {5, 2.78131}, {8, 3.69632} }
```

得到累次进行运动模糊的复合结果

In[]:=

```
blured = ComposeList[
  Table[With[{ker = ker},
    ImageConvolve[#, ker, Padding  $\rightarrow$  "Reversed"] &],
    {ker, kers}
  ],
  img
];
Multicolumn[MapIndexed[Labeled[#1, First@#2 - 1] &, blured], 2, Appearance  $\rightarrow$  "Horizontal"]
```

Out[]:=



对各次运动模糊结果进行反卷积复原

In[]:=

```
deblured = ParallelTable[
  ImageClip@* (Composition@@ Table[With[{ker = ker},
    ImageDeconvolve[#, ker, Method  $\rightarrow$  method] &],
    {ker, kers[[i ;; 1 ;; -1]]}
  ])@blured[[i + 1]],
  {method, methods}, {i, 3}
];
tabcnt++;
Labeled[#, StringForm["表` 模糊次数及反卷积算法效果对比", tabcnt], Top] &@
TableForm[
  Map[Labeled[#, TemplateApply["MSE = `"],
    ImageDistance[img, #, DistanceFunction  $\rightarrow$  "MeanSquaredEuclideanDistance"]]
  ] &, deblured, {2}],
  TableHeadings  $\rightarrow$  {methods, StringTemplate["模糊 ` 次"] /@ Range[3]},
  TableAlignments  $\rightarrow$  Center
]
```

Out["j"] =

























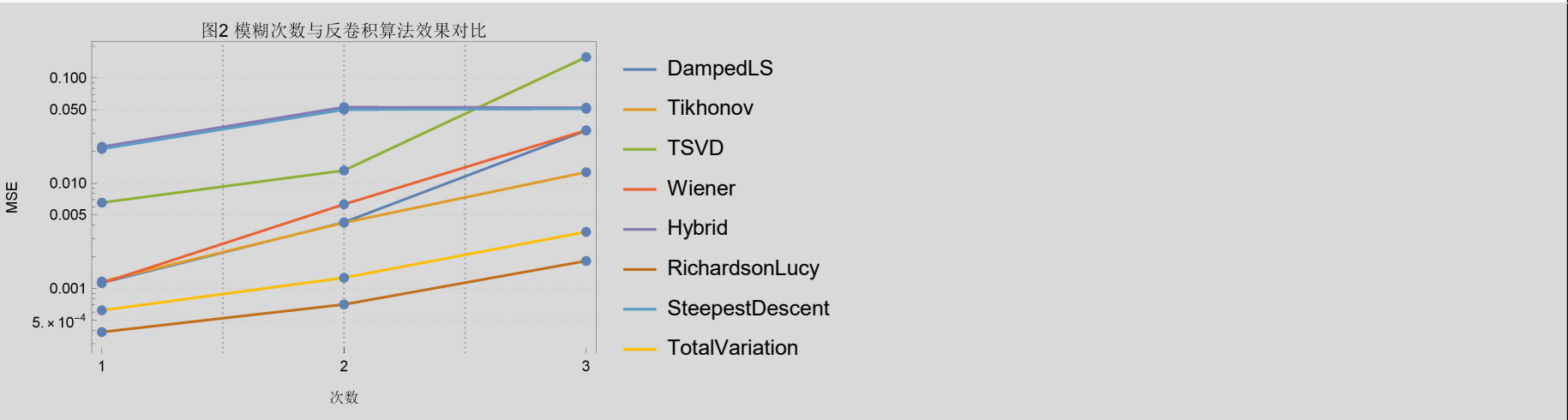
表1 模糊次数及反卷积算法效果对比			
	模糊 1 次	模糊 2 次	模糊 3 次
DampedLS Tikhonov TSVD Wiener Hybrid RichardsonLucy SteepestDescent TotalVariation	 MSE = 0.00113632	 MSE = 0.00426599	 MSE = 0.0316975
	 MSE = 0.00116881	 MSE = 0.00422802	 MSE = 0.0127605
	 MSE = 0.00655382	 MSE = 0.0132441	 MSE = 0.157956
	 MSE = 0.0011275	 MSE = 0.00631919	 MSE = 0.0318651
	 MSE = 0.0222356	 MSE = 0.052875	 MSE = 0.0522343
	 MSE = 0.000387313	 MSE = 0.000707794	 MSE = 0.00183211
	 MSE = 0.0212276	 MSE = 0.050053	 MSE = 0.0510021
	 MSE = 0.000622372	 MSE = 0.00126805	 MSE = 0.00346048

表1给出了模糊次数及反卷积算法直观的效果对比，并给出了各复原图像与原图像间的均方误差（MSE）。从表中可以明显看到截断奇异值分解（TSVD）算法在模糊3次时出现了剧烈的失真，阻尼最小二乘（DampedLS）算法也有类似的现象。下面的图2和图3则将均方误差作为参考进行量化比较。

In[]:=

```
figcnt++;
ListLogPlot[Map[
  ImageDistance[img, #, DistanceFunction -> "MeanSquaredEuclideanDistance"] &,
  deblured, {2}],
PlotTheme -> "Detailed", PlotLegends -> methods, DataRange -> {1, 3}, Joined -> True, Mesh -> Full,
FrameLabel -> {"次数", "MSE"}, FrameTicks -> {{Automatic, None}, {Range[3], None}},
PlotLabel -> StringForm["图` ` 模糊次数与反卷积算法效果对比", figcnt]
]
```

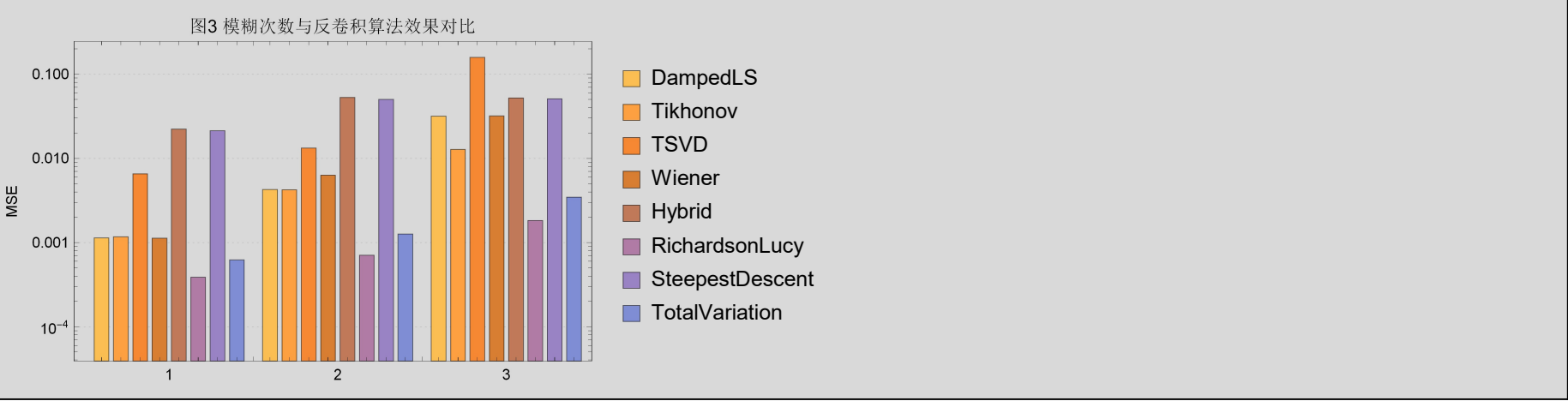
Out[]:=



In[]:=

```
figcnt++;
BarChart[Map[
  ImageDistance[img, #, DistanceFunction -> "MeanSquaredEuclideanDistance"] &,
  deblured, {2}]]^T,
ScalingFunctions -> "Log10", ChartLegends -> methods, ChartLabels -> {Range[3], None},
PlotTheme -> "Detailed", FrameLabel -> {None, "MSE"},
PlotLabel -> StringForm["图` ` 模糊次数与反卷积算法效果对比", figcnt]
]
```

Out[]:=



从图2可以看到，随着模糊作用的次数增加，相应反卷积的效果基本逐渐下降。同样，从表1中也可以明显地看出，无论哪种方法，模糊累积了3次后再进行复原都出现了较为明显的失真。

从图2和图3都可以比较明显地看到Richardson-Lucy迭代算法效果是各种算法中最好的。

一次曝光中速度变化多次

一次曝光中速度变化多次对应的数学模型是单个模糊核与图像卷积，只是卷积核的形状表现为平面上折线段，对应于运动路径。
作为对比，速度变化的长度和角度与前一部分相同

In[]:=

```
kers = motionMatrix@Take[movePath, #] & /@ Range[3];
ImageAdjust@* Image /@ kers
```

Out[]:=





{

得到速度变化各次的模糊图像

In[]:=

```
blured = ImageConvolve[img, #, Padding -> "Reversed"] & /@ kers;
Multicolumn[MapIndexed[Labeled[#1, First@#2 - 1] &, blured~Prepend~img], 2, Appearance -> "Horizontal"]
```

Out[]:=

	
0	1
	
2	3

对各次运动模糊进行反卷积复原

In[]:=

```
deblured = ParallelTable[
  MapThread[ImageClip@ImageDeconvolve[#1, #2, Method -> method] &, {blured, kers}],
  {method, methods}
];
tabcnt++;
Labeled[#, StringForm["表`` 速度变化次数及反卷积算法效果对比", tabcnt], Top] &@
TableForm[
  Map[Labeled[#, TemplateApply["MSE = ``",
    ImageDistance[img, #, DistanceFunction -> "MeanSquaredEuclideanDistance"]
  ] &, deblured, {2}],
  TableHeadings -> {methods, StringTemplate["速度变化 `` 次"] /@ Range[3]},
  TableAlignments -> Center
]
```




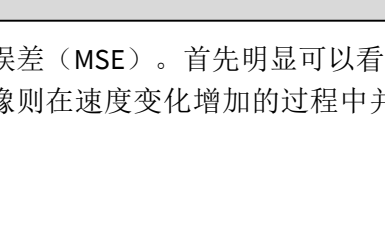


表2 速度变化次数及反卷积算法效果对比			
	速度变化 1 次	速度变化 2 次	速度变化 3 次
DampedLS			
	MSE = 0.00113632	MSE = 0.00272123	MSE = 0.00326247
			
	MSE = 0.00116881	MSE = 0.00201836	MSE = 0.00236076
			
	MSE = 0.00655382	MSE = 0.0164262	MSE = 0.0156302
			
	MSE = 0.0011275	MSE = 0.00179139	MSE = 0.00208841
Tikhonov			
	MSE = 0.0222356	MSE = 0.00814851	MSE = 0.0353213
			
	MSE = 0.000387313	MSE = 0.000457252	MSE = 0.000776921
			
	MSE = 0.0212276	MSE = 0.0083178	MSE = 0.17024
			
	MSE = 0.000622372	MSE = 0.000676591	MSE = 0.000949824
TSVD			
	MSE = 0.00116881	MSE = 0.00201836	MSE = 0.00236076
			
	MSE = 0.00113632	MSE = 0.00272123	MSE = 0.00326247
			
	MSE = 0.00655382	MSE = 0.0164262	MSE = 0.0156302
	MSE = 0.0011275	MSE = 0.00179139	MSE = 0.00208841
Wiener			
	MSE = 0.0222356	MSE = 0.00814851	MSE = 0.0353213
	MSE = 0.000387313	MSE = 0.000457252	MSE = 0.000776921
	MSE = 0.0212276	MSE = 0.0083178	MSE = 0.17024
	MSE = 0.000622372	MSE = 0.000676591	MSE = 0.000949824
Hybrid			
	MSE = 0.00116881	MSE = 0.00201836	MSE = 0.00236076
	MSE = 0.00113632	MSE = 0.00272123	MSE = 0.00326247
	MSE = 0.00655382	MSE = 0.0164262	MSE = 0.0156302
	MSE = 0.0011275	MSE = 0.00179139	MSE = 0.00208841
RichardsonLucy			
	MSE = 0.0222356	MSE = 0.00814851	MSE = 0.0353213
	MSE = 0.000387313	MSE = 0.000457252	MSE = 0.000776921
	MSE = 0.0212276	MSE = 0.0083178	MSE = 0.17024
	MSE = 0.000622372	MSE = 0.000676591	MSE = 0.000949824
SteepestDescent			
	MSE = 0.00116881	MSE = 0.00201836	MSE = 0.00236076
	MSE = 0.00113632	MSE = 0.00272123	MSE = 0.00326247
	MSE = 0.00655382	MSE = 0.0164262	MSE = 0.0156302
	MSE = 0.0011275	MSE = 0.00179139	MSE = 0.00208841
TotalVariation			
	MSE = 0.0222356	MSE = 0.00814851	MSE = 0.0353213
	MSE = 0.000387313	MSE = 0.000457252	MSE = 0.000776921
	MSE = 0.0212276	MSE = 0.0083178	MSE = 0.17024
	MSE = 0.000622372	MSE = 0.000676591	MSE = 0.000949824

表2给出了速度变化次数与反卷积算法的直观的效果对比，以及复原图像与原图像间的均方误差（MSE）。首先明显可以看到改进的残差范数最速下降（SteepestDescent）算法在速度变化3次时没有正确收敛。其它各组算法得到的复原图像则在速度变化增加的过程中并没有明显的改变。下面，图4和图5基于均方误差给出了定量的比较。

Out[]=



Out[•]=

