工作日志

lwj

2021 年 12 月 25 日

目录

1 wildDfd论文阅读

1.1 wildDFD

wild DFD[4]的算法框架如图1，通过建立目标函数，设计相关优化算法，进行求解深度信息。其特点是两帧的模糊图像之间可以有相对运动的存在（算法中设计了光流估计环节进行补偿），算法分为两个过程，$top-downlikelihood$和$bottom-uplikelihood$，其中$top-downlikelihood$通过对图像进行分割，利用分割信息对后续的深度补全提供线索信息，建立损失函数E_{DFD}。$bottom-uplikelihood$将图像分为多个块（patch），在每个块内，利用对均衡滤波模型[5]进行改进，建立优化模型，求解得到深度信息，在通过局部的平滑先验建立损失函数E_{prior}。对于建立好的模型，通过求解马尔科夫随机场问题和二次型问题，对深度进行求解。

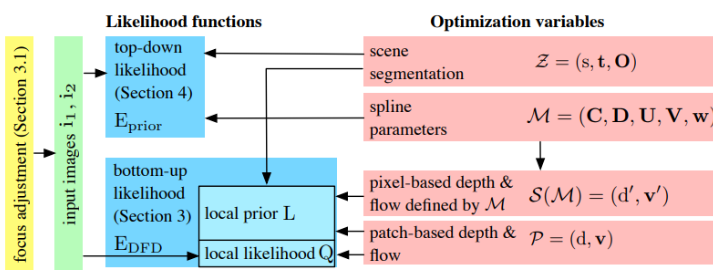


图 1: wildDfd算法框架[4]

2 用于图像分割的特征提取

2.1 Bilateral Affinity Matrix

双边仿射矩阵S的元素$S(p,q)$表征的是图像不同位置p和q之间的关联性。其中其中p和q可以理解成将图像张成向量之后的坐标。S的拉普拉斯矩阵对应的的特征向量，可以包含一定的图像分割信息。相似性通过式(1)(2)建立[4]，其中l,a,b是图像LAB色域内的三通道信息，

$$S(p,q) = \begin{cases} exp(-\Delta_{pq}) + exp(-\Delta_{qp}) \quad if\,|p-q| < 3\sigma_s \\ 0 \qquad\qquad\qquad\qquad\qquad\qquad\quad , others \end{cases} \tag{1}$$

$$\Delta_{pq} = min(\frac{|l_p - l_q|^2}{2\sigma_{pl}^2} + \frac{|a_p - a_q|^2}{2\sigma_{pa}^2} + \frac{|b_p - b_q|^2}{2\sigma_{pb}^2}, \epsilon) + \frac{|p - q|^2}{2\sigma_s^2} \tag{2}$$

S 的拉普拉斯矩阵如下:

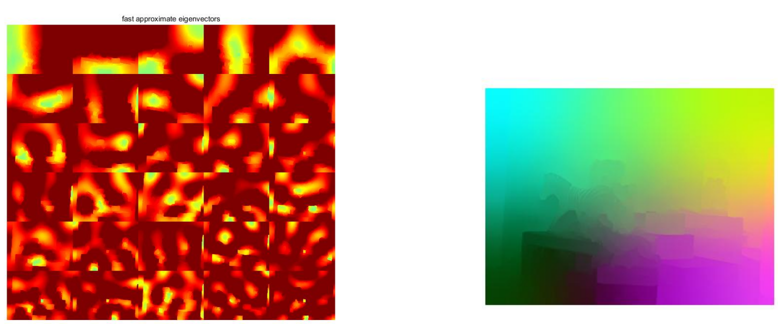
$$L = Diag(sum_{percol}(S)) - S \tag{3}$$

2.2 DNCuts

仿射矩阵S的拉普拉斯矩阵L（见(3)）是一个超高维的矩阵，其特征值的计算复杂度很高，因此Arbelaez[1]提出了DNCuts算法进行优化求解。DNCuts算法基于以下的知识：

- 如果S按行或者按列求和为1，则S和S^2的拉普拉斯矩阵的特征向量是相等的。
- 对S进行降采样（隔行丢或者隔列丢）后求解特征向量近似于对S的特征向量进行降采样。

直接对S进行降采样效果不理想，因此对S^2进行降采样。Arbelaez[1]给出的解释是，通过计算S^TS,S^2的每一列都包含了其他列的信息。因此降采样不会造成太多的信息丢失。计算得到的结果如图2，可以看出拉普拉斯仿射矩阵的特征值和图像分割具有相关性。



(a) 30个特征值图像 (b) 特征值融合之后的伪彩图

图 2: DNCuts算法计算图像分割特征

3 基于模糊均衡的DFD技术

3.1 S Transform

Spatial-doman convolution/Deconvolution Transform[3]简称（S Transform)技术，是一种计算多项式卷积和去卷积的快速算法。图像可以利用三次二维的多项式(4)进行拟合

$$f(x,y)=\sum_{m=0}^{3}\sum_{n=0}^{3}a_{mn}x^m y^n \tag{4}$$

记$h(x,y)$为旋转对称的点扩散函数（PSF),相应的模糊退化模型为(5)，其中$*$ 表示卷积

$$g(x,y)=f(x,y)*h(x,y) \tag{5}$$

如式(6)所示，对于旋转不变的PSF，可以用参数σ_h进行表征，

$$\sigma_h^2=\iint_{-\infty}^{+\infty}(x^2+y^2)h(x,y)dxdy \tag{6}$$

利用S变换[3]可得，其去卷积(deconvolution)过程可以用式(7)表示,其中∇^2为拉普拉斯算子，式(7)也经常用来进行图像边缘的增强

$$f(x,y)=g(x,y)-\frac{\sigma_h^2}{4}\nabla^2 g(x,y) \tag{7}$$

对于利用两种对焦参数拍摄的图像，其点扩散函数的参数σ_1,σ_2的关系如式(8),α表征两幅图像的放大率变化:

$$\sigma_1=\alpha\sigma_2+\beta \tag{8}$$

由(7)可以得到：

$$g_1-g_2=\frac{1}{4}G\nabla^2 g \tag{9}$$

其中

$$\nabla^2=\nabla^2 g1=\nabla^2 g2 \tag{10}$$

可以得到

$$G=\sigma_1^2-\sigma_2^2=\frac{4(g_1-g_2)}{\nabla^2 g} \tag{11}$$

代入式(8)得到

$$\sigma_2^2(\alpha^2-1)+2\alpha\beta\sigma_2+\beta^2=G \tag{12}$$

如果放大率不变也即$\alpha = 1$,则式(12)可以写成

$$\sigma_2 = \frac{G}{2\beta} - \frac{\beta}{2} \tag{13}$$

利用式(7)进行去卷积会放大图像的高频部分，进而降低信噪比，同时(9)要求$\nabla^2 = \nabla^2 g1 = \nabla^2 g2$(对于理想的二维三次多项式模型的STM变换，成立[3]),实际的图像由于噪声的影响，式(10)很难成立。因此Xian等提出了BET方法来替代STM进行深度估计[5]。通过将两张模糊图像分别和适当的PSF（可以已知高频噪声）进行卷积，进一步的，通过BET可以减少对式(10)的依赖。对g_1, g_2分别用各自的卷积核h_1, h_2进行卷积。得到式(14)

$$\begin{aligned} g_1(x,y) * h_2(x,y) &= [f(x,y) * h_1(x,y)] * h_2(x,y) \\ g_2(x,y) * h_1(x,y) &= [f(x,y) * h_2(x,y)] * h_1(x,y) \end{aligned} \tag{14}$$

由卷积运算的交换性可以得到

$$g_1(x,y) * h_2(x,y) = g_2(x,y) * h_1(x,y) \tag{15}$$

利用STM前向变换计算卷积得到：

$$\begin{aligned} g_1(x,y) * h_2(x,y) &= g_1(x,y) + \tfrac{\sigma_2^2}{4}\nabla^2 g_1(x,y) + \tfrac{\sigma_2^4}{24}(\nabla^2)^2 g_1(x,y) + R(O^6) \\ g_2(x,y) * h_1(x,y) &= g_2(x,y) + \tfrac{\sigma_1^2}{4}\nabla^2 g_2(x,y) + \tfrac{\sigma_1^4}{24}(\nabla^2)^2 g_2(x,y) + R(O^6) \end{aligned} \tag{16}$$

省去高阶项（$R(O^4, O^6)$），可得

$$g_1(x,y) + \frac{\sigma_2^2}{4}\nabla^2 g_1(x,y) = g_2(x,y) + \frac{\sigma_1^2}{4}\nabla^2 g_2(x,y) \tag{17}$$

结合式(8)，可以得到

$$a_1\sigma_1^2 + b_1\sigma_1 + c_1 = 0 \tag{18}$$

其中

$$\begin{aligned} a_1 &= \tfrac{\nabla^2 g_2}{\nabla^2 g_1} - 1 \\ b_1 &= 2\beta \\ c_1 &= -\left[\tfrac{4(g_1 - g_2)}{\nabla^2 g_1} + \beta^2\right] \end{aligned} \tag{19}$$

同样的可以得到

$$a_2\sigma_2^2 + b_2\sigma_2 + c_2 = 0 \tag{20}$$

其中

$$\begin{aligned} a_2 &= -\tfrac{\nabla^2 g_1}{\nabla^2 g_2} + 1 \\ b_2 &= 2\beta \\ c_2 &= -\left[\tfrac{4(g_1 - g_2)}{\nabla^2 g_2} - \beta^2\right] \end{aligned} \tag{21}$$

考虑到图像信噪比SNR越小，计算精度越差，交替利用利用式(18)和式(20)（如果$\nabla^2 g_1$¿$\nabla^2 g_2$则计算σ_1,否则计算σ_2）同时将SNR（利用拉普拉斯变换结果的大小来表征）低于阈值的结果过滤掉。即可计算得到相应的σ_1, σ_2

$$\begin{cases} a_1\sigma_1^2 + b_1\sigma_1 + c_1 & = 0, \quad L_1 >= L_2 \\ \sigma_1 & = \sigma_2 + \beta \\ a_2\sigma_2^2 + b_2\sigma_2 + c_2 & = 0, \quad L_1 < L_2 \end{cases} \tag{22}$$

其中L_i如式(23)表示的在一个小的窗口内（距离可以看成不变，psf不变），的拉普拉斯变换的和。

$$L_i = \sum_x \sum_y \left|\nabla^2 g_i(x,y)\right|, \quad i = 1,2 \tag{23}$$

BET方法原理简单，形式简洁，计算简单。但是依赖已知的参数β,同时对噪声污染严重，和SNR较低，图像纹理缺少的区域，其效果不理想。

4 基于快速迭代最小化算法的多通道盲去模糊

Filip[2]提出基于交替最小化优化策略的盲去卷积算法。该算法利用多通道的数据，将去卷积分为两个步骤：1).$U-Step$图像恢复；2).$H-Step$进行PSF估计；通过TV正则化和基于R_Δ的PSF正则化以及保真项最小化建立优化模型。利用交替乘子法进行迭代优化计算。

4.1 U-Step

利用H-Step得到的PSF进行图像恢复，其目标函数如式(24)，图像运算使用矩阵向量的形式。其中$\|Hu-g\|^2$ 表征的是保真项，H为PSF生成的卷积矩阵。u是清晰未退化图像张成的向量，g是多个通道模糊图像张成的向量。

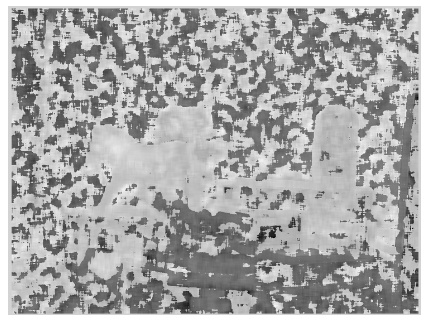
$$\begin{gathered} \min_u \tfrac{\gamma}{2}\left\|Hu-g\right\|^2 + \Phi(D_x u, D_y u) \\ \Phi(D_x u, D_y u) = \sum_i \sqrt{\left|D_x u\right|_i^2 + \left|D_y u\right|_i^2} \end{gathered} \tag{24}$$

将式(24)使用交替方向乘子法做适当松弛之后得到

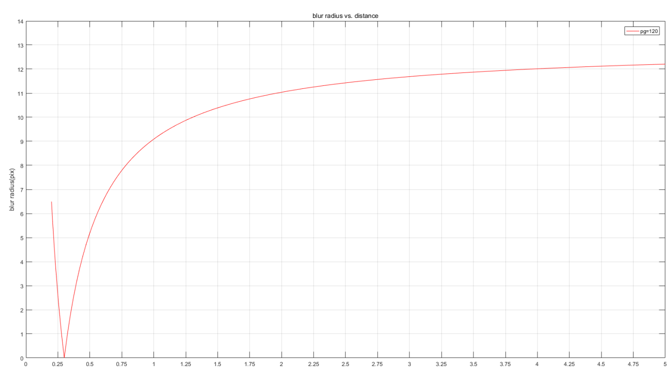
$$\min_{u,v_x,v_y} \mathcal{L}(u,v_x,v_y) = \frac{\gamma}{2}\left\|Hu-g\right\|^2 + \Phi(v_x,v_y) + \frac{\alpha}{2}\left\|D_x u - v_x - a_x\right\|^2 + \frac{\alpha}{2}\left\|D_y u - v_y - a_y\right\|^2 \tag{25}$$

其优化算法如下： 其中对于H的构建需要考虑几点：

(a) 原始图像



(b) σ_2（最亮处为7，最暗处为0）



(c) $blurradius(\sigma = R\sqrt{2})vsdistance$

图 3: BET效果

Algorithm 1 $\hat{u} = u - step(u^0)$

Initialization: $Setv_x^0 = v_y^0 = a_x^0 = a_y^0 = 0,\ j = 0$

Repeat:

Caculate $\boldsymbol{u^{j+1} = arg \min_{u} \mathcal{L}(u, v_x^j, v_y^j)}$:

$$\begin{aligned} u^{j+1} &= A^{-1}\left(H^T g + \frac{\alpha}{\gamma}\left[D_x^T(v_x^j + a_x^j) + D_y^T(v_y^j + a_y^j)\right]\right) \\ A &= H^T H + \frac{\alpha}{\gamma}\left(D_x^T D_x + D_y^T D_y\right) \end{aligned} \tag{26}$$

Caculate $\boldsymbol{v_x^{j+1}, v_y^{j+1} = arg \min_{v_x, v_y} \mathcal{L}(u^{j+1}, v_x, v_y)}$:

$$\begin{aligned} [v_x^{j+1}]_i &= \left[D_x u^{j+1} - a_x^j\right]_i [s]_i^- 1max([s]_i - \frac{1}{\alpha}, 0), \\ [v_y^{j+1}]_i &= \left[D_y u^{j+1} - a_y^j\right]_i [s]_i^- 1max([s]_i - \frac{1}{\alpha}, 0), \\ [s]_i &= \sqrt{\left[D_x u^{j+1} - a_x^j\right]_i^2 + \left[D_y u^{j+1} - a_y^j\right]_i^2} \end{aligned} \tag{27}$$

Caculate $\boldsymbol{a_x^{j+1}, a_y^{j+1}}$:

$$\begin{aligned} a_x^{j+1} &= a_x^j - D_x u^{j+1} + v_x^{j+1} \\ a_y^{j+1} &= a_y^j - D_y u^{j+1} + v_y^{j+1} \end{aligned} \tag{28}$$

$j = j + 1$

until $\|u^j - u^{j-1}\| / \|u^j\| \le tol$:

- 矩阵H的构造，如果构造成普通矩阵，则相关的运算量会巨大，通过构造循环矩阵，可以通过FFT减少运算量;
- 如果将H构造成循环矩阵，则需要考虑边界条件，反射填充和常数填充具有不同的效果；
- g和u向量需要考虑数据对齐以及边界条件

循环矩阵（circulant matrix)是一类特殊的toeplitz矩阵，

$$C=\begin{pmatrix} z_0 & z_{n-1} & \cdots & z_1 \\ z_1 & z_0 & \cdots & z_2 \\ \vdots & \vdots & & \vdots \\ z_{n-2} & z_{n-3} & \cdots & z_{n-1} \\ z_{n-1} & z_{n-2} & \cdots & z_0 \end{pmatrix} \triangleq C(z) \tag{29}$$

式(29)可以写成$downshiftpermutation$矩阵的形式。见式(30)

$$C=\sum_{k=0}^{n} z_k L^k, \tag{30}$$

$$L=\begin{pmatrix} 0 & 0 & \cdots & 0 & 1 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \ddots & 0 & 0 \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{pmatrix} \tag{31}$$

利用式(30)可以简化相关的矩阵运算。对于h_{mn}, u_{MN}，采用'full'输出的卷积，其输出的大小为$(M+m-1)\times(N+n-1)$，首先将h边缘(t,b,l,r)分别填充$(M-1)/2,(N-1)/2$个零。得到h'，将$h\c{}$从中心位置$(M+m-1)/2,(N+n-1)/2$处拆开，张成向量，作为第一列。构成H。同样的现将u的上下左右边界进行填充使其大小和卷积输出大小一致，然后直接张成向量。以$m=n=M=N=3$为例进行说明。张成的H如式(34)所示。

$$h=\begin{pmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{pmatrix} \tag{32}$$

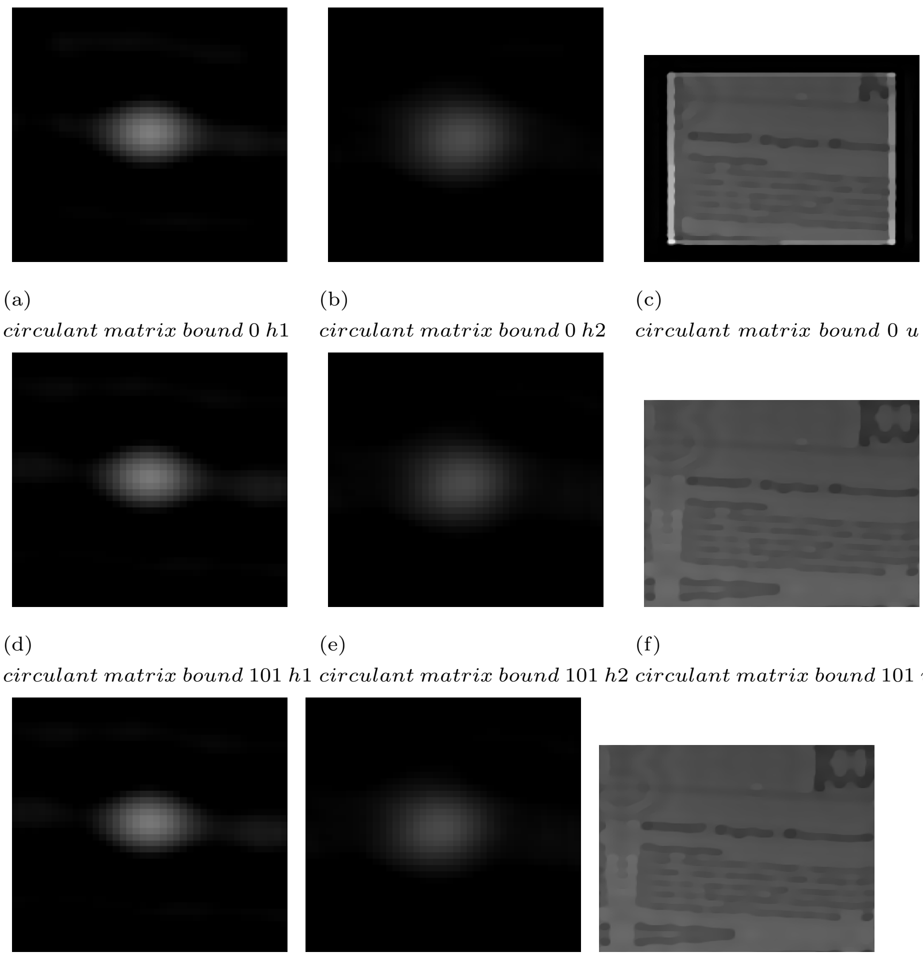
$$
h' = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & h_{00} & h_{01} & h_{02} & 0 \\ 0 & h_{10} & h_{11} & h_{12} & 0 \\ 0 & h_{20} & h_{21} & h_{22} & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{33}
$$

$$
\mathcal{H} = \left(\begin{array}{ccccc|ccccc|ccc|ccccc}
h_{11} & h_{12} & 0 & 0 & h_{20} & h_{21} & h_{22} & 0 & 0 & 0 & & \cdots & h_{00} & h_{01} & h_{02} & 0 & 0 & h_{10} \\
h_{10} & h_{11} & h_{12} & 0 & 0 & h_{20} & h_{21} & h_{22} & 0 & 0 & & & 0 & h_{00} & h_{01} & h_{02} & 0 & 0 \\
0 & h_{10} & h_{11} & h_{12} & 0 & 0 & h_{20} & h_{21} & h_{22} & 0 & & & 0 & 0 & h_{00} & h_{01} & h_{02} & 0 \\
0 & 0 & h_{10} & h_{11} & h_{12} & 0 & 0 & h_{20} & h_{21} & h_{22} & & & 0 & 0 & 0 & h_{00} & h_{01} & h_{02} \\
h_{02} & 0 & 0 & h_{10} & h_{11} & h_{12} & 0 & 0 & h_{20} & h_{21} & h_{22} & & 0 & 0 & 0 & 0 & h_{00} & h_{01} \\
\hline
h_{01} & h_{02} & 0 & 0 & h_{10} & & & & & & & & & & & & & h_{00} \\
h_{00} & h_{01} & h_{02} & 0 & 0 & & & & & & & & & & & & & \\
0 & & & & & & & & & & & & & & & & & \\
\vdots & & & & & & & & & & & & & & & & & \\
h_{22} & & & & & & & & & & & & & & & & & \\
h_{21} & & & & & & & & & & & & & & & & & \\
h_{20} & & & & & & & & & & & & & & & & & \\
0 & & & & & & & & & & & & & & & & & \\
0 & & & & & & & & & & & & & & & & & \\
h_{12} & & & & & & & & & & & & & & & & &
\end{array}\right) \tag{34}
$$

$$
u = \begin{pmatrix} u_{00} & u_{01} & u_{02} \\ u_{10} & u_{11} & u_{12} \\ u_{20} & u_{21} & u_{22} \end{pmatrix} \tag{35}
$$

$$
u' = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & u_{00} & u_{01} & u_{02} & 0 \\ 0 & u_{10} & u_{11} & u_{12} & 0 \\ 0 & u_{20} & u_{21} & u_{22} & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{36}
$$

使用循环矩阵优化的结果如图4.1其中circulant 表示用循环矩阵构造H(如式(34))进行计算，BCCB表示用块循环矩阵进行构造H(如式(38))进行计算。bound 0 表示0填充边界，bound 101 表示用镜像填充。 也可以构造



(a) *circulant matrix bound* 0 $h1$
(b) *circulant matrix bound* 0 $h2$
(c) *circulant matrix bound* 0 u
(d) *circulant matrix bound* 101 $h1$
(e) *circulant matrix bound* 101 $h2$
(f) *circulant matrix bound* 101 u
(g) *BCCB matrix bound* 101 $h1$
(h) *BCCB matrix bound* 101 $h2$
(i) *BCCB matrix bound* 101 u

图 4: 不同矩阵构造和边界效果

块循环矩阵(BCCB,Block circulant matrix with circulant blocks)。

$$BCCB = \begin{pmatrix} C_0 & C_{n-1} & \cdots & C_1 \\ C_1 & C_0 & \cdots & C_2 \\ \vdots & \vdots & & \vdots \\ C_{n-2} & C_{n-3} & \cdots & C_{n-1} \\ C_{n-1} & C_{n-2} & \cdots & C_0 \end{pmatrix} \triangleq BCCB(Z) \tag{37}$$

其中$C_i = C(Z(:,i))$。为Z的第i列生成的循环矩阵。利用BCCB来构造可以使用反射边界，从而减少边界的振铃现象。具体构造方法也是先将h填充边界成h'，然后将每行从中间拆开张成向量作为字块的生成向量z，从中间行拆开，组成块循环矩阵，见代码。以$m = n = M = N = 3$为例进行说明。张成的H如式(38)所示。

$$\mathcal{H} = \left(\begin{array}{ccccc|ccccc|cc|ccccc}
h_{11} & h_{12} & 0 & 0 & h_{10} & h_{21} & h_{22} & 0 & 0 & h_{20} & \cdots & 0 & h_{01} & h_{02} & 0 & 0 & h_{00} \\
h_{10} & h_{11} & h_{12} & 0 & 0 & h_{20} & h_{21} & h_{22} & 0 & 0 & & 0 & h_{00} & h_{01} & h_{02} & 0 & 0 \\
0 & h_{10} & h_{11} & h_{12} & 0 & 0 & h_{20} & h_{21} & h_{22} & 0 & & 0 & 0 & h_{00} & h_{01} & h_{02} & 0 \\
0 & 0 & h_{10} & h_{11} & h_{12} & 0 & 0 & h_{20} & h_{21} & h_{22} & & 0 & 0 & 0 & h_{00} & h_{01} & h_{02} \\
h_{12} & 0 & 0 & h_{10} & h_{11} & h_{22} & 0 & 0 & h_{20} & h_{21} & & 0 & h_{02} & 0 & 0 & h_{00} & h_{01} \\
\hline
h_{01} & h_{02} & 0 & 0 & h_{00} & & & & & & & & & & & & 0 \\
h_{00} & h_{01} & h_{02} & 0 & 0 & & & & & & & & & & & & \\
0 & & & & & & & & & & & & & & & & \\
\vdots & & & & & & & & & & & & & & & & \\
h_{22} & & & & & & & & & & & & & & & & \\
h_{21} & & & & & & & & & & & & & & & & \\
h_{20} & & & & & & & & & & & & & & & & \\
0 & & & & & & & & & & & & & & & & \\
0 & & & & & & & & & & & & & & & & \\
h_{22} & & & & & & & & & & & & & & & &
\end{array}\right) \tag{38}$$

4.2 H-Step

利用$U - Step$得到的恢复好的清晰图像u，进行模糊核函数的估算。优化目标函数如式(39)。使用的参数为

$\gamma = 1e2; \alpha = 1e{-}1{*}gamma; \beta = 1e4{*}gamma; \delta = 1e3{*}gamma; maxLoop = 100; tol = 1e{-}1;$

$$
\begin{aligned}
\min_{h}(L)(h) &= \tfrac{\gamma}{2}\left\|Uh-g\right\|^{2}+\tfrac{\delta}{2}h^{T}R_{\Delta}h+\Psi(h)\\
R_{\Delta} &= \left[\Delta G_{2},-\Delta G_{1}\right]^{T}\left[\Delta G_{2},-\Delta G_{1}\right]\\
\Psi(h) &= \sum_{k=1}^{K}\sum_{i=1}^{\hat{L}}\psi(h_{k}(i)),\psi(t)=\begin{cases} t, & if\ t\geq 0\\ +\infty, & others\end{cases}
\end{aligned}
\tag{39}
$$

利用ALM算法(39)可以转变为(40)

$$
\min_{h}\mathcal{L}(h,w)=\frac{\gamma}{2}\left\|Uh-g\right\|^{2}+\frac{\delta}{2}h^{T}R_{\Delta}h+\Psi(w)+\frac{\beta}{2}\left\|h-w-b\right\|^{2} \tag{40}
$$

按照如下的优化算法进行迭代求解 需要注意一下几点：

Algorithm 2 $\hat{h}=h-step(h^{0})$

$\boldsymbol{Initialization}$: $Setw^{0}=b^{0}=0,\ j=0$

$\boldsymbol{Repeat}$:

$\boldsymbol{Caculate\ h^{j+1}=arg\min_{u}\mathcal{L}(h,w^{j})}$:

$$
\begin{aligned}
h^{j+1} &= B^{-1}\left(U^{T}g+\frac{\beta}{\gamma}\left[w^{j}+b^{j}\right]\right)\\
B &= U^{T}U+\frac{\delta}{\gamma}R_{\Delta}+\frac{\beta}{\gamma}I
\end{aligned}
\tag{41}
$$

$\boldsymbol{Caculate\ w^{j+1}=arg\min_{w}\mathcal{L}(h^{j+1},w)}$:

$$
[w^{j+1}]_{i}=max([h^{j+1}-b^{j}]_{i}-\frac{1}{\beta},0), \tag{42}
$$

$\boldsymbol{Caculate\ b^{j+1}}$:

$$
b^{j+1}=b^{j}-h^{j+1}+w^{j+1} \tag{43}
$$

$j=j+1$

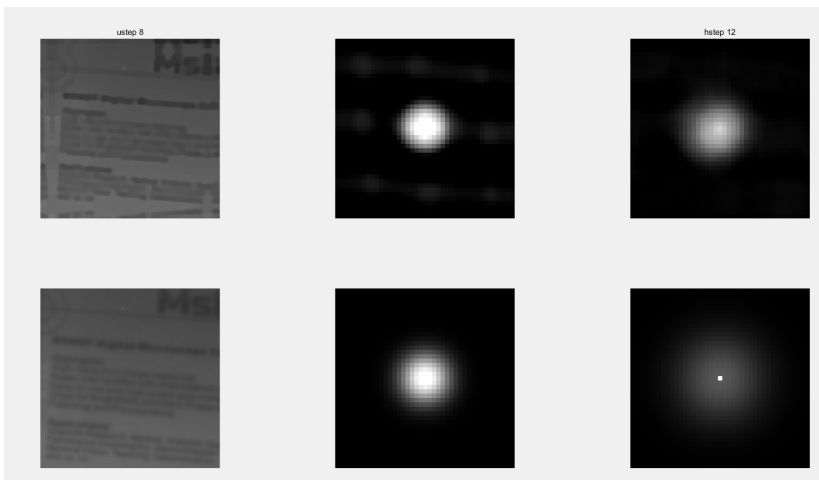
$\boldsymbol{until}\ \|h^{j}-h^{j-1}\|\,/\,\|h^{j}\|\leq tol$:

- U,R_{Δ} 的构造需要考虑不同卷积输出模式的影响，论文推荐的是valid模式
- U,R_{Δ}矩阵相关运算，可以使用FFT运算加速，代码中使用的是opencv的API matchTemplate(内部已经用fft做了优化)
- 进行相关的处理之前，需要对图像进行预处理，代码中实现的预处理包括用一个比较小的高斯核进行滤波，进行归一化

4.3 实验结果

实验结果见图4.3，其中第一行的图片是迭代处理后的图像和psf，第二行是原始的图片$(g1+g2)/2$，以及根据实际情况大致预测的PSF。图示总计迭代了6次，参数为

$$L = 41, imagesize = 256, \gamma = 1e2, \alpha = 1e0 * \gamma, \beta = 1e4 * \gamma, \delta = 1e3 * \gamma,$$



5 接下来的工作

- 进一步完善MBD算法的实验,考虑使用TGV进行图像恢复
- 复现wild dfd

参考文献

[1] Pablo Arbelaez, Jordi Pont-Tuset, Jon Barron, Ferran Marques, and Jitendra Malik. Multiscale combinatorial grouping. In *2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[2] F. Sroubek and P. Milanfar. Robust multichannel blind deconvolution

via fast alternating minimization. *IEEE Transactions on Image Processing*, 21(4):1687–1700, 2012.

[3] Muralidhara Subbarao. Spatial-domain convolution/deconvolution transform. *Technique Report No. 91.07. 03*, 1991.

[4] H. Tang, S. Cohen, B. Price, S. Schiller, and K. N. Kutulakos. Depth from defocus in the wild. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[5] T. Xian and M. Subbarao. Depth-from-defocus: blur equalization technique. *Proceedings of SPIE - The International Society for Optical Engineering*, pages 63820E–63820E–10, 2006.