实时图像去雾算法及其在GPU上的实现

张 军

zhangjunman@sjtu.edu.cn

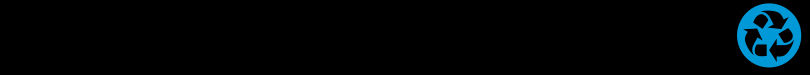
上海交通大学

SHANGHAI JIAO TONG UNIVERSITY

提　纲

1. 问题描述
2. 相关工作
3. 数学模型&去雾算法
4. GPU程序实现
5. 实验结果及分析
6. 存在问题
7. 改进方向

1. 问题描述



- 图像去雾(Haze removal or De-hazing)算法是计算机视觉领域中一个重要的研究课题，在诸如自动监控系统、自动驾驶、室外目标识别和低能见度环境中可视导航等领域内有着广泛的应用。
- 由于单幅图像去雾问题的求解依赖于未知的场景深度信息，使之成为一个具有挑战性的研究课题。

1.1 雾霾图像 （Hazy Images）





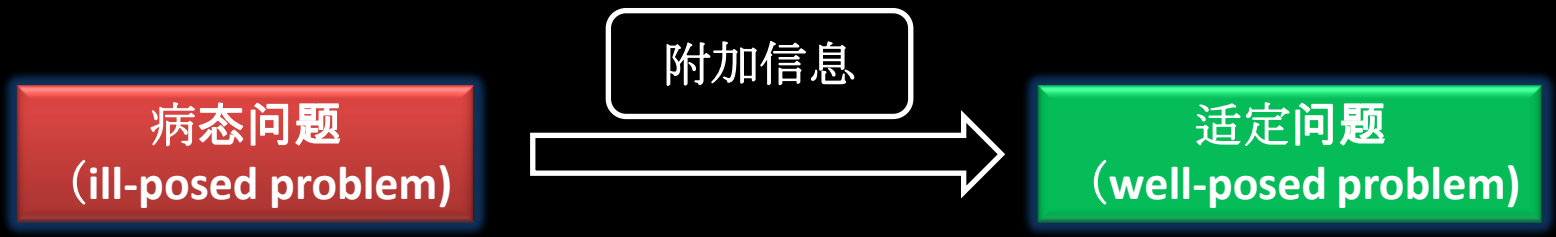
- 低能见度
- 暗淡色彩

1.2 图像去雾的目标

- 细节恢复
- 对比度增强
- 饱和度增强
- 维持色度不变

2 相关工作

- 使用附加信息 （additional information）
  - Polarization filter [Shwartz et al., CVPR'06]
  - Multiple images [Narasimhan& Nayar, CVPR'00]
  - Known 3D model [Kopf et al., Siggraph Asia'08]
  - User-assistance [Narasimhan& Nayar, CPMCV'03]



2 相关工作

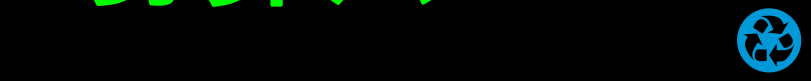
- 使用 先验信息 （stronger priors）
  - Maximize local contrast[Tan, CVPR 08]
  - Independent Component Analysis [Fattal, Siggraph 08]
  - Dark channel prior [He, CVPR 09]
  - Fusion-based strategy [Ancuti, ICIP 2010]

2.1 现有方法局限性

- 计算效率不能适应实时应用
- 鲁棒性不足，对夜景适应性差

[He, CVPR 09]

3 数学模型&去雾算法



Atmospheric light

$$\mathbf{I}(x) = e^{-\beta d(x)}\mathbf{J}(x) + \mathbf{A}(1 - e^{-\beta d(x)})$$

Hazy image

Scene radiance

Scene depth map

3.1 反演方程

$$\mathbf{J}(x)=\mathbf{I}(x)+(\mathbf{I}(x)-\mathbf{A})\frac{(1-e^{-\beta d(x)})}{e^{-\beta d(x)}}$$

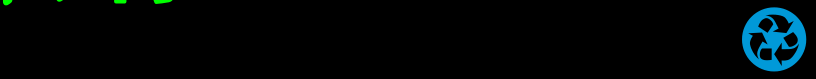
Observed Data

Global Constant

Key Unknown

去雾算法需要从观测图像中推测场景实际亮度

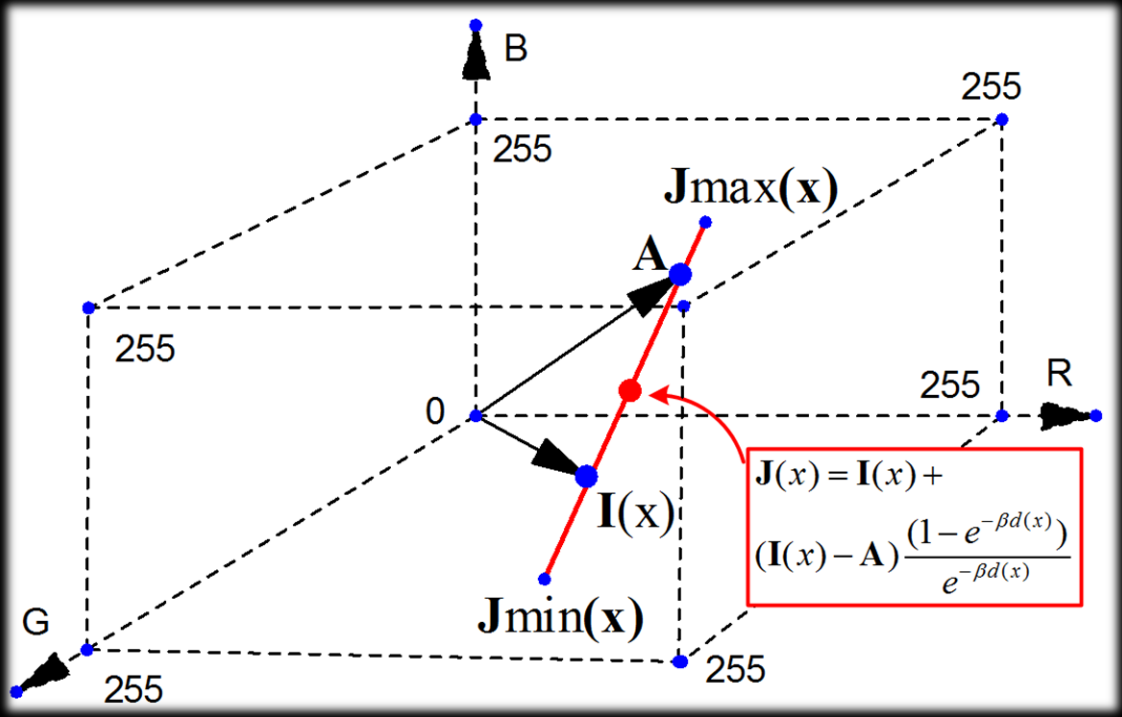
3.2 研究目标



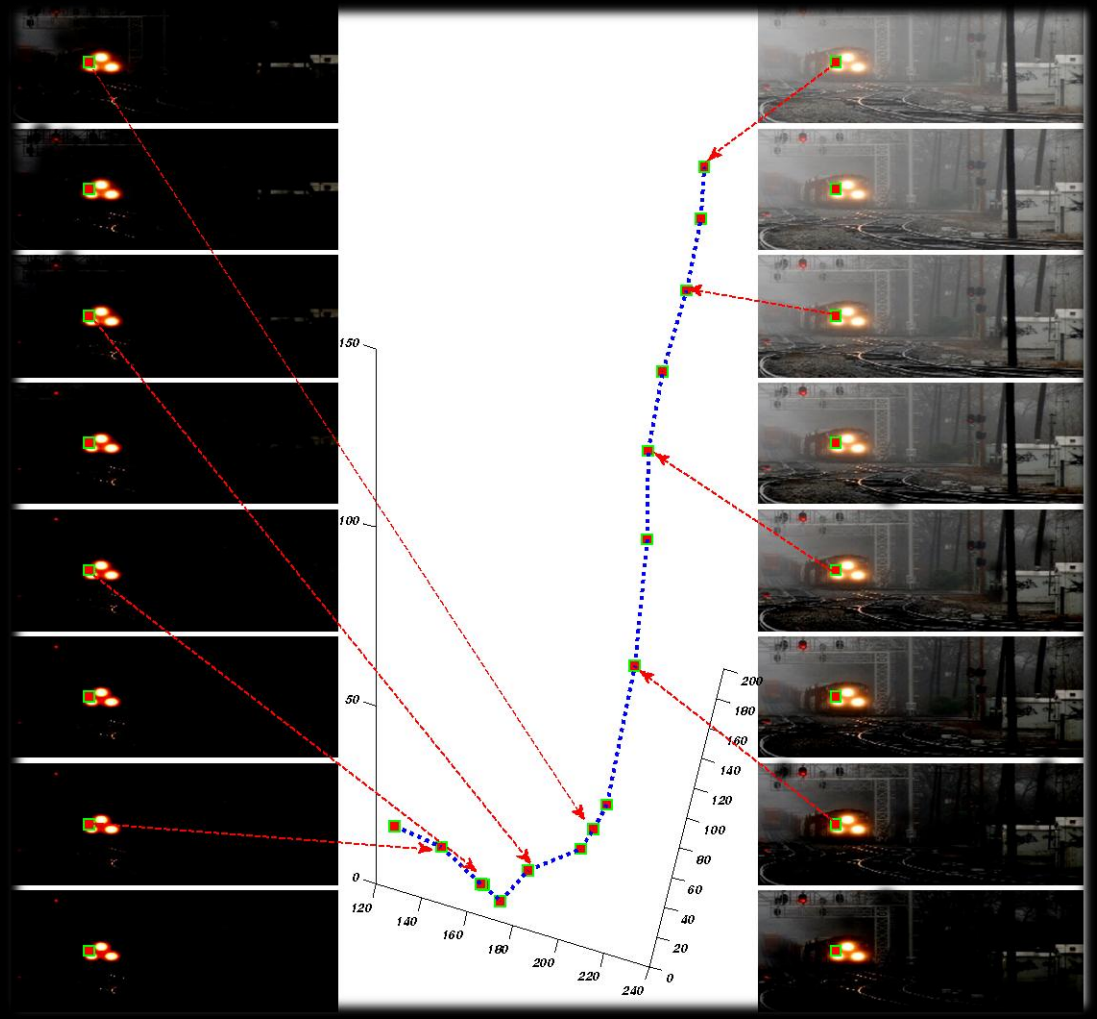
- 提出一种基于虚拟去雾候选图像序列(Virtual Haze-free Candidate Image Sequence， **VHCIs**)的像素级图像增强算法，可对大尺寸图像进行实时消除雾效的计算。
- 该去雾算法通过离散化穷举所有可能的场景深度值，通过并发的快速局部计算而减少耗时的全局串行计算，非常适合在GPU上进行并行加速(Hardware friendly)。

3.3 虚拟去雾候选图像序列 VHCIs

- 数字图像的像素值只能取有限个整数值 (0~255 for 8-bit format), 故场景亮度值J(x)是可以从有限个对场景深度值d(x)离散化穷举中得到的。



3.3 虚拟去雾候选图像序列 VHCIs



$$\mathbf{J}_{d_k}(x) = \mathbf{I}(x) + (\mathbf{I}(x) - \mathbf{A})\frac{(1 - e^{-\beta(k\Delta d)})}{e^{-\beta(k\Delta d)}}$$

$$VHCIs := \{\mathbf{J}_{d_k}(x)\}_{k=0}^{K}$$

$$d_k = k\Delta k$$

全局**尝试**
局部精**选**
Global-to-Local

3.4 全局尝试、局部精选

- 最优去雾准则：如何从VHCIs中挑选出合适的像素值，重新组成一幅消除雾效的图像。
- 图像质量评价准则(image quality assessment)成为局部精选的关键。
- 为保证可完全被GPU并行加速，局部精选操作涉及范围越小越好——单个像素位置！

全局尝试
局部精选
Global-to-Local

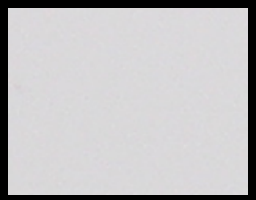
3.5 像素值曲线的累加弦长参数化

- 减少分析计算强度
- 增强几何直观性
- 抵制图像噪声映像

$$\left\{ s_k(x) = \sum_{i=1}^{k} \left\| \mathbf{J}_{d_i}(x) - \mathbf{J}_{d_{i-1}}(x) \right\| \right\}_{k=1}^{K}$$

$$s_0(x) = 0$$

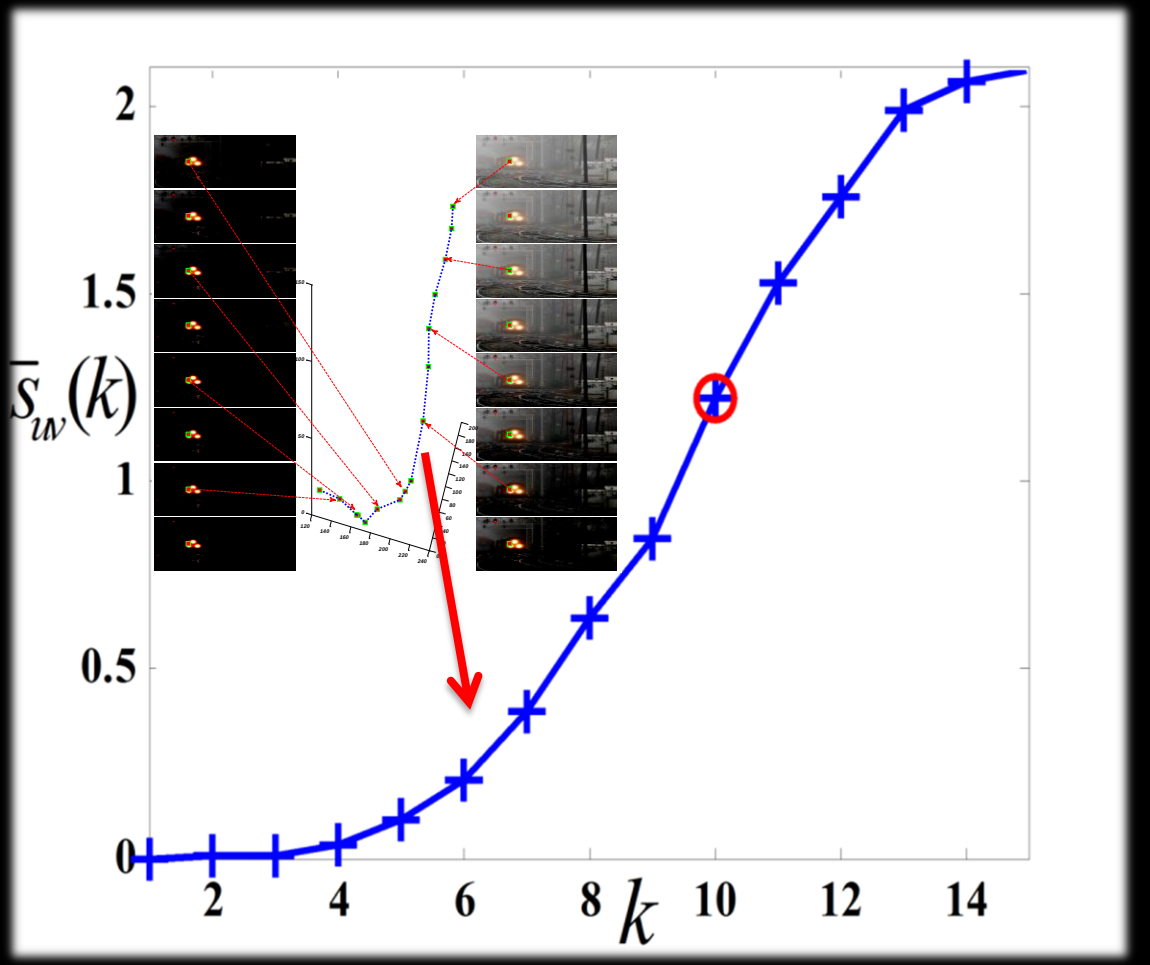
$$\overline{s}_k(x) = \left\{ \left[(s_k * G_{2D})(x) \right] * G_{1D} \right\}(k)$$



原图

放大噪声

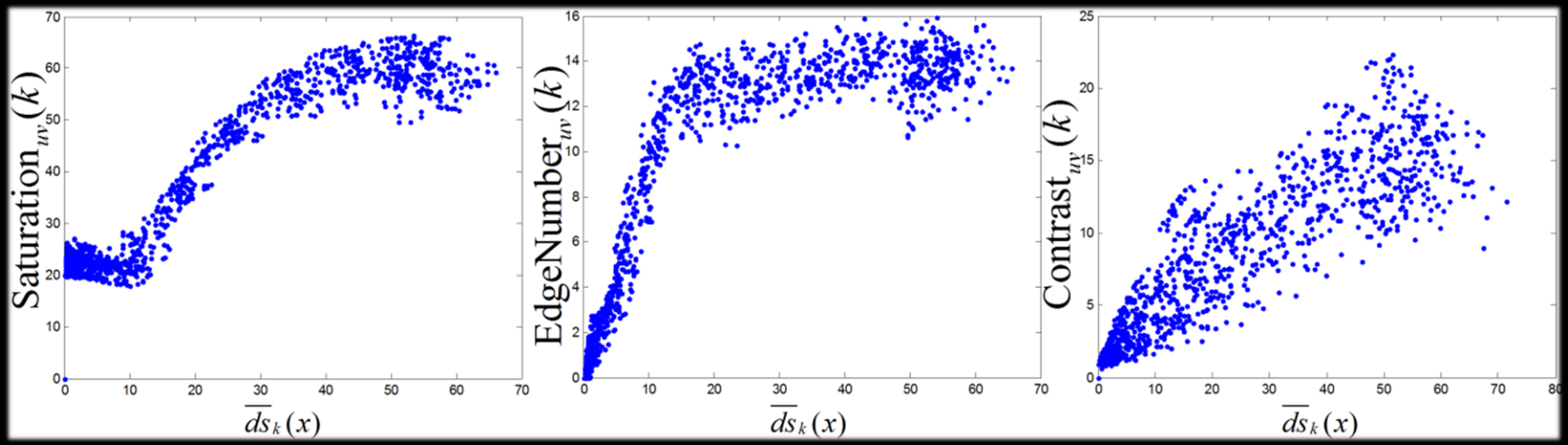
去雾后



3.6 图像局部质量分析实验

- 图像局部质量与像素值曲线点的稠密型正相关；
- 像素值曲线点的稠密可由曲线一阶导数值所表示。

$$\overline{ds_k}(x) = \frac{d\overline{s}_k(x)}{dk} \approx \frac{\overline{s}_{k+1}(x) - \overline{s}_{k-1}(x)}{2}$$



3.7 像素级最优去雾准则

- 随着$d_k(x)$值的连续变化，虚拟去雾图像$J_k(x)$将从“去雾过度”逐渐变化到“去雾不足”。考虑到$J_k(x)$是连续函数，其值必定会在两个极值点间存在一个平衡点（零点定理），该值即为“最佳去雾点”。

- 数学上，这种发生“质变”的点，一般都为曲线的“拐点”。

- 结合上述实际实验，确定最优去雾准则为（Pixel-level Optimal De-hazing Criterion, PODC）：

$$\overline{ds}(x)_{index} = \arg\max_{0 \le k \le K} \overline{ds}_k(x)$$

3.8 图像序列融合的数学模型

- 采用加权求和方式，在PODC的指导下综合VCHIs的像素值信息：

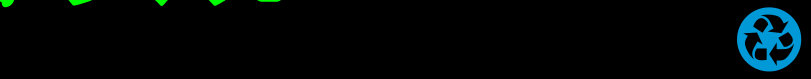
$$\mathbf{J}(x)=\frac{\sum_{k=1}^{K}\mathbf{J}_{d_k}(x)\cdot w_k(x)}{\sum_{k=1}^{K}w_k(x)}$$

$$w_k(x)=\left(\overline{ds}_k(x)/\overline{ds}(x)_{index}\right)^{\alpha}$$

3.9 去雾效果（原始图像）

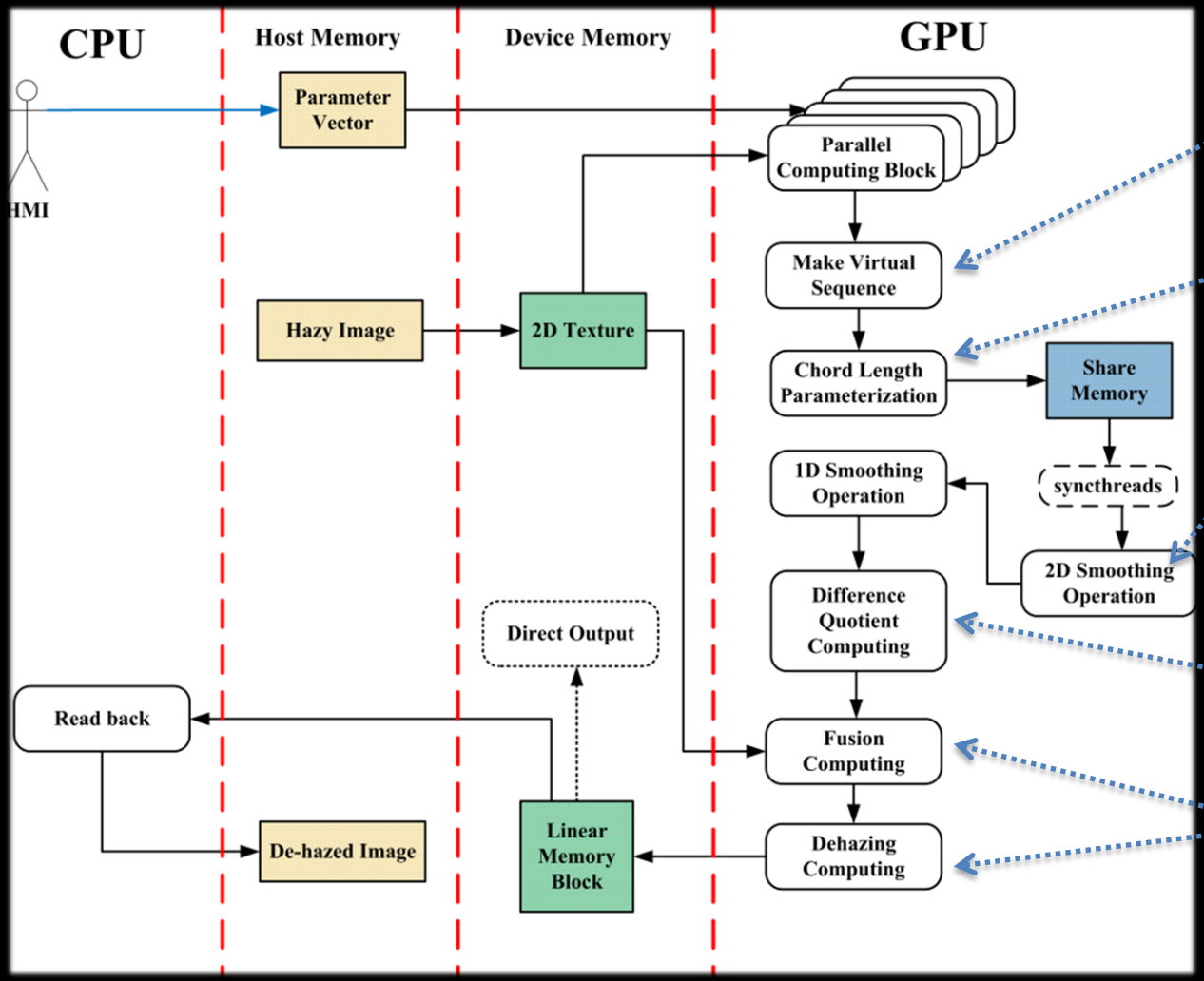
3.9 去雾效果（处理结果）

4 GPU程序实现



- 减少数据传输规模
- 减少中间数据存储规模
- 保持局部计算独立性，降低线程间同步性
- 使用硬件缓存加快数据传输速度
- 利用局部独立计算替代数据读取
- 充分发挥CPU和GPU各自优势

4.1 GPU程序实现方案（I）



$$\mathbf{J}_{d_k}(x)=\mathbf{I}(x)+(\mathbf{I}(x)-\mathbf{A})\frac{(1-e^{-\beta(k\Delta d)})}{e^{-\beta(k\Delta d)}}$$

$$\left\{s_k(x)=\sum_{i=1}^{k}\left\|\mathbf{J}_{d_i}(x)-\mathbf{J}_{d_{i-1}}(x)\right\|\right\}_{k=1}^{K}$$

$$s_0(x)=0$$

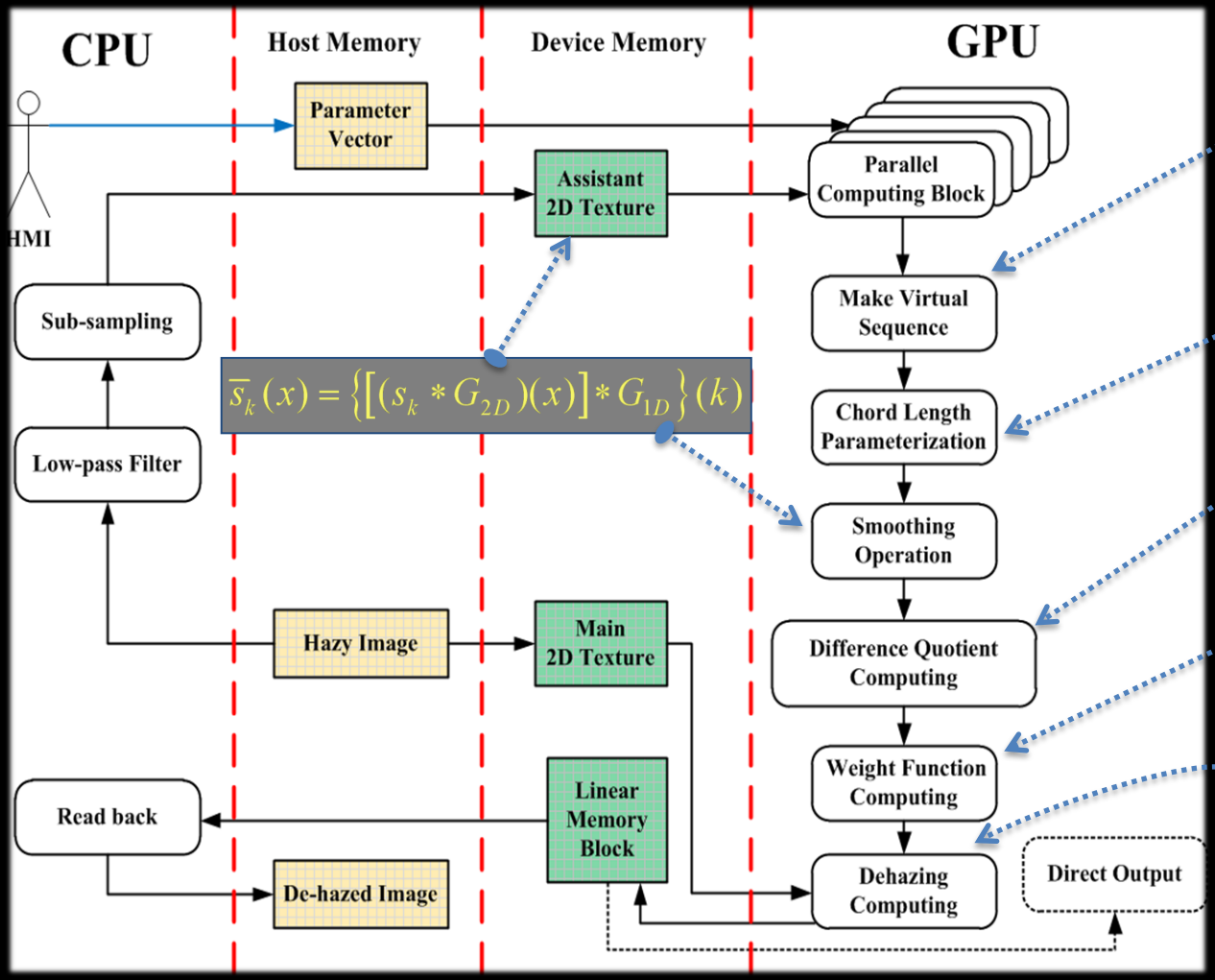
$$\overline{s}_k(x)=\left\{\left[(s_k * G_{2D})(x)\right] * G_{1D}\right\}(k)$$

$$\overline{ds}_k(x)=\frac{d\overline{s}_k(x)}{dk}\approx\frac{\overline{s}_{k+1}(x)-\overline{s}_{k-1}(x)}{2}$$

$$w_k(x)=\left(\overline{ds}_k(x)/\overline{ds}(x)_{index}\right)^{\alpha}$$

$$\mathbf{J}(x)=\frac{\sum_{k=1}^{K}\mathbf{J}_{d_k}(x)\cdot w_k(x)}{\sum_{k=1}^{K}w_k(x)}$$

4.2 GPU程序实现方案（II）



$$\mathbf{J}_{d_k}(x) = \mathbf{I}(x) + (\mathbf{I}(x) - \mathbf{A})\frac{(1 - e^{-\beta(k\Delta d)})}{e^{-\beta(k\Delta d)}}$$

$$\left\{ s_k(x) = \sum_{i=1}^{k} \left\| \mathbf{J}_{d_i}(x) - \mathbf{J}_{d_{i-1}}(x) \right\| \right\}_{k=1}^{K}$$

$$s_0(x) = 0$$

$$\overline{ds}_k(x) = \frac{d\overline{s}_k(x)}{dk} \approx \frac{\overline{s}_{k+1}(x) - \overline{s}_{k-1}(x)}{2}$$

$$w_k(x) = \left(\overline{ds}_k(x) / \overline{ds}(x)_{index} \right)^{\alpha}$$

$$\phi(x) = \frac{\sum_{k=1}^{K} \frac{(1 - e^{-\beta d_k})}{e^{-\beta d_k}} \cdot w_k(x)}{\sum_{k=1}^{K} w_k(x)}$$

$$\mathbf{J}^{opt}(x) = \mathbf{I}(x) + (\mathbf{I}(x) - \mathbf{A})\phi(x)$$

4.3 实现方案对比

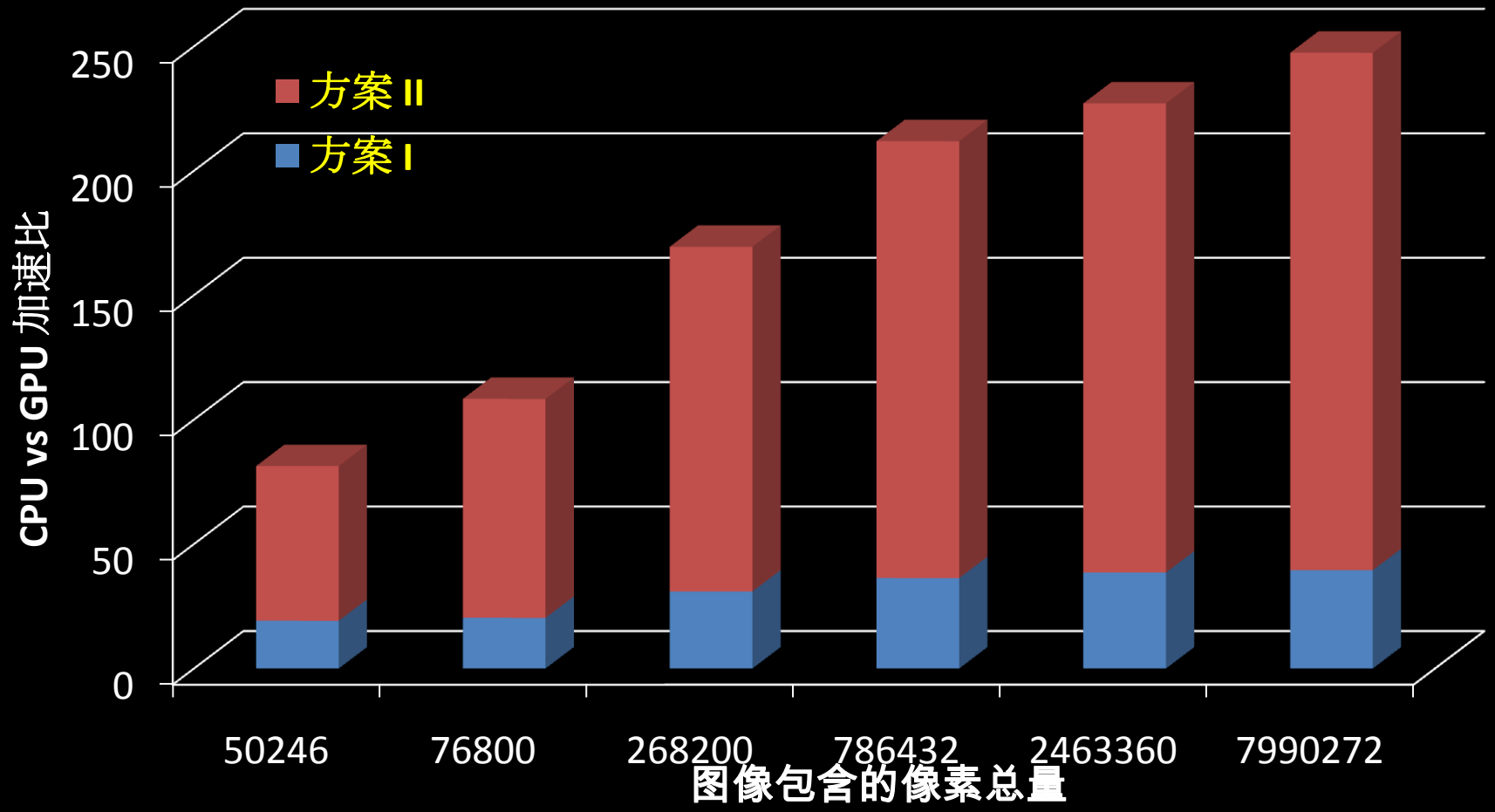
- 方案I

1. 需要共享内存
2. 需要线程同步
3. 两次2D纹理访问或存储三维向量组
4. CPU闲置
5. 一次纹理数据传输

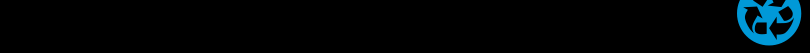
- 方案II

1. 无需共享内存
2. 无需线程同步
3. 两次2D纹理访问
4. CPU充分利用中
5. 两次纹理数据传输

4.4 加速比实验


方案 II
方案 I
CPU vs GPU 加速比
250
200
150
100
50
0
50246
76800
268200
786432
2463360
7990272
图像包含的像素总量

5 实验结果及分析



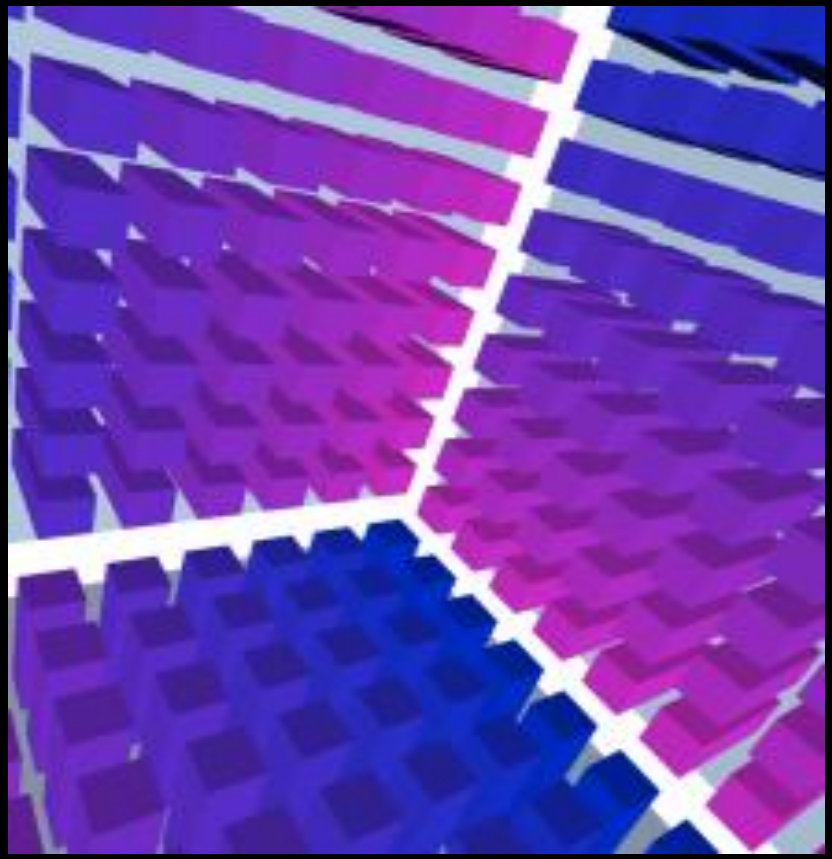
方案 II 运行时间

INTEL Core2 Duo 2.93GHz CPU 和 NVIDIA GTX 460 GPU 的PC机

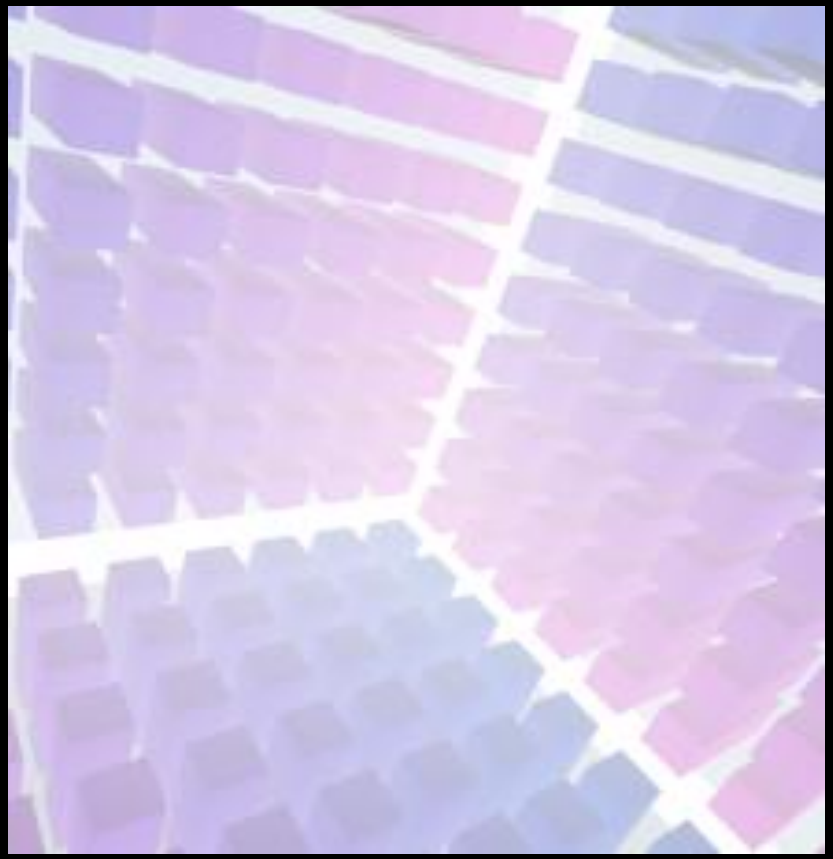
软件编译环境是Visual Studio 2008和CUDA C 3.2

Number of pixels	*GPU Run-times (second)*	*Speedup factor*
50 246	0.002	62.15
76 800	0.002	87.98
268 200	0.004	138.50
786 432	0.011	175.55
2 463 360	0.032	188.51
7 990 272	0.102	207.92

5.1 虚拟图像实验结果

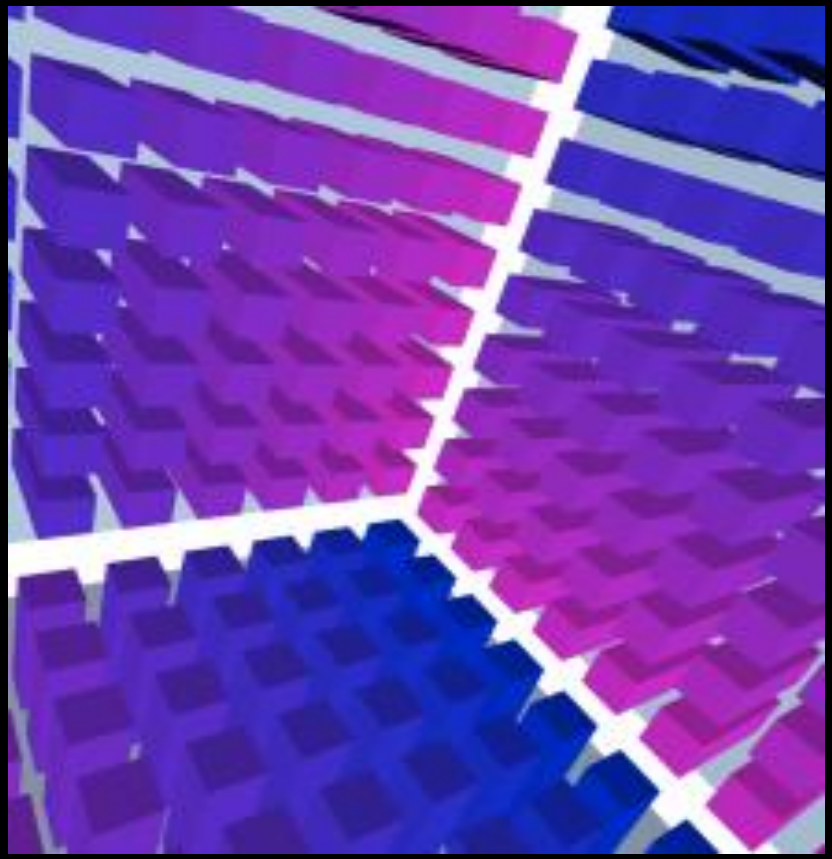


haze-free image

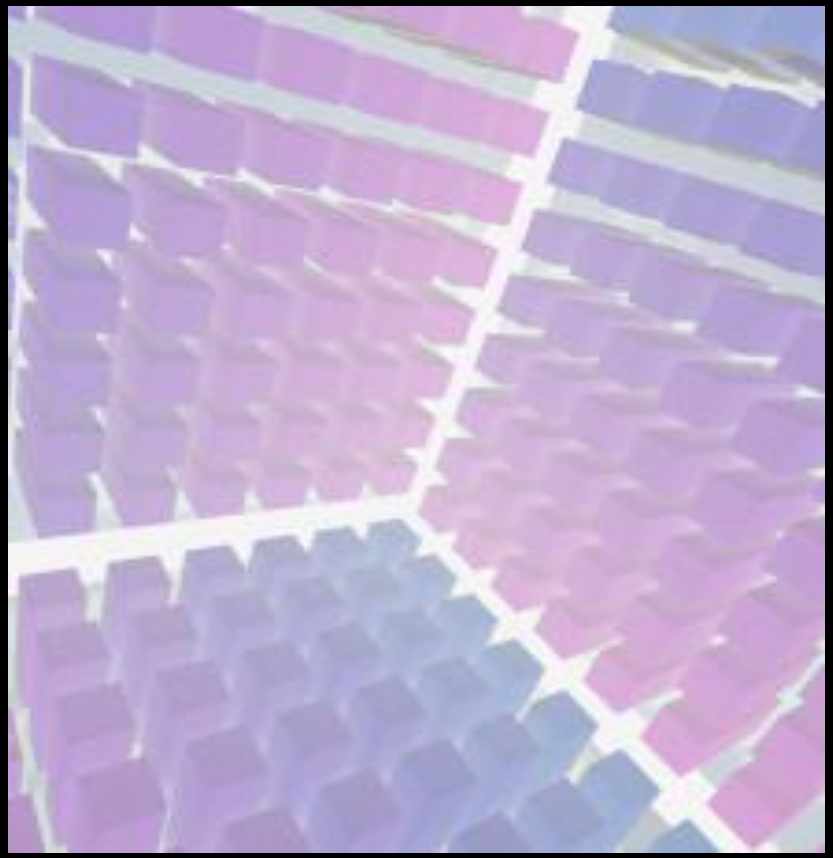


hazy image

5.1 虚拟图像实验结果

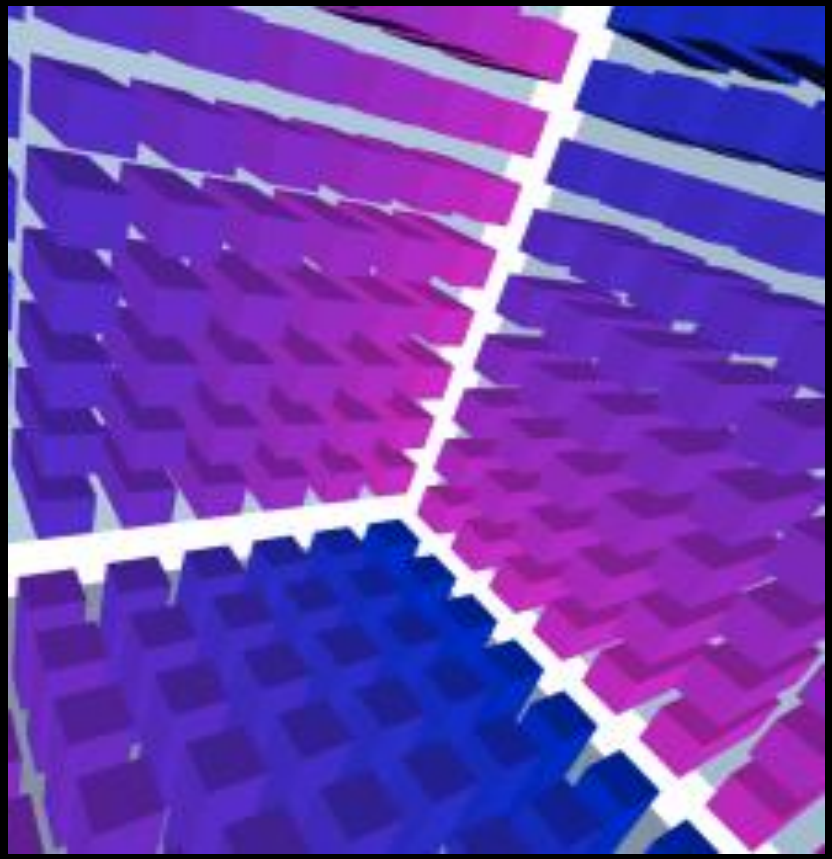


haze-free image

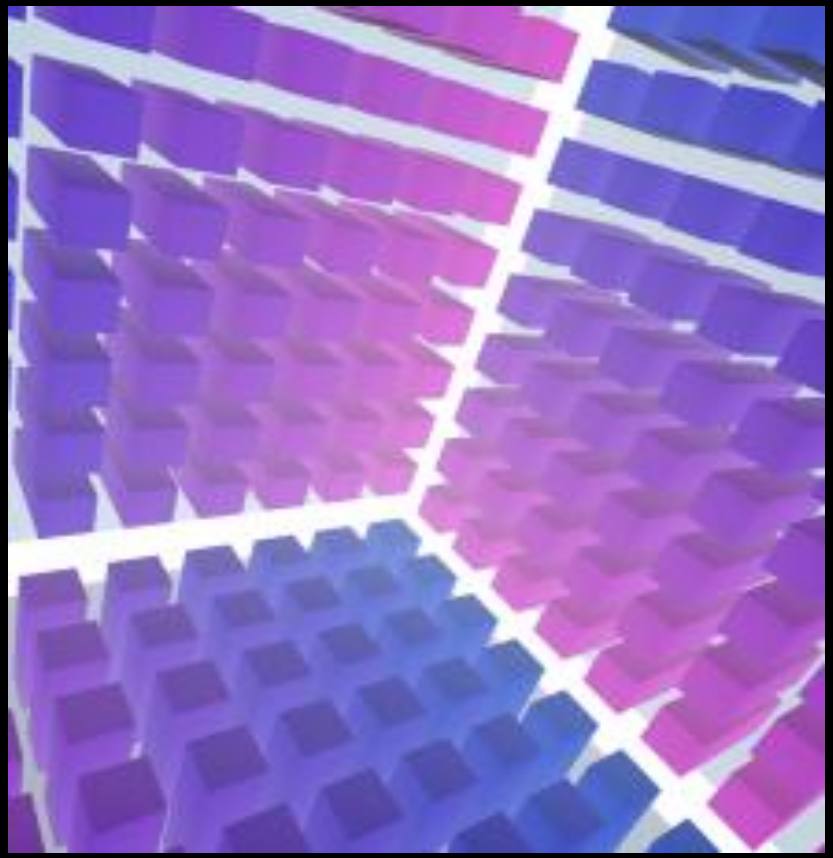


Tarel's result

5.1 虚拟图像实验结果

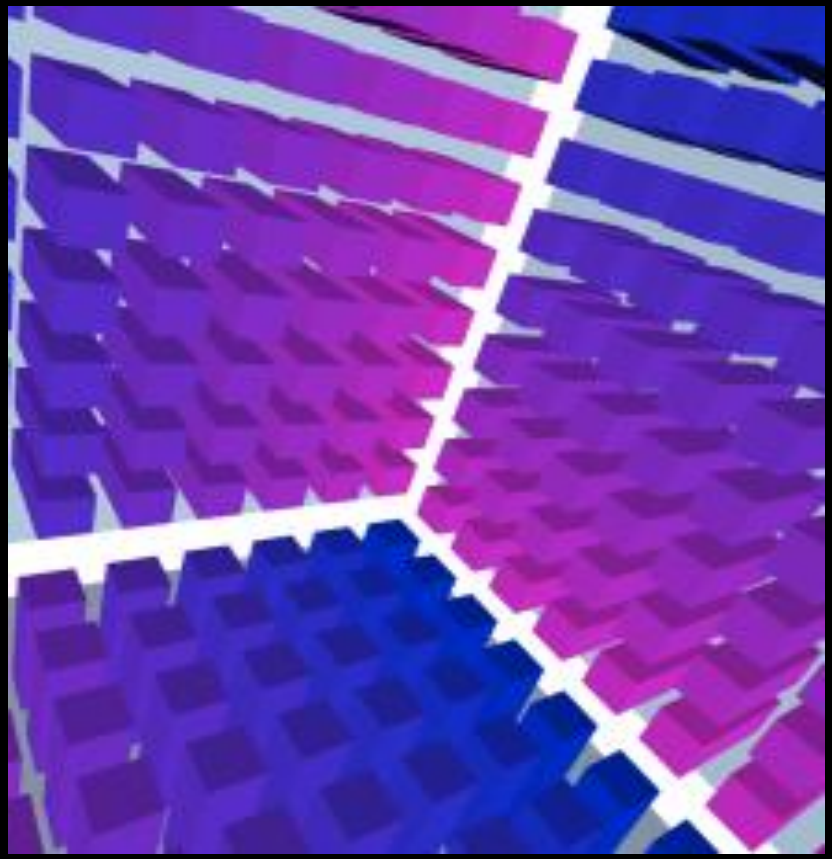


haze-free image

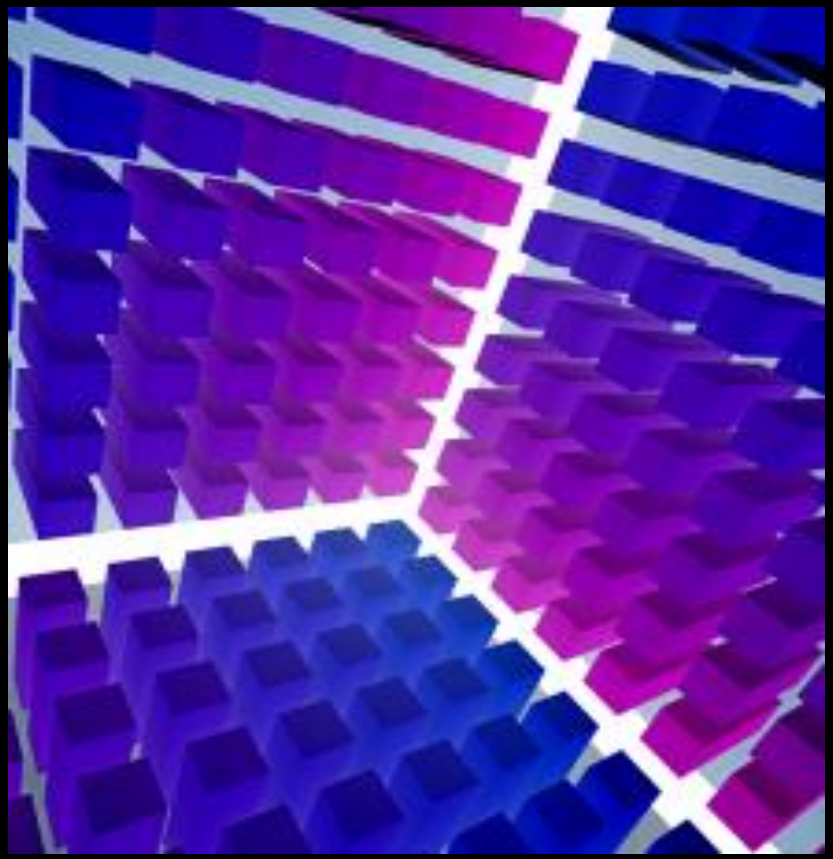


He’s result

5.1 虚拟图像实验结果



haze-free image



Our result

5.2 真实图像实验结果

hazy image

Fattal's result

5.2 真实图像实验结果

hazy image

Our result

5.2 真实图像实验结果

hazy image

He’s result

5.3 色差实验结果

hazy image

5.3 色差实验结果

Our Result

5.4 夜景实验结果

hazy image

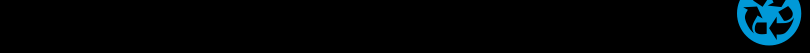
5.4 夜景实验结果

He’s result

5.4 夜景实验结果

Our result

6 存在问题



- 过度的2D平滑操作有可能造成图像边缘处出现人工痕迹（Halo Artifacts）

7 改进方向

- Paris S, Hasinoff S W, Kautz J. Local Laplacian Filters: Edge-aware Image Processing with a Laplacian Pyramid[J]. ACM Transactions on Graphics (Proc. of SIGGRAPH). 2011, 30(4): Article 68.

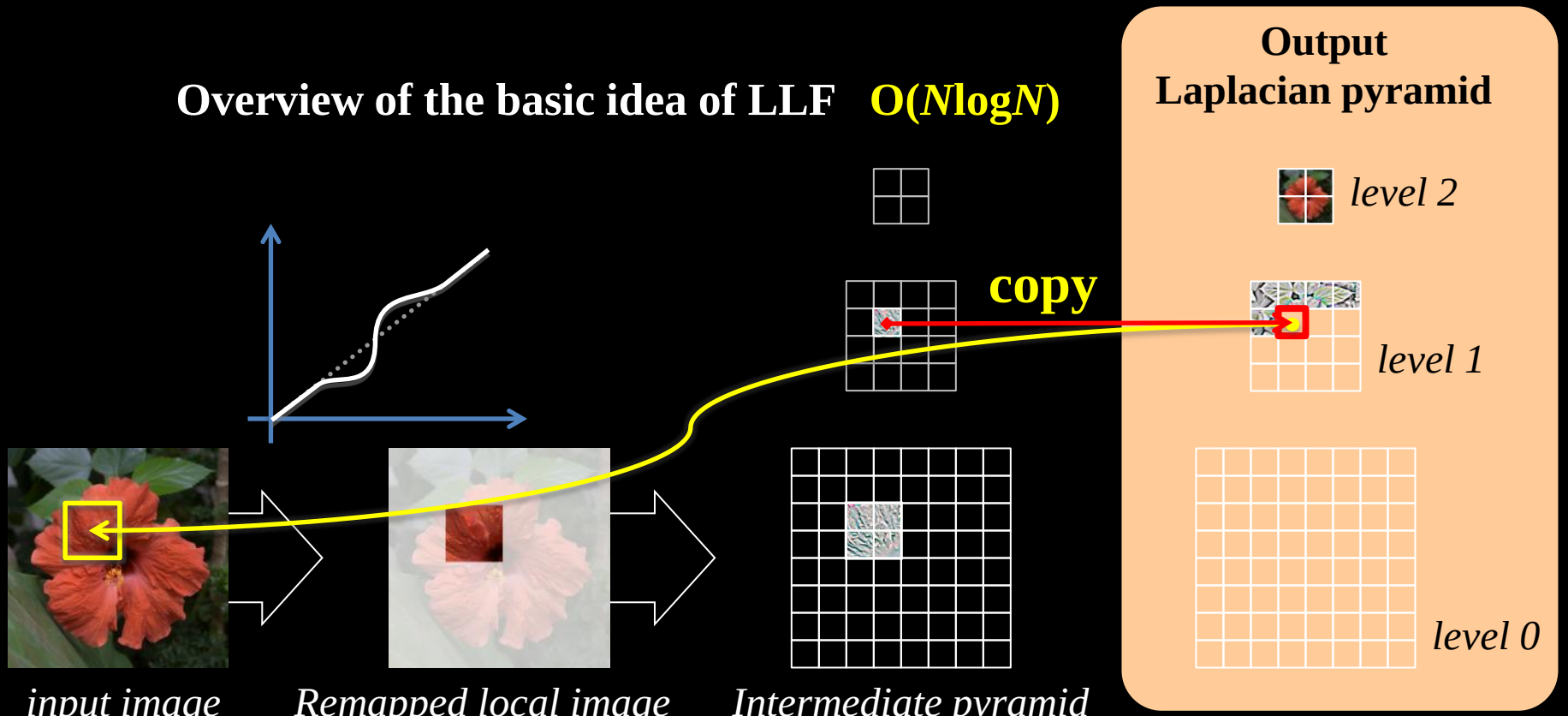
input image　　details enhanced

7 改进方向

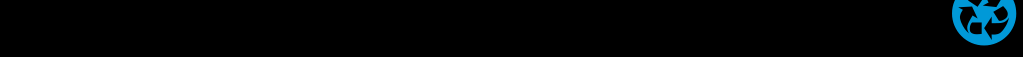
- Paris S, Hasinoff S W, Kautz J. Local Laplacian Filters: Edge-aware Image Processing with a Laplacian Pyramid[J]. ACM Transactions on Graphics (Proc. of SIGGRAPH). 2011, 30(4): Article 68.

input image　　details enhanced

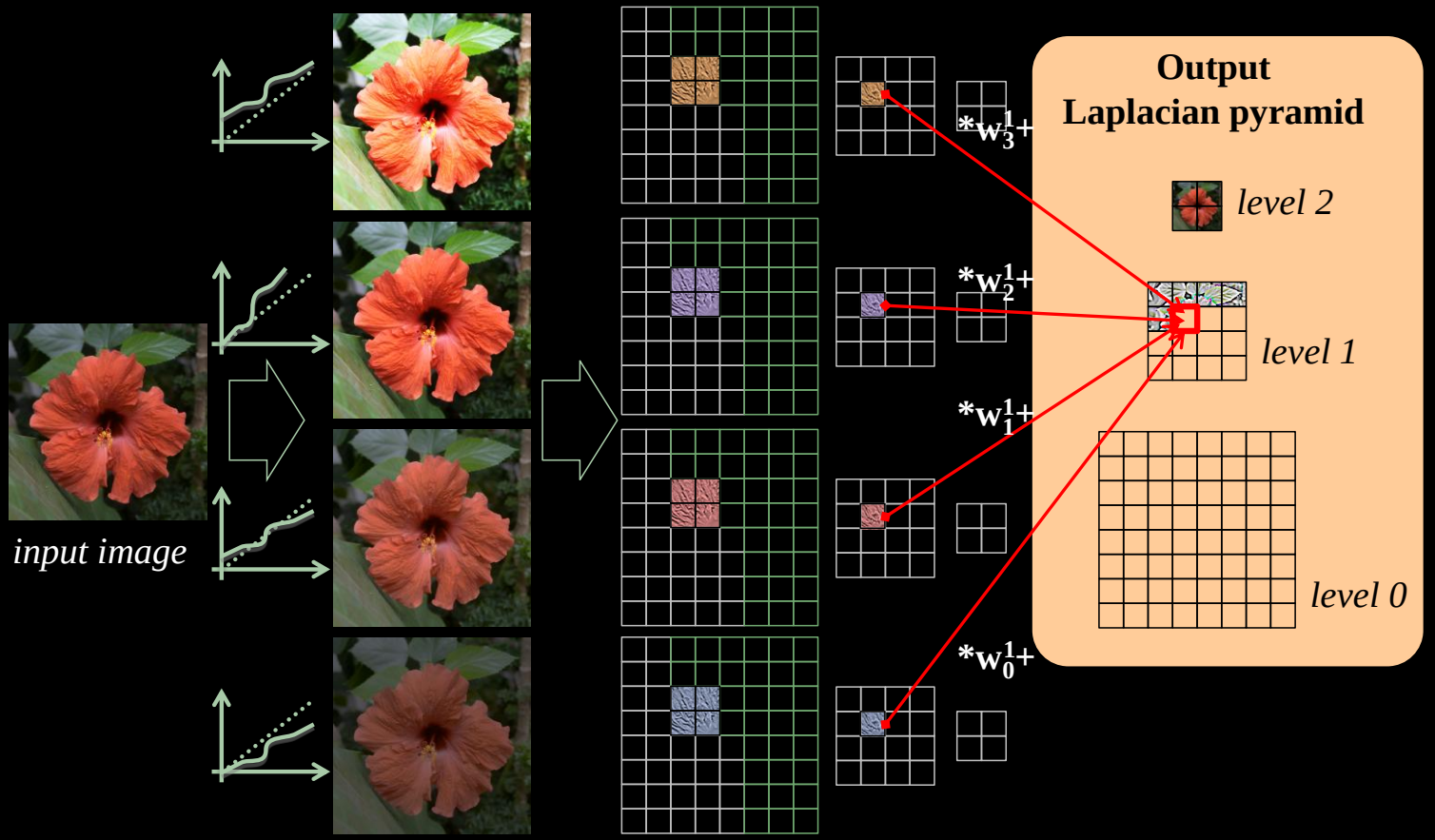
7.1 Naïve Local Laplacian Filters



7.2 Fast Local Laplacian Filters



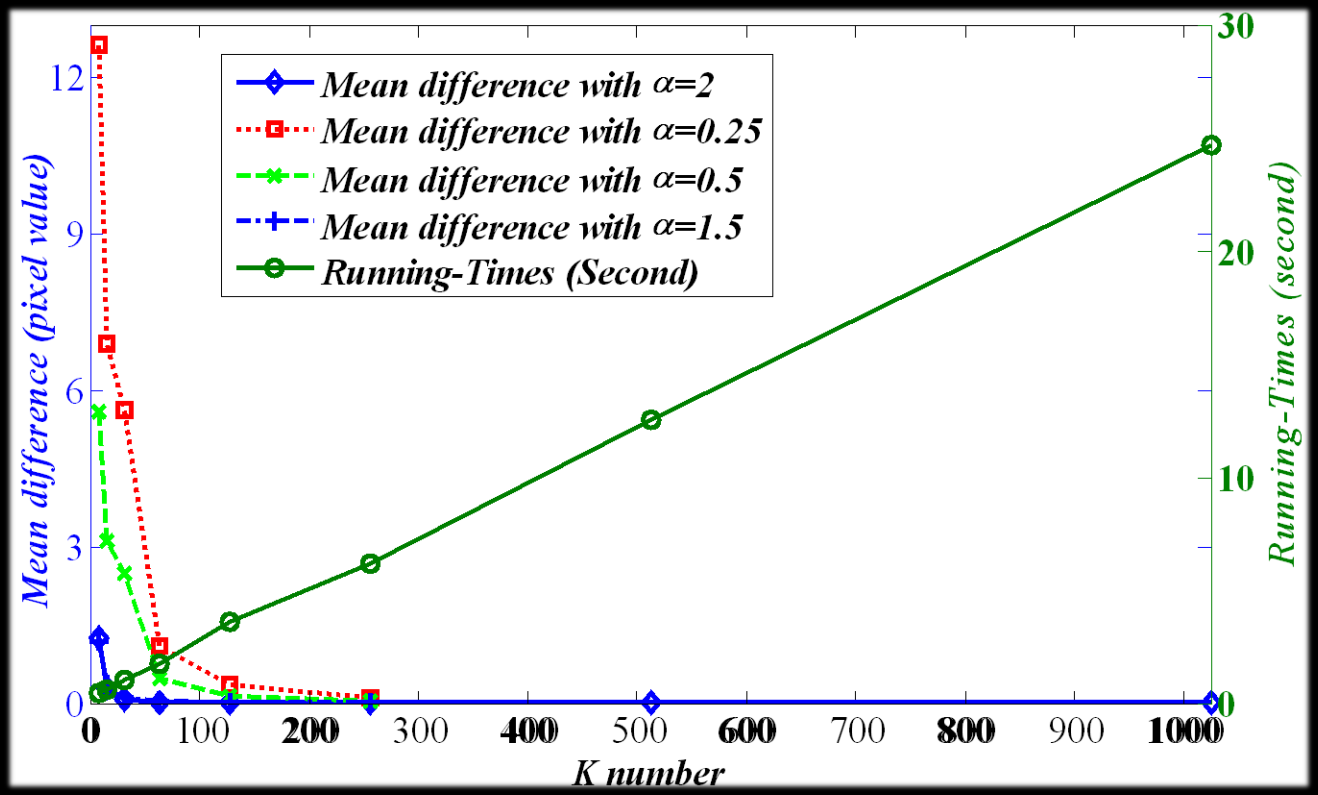
Overview of the basic idea of Fast LLF O(*N*)



全局尝试局部精选
Global-to-Local

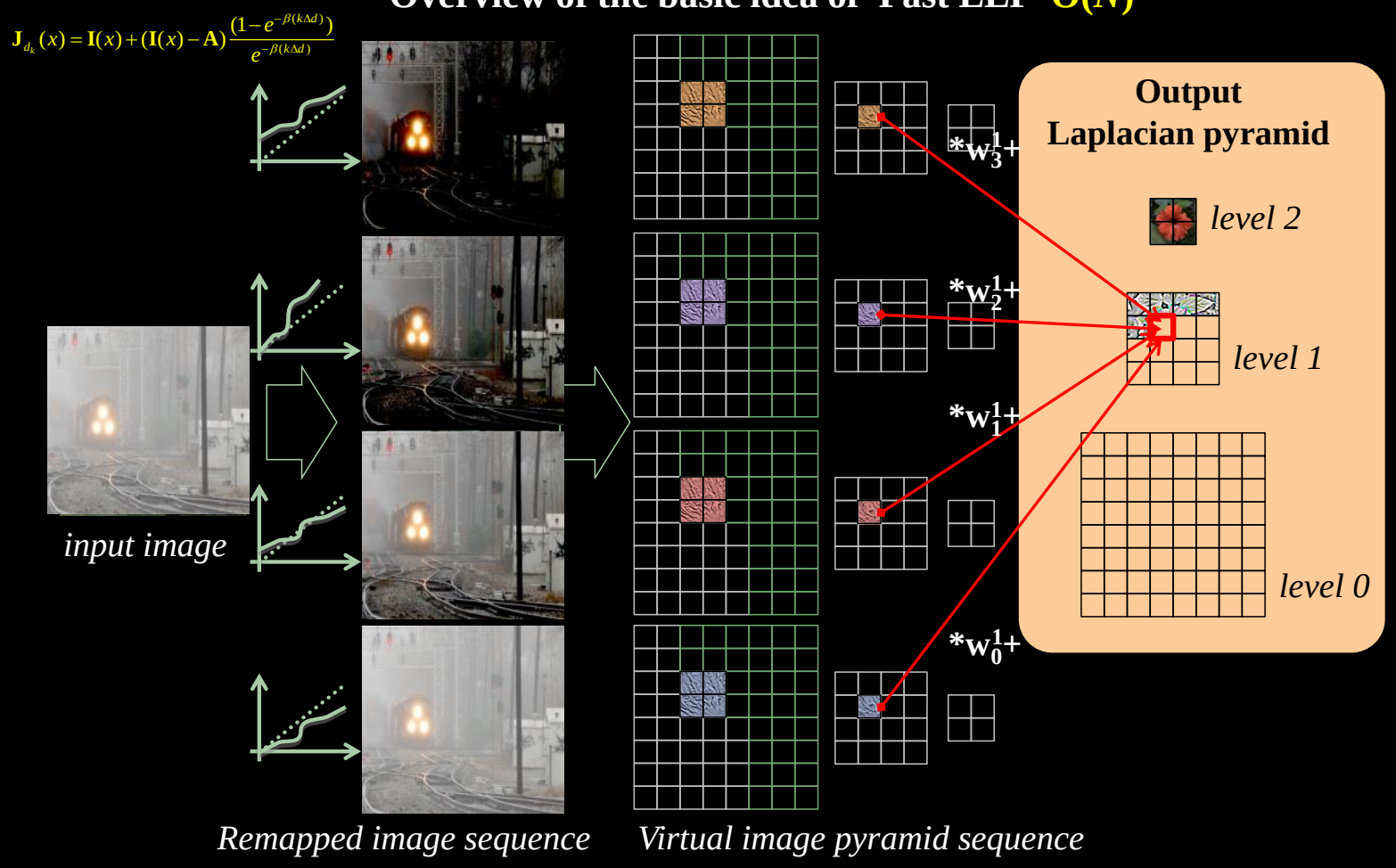
7.3 逼近精度

$$\int_{\Omega_w} \| f(x;q) - f(x;q_0) \| dx \le d \| q - q_0 \|^{\min(1,a)}$$

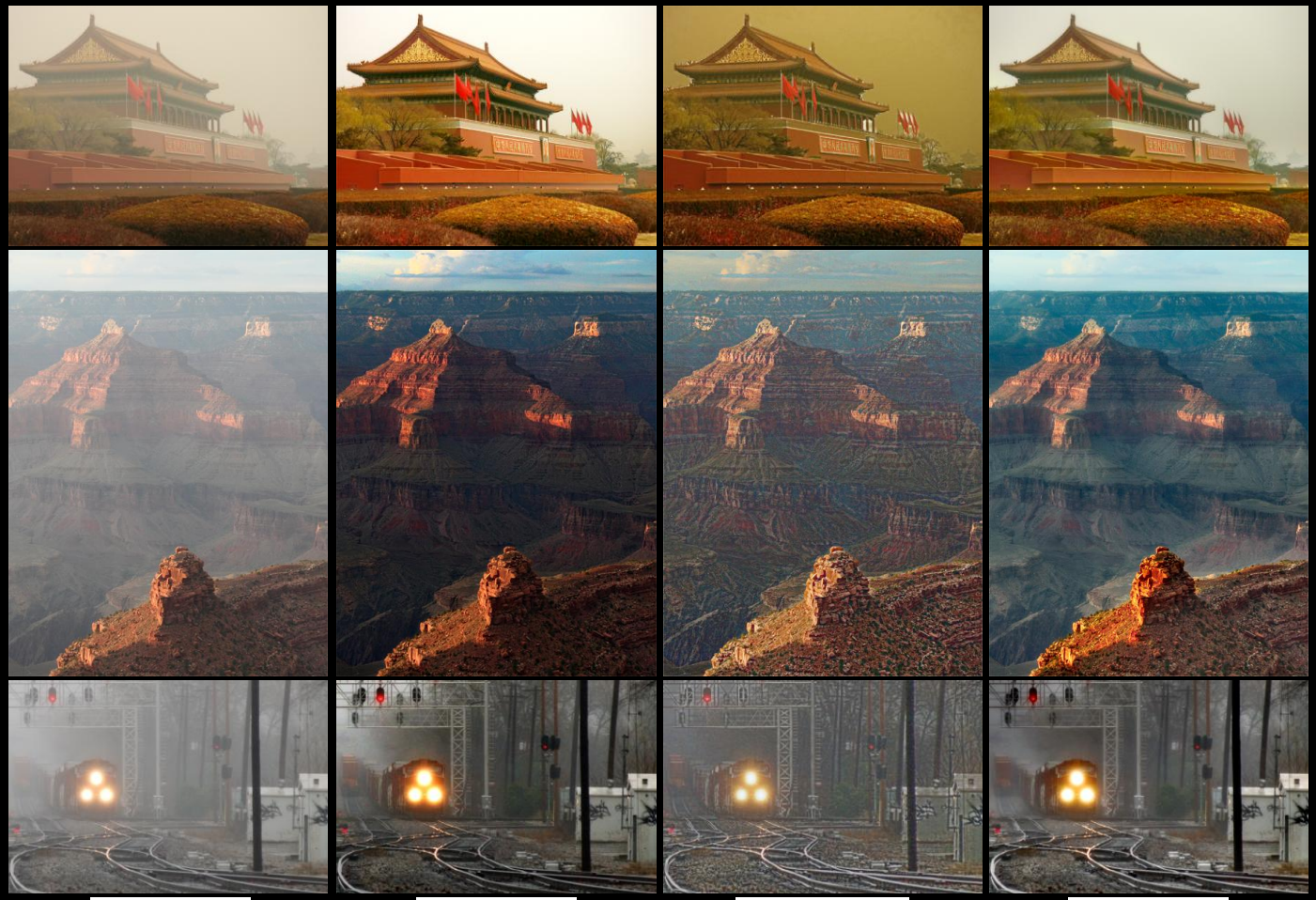


全局尝试局部精选
Global-to-Local

7.5 应用到去雾算法


Overview of the basic idea of Fast LLF O(N)
$J_{d_k}(x) = I(x) + (I(x) - A)\frac{(1 - e^{-\beta(k\Delta d)})}{e^{-\beta(k\Delta d)}}$
input image
Remapped image sequence
Virtual image pyramid sequence
*w_3^1+
*w_2^1+
*w_1^1+
*w_0^1+
Output Laplacian pyramid
level 2
level 1
level 0
全局尝试局部精选
Global-to-Local

7.6 初步实验结果



(a) hazy images (b) He et al. 2009 (c) Tarel et al. 2009 (d) VIPs Fusion