Kernel Regression as a Modern Tool for Image Processing and Vision

Summary

• **Motivation:**
  – Existing methods make strong assumptions about signal and noise models.
  – Develop “universal”, robust methods based on adaptive non-parametric statistics

• **Goal:**
  – Develop the adaptive Kernel Regression framework for a wide class of problems, producing algorithms competitive with state of the art.
The Many Problems of Imaging

- Digital imaging system

Digital still or video camera → Measurements

A real scene → Atmosphere → Lens → A limited number of pixels → Image sensor

Blurring effect → Down-sampling effect → Noise effect

Deblurring problem → Denoising problem

Upscaling problem (or Interpolation)

Data Sampling Scenarios

- Full samples
- Incomplete samples
- Irregular samples

Possibly random positions

Denoising → Denoising + interpolation → Denoising + reconstruction

Want a unified method for treating all these scenarios

Motivating example: Super-resolution

Resolution enhancement from video frames captured by a commercial webcam (3COM Model No.3719)

Irregular Samples by Frame Interlace

Sparse and noisy data
Kernel Regression Framework

• The data model

Zero-mean, i.i.d noise (No other assump.)

\[ y_i = z(x_j) + \varepsilon_i, \quad i = 1, 2, \ldots, P \]

A sample The sampling position The number of samples

The regression function

• The specific form of \( z(x) \)
may remain unspecified.

Kernel Regression: The Local Model

• Local polynomial approximation (1-D)

\[
z(x_i) \approx z(x) + z'(x)(x_i - x) + \cdots + \frac{1}{N!} z^{(N)}(x)(x_i - x)^N
\]

\[ = \beta_0 + \beta_1(x_i - x) + \cdots + \beta_N(x_i - x)^N \]

• ...... and in 2-D

\[
z(x_i) \approx z(x) + \nabla z(x)^T(x_i - x) + \frac{1}{2!} (x_i - x)^T \mathcal{H} z(x) (x_i - x) + \cdots
\]

\[ = \beta_0 + \beta_1^T (x_i - x) + \beta_2 \text{ vech } \{ (x_i - x)(x_i - x)^T \} + \cdots, \]

Unknowns
Optimization Problem

- We have a local representation with respect to each sample:
  \[ y_1 = \beta_0 + \beta_1^T (x_1 - x) + \beta_2^T \text{vech} \left\{ (x_1 - x)(x_1 - x)^T \right\} + \cdots + \varepsilon_1, \]
  \[ y_2 = \beta_0 + \beta_1^T (x_2 - x) + \beta_2^T \text{vech} \left\{ (x_2 - x)(x_2 - x)^T \right\} + \cdots + \varepsilon_2, \]
  \[ \vdots \]
  \[ y_p = \beta_0 + \beta_1^T (x_p - x) + \beta_2^T \text{vech} \left\{ (x_p - x)(x_p - x)^T \right\} + \cdots + \varepsilon_p, \]

- Optimization

\[
\min_{\{\beta_n\}_{n=0}^P} \sum_{i=1}^P \left[ y_i - \beta_0 - \beta_1^T (x_i - x) - \beta_2^T \text{vech} \left\{ (x_i - x)(x_i - x)^T \right\} - \cdots \right]^2 K(x_i - x)
\]

\[ \hat{z}(x) = \sum_{i=1}^P W_i(x, K, h, N) y_i \]

The Influence of Regression Order (1-D)

- Zeroth order (N = 0): Constant model \( \beta_0 \)
- First order (N = 1): Linear model \( \beta_0 + \beta_1 (x - x_i) \)
- Second order (N = 2): Quadratic model \( \beta_0 + \beta_1 (x - x_i) + \beta_2 (x - x_i)^2 \)

- You may have a very different estimated value
Kernel

• **Conditions**
  – Necessary: Non-negative, Uni-modal, Symmetric, Decaying away from the center
  – Desirable: **Small** footprints in densely sampled areas and **larger** ones in sparse areas.

“Classic” L₂ Kernel Regression

• A *local* weighted least square estimator where the weights depend on the **positions** of the nearby data.

\[
\hat{b} = \arg \min_b (y - X_x b)^T W_x (y - X_x b)
\]

\[
b = [\beta_0, \beta_1^T, \ldots, \beta_N^T]^T \quad y = [y_1, y_2, \ldots, y_P]^T
\]

\[
W_x = \text{diag } [K_{h_1}(x_1 - x), K_{h_2}(x_2 - x), \ldots, K_{h_N}(x_P - x)]
\]

\[
X_x = \begin{bmatrix}
1 & (x_1 - x)^T & \text{vech}^T \left\{(x_1 - x)(x_1 - x)^T\right\} & \cdots \\
1 & (x_2 - x)^T & \text{vech}^T \left\{(x_2 - x)(x_2 - x)^T\right\} & \cdots \\
\vdots & \vdots & \vdots & \ddots \\
1 & (x_P - x)^T & \text{vech}^T \left\{(x_P - x)(x_P - x)^T\right\} & \cdots 
\end{bmatrix}
\]
“Classic” $L_2$ Kernel Regression

- We are often only interested in the pixel value. So the estimator is simplified to

$$
\hat{z}(x) = \hat{\beta}_0 = \left[1, 0, 0, \cdots \right] \left( X_x^T W_x X_x \right)^{-1} X_x^T W_x y,
$$

The case of $N=0$:

$$
\hat{z}(x) = \sum_{i=1}^{P} K_{h_i}(x - x_i) y_i
$$

Nadaraya-Watson Estimator (’69)

Locally Linear Estimator

- The optimization yields a pointwise estimator:

$$
\hat{z}(x) = \sum_{i=1}^{P} W_i(x_i, x, K, h, N) y_i
$$

- The bias and variance are related to the regression order and the smoothing parameter:
  - Large $N \rightarrow$ small bias and large variance
  - Large $h \rightarrow$ large bias and small variance
Classic Kernel Regression: What do the weights look like?

- The Net Effect of Classic L₂ KR

\[ \tilde{y}(x) = \sum_{i=1}^{P} W_i(x, K, h, N) y_i \]

Equivalent kernel

Equivalent Kernels: Top View

<table>
<thead>
<tr>
<th>N=0</th>
<th>N=1</th>
<th>N=2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regularly sampled Data case</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Irregularly sampled Data case</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
To summarize so far ……

- **Classic Kernel Regression:**
  Locally **Linear, Shift-varying** Filters:

  \[
  \hat{z}(x_j) = \sum_i W(x_i, x_j) y_i
  \]

- **Data-Adaptive Kernel:**
  Locally **Non-Linear, Shift-varying** Filters:

  \[
  \hat{z}(x_j) = \sum_i W(x_i, x_j; y_i, y_j) y_i
  \]

---

**0-th order Kernel Regression:**

**More general formulation**

- The data:

  \[ y_i = z(x_i) + e_i, \quad \text{for } i = 1, \ldots, n, \]

- The (point) estimate:

  \[
  \hat{z}(x_j) = \arg \min_{z(x_j)} \sum_{i=1}^n [y_i - z(x_j)]^2 \quad \rightarrow \quad \hat{z}(x_j) = \sum_i \frac{1}{N} y_i
  \]

Measure of similarity between two data points i and j.
The More General Framework

• A data-fitting problem

Given samples

The regression function

Zero-mean, i.i.d noise (No other assump.)

The sample position

The number of samples

• The particular form of \( z(x) \)
  may remain unspecified.

(Point) Estimate: Matrix Formulation

• Weighted Least Squares problem:

\[
\hat{z}(x_j) = \arg \min_{z(x_j)} [y - z(x_j)1_n]^T K_j [y - z(x_j)1_n]
\]

where

\[
y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}, \quad 1_n = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}, \quad K_j = \text{diag} \begin{bmatrix} K(x_1, x_j, y_1, y_j) \\ K(x_2, x_j, y_2, y_j) \\ \vdots \\ K(x_n, x_j, y_n, y_j) \end{bmatrix}
\]
Solution: Locally Adaptive Filters

\[ \hat{z}(x_j) = \arg \min_{z(x_j)} \left[ y - z(x_j)1_n \right]^T K_j \left[ y - z(x_j)1_n \right] \]

\[ \downarrow \]

\[ \hat{z}(x_j) = \left( 1_n^T K_j 1_n \right)^{-1} 1_n^T K_j y \]

\[ = \sum_i \frac{K(x_i, x_j, y_i, y_j)}{\sum_i K(x_i, x_j, y_i, y_j)} y_i \]

\[ = \sum_i W_{i, j} y_i \]

= \text{Convex combination of all the data.}

Generalization of:

\[ \hat{z}(x_j) = \sum_i \frac{1}{N} y_i \]

Some Special Cases

- Classical Gaussian Linear Filters:

\[ K(x_i, x_j, y_i, y_j) = \exp \left( \frac{-||x_i - x_j||^2}{h_x^2} \right) \]
Some Special Cases

- **Bilateral Filter** (Tomasi, Manduchi, ‘98)
- **Non-local Means** (Buades et al. ‘05)
- **LARK** (Takeda, Farsiu, Milanfar, ‘07)

\[
K(x_i - x) \cdot K(y_i - y)
\]

- The photometric distance
  \[ \delta y = |y_i - y| \]
- The spatial distance
  \[ \delta x = |x_i - x| \]
- The Euclidean distance
  \[ \sqrt{\delta x^2 + \delta y^2} \]
- The geodesic distance
  \[ \text{(LARK)} \]

**Bilateral Kernel (BL) [Tomasi et al. ‘98]**

\[
K(x_t, x, y_t, y) = \exp \left\{ \frac{||y_t - y||^2}{h_r^2} - \frac{||x_t - x||^2}{h_d^2} \right\}
\]

Pixel similarity = Spatial similarity
Non-local Means (NLM) [Buades et al. ‘05]

\[
K(x_l, x, y_l, y) = \exp \left\{ -\frac{\|y_l - y\|^2}{h_f^2} - \frac{\|x_l - x\|^2}{h_d^2} \right\}
\]

Patch similarity \quad Spatial similarity

→ Smoothing effect

Steering Matrix

\[
J_i = \begin{bmatrix}
\frac{\partial z_{11}(x_j)}{\partial x_i} \\
\frac{\partial z_{21}(x_j)}{\partial x_i} \\
\vdots \\
\frac{\partial z_{12}(x_j)}{\partial x_i} \\
\frac{\partial z_{22}(x_j)}{\partial x_i} \\
\end{bmatrix},
\]

\[
\hat{C}_i^{\text{naive}} = J_i^T J_i
\]

\[x_j \in \xi_i\]
Bilateral and NLM cases: The “Separable” Kernels

- Consider the denoising problem with regularly sampled data

\[ K(x_i - x_j, y_i - y_j) = K_{hs}(x_i - x_j)K_{hr}(y_i - y_j) \]

- Special case: N = 0, m=2

\[ \tilde{z}(x_j) = \frac{\sum_{i=1}^{P} K_{hs}(x_i - x_j)K_{hr}(y_i - y_j)y_i}{\sum_{i=1}^{P} K_{hs}(x_i - x_j)K_{hr}(y_i - y_j)} \]

With Gaussian Kernels, this is just the Bilateral filter! (Tomasi, Elad)

N>0 can generalize the bilateral filter!

Bilateral Filter: General Idea

- Every sample is replaced by a weighted average of its neighbors (as in the WLS),

- These weights reflect two forces
  - How close are the neighbor and the center sample, so that larger weight to closer samples,
  - How similar are the neighbor and the center sample – larger weight to similar samples.

- All the weights should be normalized to preserve the local mean.
In an Equation

The result at the $k$th sample

The weight

The neighbor sample

Normalization of the weighting

Averaging over the $2N+1$ neighborhood

Bilateral Filter: The Weights

$$W_s[k, n] = \exp \left\{ -\frac{d_s^2 [k, [k - n]]}{2\sigma_s^2} \right\} = \exp \left\{ -\frac{n^2}{2\sigma_s^2} \right\}$$

$$W_r[k, n] = \exp \left\{ -\frac{d_r^2 [Y[k], Y[k - n]]}{2\sigma_r^2} \right\} = \exp \left\{ -\frac{(Y[k] - Y[k - n]^2)}{2\sigma_r^2} \right\}$$

$$W[k, n] = W_s[k, n] \cdot W_r[k, n]$$
It is clear that in weighting this neighborhood, we would like to preserve the step.

Bilateral Filter: The Weights

\[ W_R[k, n] = \exp \left\{ \frac{[Y[k] - Y[k - n]]^2}{2\sigma_R^2} \right\} \]

\[ W_S[k, n] = \exp \left\{ -\frac{n^2}{2\sigma_S^2} \right\} \]
It appears that the weight is inversely proportional to the **Total-Distance** (both horizontal and vertical) from the center sample.

\[
W[k, n] = \exp \left\{ \frac{-\sigma_s^2 d^2_s[k, k-n] + \sigma_s^2 d^2_s[y[k], y[k-n]]}{2\sigma_s^2 \sigma_R^2} \right\}
\]

**Bilateral Kernel Properties**

- Per each sample, we can define a 'Kernel' that averages its neighborhood

\[
\sum_{n=-N}^{N} W[k, n]
\]

- This kernel changes from sample to sample!
- The sum of the kernel entries is 1 due to the normalization,
- The center entry in the kernel is the largest,
- Subject to the above, the kernel can take any form (as opposed to filters which are monotonically decreasing).
3.9 Filter Parameters

The filter is controlled by 3 parameters:

\[ \begin{align*} 
N & \quad \text{The size of the filter support,} \\
\sigma_s & \quad \text{The variance of the spatial distances,} \\
\sigma_R & \quad \text{The variance of the spatial distances, and} \\
I_t & \quad \text{The filter can be applied for several iterations in order to further strengthen its edge-preserving smoothing.} 
\end{align*} \]

Additional Comments

The bilateral filter is a powerful filter:

- One application of it gives the effect of numerous iterations using traditional local filters,
- Can work with any reasonable distances \( d_s \) and \( d_R \) definitions,
- Easily extended to higher dimension signals, e.g. Images, video, etc.
- Easily extended to vectored-signals, e.g. Color images, etc.
- But it can be improved upon!
Bilateral Kernels (Low Noise)

\[ K_{\text{bilat}}(x_i - x, y_i - y) = K_{h_s}(x_i - x) \cdot K_{h_r}(y_i - y) \]

Bilateral Kernels (High Noise)

\[ K_{\text{bilat}}(x_i - x, y_i - y) = K_{h_s}(x_i - x) \cdot K_{h_r}(y_i - y) \]

Effectively useless
Important Special Cases

- **Bilateral Filter** (Tomasi, Manduchi, ‘98)
  \[
  K(x_i, x_j, y_i, y_j) = \exp\left\{ \frac{-\|x_i - x_j\|^2}{h_x^2} + \frac{-(y_i - y_j)^2}{h_y^2} \right\}
  \]

- **Non-local Means** (Buades. et al. ‘05)
  \[
  K(x_i, x_j, y_i, y_j) = \exp\left\{ \frac{-\|x_i - x_j\|^2}{h_x^2} + \frac{-\|y_i - y_j\|^2}{h_y^2} \right\}
  \]

- **LARK** (Takeda, Farsiu, Milanfar ‘07)
  \[
  K(x_i, x_j, y_i, y_j) = \exp\left\{ -(x_i - x_j)^T \hat{C}_{ij}(y)(x_i - x_j) \right\}
  \]

“Learned” (Geodesic) Distance Metric

Special Cases: LARK

\[
K(x_i, x_j, y_i, y_j) = \exp\left\{ -(x_i - x_j)^T \hat{C}_{ij}(x_i - x_j) \right\}
\]

Estimated Local gradient covariance
- “Structure Tensor”
- “Metric Tensor”

\[
\hat{C}_{ij} = \sum_j \begin{bmatrix}
\tilde{z}_{i,1}(x_j) & \tilde{z}_{i,2}(x_j) \\
\tilde{z}_{i,1}(x_j) & \tilde{z}_{i,2}(x_j)
\end{bmatrix}
\]
Comparisons

Bilateral  Non-local Means  LARK

Shown in non-overlapping patches
(for convenience of illustration only)

Gradient Covariance Matrix and Local Geometry

Gradient matrix over a local patch:

\[
\hat{C}_x - J_i^T J_i - \sum_j \begin{bmatrix}
\frac{\partial^2}{\partial x_1^2}(x_j) & \frac{\partial^2}{\partial x_1 \partial x_2}(x_j) & \frac{\partial^2}{\partial x_2^2}(x_j)
\end{bmatrix}
\]

\[
= \lambda_1 v_1 v_1^T + \lambda_2 v_2 v_2^T
\]

\[
J_i = \begin{bmatrix}
\frac{\partial}{\partial x_1}(x_j) & \frac{\partial}{\partial x_2}(x_j)
\end{bmatrix} = U_i S_i V_i^T, \quad x_j \in \omega_i
\]

Capturing locally dominant orientations
Locally Dominant Orientation

- Principal component analysis of the local gradients
  \[ \hat{C}_i = J_i^T J_i - \sum_j \begin{bmatrix} z_{x_1}^2(x_j) & z_{x_1}(x_j)z_{x_2}(x_j) \\ z_{x_1}(x_j)z_{x_2}(x_j) & z_{x_2}^2(x_j) \end{bmatrix} \]
  \[ = \lambda_1 v_1 v_1^T + \lambda_2 v_2 v_2^T \]

  The second eigenvector is the local edge orientation.

Interpreting the eigenvalues

Classification of image points using eigenvalues of \( C \):

- \( \lambda_1 \) and \( \lambda_2 \) are small
- \( \lambda_1 > > \lambda_2 \)
- \( \lambda_1 \) and \( \lambda_2 \) are large
- \( \lambda_1 \sim \lambda_2 \)
- \( \lambda_1 > > \lambda_2 \)
- \( \lambda_1 \)

“Corner”

“Edge”

“Flat” region

“Edge”
Corner response function
\[ R = \text{det}(C) - \alpha \text{trace}(C)^2 = \lambda_1\lambda_2 - \alpha(\lambda_1 + \lambda_2)^2 \]

\( \alpha \): constant (0.04 to 0.06)

LARK Kernels

Compute
Locally Adaptive Regression Kernels (LARK)

\[
K(x_i, x_j, y_i, y_j) = \exp \left\{ - (x_i - x_j)^T \hat{C}_{i,j} (x_i - x_j) \right\}
\]

\[
\hat{C}_{i,j} = J_i^T J_i = \sum_j \begin{bmatrix} z_{x,x}(x_j) & z_{x,y}(x_j) \\ z_{y,x}(x_j) & z_{y,y}(x_j) \end{bmatrix}
\]

Comparisons

- Bilateral
- Non-local Means
- LARK
Robustness of LARK Descriptors

Original image  Brightness change  Contrast change  WGN sigma = 10

1

2

3

Some Applications

- Denoising
- Interpolation
- Super-resolution
- Deblurring
Gaussian Noise Removal

Noisy image, RMSE=24.87

LARK
RMSE=6.63

KSVD
Elad, et al. (2007)
RMSE=6.89

BM3D
Foi, et al. (2007)
RMSE=6.35
State of Art for Gaussian Noise

Film Grain Reduction (Real Noise)

Noisy image
Film Grain Reduction (Real Noise)

LARK

KSVD

BM3D
The State of the Art

- "Is Denoising Dead?" [Chatterjee, Milanfar, TIP 2010]
  - For complex images, yes.
  - For simpler images, no!

Adaptive Kernels for Interpolation

- What about missing pixels?
  - Local metric (covariance) is undefined!
    \[ K(C, x_l, x) = \sqrt{\det C_l} \exp \left\{ -\frac{(x_l - x)^T C_l(x_l - x)}{2} \right\} \]
  - Using a "pilot" estimate, fill the missing pixels:
Adaptive Sharpening/Denoising

- Sharpening the LARK Kernel

\[ S = K - \kappa L \otimes K \]
Automatically setting the local sharpness

• Local Measure of Sharpness: \[ Q = \frac{s_1 - s_2}{s_1 + s_2} \]


LARK-based Simultaneous Sharpening/Denoising

• Net effect:
  – aggressive denoising in “flat” areas
  – Selective denoising and sharpening in “edgy” areas

Locally adaptive denoise/deblur filters
Examples

original image  proposed method

original image

State-of-the-art Methods
Experimental Results 2

Super-resolution

Motion Estimation → Adaptive Kernel Regression (2D)
The Matrix Formulation

- Collect the vector formulation for all \( j = 1, \ldots, n \)

\[
\tilde{z}(x_j) = \sum_i W_{i,j} y_i = w_j^T y.
\]

\[
\tilde{z} = \begin{bmatrix}
w_1^T \\
w_2^T \\
\vdots \\
w_n^T
\end{bmatrix} y = W y.
\]

Generally data-dependent

The Matrix \( W \)

- \( W \) is very special:

\[
w_j^T y = \sum_i W_{i,j} y_i = \sum_i \frac{K_{i,j}}{\sum_i K_{i,j}} y_i
\]

\[
W = D^{-1} K, \quad \text{where } D_{jj} = \text{diag}\{\sum_i K_{i,j}\}
\]

\[
W = D^{-1} K = D^{-1/2} D^{-1/2} K D^{-1/2} D^{1/2}
\]

Positive Definite

- \( W \) is positive definite, (almost) symmetric
Properties of $W$

- Key properties (Perron-Frobenius):
  - $W$ is positive definite and row-stochastic ($w_j^T 1_n = 1$)
  - $W$ has spectral radius $\lambda_1(W) = 1$
  - Dominant eigen-vector: $v_1 = \frac{1}{\sqrt{n}} 1_n$
  - Ergodicity: $\lim_{k \rightarrow \infty} W^k = v_1 u_1^T = 1_n u_1^T$
Other Interpretations of $W$

- Probability Transition Matrix for a Markov Chain
- Graphical Models of Data
- Spectral Methods

\[
W = D^{-1} K
\]

\[
\mathcal{L} = D^{1/2} W D^{-1/2} - I
\]

“Graph Laplacian”

Properties of $W$ and Filter Performance

- Spectral Decomposition of $W$

\[
W = VSV^T
\]

where $S = \text{diag} [\lambda_1, \cdots, \lambda_n]$

\[0 \leq \lambda_n \leq \cdots \leq \lambda_1 = 1.\]

\[
W^k = VS^kV^T = VS^kV^T = \sum_{i=1}^{n} \lambda_i^k v_i v_i^T
\]

- Use this to study the statistical performance
Statistical Analysis of Filters

• Bias  \[ \|\text{bias}\|^2 \approx \|(W - I)z\|^2 \]

• Variance

\[ \text{cov}(\tilde{z}) = \text{cov}(Wy) \approx \text{cov}(We) = \sigma^2 W W^T \]

• Mean-Squared Error

\[ \text{MSE} = \|\text{bias}\|^2 + \text{tr}(\text{cov}(\tilde{z})) \]

Statistical Analysis of Filters

• With \( W = VSV^T \) write \( z = Vb_0 \)

\[ \text{MSE} = \sum_{i=1}^{n} \left( \lambda_i - 1 \right)^2 b_{0i}^2 + \sigma^2 \lambda_i^2 \]

Signal coefficients in the basis given by eigenvectors of \( W \)

Bias\(^2\)  Variance
An Observation

• What is the “ideal” spectrum for \( W \)?

• Minimize the Mean-Squared Error w.r.t. \( \lambda_i \)

\[
\text{MSE}(\lambda_i) = \sum_{i=1}^{n} (\lambda_i - 1)^2 b_{0i}^2 + \sigma^2 \lambda_i^2
\]

• Optimal Spectrum:

\[
\lambda_i^* = \frac{b_{0i}^2}{b_{0i}^2 + \sigma^2} = \frac{1}{1 + \text{snr}_i^{-1}}
\]

“Ideal” Wiener Filter

• Explains performance of state of the art denoising

Improving the “non-ideal” Filter \( \hat{z} = Wy \)

• Diffusion (Perona, Malik ’90, Coifman et al ’06, ….)

\[
\hat{z}_k = W \hat{z}_{k-1}
\]

\[
\hat{z}_k - \hat{z}_{k-1} = (W - I) \hat{z}_{k-1}
\]

“Laplacian”

• Repeated Application of \( W \)
  – Result: Ultimately an over-smoothed image

\[
\text{MSE}_k = \sum_{i=1}^{n} (\lambda_i^k - 1)^2 b_{0i}^2 + \sigma^2 \lambda_i^{2k}
\]

Bias ↑ Variance ↓
Improving the Estimates II

- **Residual Iterations**: Adding “roughness” to the estimate
  \[
  \hat{z}_k = \hat{z}_{k-1} + W(y - \hat{z}_{k-1})
  \]

- **Example**:
  \[
  \hat{z}_1 = \hat{z}_0 + W(y - \hat{z}_0)
  = Wy + W(y - Wy)
  = (2I - W)Wy
  \]

Statistical Performance Analysis

- **Diffusion**
  \[
  \text{MSE}_k = \sum_{i=1}^{n} \left( \frac{\lambda_i^k - 1}{b_{0i}} \right)^2 + \sigma^2 \lambda_i^{2k}
  \]

- **Residual**
  \[
  \text{MSE}_k = \sum_{i=1}^{n} (1 - \lambda_i)^{2k+2} b_{0i}^2 + \sigma^2 (1 - (1 - \lambda_i)^{k+1})^2
  \]
Examples (LARK Filter):

Examples (NLM Filter):
Which to Use?

- Depends on the filter, the underlying image, and the noise variance.

  - **Diffusion** better if filter is “weak”
    - Doesn’t do much denoising

  - **Residual** better if filter is “aggressive”
    - Over-smooths (signal left in residuals)

Relations to Empirical Bayes

- Regularization

\[
\hat{z} = \arg\min_z \frac{1}{2} \|y - z\|^2 + \frac{\lambda}{2} R(y, z)
\]

  - **Empirical log-Prior**

- Steepest Descent Iteration:

\[
\hat{z}_k = \hat{z}_{k-1} - \mu \left[ (\hat{z}_{k-1} - y) + \lambda \nabla R(y, z_{k-1}) \right]
\]

  - **Step size**
  - **Gradient**
Relations to Empirical Bayes

1. MAP SD: \[ \hat{z}_{k+1} = \hat{z}_k - \mu \left[ (\hat{z}_k - y) + \lambda \nabla R(\hat{z}_k) \right] \]

2. Residuals: \[ \hat{z}_{k+1} = \hat{z}_k + W(y - \hat{z}_k) \]

3. Diffusion: \[ \hat{z}_{k+1} = \hat{z}_k + (W - I) \hat{z}_k \]

\[ \nabla R(z_k) = \frac{-1}{\mu \lambda} (W - \mu I)(y - \hat{z}_k) \]

\[ \nabla R(z_k) = \frac{1}{\mu \lambda} (W - (1 - \mu)I)(y - \hat{z}_k) - \frac{1}{\mu \lambda} (I - W) \tilde{y} \]
Empirical (log-) Priors

- **Residuals:**
  \[
  \hat{R}(z) = \frac{1}{2\mu\lambda} (y - \hat{z})^T (W - \mu I) (y - \hat{z})
  \]

- **Diffusion:**
  \[
  \hat{R}(z) = \frac{1}{2\mu\lambda} (y - \hat{z})^T ((1 - \mu)I - W)(y - \hat{z}) + \frac{1}{\mu\lambda} y^T (I - W) \hat{z}
  \]

\[
\hat{p}(z) = c \exp \left[ -\hat{R}(z) \right]
\]

3-D Space-Time processing
Space–Time Descriptors

- Setup is similar to 2-D, but....
- Samples from nearby frames
- Covariance matrix is now 3x3
  - Contains implicit motion information
- Space-time processing

\[ J = \begin{bmatrix}
  z_x(x_1) & z_x(x_2) & z_x(x_3) \\
  z_p(x_1) & z_p(x_2) & z_p(x_3)
\end{bmatrix} \]

Spatial gradients
Temporal gradients

Super-resolution without Motion Estimation

Adaptive Kernel Regression (3D)

Original
(QCIF, 144 x 176, 12 frames)

x3 in space, x2 in time
Upscaled video
(432 x 528, 24 frames)
Space-Time Upscaling + Deblur

Another Application:
Removing Atmospheric Turbulence
What is the Problem?

Assumption:
Both scene and camera are static.
Atmospheric Turbulence

Effects of atmospheric turbulence:
1. Geometric distortion
2. Space and time-varying blur

Degradation model for the $k$-th frame:

$$G_k[x] = (F \ast h_{k,x} \ast h_l)[x] + N_k[x]$$

Goal: to restore a single high quality image from the observed sequence $\{G_k\}$, $k = 1, \ldots, n$

Alternative: Adaptive Optics

Even more Expensive, Large Systems
**Isoplanatic Patches**

Global imaging model:

\[ G_k[x] = (F \otimes h_{k,x} \otimes h)[x] + N_k[x] \]

Local imaging model (patch-wise)

Turbulence-caused PSF (spatially invariant)

\[ g_k = f \otimes h_k \otimes h + n_k \]

= \[ f \otimes h \otimes h_k + n_k \]

= \[ n \otimes h_k + n_k \]

The PSFs are locally constant (isoplanatic.)

**Proposed Framework**

- Observed sequence \( \{G_k\} \)
- Near-diffraction-limited image \( Z \)
- Non-Rigid Image Registration
- Near-Diffraction-Limited Image Reconstruction
- Single Image Blind Deconvolution
- Output \( \hat{f} \)
Non-Rigid Image Deformation

The movement of a given pixel $x = (x, y)^T$ can be described through the motion of the control points:

$$W(x; \mathbf{p}) = x + A(x) \mathbf{p}$$

Motion vector

Deformation vector $\mathbf{p} = [\Delta x_1, \ldots, \Delta x_n, \Delta y_1, \ldots, \Delta y_n]^T$ denotes control points’ movement.

$$A(x) = \begin{bmatrix}
    c_1 & \cdots & c_n & 0 & \cdots & 0 \\
    0 & \cdots & 0 & c_1 & \cdots & c_n
\end{bmatrix}$$

is a spline function-based weight matrix.

Non-Rigid Image Registration

To improve the estimation forward deformation in $\mathbf{p}$, the following:

$$C(\mathbf{p}, \mathbf{\hat{p}}) = \sum_x |G(W(x; \mathbf{p}))/\mathbf{p}| \mathbf{x} \sum_x [\sum_x |G(W(x; \mathbf{p}))/\mathbf{p}| \mathbf{x}]^2 + \gamma(\mathbf{p} + \mathbf{\hat{p}})^T(\mathbf{p} + \mathbf{\hat{p}})$$

Forward fidelity term

Forward fidelity term

Symmetry constraint

We obtain a registered frame sequence $\{R_k\}$ with minimal geometric distortion.
What Happens to the Local PSFs?

Consider the registered frame sequence \( \{ R_k \} \) locally:

\[
\begin{align*}
    r_k &= \delta_{\Delta x} \otimes g_k \\
    \text{Registered isoplanatic patch} &= \delta_{\Delta x} \otimes (h_k \otimes z + n_k) \\
    &= \tilde{h}_k \otimes z + \tilde{n}_k \\
    &= q_k + \tilde{n}_k
\end{align*}
\]

where \( \tilde{h}_k = \delta_{\Delta x} \otimes h_k \) is the registered (shifted) PSF,

and \( \tilde{n}_k = \delta_{\Delta x} \otimes n_k \) is the shifted noise.

After registration, the new PSF's are shifted versions of the originals.

Blurry Image Reconstruction

Recover the image \( (z) \) that is \( \sim \) uniformly blurry.
Sharpest Patch Detection

Once we have sufficient observations, the sharpest patch (in the frame) can be viewed as:

\[ r_\ell = \hat{r}_\ell + \tilde{n}_\ell \approx z + \tilde{n}_\ell \]

near-diffraction-limited patch

Sharpness can be measured through patch intensity variance:

\[ \text{var}(r_k) = \frac{1}{L^2 - 1} \sum_x (r_k[x] - \bar{r}_k)^2 \]

The pixel values \( n_\ell \) are contaminated by noise, which can cause artifacts in the subsequent deconvolution step.

Temporal Kernel Regression

\[ \hat{z}(x) = \arg \min_{q_i(x)} \sum_j |r_j(x) - q_\ell(x)|^2 U(\ell, j) \]

Weights measure the similarity of collocated patches

\[ U(\ell, j) = \exp \left( -\frac{||r_\ell - r_j||^2}{N^2 \sigma^2} \right) \]

Patch size Smoothing parameter

The solution is:

\[ \hat{z}(x) = \frac{\sum_j U(\ell, j)r_j(x)}{\sum_j U(\ell, j)} \]
Blind Deconvolution

Once near-diffraction-limited image \( Z = F \otimes h + \varepsilon \) is obtained, the latent image \( \hat{F} \) can be estimated through:

\[
< \hat{F}, \hat{h} > = \arg \min_{\hat{F}, \hat{h}} \| Z - F \otimes h \|^2 + \lambda_1 \Phi_1(F) + \lambda_2 \Phi_2(h)
\]


Results I

“Ground truth”  
Input video (237x237x100)  
Registered video

Turbulence is caused by the hot air exhausted from a building’s vent.

*Courtesy of Dr. S. Harmeling from Max Planck Institute for Biological Cybernetics, and is also used in his CVPR 2010 paper.*
EE 264: Image Processing and Reconstruction

Results I

“Ground truth”

Hirsch et al. CVPR 2010

Proposed

Results II

Input video Water Tower*
(300x220x80)

Registered video

Top part of a water tower imaged at a (horizontal) distance of 2.4 km.

*Courtesy of Prof. Mikhail A. Vorontsov from the Intelligent Optics Lab of the University of Maryland / Dayton.
Results II

One input

Output

Additional Comparisons

Lucky region
(Vorontsov et al. 2009)

Proposed approach
Astronomical Imaging

Moon Surface (410x380x80)

Moon surface captured by a ground-based telescope. Video courtesy of NASA Langley Research Center.

Registered Sequence
Output
Proposed Approach

Computer Vision Applications:
Object (2-D) and Action (3-D) Detection and Recognition

......... from one example
Take a look at this:

See it here?
How about here?

Or here?
Single Example, No Training!

(Most) people can find the Dragon Fruit from one look

Even if they’ve never seen it before.

Holy Grail of Computer Vision:
“Visual Search”

“Visual Search”: Robustly detect objects/actions of interest within images/videos from a single query

1) Whether objects (actions) are present or not,
2) How many objects (actions)?
3) Where are they located?
Summary: Object Detection with Local Regression Kernels

Query
Target

Descriptors
- LARK
- LSK
- NLM
- BL
- SSIM
- SIFT
- HOG

Compare Visual Features
Target with bounding box

Some sample results:
Some Examples

Query

Target

Target

query

target

Experimental Results

query

Target

Target

Target
Some Examples

Hand-drawn Query

Targets
Action Detection Example

- No Motion Estimation
- No Segmentation
- No Learning
- No Prior Information

(Multiple Actions)
A Final Thought

• **Non-parametric Regression** is a SLAM
  
  – Simple (Weighted Least-Squares)
  – Local (Parallelizable)
  – Adaptive (Data-dependent processing)
  – Many uses (State of the art)

• This is “Modern Image Processing”
  – Use it!