Iterative & Multiframe

by Erol Seke

For the course “Image Restoration”

OSMANGAZI UNIVERSITY
Five Amendments (thou shall not ignore them)

Things to ponder before we delve in

1. By restoration we do not mean visual enhancement
2. A restored image has more information than the one not restored.
3. In order to perform restoration we need additional information beside the image.
4. Additional information sources can be
   a. Expectations
   b. Known/estimated imaging model (blur, noise, spatial/freq. characteristics etc).
   c. Other images from the same sensor and/or same scene.
5. Amount of additional information determines the amount of possible restoration but it is limited/finite even if the information is utilized optimally.

Summary:
Absence of additional information means no restoration.
In such a case, what is done is just visual enhancement.
Imaging Model

actual scene

acquired digital image(s)

Estimated Model

estimated image

calculated image(s)

feedback

image(s) in hand

iterate until this quantity is small or not changing
Motive in considering multi-frame restoration:
Multiple somewhat independent information source

Simplest model has
- **Shift**: spatial translations on the image plane
- **Rotate**: rigid rotations about sensor axis
- **Blur**: usually assumed common/spatially invariant
- **Noise**: additive and uncorrelated
- **Sample**: continuous to discrete
Consider a zero mean ergodic random process

assume $n_2 >> n_1$

assume $i_2 >> i_1$

assume $N$ is sufficiently large

any randomly selected set from these random samples makes up another random sequence with the same characteristics.

(definition of ergodicity)
Consider another ergodic random source with mean $m$

$$N = 1 + n_2 - n_1$$

$$\frac{1}{N} \sum_{n=n_1}^{n_2} x_i(n) \rightarrow m$$

assuming $n_2 \gg n_1$

**Corollary:**
the average of a set of samples gets closer to mean $m$
as the number of samples in the set increases.
(That is the definition of mean for the RPs, anyway.)
Noise Reduction via Averaging

\[ I_R = \frac{1}{N} \sum_{i=1}^{N} I_i \]

(example has 8 noisy images)
Images shifted by 0,1,2 pixels in both direction

Clearly, the images must be properly registered before averaging. Otherwise: The noise is reduced again, but the image is blurred.

\[ I_S = \frac{1}{N} \sum_{i=1}^{N} I_i \]

The sum of not-shifted images (from previous slide)
How do we have such image sets?

| aerial photos | brain PET data† | avg of registered images |

so we need to have some kind of interpolation in order to calculate the value of the desired image pixels
Interpolation

missing data (Q: what was it before we lost it?)

\[ x(n) \]

\[ n \]

answer 1

\[ x(n) \]

\[ n \]

answer 2

\[ x(n) \]

\[ n \]

a) it was the same as the previous one.
b) it was the same as the next one.

closest one: nearest neighbor, zero order interpolation

it was on the line passing through two nearest known points

linear / first order interpolation
answer 3

2nd order polynomial passing through three points

2nd order polynomial curve fit
a) use 2 points before and 1 point after
b) use 1 point before and 2 points after
c) no, use 3 points before the null point.
d) certainly the possibilities are increasing

answer 4

3rd order polynomial passing through 4 points

The genuine question: Why polynomials?
The inevitable answer: there is no reason

it just looks simple and understandable, but may be time consuming on the other hand.

We need an equally simple but accurate(?) technique which is flexible enough to allow many missing points.
weighted average: Neighboring points are given weights according to their distance from the unknown point and weighted average is calculated.

\[ \hat{x}(p) = \frac{\sum_{i=n_1}^{n_2} w_i x(i)}{\sum_{i=n_1}^{n_2} w_i} \quad \text{and usually} \quad \sum_{i=n_1}^{n_2} w_i = 1 \]

The technique is simple to apply in 2D too.

\[ \hat{x}(p) = \sum_{i=1}^{7} w_i x(i) \]
All goes back to the sampling theorem

"Any signal can be reconstructed from its samples if it is sampled at a rate higher than Nyquist rate"

and this is how

\[ x(t) = \sum_{n=-\infty}^{n=\infty} x(nT_s) \text{sinc}(2f_m(t-nT_s)) \]  

(reeception formula)

\[ f_m = \frac{1}{2T_s} \]

\( f_m \) must be higher than the highest frequency component in the signal

If we want the value at a point between available samples: Ideally, we should reconstruct the signal using all the samples and find the value(s) everywhere. Is it feasible? Nope.

So, we just assume that the signal is sampled at a rate much higher than the Nyquist rate and that the value of a point can be approximated using the sinc(.)s nearby.
it is assumed that other \( \text{sinc}(.) \) functions are too small to bother here

Calculating several \( \text{sinc}(.) \) values for a given point is expensive, although best. Therefore we use other functions (interpolation kernels) emulating such a summation.
Interpolation during image resize

original

enlarged

bilinear interpolation

pixel duplication (nearest neighbor)
\[ N = \text{imresize}(I, [202, 202], 'nearest', 'bilinear', 'bicubic') \]

\[ F = \text{abs}(\text{ifft2}((\text{fftshift}(\text{fft2}(I)), 202, 202))) \]
• registration is in sub-pixel level
• translations require interpolation
subpixel registration is performed using;

now we can sharpen a little bit
Spatial Transformations

The task of finding

\[ I_2(x', y') = g(I_1(x, y)) \]

intensity relation

and

\[ (x', y') = (f_x(x, y), f_y(x, y)) \]

spatial coordinate relation

between two or more images is called image registration.

One of the simplest spatial transformation is the one with rotation and translation only.

\[
\begin{bmatrix}
  x' \\
  y'
\end{bmatrix} = \begin{bmatrix}
  \cos(\alpha) & -\sin(\alpha) \\
  \sin(\alpha) & \cos(\alpha)
\end{bmatrix} \begin{bmatrix}
  x \\
  y
\end{bmatrix} + \begin{bmatrix}
  x_o \\
  y_o
\end{bmatrix}
\]

\[ I_1 \quad f_x, f_y \quad I_2 \]
Scale can be included as a scalar in front of the 4x4 matrix. Replacing the matrix elements with independent scalars as

\[
\begin{bmatrix}
  x' \\
  y'
\end{bmatrix} =
\begin{bmatrix}
  a_1 & a_2 \\
  a_4 & a_5
\end{bmatrix}
\begin{bmatrix}
  x \\
  y
\end{bmatrix} +
\begin{bmatrix}
  a_3 \\
  a_6
\end{bmatrix}
\]

we can represent other spatial operations like shear, skew and aspect ratio changes. In such a transformation parallel lines remain parallel lines, but angles may change.

Such transformations are called **affine transformations**

Homework: create N random points in 2D, apply translation+rotation+scaling, display both sets
the operation is no longer rigid
skew and stretch (squeeze) is applied

parallel lines are still parallel

angles are changed

Since we have 6 unknowns in
\[
\begin{bmatrix}
x' \\
y'
\end{bmatrix} = \begin{bmatrix}
a_1 & a_2 \\
a_4 & a_5
\end{bmatrix} \begin{bmatrix}
x \\
y
\end{bmatrix} + \begin{bmatrix}
a_3 \\
a_6
\end{bmatrix}
\]
or

\[X' = RX + T\]

we only need 3 corresponding point pairs in two images in order to determine the transformation.
Three example points are marked on images with red, green and blue.
Any three point pairs would do, according to the analytic calculus. Another example triplet is marked with magenta on the images. The problem with these points however, is that they are too close, therefore the results are highly sensitive to errors in coordinates.

**Homework**: calculate the 6 parameters for given 3 point pairs.

How do we select these 3 point pairs?

- **Manual**
- **Automatic**
- Accuracy?

\[
\begin{bmatrix}
\sum x_j^2 & \sum x_j y_j & \sum x_j & 0 & 0 & 0 \\
\sum x_j y_j & \sum y_j^2 & \sum y_j & 0 & 0 & 0 \\
\sum x_j & \sum y_j & \sum 1 & 0 & 0 & 0 \\
0 & 0 & 0 & \sum x_j^2 & \sum x_j y_j & \sum x_j \\
0 & 0 & 0 & \sum x_j y_j & \sum y_j^2 & \sum y_j \\
0 & 0 & 0 & \sum x_j & \sum y_j & \sum 1
\end{bmatrix}
\begin{bmatrix}
a_1 \\
a_2 \\
a_3 \\
a_4 \\
a_5 \\
a_6
\end{bmatrix}
= 
\begin{bmatrix}
\sum x_j x_j \\
\sum x_j y_j \\
\sum x_j \\
\sum y_j x_j \\
\sum y_j y_j \\
\sum y_j
\end{bmatrix}
\]

- **Find many points (corners, other feature points)**
- **Use least-squares method**
- **Spurious points?**
- **Missing points?**
- **Finding correspondence?**

Is this transformation realistic?

**Homework**: check how to find least squares solutions using MATLAB
Correct determination of correspondence between feature point sets extracted from two images is very crucial. But the difficulties are;
1. Some points have no corresponding points on the other set
2. Point locations may have errors because of noise.
3. Points may be too close preventing the detection of correspondence
4. Too many points mean increasing complexity

example: 1000± points in each set

\[ \begin{align*}
\text{Angle} &= 50^\circ \\
\text{Scale} &= 2.2 \\
X_s &= 40 \\
Y_s &= 30
\end{align*} \]
Select 3† points from each set

Calculate transformation using 3 points

Validated?

Yes

Calculate transformation using all corresponding point pairs (least squares)

Transform parameters

No

For each point in first set

Apply transformation

Find closest point in the other set

Reject outliers

Enough number of correspondences

Some intelligence in many steps would fasten the algorithm

† Planar data. For higher dimensions one needs more points
Detection of Corners as Feature Points

Even the images with simplest geometric shapes will have many corners. It is not unusual to have thousands of feature points detected in an image.

But sometimes not enough corners exist.

Resolution affects the number of corners detected.

Basic Corner Detection Principle

All directional derivatives change for the points neighboring the examined point if it is a corner.

Some directional derivatives do not change if the examined point is not a corner.

When 3D scene is projected onto a 2D plane, distant objects appear smaller.

General 8 parameter projective transformation:

\[
\begin{align*}
(x', y') &= (\frac{fx}{f - z}, \frac{fy}{f - z}) \\
x' &= \frac{a_1 x + a_2 y + a_3}{a_7 x + a_8 y + 1} \\
y' &= \frac{a_4 x + a_5 y + a_6}{a_7 x + a_8 y + 1}
\end{align*}
\]
In order to speed up the conversion smaller versions of the images (resized with ‘nearest’) can be used.
Expand \( \sin(.) \) and \( \cos(.) \) terms to their Taylor series and use only first two terms

\[
g(x, y) \approx f(x + x_o - y\alpha - \frac{x\alpha^2}{2}, y + y_o + x\alpha - \frac{y\alpha^2}{2})
\]

Expand \( f \) to Taylor series and use first two terms

\[
g(x, y) \approx f(x, y) + \left( x_o - y\alpha - \frac{x\alpha^2}{2} \right) \frac{\partial f}{\partial x} + \left( y_o - x\alpha - \frac{y\alpha^2}{2} \right) \frac{\partial f}{\partial y}
\]

The difference between \( f \) and \( g \) is

\[
E(x_o, y_o, \alpha) = \sum \left( f(x, y) + \left( x_o - y\alpha - \frac{x\alpha^2}{2} \right) \frac{\partial f}{\partial x} + \left( y_o - x\alpha - \frac{y\alpha^2}{2} \right) \frac{\partial f}{\partial y} - g(x, y) \right)^2
\]

\[
\frac{\partial}{\partial x_o} E(x_o, y_o, \alpha) = 0
\]

\[
\frac{\partial}{\partial y_o} E(x_o, y_o, \alpha) = 0
\]

\[
\frac{\partial}{\partial \alpha} E(x_o, y_o, \alpha) = 0
\]

This technique is useful only if the motion/rotation parameters are very small. Iterative/multi-resolution techniques are proposed to overcome this limitation.
1. Select one of the images as reference
2. Find rotations of the other images
3. Undo rotations
4. Find translations of other images
5. Undo translations
6. Until satisfaction repeat from 2.
7. Using rotation and translation parameters construct an irregular sample field.
8. Interpolate regular sample points from the irregular sample points

Translations must be found in sub-pixel level, therefore it is a good idea to up-sample (resize-bilinear) the images first. Rotations must be as accurate as necessary that the errors at the image edges are less than half a pixel size.
Bigger Picture

A set of frames
(a movie sequence?)

Model

Registration

Superresolution

Restoration

Frequency Domain Techniques
- alias calculation
- best fit

Spatial Domain Techniques
- interpolation
- backprojection
- probabilistic
- best fit

(…and their famous variants / hybrids)
Aliasing

\[ x(t) \]

\[ x(n) \]

\[ |X(f)| \]

\[ |X(k)| \]

Aliased components

\[ \ldots \]

\[ \ldots \]
Since the unaliased components must be the same the average is taken for them. The (complex) sum of aliased terms are known. If there are enough $x_i(n)$s then aliased terms can be calculated provided that the relative sample locations (phase) are known (registered).

Recursive

Let $Y$ be N.M.px1 matrix of pixels obtained by canonically ordering p NxM LR image(s).

$X$ be $r^2$.N.Mx1 matrix of pixels of HR image where $r$ is the enlargement factor.

$D$ be N.M.pxr$^2$.N.M matrix representing downsampling, blur, shift etc. (imaging model)

$n$ noise term

\[ Y = DX + n \]  

($\hat{X}$ is the term we want)

Obviously, it is not feasible to try to solve the system by means of matrix inversions because of the size, non-uniqueness and possibility of singularity.

Iterative aproaches help;

Estimate an initial HR image and recursively modify it to minimize the error $n = |Y - DX|$

\[ \hat{X}_{i+1} = \hat{X}_i + D_{BP}(Y - D\hat{X}_i) \]

$D_{BP}$, the backpropagation matrix determines how the calculated errors are used to update estimation and its choice strictly affects the stability and convergence speed of the recursions.
In reality, samples are not delta samples like $n \times (n)$ but averages of values in a region around the sampling point, and the average is not equal to the value at the sampling point.

CCD and CMOS cells have such nonzero physical dimensions too.

They also have nonzero aperture time.

So the value generated for a pixel is

$$V_{cell} = Q \left( \int_{\Delta x} \int_{\Delta y} \int_{\Delta t} F(x, y, t) dt dy dx \right)$$

where $F(x,y,t)$ is continuous light irradiance field, $Q$ represents the digitization/quantization process.
Each LR image cell can be assumed to be obtained using a weighted sum of HR cell values

$$
\sum_i w_i \text{int}_i
$$

and the weights are the areas where LR cell and HR cells coincide

Obviously this requires very accurate registration.
This schema can be used to determine translations

\[ P_L = \sum_{i=1}^{9} p_H(i)w(i) \]  

or in the matrix form  
\[ I_{Lk} = I_H W_k \]  

\((k\) is the image number\)

If we consider \(W_k\) as a translation operator
\[ I_{L1} W_2 = I_{L2} W_1 \]

That is translating \(I_{L1}\) with \(W_2\) and translation \(I_{L2}\) with \(W_1\) would give the same LLR image.

or
\[ \begin{bmatrix} I_{L1} - I_{L2} \end{bmatrix} \begin{bmatrix} W_2 \\ W_1 \end{bmatrix} = 0 \]

homogenous equation needs non-homogenous constraint equations
Illustration of the operation

\[ I_{L1} W_2 = I_{L2} W_1 \]

in \( N \times M \) dimensional image space

\((N \times M : \text{image size})\)
We have the constraint set

\[ w(1) + w(3) + w(7) + w(9) = 1 \]
\[ w(1) + w(6) + w(8) - w(9) = 1 \]
\[ w(2) + w(8) = 1 \]
\[ w(3) - w(6) + w(9) = 0 \]
\[ w(4) + w(6) = 1 \]
\[ w(5) = 1 \]

And assuming that the first image has zero translation

\[ w(1) = w(3) = w(7) = w(9) = 0.25 \]
\[ w(2) = w(4) = w(6) = w(8) = 0.5 \]
\[ w(5) = 1 \]

We have an equality constrained least-squares system

\[ PL_2W_1 = b \]
\[ CW_1 = d \]

where \( b = PL_1W_2 \)

\[ C = \begin{bmatrix}
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -1 \\
0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 1 \\
\end{bmatrix} \]
\[ d = \begin{bmatrix}
1 \\
1 \\
1 \\
1 \\
1 \\
0 \\
\end{bmatrix} \]
\[ W_1 = \begin{bmatrix}
w(1) \\
w(2) \\
w(3) \\
w(4) \\
w(5) \\
w(6) \\
w(7) \\
w(8) \\
\end{bmatrix} \]

We have the constraint set

\[ w(1) + w(3) + w(7) + w(9) = 1 \]
\[ w(1) + w(6) + w(8) - w(9) = 1 \]
\[ w(2) + w(8) = 1 \]
\[ w(3) - w(6) + w(9) = 0 \]
\[ w(4) + w(6) = 1 \]
\[ w(5) = 1 \]

And assuming that the first image has zero translation

\[ w(1) = w(3) = w(7) = w(9) = 0.25 \]
\[ w(2) = w(4) = w(6) = w(8) = 0.5 \]
\[ w(5) = 1 \]

We have an equality constrained least-squares system

\[ PL_2W_1 = b \]
\[ CW_1 = d \]

where \( b = PL_1W_2 \)

\[ C = \begin{bmatrix}
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -1 \\
0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 1 \\
\end{bmatrix} \]
\[ d = \begin{bmatrix}
1 \\
1 \\
1 \\
1 \\
1 \\
0 \\
\end{bmatrix} \]
\[ W_1 = \begin{bmatrix}
w(1) \\
w(2) \\
w(3) \\
w(4) \\
w(5) \\
w(6) \\
w(7) \\
w(8) \\
\end{bmatrix} \]
One-D Example

aliased barcode

deburred, enhanced

resampled

straightened

thresholded

D.G. Bailey, Journal of Electronic Imaging, 10 (1), pp 213-221 (January 2001)
Coloring

Registration + Coloring

Fusion of LR color and HR gray level images
Images with different exposures (fusing)

Dynamic range is adjusted everywhere
Images with different blur/focus (fusing)

sharp image everywhere
Image Mosaicking
END