Master SIF - REP (Part 6)
Advanced filtering

Thomas Maugey (courtesy of Olivier Le Meur)
thomas.maugey@inria.fr

Université de Rennes

Inria

Fall 2023

Outline

Outline

Motivations to use region Covariances

Objective:
To analyze the textural information surrounding each pixel.
To define a robust descriptor to compare patches.

Region covariance captures the texture by computing a small set of second order statistics on specific image features.

Matrix covariance has high discriminative power and good robustness to the changes in illumination, view, and pose.

References:
(Arbelot et al., 2015, Karacan et al., 2013, Tuzel et al., 2006)

Region Covariances

Let us consider a pixel $\mathbf{p}$, described by a d-dimensional features vector $\mathbf{z}(\mathbf{p})$.
The region covariance is defined as the following $d \times d$ covariance matrix:

$$\mathbf{C}_r(\mathbf{p}) = \frac{1}{W} \sum_{\mathbf{q} \in N_r^{\mathbf{p}}} w_r(\mathbf{p}, \mathbf{q}) \left(\mathbf{z}(\mathbf{q}) - \mu_r\right) \left(\mathbf{z}(\mathbf{q}) - \mu_r\right)^T \qquad (1)$$

Where:

- $N_r^{\mathbf{p}}$ is a square neighborhood centered on $\mathbf{p}$ of size $(2r+1) \times (2r+1)$;
- μ_r is a vector containing the mean of each feature inside this region;
- $w_r(\mathbf{p}, \mathbf{q})$ is a regularization term: $w_r(\mathbf{p}, \mathbf{q}) = exp(-\frac{9\|\mathbf{p}-\mathbf{q}\|^2}{2r^2})$, a Gaussian weighting function with standard deviation $r/3$ $(r \in [20, 30])$;
- W is the normalization factor.

Region Covariances

The d-dimensional feature vector $\mathbf{z}(\mathbf{p})$ can embed different properties:

- $d = 9$ dimensions (Tuzel et al., 2006):

$$\mathbf{z}(\mathbf{p}) = \left[R(\mathbf{p}) \; G(\mathbf{p}) \; B(\mathbf{p}) \; \left|\frac{\partial L(\mathbf{p})}{\partial x}\right| \; \left|\frac{\partial L(\mathbf{p})}{\partial y}\right| \; \left|\frac{\partial^2 L(\mathbf{p})}{\partial x^2}\right| \; \left|\frac{\partial^2 L(\mathbf{p})}{\partial y^2}\right| \; x \; y \right]^T$$

- $d = 7$ dimensions (Karacan et al., 2013):

$$\mathbf{z}(\mathbf{p}) = \left[L(\mathbf{p}) \; \left|\frac{\partial L(\mathbf{p})}{\partial x}\right| \; \left|\frac{\partial L(\mathbf{p})}{\partial y}\right| \; \left|\frac{\partial^2 L(\mathbf{p})}{\partial x^2}\right| \; \left|\frac{\partial^2 L(\mathbf{p})}{\partial y^2}\right| \; x \; y \right]^T$$

- $d = 6$ dimensions (Arbelot et al., 2015):

$$\mathbf{z}(\mathbf{p}) = \left[L(\mathbf{p}) \; \frac{\partial L(\mathbf{p})}{\partial x} \; \frac{\partial L(\mathbf{p})}{\partial y} \; \frac{\partial^2 L(\mathbf{p})}{\partial x^2} \; \frac{\partial^2 L(\mathbf{p})}{\partial y^2} \; \frac{\partial^2 L(\mathbf{p})}{\partial x \partial y} \right]^T$$

where L is the intensity, RGB components and the first and second order derivatives of the intensities with respect to x and y.

Region Covariances

How to proceed?

$$\mathbf{C}_r(\mathbf{p}) = \frac{1}{W} \sum_{\mathbf{q} \in N_r^{\mathbf{p}}} w_r(\mathbf{p}, \mathbf{q}) \left(\mathbf{z}(\mathbf{q}) - \mu_r\right) \left(\mathbf{z}(\mathbf{q}) - \mu_r\right)^T \qquad (2)$$

For a given patch $N_r^{\mathbf{p}}$ centred at the spatial location $\mathbf{p}$:

1. Compute the feature vectors $\mathbf{z}(\mathbf{q})$, $\mathbf{q} \in N_r^{\mathbf{p}}$;
2. Compute the $d \times d$ covariance matrices, $\mathbf{q} \in N_r^{\mathbf{p}}$: $\left(\mathbf{z}(\mathbf{q}) - \mu_r\right) \left(\mathbf{z}(\mathbf{q}) - \mu_r\right)^T$.
   For instance, if $\mu_r = 0$ and $d = 3$, $\mathbf{z}(\mathbf{q}_0) = [a \ b \ c]^T$:
   $$\begin{bmatrix} a \\ b \\ c \end{bmatrix} \begin{bmatrix} a & b & c \end{bmatrix} = \begin{bmatrix} a^2 & ab & ac \\ ba & b^2 & bc \\ ca & cb & c^2 \end{bmatrix}$$
3. Multiply the covariance matrix $\mathbf{C}(\mathbf{q})$ by the weight $w_r(\mathbf{p}, \mathbf{q})$ (computed off-line), $\mathbf{q} \in N_r^{\mathbf{p}}$;
4. Sum all the matrices, and normalized by $1/W$, $W = \sum_{\mathbf{q} \in N_r^{\mathbf{p}}} w_r(\mathbf{p}, \mathbf{q})$.

Region Covariances

Example of region covariance matrices

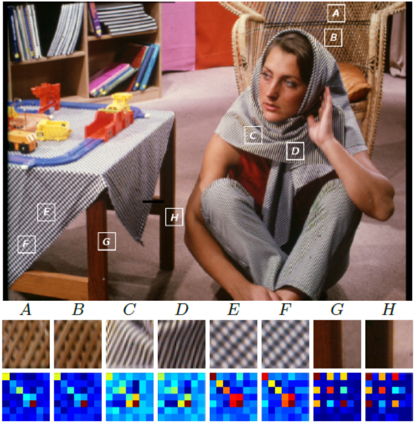


Figure 2: *Region covariance descriptors for different regions of the publicly available Barbara image. Regions having similar visual characteristics are represented by similar covariance descriptors. In this example, the covariance representations are based on very simple image features, namely intensity, orientation, and pixel coordinates (Equation 6).*

(Extracted from (Karacan et al., 2013).)

Matching region covariances descriptors

The covariance matrices do not lie on Euclidean space.

Non-trivial similarity measures...

The dissimilarity between two covariance matrices $d \times d$ can be measured by an affine-invariant metric (Förstner and Moonen, 2003)

$$\rho(\mathbf{C}_1, \mathbf{C}_2) = \sqrt{\sum_{i=1}^{d} ln^2 \lambda_i(\mathbf{C}_1, \mathbf{C}_2)} \tag{3}$$

where, $\{\lambda_i(\mathbf{C}_1, \mathbf{C}_2)\}_{i=1..d}$ are the generalized eigenvalues of $\mathbf{C}_1$ and $\mathbf{C}_2$.

(Hong et al., 2009) proposed a method to transform covariance matrices into Euclidean Space!

Matching region covariances descriptors

We need to transform covariance matrices into an Euclidean vector space... $\Rightarrow$ thanks to the Cholesky decomposition

Covariance matrices are symmetric and positive semi-definite matrix and have **a unique Cholesky decomposition**.

Definition

A symmetric matrix $\mathbf{A} \in \mathcal{R}^{n \times n}$ is called positive semi-definite if $\mathbf{x}^T\mathbf{A}\mathbf{x} \geq 0$ for all $\mathbf{x} \in \mathcal{R}^n$, and is called positive definite if $\mathbf{x}^T\mathbf{A}\mathbf{x} > 0$ for all nonzero $\mathbf{x} \in \mathcal{R}^n$.

Properties:

if $\mathbf{A}$ is positive semi-definite, then

- All eigenvalues of $\mathbf{A}$ are positive;
- $\mathbf{A}$ has a unique Cholesky decomposition $\mathbf{A} = \mathbf{B}\mathbf{B}^T$;

Matching region covariances descriptors

Cholesky decomposition

- Cholesky decomposition:

$$\mathbf{A} = \mathbf{B}\mathbf{B}^T \tag{4}$$

where **B** is the lower triangular matrix with positive diagonal entries.

Example:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 5 & 5 & 5 \\ 1 & 5 & 14 & 14 \\ 1 & 5 & 14 & 15 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 \\ 1 & 2 & 3 & 0 \\ 1 & 2 & 3 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 2 & 2 & 2 \\ 0 & 0 & 3 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- The Cholesky-Banachiewicz algorithm or the Cholesky-Crout algorithm (see Wikipedia, OpenCV code).

Matching region covariances descriptors

Let **C** be a $d \times d$ covariance matrix, a unique set of $2d$ points $\mathcal{S} = \{s_i\}$, referred to as Sigma Points, can be computed as:

$$s_i = \begin{cases} \mu_r + \mathbf{L}_i & \text{if } 1 \leq i < d \\ \mu_r - \mathbf{L}_i & \text{if } d+1 \leq i \leq 2d \end{cases} \tag{5}$$

where $\mathbf{L}_i$ is the i^{th} column of the lower triangular matrix **L** obtained with the Cholesky decomposition $\mathbf{C} = \mathbf{L}\mathbf{L}^T$ and α a scalar value.

The set of columns of **L** has the same second order statistics as the original covariance matrix **C** (Hong et al., 2009).

$\mathcal{S}$ can be enriched with first-order statistics:

$$\mathcal{S} = \{\{s_i\}_{i=1..2d},\ \mu_r\} \tag{6}$$

Matching region covariances descriptors

Example of texture descriptors (sigma points):

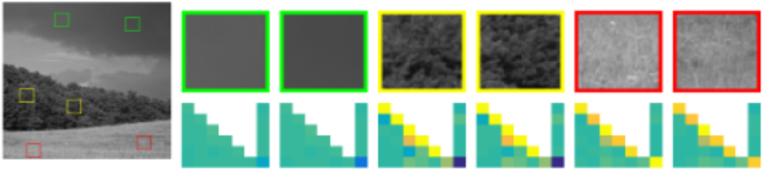


Figure 2: **Texture descriptors.** Patches taken from several regions of the image in Figure 1 (top) and their respective descriptors computed for the central pixel of the window (bottom). Patches from similar regions have similar descriptors.

(Adapted from (Arbelot et al., 2015))

Be careful, the implementation is here not exactly the same: only d vectors, not $2d$ vectors as previously mentioned.

Matching region covariances descriptors

Using the set $\mathcal{S}$, a vectorial representation of a covariance matrix can be obtained by simply concatenating the elements of $\mathcal{S}$.

This enriched feature vector denoted by $\Psi(\mathbf{C})$ is defined as:

$$\Psi(\mathbf{C}) = [s_1, ..., s_d, s_{d+1}, ..., s_{2d}, \mu_r]^T \tag{7}$$

where s_i is a column vector $d \times 1$.

The similarity between two covariance descriptors $\mathbf{C}_p$ and $\mathbf{C}_q$ extracted from two patches centered at pixels **p**and **q**, respectively can be computed as:

$$w_{pq} = exp\left(-\frac{\|\Psi(\mathbf{C}_p) - \Psi(\mathbf{C}_q)\|^2}{2\sigma^2}\right) \tag{8}$$

Applications

Used for smoothing / abstraction / details boosting ...

$$\overline{I}(\mathbf{x}) = \frac{1}{C(\mathbf{x})} \sum_{\mathbf{y} \in \mathcal{N}(\mathbf{x})} I(\mathbf{y}) w(\mathbf{x}, \mathbf{y})$$

where,

- $\mathbf{C}$ is the $d \times d$ covariance matrix,
- $\Psi(\mathbf{C}_x)$ and $\Psi(\mathbf{C}_y)$ are the descriptors of the covariance matrices located at $\mathbf{x}$ and $\mathbf{y}$, respectively,
- Weights are given by:

$$w(\mathbf{x}, \mathbf{y}) = exp\left(-\frac{\|\Psi(\mathbf{C}_x) - \Psi(\mathbf{C}_y)\|^2}{2\sigma^2}\right))$$

Applications

Smoothing:

Smoothing:

Abstraction:

Applications

Abstraction:

Conclusion

A novel structure-preserving image smoothing approach:

- a rather simple approach
- 1st and 2nd order region statistics of simple image features
- Good performance on separation of structure from texture

More details on `https://web.cs.hacettepe.edu.tr/~karacan/projects/regcovsmoothing/`

Outline

Introduction (1/2)

Image retargeting is related to the problem of displaying images without distortion on media of various sizes (cell phones, projection screens).

Adapted from Michael Rubinstein - MIT CSAIL.

Introduction (2/2)

Problem statement:

- $I : \Omega \subset \mathcal{R}^2 \rightarrow \mathcal{R}^3$ an input image of size $M_I \times N_I$.
- $J : \Omega \subset \mathcal{R}^2 \rightarrow \mathcal{R}^3$ an output image of size $M_J \times N_J$.

How to determine the output image J which will be *good representative* of the original image I?

We would expect:

- Adhere to the geometric constraints (display/aspect ratio)
- Preserve the important content and structures
- Limit artifacts

Seam carving (1/2)

A very nice solution proposed by Avidan & Shamir:

Avidan, S., & Shamir, A. (2007, August). Seam carving for content-aware image resizing. In ACM Transactions on graphics (TOG) (Vol. 26, No. 3, p. 10). ACM.

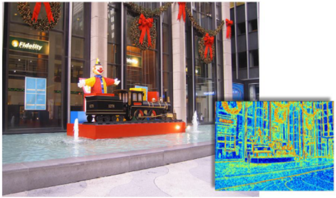
- Basic idea:
  remove unimportant pixels from the image, i.e. pixels with less energy.
- The use of gradient-based energy:

$$E(I(\mathbf{x})) = |\frac{\partial}{\partial x}I(\mathbf{x})| + |\frac{\partial}{\partial y}I(\mathbf{x})|$$

- Preserve strong contours
- Human vision more sensitive to edges

Seam carving (2/2)

How can we remove pixels from an image?



Optimal

Least-energy pixels (per row)

Least-energy columns

Seam definition (1/2)

$I : \Omega \subset \mathcal{R}^2 \rightarrow \mathcal{R}^3$ an input image of size $M_I \times N_I$.

A Top to Bottom (or Left to Right) seam is a connected path of pixels from Top to Bottom (resp. Left to Right). The seam is composed by only one pixel per row (resp. per column).

- Top to Bottom seam:

$$s^{TB} = \{s_i^{TB}\}_{i=1}^{N_I} = \{(x(i), i)\}_{i=1}^{N_I}$$

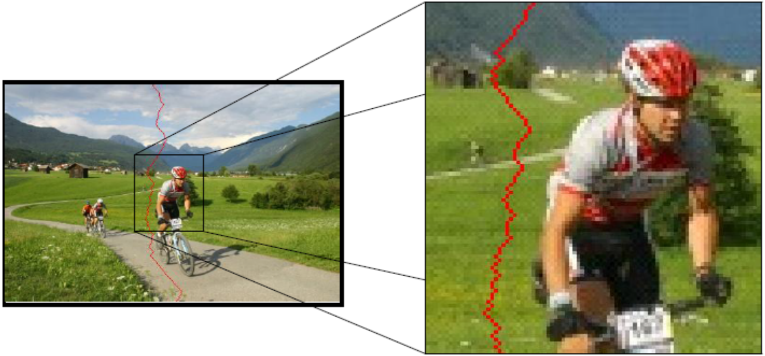
such that $\forall i, |x(i) - x(i-1)| \leq 1$.

- Left to Right seam:

$$s^{LR} = \{s_j^{LR}\}_{j=1}^{M_I} = \{(j, y(j))\}_{j=1}^{M_I}$$

such that $\forall j, |y(i) - y(i-1)| \leq 1$.

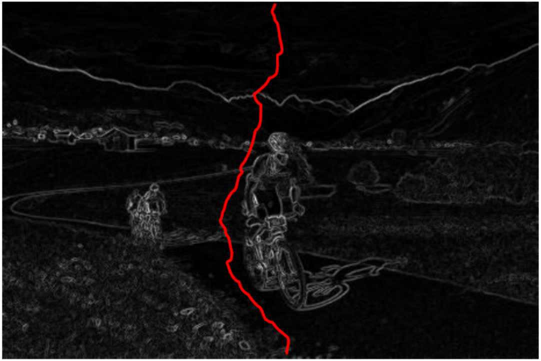
Seam definition (2/2)



Optimal seam (1/4)

How to define an optimal seam, i.e. a seam preserving as much as possible the content?

Optimal seam (2/4)



$$s^{TB^*} = \arg\min_{s^{TB}} E(s^{TB})$$

Optimal seam (3/4)

Optimal seam can be found by the following recursion relation:

$$E_c(j,i) = E(j,i) + \min(E_c(j-1,i-1), E_c(j,i-1), E_c(j+1,i-1))$$

This can be solved efficiently by using dynamic programming.

- Dynamic programming is a programming method that stores the results of sub-calculations in order to simplify calculating a more complex result.

Optimal seam (4/4)

- $I : \Omega \subset \mathcal{R}^2 \rightarrow \mathcal{R}^3$ an input image of size $M_I \times N_I$.
- $J : \Omega \subset \mathcal{R}^2 \rightarrow \mathcal{R}^3$ an output image of size $M_J \times N_J$.

Seam-carving algorithm: $(N_I = N_J$ and $M_J < M_I)$

Do $M_I - M_J$ times

1. $E \leftarrow$ compute energy map of I;
2. s^{TB} find optimal seam in E (required to compute E_c);
3. $E \leftarrow$ remove s^{TB} from I.

For vertical seam, we just need to transpose the image.

Outline

Gaussian and Laplacian pyramid (1/3)

An image pyramid is a collection of images - all arising from a single original image - that are successively downsampled until some desired stopping point is reached.

- Gaussian pyramid: Sequence of low-pass, down-sampled images. Usually constructed with a separable 1D kernel and a down-sampling factor of 2 (in each direction) (Burt and Adelson, 1983).

- Laplacian pyramid: Each level of the Laplacian pyramid is the difference between two adjacent low-pass images of the Gaussian pyramid, $[g_0, g_1, \cdots g_{N_1}]$:

$$l_k = g_k - Up(g_{k+1}) \tag{9}$$

  where $Up(g_{k+1})$ is an up-sampled, smoothed version of g_{k+1}.

Good lecture: https://www.youtube.com/watch?v=NiGcuurpV5o&feature=plcp

Gaussian and Laplacian pyramid (2/3)

Gaussian pyramid

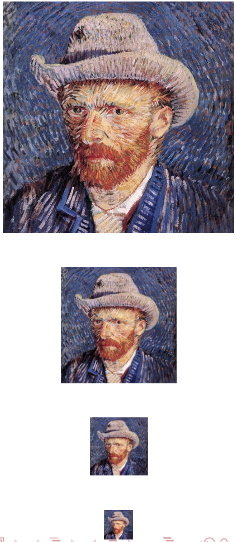
The Gaussian pyramid is built from the finest to the coarsest levels:

Level 0 (highest resolution) = original image.

Level N-1 (coarsest resolution)

```
int level = 4;
Copy input image into imgTemp1;

// Form a Gaussian pyramid
for(int i=0; i<level; i++){
  Filtering and decimation of imgTemp1
  Save the result, this is the ith level of the pyramid
}
```



Gaussian and Laplacian pyramid (3/3)

Laplacian pyramid

- The coarsest scale layer of a Laplacian pyramid is the same as the coarsest scale layer of a Gaussian pyramid.
- Each of the finer scale layers of a Laplacian pyramid is a difference between a layer of the Gaussian pyramid and a prediction obtained by upsampling the next coarsest layer of the Gaussian pyramid (Forsyth and Ponce, 2003).

Recovering the image from its Laplacian pyramid is the synthesis process.

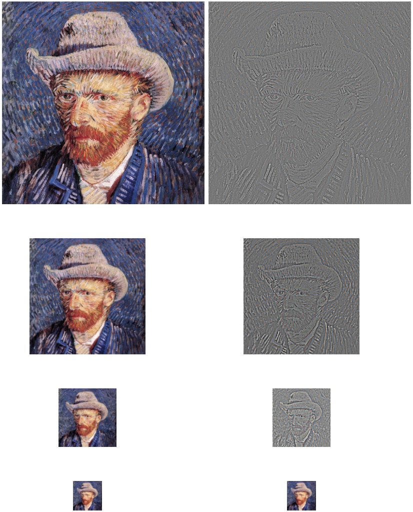


Image blending (1/4)

Feathering

- Feathering: the pixel values in the blended regions are a linear combination of pixel values from the two overlapping images.

$$blended(x, y) = \alpha I_1(x, y) + (1 - \alpha) I_2(x, y)$$

(a) I_1 (b) I_2

(c) $\alpha = 0.35$ (d) See mask (e) (e)

Image blending (2/4)

- Pyramid blending: Seamlessly stitch together images by smoothing the boundary in a scale-dependent way to avoid boundary artifacts.

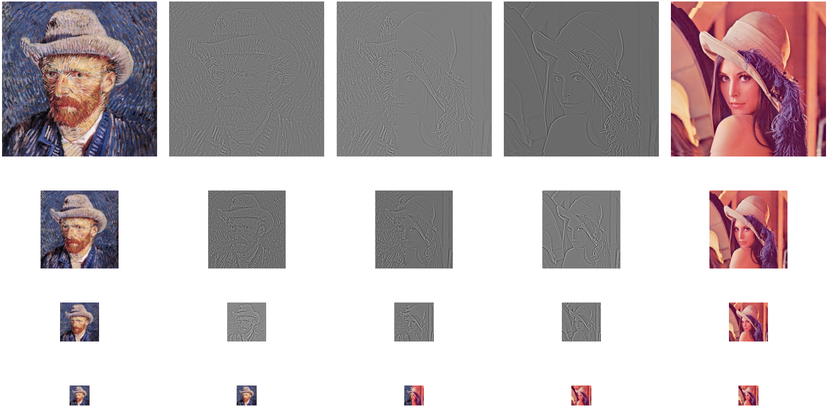
Algorithm:

1. Generate Laplacian pyramid $L1$ for image I_1;
2. Generate Laplacian pyramid $L2$ for image I_2;
3. Generate Gaussian pyramid GM for the mask m;
4. Generate a blended Laplacian pyramid LR:

$$LR(x,y)^k = GM(x,y)^k \times L1^k(x,y) + (1 - GM(x,y)^k) \times L2^k(x,y)$$

   where, k represents the pyramid level.
5. Collapse the LS pyramid to get the final blended image.

Image blending (3/4)



Blended Laplacian pyramid

Image blending (4/4)

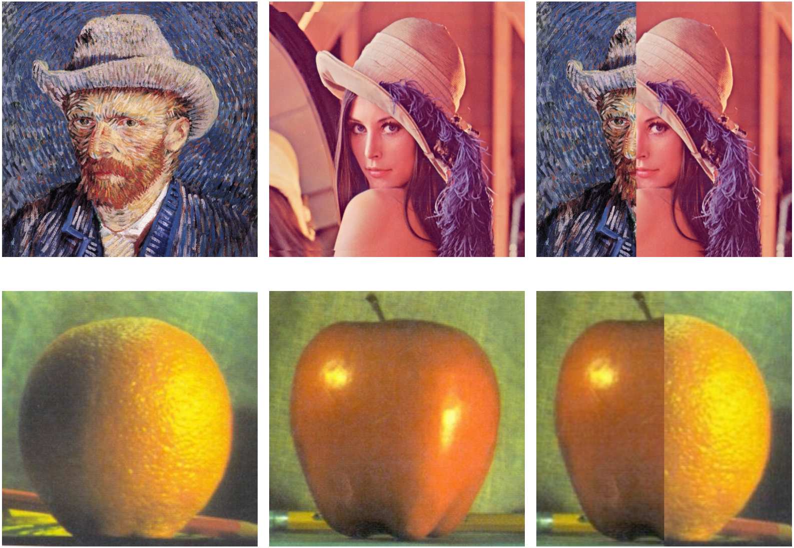
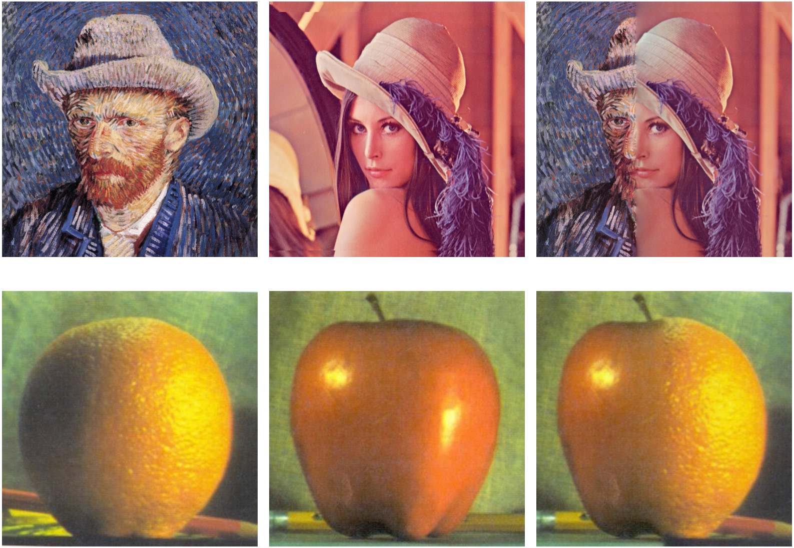


Image blending (4/4)



Outline

Introduction (1/5)

Diffusion is the net movement of molecules or atoms from a region of high concentration to a region of low concentration. This is also referred to as the movement of a substance down a concentration gradient.

Adapted from Wikipedia.

Introduction (2/5)

Diffusion aims at minimizing differences in the spatial concentration $u(\mathbf{x};t)$ of a substance without creation or destruction of the mass. This process can be described by two equations:

1. Flick's law states that the flow $\mathbf{J}$ goes from regions of high concentration to regions of low concentration, with a magnitude that is proportional to the concentration gradient:

$$\mathbf{J} = -g\nabla u(\mathbf{x};t) \tag{10}$$

   where g is the diffusion coefficient and $\mathbf{J}$ is the diffusion flow, i.e. the amount of substance per unit area per unit time.

2. The continuity equation describes the transport of a conserved quantity (conservation law):

$$\frac{\partial u}{\partial t} + div(\mathbf{J}) = 0 \tag{11}$$

Reminder: if $\mathbf{F} = [P, Q, R]$, $div(\mathbf{F}) = \frac{\partial P}{\partial x} + \frac{\partial Q}{\partial y} + \frac{\partial R}{\partial z}$.

Introduction (2/5)

Diffusion aims at minimizing differences in the spatial concentration $u(\mathbf{x};t)$ of a substance without creation or destruction of the mass. This process can be described by two equations:

1. Flick's law states that the flow $\mathbf{J}$ goes from regions of high concentration to regions of low concentration, with a magnitude that is proportional to the concentration gradient:

$$\mathbf{J} = -g\nabla u(\mathbf{x};t) \tag{10}$$

   where g is the diffusion coefficient and $\mathbf{J}$ is the diffusion flow, i.e. the amount of substance per unit area per unit time.

2. The continuity equation describes the transport of a conserved quantity (conservation law):

$$\frac{\partial u}{\partial t} + div(\mathbf{J}) = 0 \tag{11}$$

Reminder: if $\mathbf{F} = [P, Q, R]$, $div(\mathbf{F}) = \frac{\partial P}{\partial x} + \frac{\partial Q}{\partial y} + \frac{\partial R}{\partial z}$.

Introduction (3/5)

Diffusion aims at minimizing differences in the spatial concentration $u(\mathbf{x};t)$ of a substance without creation or destruction of the mass.

$$\mathbf{J} = -g\nabla u(\mathbf{x};t) \qquad \frac{\partial u}{\partial t} + div(\mathbf{J}) = 0$$

From these two equations, we get the diffusion equation:

$$\boxed{\frac{\partial u}{\partial t}(\mathbf{x}) = div(g \times \nabla u(\mathbf{x};t))}$$

where, g is the diffusivity.

- if $g = const$, the diffusion is linear, isotropic and homogeneous;
- if $g = f(\mathbf{x})$, the diffusion is space-dependent, in-homogeneous diffusion;
- if $g = f(u)$, the diffusion is non linear;
- if g is matrix-valued, the diffusion is anisotropic.

Introduction (4/5)

Example: let $I : \Omega \subset \mathcal{R}^2 \rightarrow \mathcal{R}$, a scalar image. $\forall \mathbf{x} \in \Omega$,

$$\frac{\partial I}{\partial t}(\mathbf{x}) = \beta^t(\mathbf{x}) \tag{12}$$

- This PDE indicates how the pixel values of the image are evolving, between given times t and $t + dt$ ($dt \rightarrow 0$). $\beta^t(\mathbf{x})$ is the evolution velocity at time t and position $\mathbf{x}$.
- t is a virtual variable which stands for the evolution time. One generally stops the evolution after a finite time t_{end}, or when $\beta^t(\mathbf{x}) = 0$, $\forall \mathbf{x} \in \Omega$ (convergence).

Introduction (5/5)

How can we implement the following PDE?

$$\boxed{\frac{\partial I}{\partial t}(\mathbf{x}) = \beta^t(\mathbf{x})}$$

This is an iterative process!

$$\begin{cases} I^{t=0} = I_0 \\ \\ \frac{\partial I}{\partial t}(\mathbf{x}) = \beta^t(\mathbf{x}) \end{cases} \qquad \begin{cases} I^{t=0} = I_0 \\ \\ REPEAT \\ I^{t+dt}(\mathbf{x}) = I^t(\mathbf{x}) + dt \times \beta^t(\mathbf{x}) \\ UNTIL\ t < t_{end} \end{cases}$$

Linear diffusion (1/4)

$$\boxed{\frac{\partial I}{\partial t}(\mathbf{x};t) = div(g \times \nabla I(\mathbf{x};t))}$$

where, g is constant.

For $\Omega \subset \mathcal{R}^2$ with boundary $\partial\Omega$:

- Initial condition:

$$I(\mathbf{x};0) = I(\mathbf{x}), \forall \mathbf{x} \in \Omega \tag{13}$$

- Boundary condition:

$$I(\mathbf{x};t) = 0, \forall \mathbf{x} \in \partial\Omega \tag{14}$$

The Neumann boundary condition can be taken into account by a symmetry procedure. If the value of a pixel outside the domain is needed, we use the value of the pixel which is symmetric with respect to the boundaries.

Linear diffusion (2/4)

$$\boxed{\frac{\partial I}{\partial t}(\mathbf{x};t) = div(g \times \nabla I(\mathbf{x};t))}$$

where, g is constant.

- As g is constant (e.g. $g = 1$), $div(\nabla I) = \Delta I$, which is the Laplacian:

$$\boxed{\frac{\partial I}{\partial t}(\mathbf{x};t) = \Delta I(\mathbf{x};t)}$$

This is the heat equation, describing heat flows through solids. This is also called the diffusion equation.

Linear diffusion (3/4)

$$\boxed{\frac{\partial I}{\partial t}(\mathbf{x};t) = \Delta I(\mathbf{x};t))}$$

where, $g = 1$.

Implementation through finite difference:

- $I(\mathbf{x};t+dt) = I(\mathbf{x};t) + dt \times \Delta I(\mathbf{x};t)$, with $dt = 1$;
- $\forall \mathbf{x} = (i,j)$,

  $$\Delta I(i,j) = -4I(i,j)+I(i+1,j)+I(i-1,j)+I(i,j+1)+I(i,j-1)$$

  assuming the Neumann boundary condition.

$$\begin{aligned} I(i,j;t+1) &= I(i,j;t) - 4I(i,j;t) + \\ & \quad I(i+1,j;t) + I(i-1,j;t) + I(i,j+1;t) + I(i,j-1;t) \end{aligned}$$

Linear diffusion (4/4)

- Source code:

```
// Copy input image into imgOut
const int nbIter = 30;
const float g = 0.2f;
int ddepth = CV_32FC3;
for (int i=0;i<nbIter;i++) {
 Laplacian(imgOut,laplacian ,ddepth);
 addWeighted(imgOut,1,laplacian ,g,0,imgOut);}
```

Noisy image and linear diffusion

Non linear diffusion (1/3)

$$\boxed{\frac{\partial I}{\partial t}(\mathbf{x};t) = div(g(\|\nabla I(\mathbf{x};t)\|)\nabla I(\mathbf{x};t))}$$

where, $g : \mathcal{R} \rightarrow \mathcal{R}$ controls the rate of diffusion and decreases with the image gradient.

- Perona-Malik proposed two functions for the diffusion coefficient:

$$\begin{aligned} g(\|\nabla I\|) &= exp(-\frac{\|\nabla I\|^2}{k^2}) \\ g(\|\nabla I\|) &= \frac{1}{1+\frac{\|\nabla I\|^2}{k^2}} \end{aligned}$$

- Charbonnier coefficient:

$$g(\|\nabla I\|) = \frac{1}{\sqrt{1+\frac{\|\nabla I\|^2}{k^2}}}$$

Non linear diffusion (2/3)

Perona-Malik

$$g(\|\nabla I\|) = exp(-\frac{\|\nabla I\|^2}{k^2})$$

Perona-Malik

$$g(\|\nabla I\|) = \frac{1}{1 + \frac{\|\nabla I\|^2}{k^2}}$$

Charbonnier

$$g(\|\nabla I\|) = \frac{1}{\sqrt{1 + \frac{\|\nabla I\|^2}{k^2}}}$$

Non linear diffusion (3/3)

- Source code:

```
const int nbIter = 30;
double lambda_g1 = 0.00025;
Mat weightMap(Size(imageIn.cols,imageIn.rows),CV_32FC1);;
for (int i=0;i<nbIter;i++) {
 // Compute Gradient (gradXColor and gradYColor)
 // Compute the diffusion rate (One channel!)
 weightMap = ....;
 // Compute the weighted gradient (Need 3 channels!)
 vector<Mat> weightMapC3;
 Mat_<Vec3f> weightMapC3Img;
 weightMapC3.push_back(weightMap);
 weightMapC3.push_back(weightMap);
 weightMapC3.push_back(weightMap);
 merge(weightMapC3, weightMapC3Img);
 multiply(gradXColor,weightMapC3Img,gradXColorWeighted);
 multiply(gradYColor,weightMapC3Img,gradYColorWeighted);
 // Compute the divergence of the weighted gradient
 Sobel(gradXColorWeighted, divX,-1,1,0,CV_SCHARR);
 Sobel(gradYColorWeighted, divY,-1,0,1,CV_SCHARR);
 // Sum the divergence sumDiv
 addWeighted(imgOut,1,sumDiv,lambda,0,imgOut);}
```

Application to inpainting

Application to isotropic non-linear inpainting

Outline

Objectives

Detect and magnify motion in videos that is imperceptible by the human eye

More formal objectives

Let $I(x,t)$ denote the image value at position x and time t.

We assume that the videos has small translation motion.
We express $I(x,t)$ with respect to a displacement function $\delta(t)$:

$$I(x,t) = f(x + \delta(t))$$

with $I(x,0) = f(x)$.

The goal of motion magnification is to synthesize a signal:

$$\hat{I}(x,t) = f(x + (1+\alpha)\delta(t))$$

with α being the amplification factor.

[Hao-Yu Wu, Michael Rubinstein, Eugene Shih, John Guttag, Frédo Durand, William T. Freeman Eulerian Video Magnification for Revealing Subtle Changes in the World ACM Transactions on Graphics, Volume 31, Number 4 (Proc. SIGGRAPH), 2012]

First order motion

We assume that the image can be approximated by a first-order Taylor expansion about x:

$$I(x,t) \approx f(x) + \delta(t)\frac{\partial f(x)}{\partial x}$$

Let us apply a temporal band-pass filter at every position x of image $I(x,t)$, such that we pick all what is different from $f(x)$.
We assume that the motion signal $\delta(t)$ is within the bandpass signal. The results is denoted by

$$B(x,t) = \delta(t)\frac{\partial f(x)}{\partial x}.$$

Then, we magnify the variation $B(x,t)$ by α and add it back to $I(x,t)$ leading to:

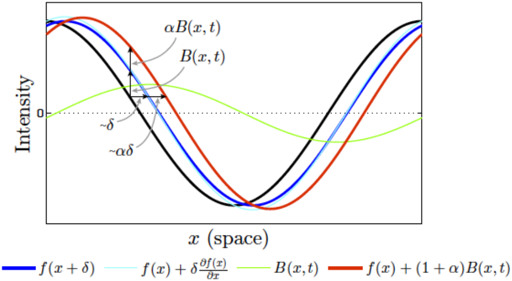
$$\tilde{I}(x,t) = f(x) + (1+\alpha)B(x,t)$$

If we assume that the Taylor expansion still holds for $(1+\alpha\delta(t))$, we have

$$\tilde{I}(x,t) \approx \hat{I}(x,t) = f(x + (1+\alpha)\delta(t))$$

Example

Sinusoid Magnification



Discussion

We have assumed that the motion signal $\delta(t)$ is within the bandpass signal.

We relax this hypothesis, and denote by $\delta_k(t)$, the different temporal spectral components of $\delta(t)$, *i.e.*, $\delta(t) = \sum_k \delta_k(t)$.

In the general case, the band-pass filter attenuate some frequencies (each one with an attenuation factor γ_k). This results in a band-pass filtered signal:

$$B(x,t) = \sum_k \gamma_k \delta_k(t) \frac{\partial f(x)}{\partial x}.$$

The motion magnification is now frequency-dependent $\alpha_k = \gamma_k \alpha$. Therefore

$$\tilde{I}(x,t) \approx f(x + \sum_k (1+\alpha_k)\delta_k(t))$$

Discussion

The Taylor expansion is true for images with reasonably small spatial variations and motions.

$$\begin{aligned} \tilde{I}(x,t) &\approx \hat{I}(x,t) \\ \Rightarrow f(x) + (1+\alpha)\delta(t)\frac{\partial f(x)}{\partial x} &\approx f(x + (1+\alpha)\delta(t)) \end{aligned}$$

Let us imagine that $f = \cos(\omega x)$ and let us denote $\beta = \alpha + 1$

$$\begin{aligned} cos(\omega x) - \omega\beta\delta(t)\sin(\omega x) &\approx \cos(\omega x + \omega\beta\delta(t)) \\ cos(\omega x) - \omega\beta\delta(t)\sin(\omega x) &\approx \cos(\omega)\cos(\omega\beta\delta(t)) - \sin(\omega)\sin(\omega\beta\delta(t)) \end{aligned}$$

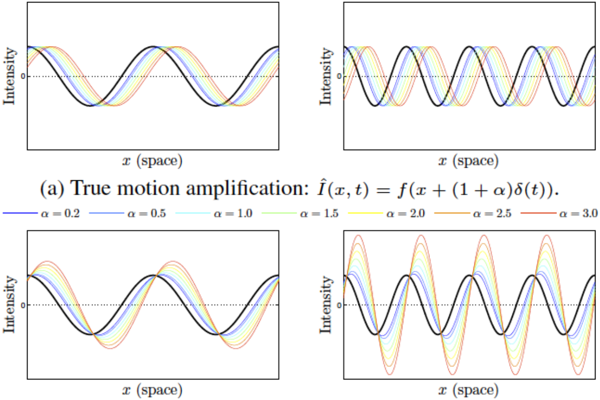
The approximation below can rewritten as:

$$\cos(\omega\beta\delta(t)) \approx 1 \text{ and } \sin(\omega\beta\delta(t)) \approx \omega\beta\delta(t)$$

A known results is that $\sin(\frac{\pi}{4}) = 0.9\frac{\pi}{4}$, which means that 10% approximation error is enabled if

$$(1+\alpha)\delta(t) < \frac{\lambda}{8} \text{ with } \lambda = \frac{2\pi}{\omega}$$

Example of too large magnifification



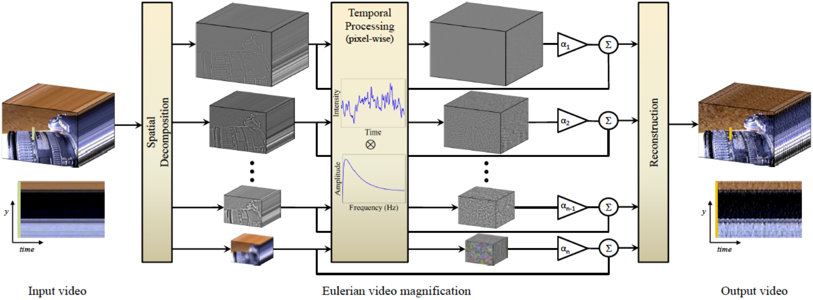
(a) True motion amplification: $\hat{I}(x,t) = f(x + (1+\alpha)\delta(t))$.

(b) Motion amplification via temporal filtering: $\tilde{I}(x,t) = I(x,t) + \alpha B(x,t)$.

The below study enable to have a multi-scale magnification

Magnification Algorithm

Overview

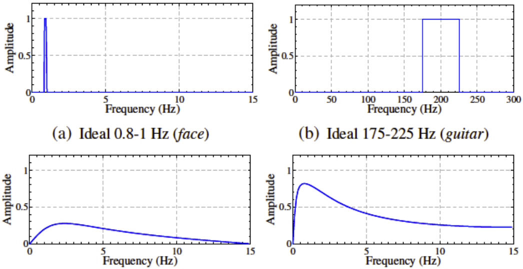


The spatial decomposition is done with a 5 steps binomial filter

Magnification Algorithm

Step1- Select a temporal band-pass filter

Depends on the applications



(a) Ideal 0.8-1 Hz (*face*)
(b) Ideal 175-225 Hz (*guitar*)
(c) Butterworth 3.6-6.2 Hz (*subway*)
(d) Second-order IIR (pulse detection)

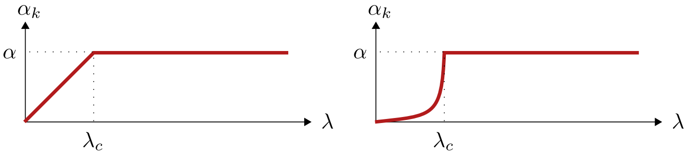
Broad band-pass filters are preferred for motion magnification with no priors on the frequency.
Narrow band-pass filters are preferred for specific task (guitars, blood, etc.).

Magnification Algorithm

Step2- select the desired magnification factor α

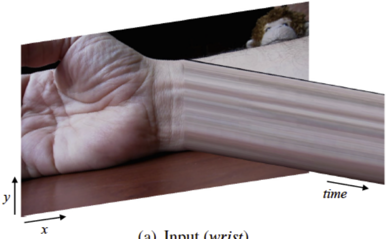
Step3- select a spatial frequency cutoff α

Step4- select the shape of the attenuation α

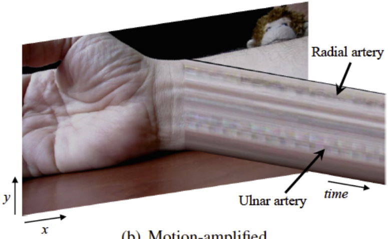


Depends if you would like to emphasize the low frequencies (task dependent magnification).

Results

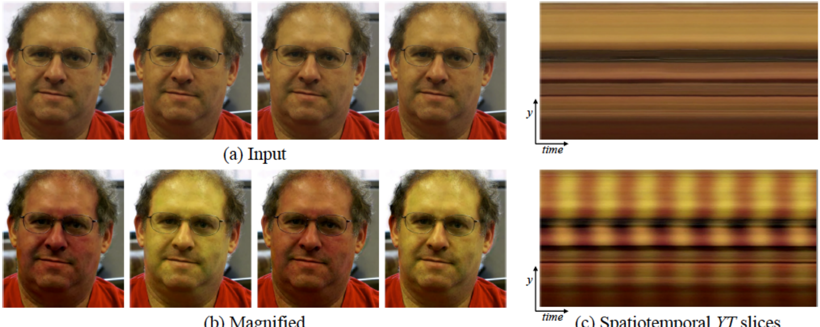


(a) Input (*wrist*)



(b) Motion-amplified

Results



(a) Input

(b) Magnified

(c) Spatiotemporal *YT* slices

Results

input video (30 fps)

Results

Magnified motion video

Results

input video (30 fps)

Magnified motion video

Results

input video (600 fps)

Results

Magnified motion video 82.4 Hz

Results

Magnified motion video 110.0 Hz

Phase-based Magnification

Instead of a Taylor expansion, we write the Fourier serie

$$f(x+\delta(t)) = \sum_{\omega=-\infty}^{\infty} A_\omega e^{i\omega(x+\delta(t))}$$

At each spatial frequency ω, the complex sinusoid is denoted by

$$S_\omega(x,t) = A_\omega e^{i\omega(x+\delta(t))}$$

Its phase $\omega(x+\delta(t))$ contains motion information.

[Neal Wadhwa, Michael Rubinstein, Frédo Durand, William T. Freeman Phase-based Video Motion Processing ACM Transactions on Graphics, Volume 32, Number 4 (Proc. SIGGRAPH) 2013.]

Phase-based Magnification

The motion information in the phase is isolated by using a DC balanced filter assuming that this temporal filter has no other effect than removing the component ωx. The result is

$$B_\omega(x,t) = \omega\delta(t)$$

The magnification is obtained by multiplying $B_\omega(x,t)$ by α, leading to:

$$\tilde{S}_\omega(x,t) := S_\omega(x,t)e^{i\alpha B_\omega(x,t)} = A_\omega e^{i\omega(x+\delta(t))}e^{i\alpha B_\omega(x,t)}$$

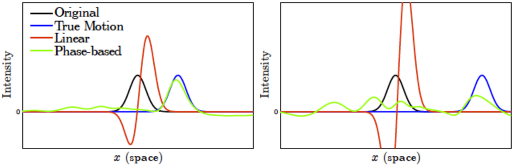
and then

$$\hat{S}_\omega(x,t) = A_\omega e^{i\omega(x+(1+\alpha)\delta(t))}$$

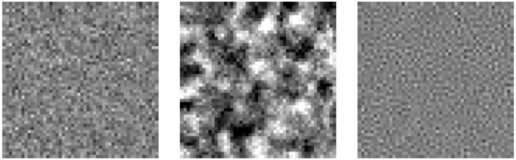
Discussion

Compared to the Linear algorithm, it is more resilient to high spatial frequencies.

$$\alpha\delta(t) < \frac{\lambda}{2}$$

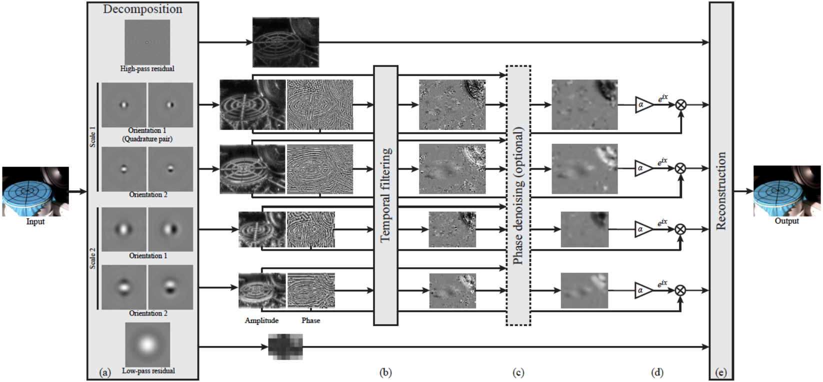


(a) $\alpha = 6$ (b) $\alpha = 14$

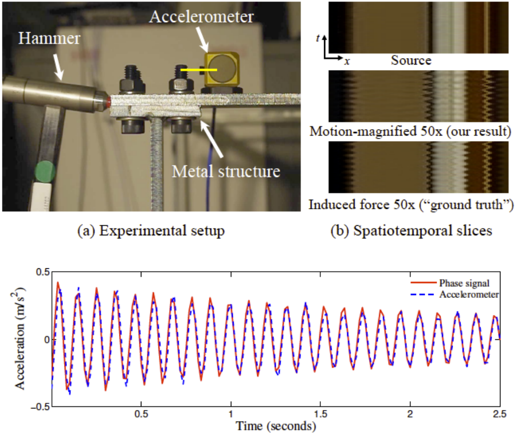


(a) Input (b) Linear (c) Phase-based

Phase-based Algorithm



Results

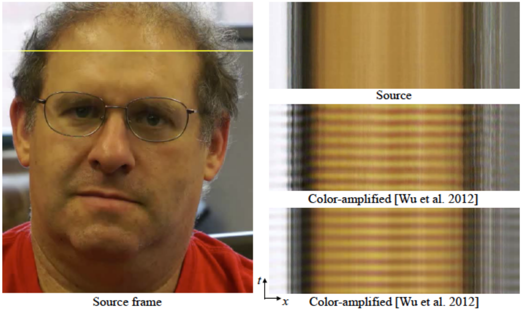


(a) Experimental setup

(b) Spatiotemporal slices

(c) Acceleration comparison

Results



Source frame

Source

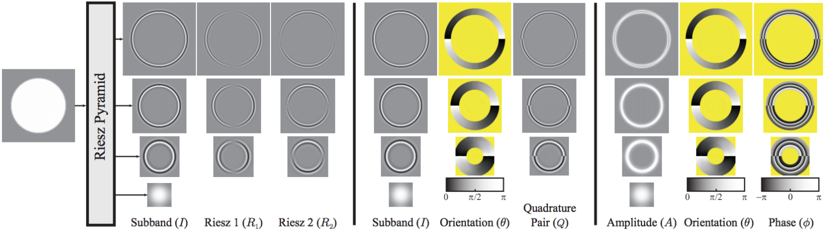
Color-amplified [Wu et al. 2012]

Color-amplified [Wu et al. 2012]
After motion cancellation

Results

Riesz Pyramid

New filters to optimize the spectral phase decomposition of the input images.



(a) Input

(b) Riesz Pyramid (Cartesian Coordinates)

(c) Riesz Pyramid (Cylindrical Coordinates)

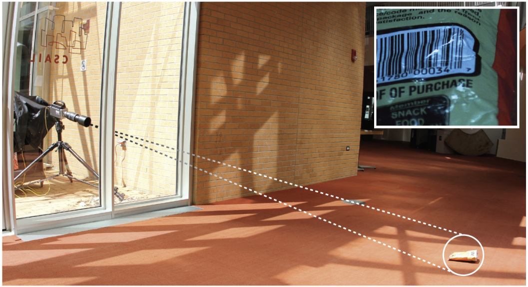
(d) Riesz Pyramid (Spherical Coordinates)

The magnification becomes more precise.

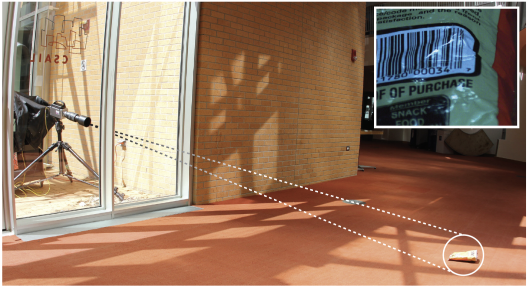
Results

Next step?

Next step?

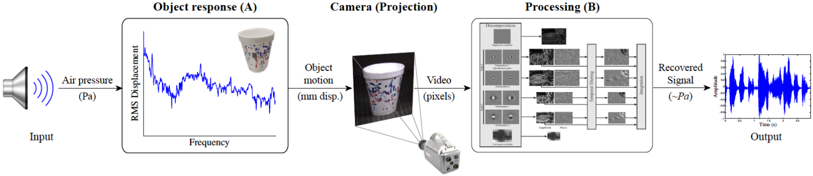


Next step?



Visual microphone: record sounds by filming and magnifying object motions

Global scheme



Camera records the video at $1-20$ kHz.

Algorithm

The input signal is decomposed thanks to multi-scale pyramidal transforms. For an orientation θ and a scale r the signal projection is equal to

$$A(r,\theta,x,y,t)e^{i\varphi(r,\theta,x,y,t)}$$

We extract the phase and compute the local phase variation:

$$\varphi_v(r,\theta,x,y,t) = \varphi(r,\theta,x,y,t) - \varphi(r,\theta,x,y,t_0)$$

This gives the motion information for a scale, orientation, time and position. We average over the position:

$$\Phi(r,\theta,t) = \sum_{x,y} A(r,\theta,x,y,t)^2\varphi_v(r,\theta,x,y,t)$$

And this gives a global motion activity for a given scale, orientation, time.

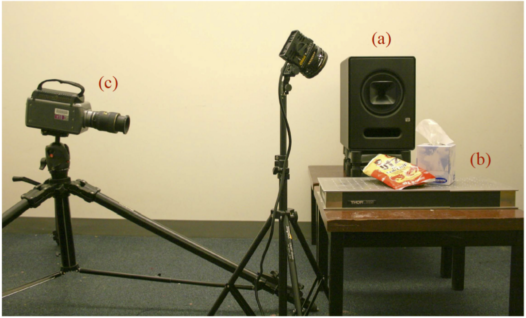
Algorithm

Let us assume that the signal was initially decomposed over a set of scales and orientations $\{r_i\}$ and $\{\theta_i\}$. The global motion is:

$$s(\hat{t}) = \sum_i \Phi(r_i, \theta_i, t - t_i)$$

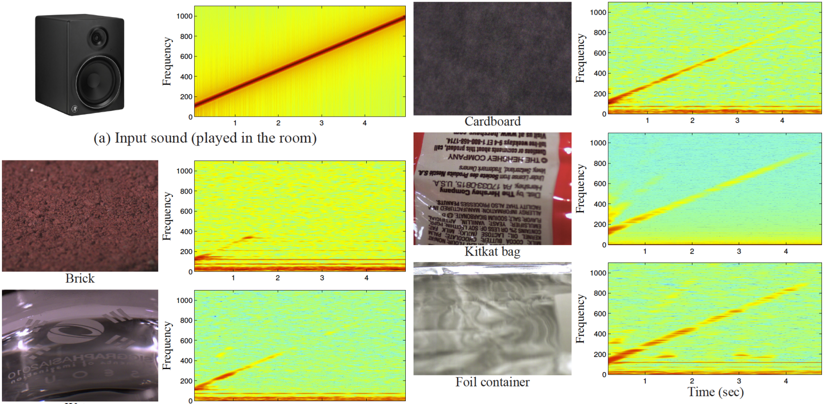
where t_i is a shift to align the phase signals.

Experiments



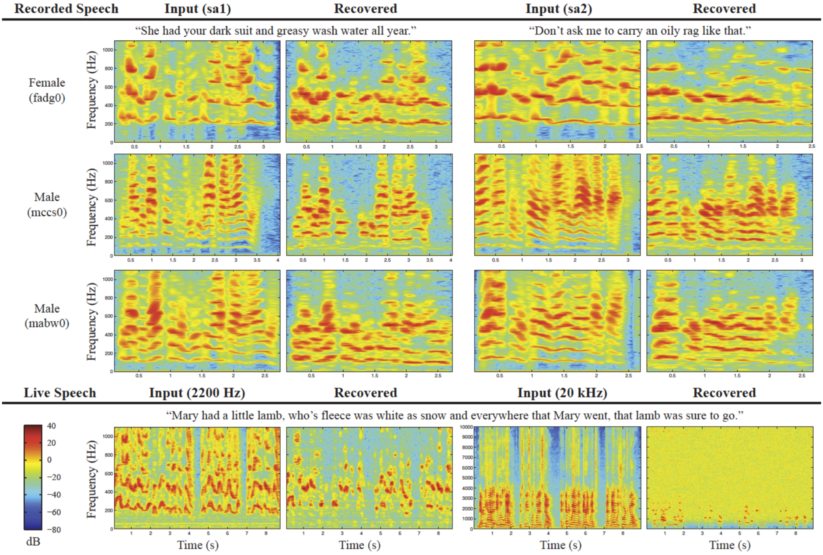
Video frames at $2-20$ kHz, 192×192 to 700×700, sounds produced at $80-110$ dB (actor voice-jet engine)

Results



(b) Reconstructed sound

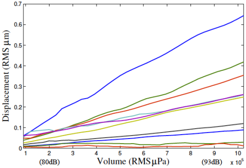
Results


Recorded Speech
Input (sa1)
Recovered
Input (sa2)
Recovered
"She had your dark suit and greasy wash water all year."
"Don't ask me to carry an oily rag like that."
Female (fadg0)
Male (mccs0)
Male (mabw0)
Frequency (Hz)
Live Speech
Input (2200 Hz)
Recovered
Input (20 kHz)
Recovered
"Mary had a little lamb, who's fleece was white as snow and everywhere that Mary went, that lamb was sure to go."
dB
Time (s)

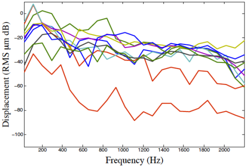
Results

	Measured displacement increase rate (µm/Pa)
crabchips	62.4
greenteabox	41.1
tissue	32.9
foiltogo	20.6
kitkat	21.6
afoil	24.6
rose	10.6
foamcup	8.6
chobani	1.2
teapot	1.1

(a) Displacement coefficients at 300Hz



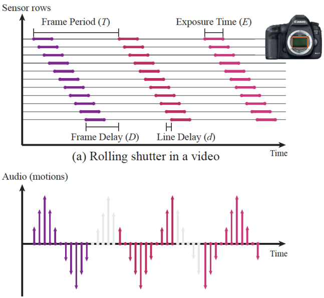
(b) Motion vs. sound volume



(c) Frequency responses

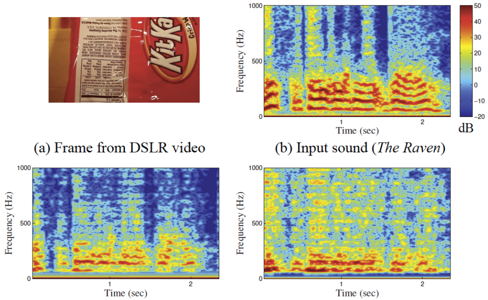
Low speed camera

Use the Rolling shutter



(a) Rolling shutter in a video

(b) Converted to audio signal



(a) Frame from DSLR video

(b) Input sound (*The Raven*)

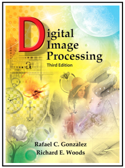
(c) Result (simulated camera: $E = 0$)

(d) Result from DSLR

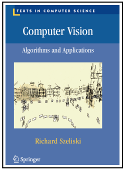
Results

References

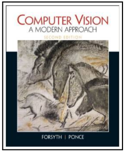
The following slides rely heavily upon the following documents:



Rafael Gonzalez and Richard Woods, *Digital Image Processing*, 3rd edition, Pearson Prentice Hall, 2008.



Richard Szeliski, *Computer Vision: Algorithms and Applications*, 2010.



David Forsyth and Jean Ponce, *Computer Vision: A Modern Approach*, 2003.

References

Benoit Arbelot, Romain Vergne, Thomas Hurtut, and Joëlle Thollot. *Color Transfer and Colorization based on Textural Properties*. PhD thesis, INRIA; LJK, 2015. 4, 6, 13

Peter J Burt and Edward H Adelson. The laplacian pyramid as a compact image code. *Communications, IEEE Transactions on*, 31 (4):532–540, 1983. 33

Wolfgang Förstner and Boudewijn Moonen. A metric for covariance matrices. In *Geodesy-The Challenge of the 3rd Millennium*, pages 299–309. Springer, 2003. 9

David A Forsyth and Jean Ponce. A modern approach. *Computer Vision: A Modern Approach*, 2003. 35

Xiaopeng Hong, Hong Chang, Shiguang Shan, Xilin Chen, and Wen Gao. Sigma set: A small second order statistical region descriptor. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1802–1809. IEEE, 2009. 9, 12

Levent Karacan, Erkut Erdem, and Aykut Erdem. Structure-preserving image smoothing via region covariances. *ACM Transactions on Graphics (TOG)*, 32(6):176, 2013. 4, 6, 8

Oncel Tuzel, Fatih Porikli, and Peter Meer. Region covariance: A fast descriptor for detection and classification. In *European conference on computer vision*, pages 589–600. Springer, 2006. 4, 6