ISSN 23499842(Online), Volume 1, Special Issue 1(NCFCSIT 15), March 2015
International Journal of Innovative Trends and Emerging Technologies

BILATERAL FILTERING OF IMAGE FOR MUTLIRESOLUTION USING BACK PROPAGATION

S.HARIHARASUDHAN[1], DR.B.RAGHU[2]
[1]Research Scholar, Bharath University, Tamilnadu, India
[2]Professor, Sri Ramanujar engineering College, Tamilnadu, India

Abstract: The back propagation and bilateral filtering process is being implemented. It is identified that the image is trained by weights which is supplied to the image for the extraction of the enhanced image of low resolution to high resolution. The back propagation algorithm looks for the minimum of the error function in weight space using the method of gradient descent. The combination of weights which minimizes the error function is considered to be a solution of the learning problem. Since this method requires computation of the gradient of the error function at each iteration step, it is guaranteed the continuity and differentiability of the error function. Back-projection method can minimize the reconstruction error efficiently by an iterative algorithm. The Bilateral filtering smoothes images while preserving edges, by means of a nonlinear combination of nearby image values. It combines gray levels or colors based on both their geometric closeness and their photometric similarity, and prefers near values to distant values in both domain and range. The bilateral back-projection method is proposed to solve the problems associated with the original one, when it is applied to single image, the original back projection algorithm can minimize the reconstruction error efficiently under certain conditions. Then, the idea of bilateral filtering is employed to guide the back-projection process. The image edge information is integrated to avoid across edge projection, thus the ringing effect and chessboard effect can be removed.

Keywords;
Bilateral filtering, alpha matting technique, back propagation, Edge functions,

Introduction:

In this paper, the back propagation and bilateral filtering process is being implemented for processing of an image for multiresolution. It is identified that the image is trained by weights which is supplied to the image for the extraction of the enhanced image of low resolution to high resolution. The back propagation algorithm

looks for the minimum of the error function in weight space using the method of gradient descent. The combination of weights which minimizes the error function is considered to be a solution of the learning problem. Since this method requires computation of the gradient of the error function at each iteration step, it is guaranteed the continuity and differentiability of the error function. Back-projection method can minimize the reconstruction error efficiently by an iterative algorithm. In each step, the current reconstruction error is back-projected to adjust the image intensity. Although this method can improve the image quality greatly, it suffers from some unsatisfying artifacts, such as the ringing effect and the chessboard effect. The underlining reason is the usage of *isotropic* back-projection kernel. It is very likely that the isotropic back-projection kernel leads to unsatisfactory results, since the edge information is totally ignored throughout the update procedure. The bilateral back-projection method is proposed to solve the problems associated with the original one, when it is applied to single image, the original back projection algorithm can minimize the reconstruction error efficiently under certain conditions. Then, the idea of bilateral filtering is employed to guide the back-projection process. The image edge information is integrated to avoid across edge projection, thus the ringing effect and chessboard effect can be removed.

Bilateral filtering smoothes images while preserving edges, by means of a nonlinear combination of nearby image values. It combines gray levels or colors based on both their geometric closeness and their photometric similarity, and prefers near values to distant values in both domain and range.

In this paper, bilateral filtering method to propagate the error according to the edge information it is explained for the process of the bilateral filtering which is a non-linear filtering technique which can combine image information from both of the space domain and the feature domain in the filtering process. The underlining idea of the bilateral filtering is to do the smoothing according to pixels not only close in the space domain, but close in the feature domain as well, thus the edge sharpness is preserved by avoiding the cross edge smoothing.

Bilateral filtering is closely related to other edge preserving techniques such as nonlinear diffusion and adaptive smoothing. The basic idea underlying bilateral filtering is to do in the range of an image what traditional filters do in its domain. Two pixels can be *close* to one another, that is, occupy nearby spatial location, or they can be *similar* to one another, that is, have nearby values, possibly in a perceptually meaningful fashion.

Each edge segment is decomposed by the alpha matting technique, which describes the neighbouring region as a linear combination of two sides of this segment through an alpha channel. Soft edge smoothness prior is applied to super resolve the alpha channel, which is further used to synthesis a high resolution smooth and sharp edge. To improve the image quality for the regions without salient edge segments, a back-projection based post-processing step is employed.

The bilateral back-projection algorithm is the same as the original one, thus the error is back- projected almost isotropically. On the other hand, for a region near a step edge, the error will be only propagated in the part on the same side of the edge as the position corresponding to the low resolution error on the high resolution image.

Here it is identified two processes reasoning for resulting follows as:
(1) The resulting local intensity fluctuation will not influence the updating procedure, thus stable edges can be produced.
(2) It also improves the efficiency, since the filter need to be computed only once without updating.
The error correction on high resolution image is back-projected according to the image edges. Impressive results illustrate the effectiveness of our algorithm.

Back propagation algorithm:

1. Propagates inputs forward in the usual way, i.e.
   - All outputs are computed using sigmoid thresholding of the inner product of the corresponding weight and input vectors.
   - All outputs at stage *n* are connected to all the inputs at stage *n*+1
2. Propagates the errors backwards by apportioning them to each unit according to the amount of this error the unit is responsible for.

We now derive the stochastic Back propagation algorithm for the general case. The derivation is simple, but unfortunately the book-keeping is a little messy.

- = input vector for unit *j* (x_{ji} = *i*th input to the *j*th unit)
- = weight vector for unit *j* (w_{ji} = weight on x_{ji})
- the weighted sum of inputs for unit *j*
- o_j = output of unit *j*
- t_j = target for unit *j*
- *Downstream*(*j*) = set of units whose immediate inputs include the output of *j*
- *Outputs* = set of output units in the final layer

Since it is update after each training example, it can simplify the notation somewhat by imagining that the training set consists of exactly one example and so the error can simply be denoted by *E*.

We want to calculate for each input weight w_{ji} for each output unit j. Note first that since z_j is a function of w_{ji} regardless of where in the network unit j is located,

$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial z_j} \cdot \frac{\partial z_j}{\partial w_{ji}} =$$

Furthermore, is the same regardless of which input weight of unit *j* we are trying to update. So we denote this quantity by .

Consider the case when . We know

$$E = 1/2 \sum_{k \in outputs} (t_k \cdot^2$$

Since the outputs of all units are independent of w_{ji}, we can drop the summation and consider just the contribution to *E* by *j*.

$$\partial_j = \frac{\partial E}{\partial z_j} = \frac{\partial}{\partial z_j} \cdot^2$$

$$= -(t_j \cdot$$

$$= -(t_j - o_j)\left(1 - \sigma(z\right.$$

$$= -(t_j -$$

Thus

$$\Delta\omega_{ij} = -\eta \frac{\partial E}{\partial w_{ij}} = \eta \delta_j x_{ji}$$

Now consider the case when *j* is a hidden unit. Like before, we make the following two important observations.

1. For each unit *k* downstream from *j*, z_k is a function of z_j
2. The contribution to error by all units i in the same layer as *j* is independent of

We want to calculate for each input weight for each hidden unit *j*. Note that influences just z_j which influences o_j which influences $z_k Vk \in Dow$ each of which influence *E*. So this can be written as,

$$\frac{\partial E}{\partial w_{ji}} = \sum_{k \in downstream(j)} \frac{\partial E}{\partial z_k} \cdot \frac{\partial z_k}{\partial o_j} \cdot \frac{\partial o_j}{\partial z_j} \cdot \frac{\partial z_j}{\partial w_{ji}}$$

$$= \sum_{k \in Downstream(j)} \frac{\partial E}{\partial z_k} \cdot \frac{\partial z_k}{\partial o_j} \cdot$$

Again note that all the terms except x_{ji} in the above product are the same regardless of which input weight of unit j we are trying to update. Like before, we denote this common quantity

By . Also note that , and

. Substituting,

$$\delta_j = \sum_{k \in Downstream(j)} \frac{\partial E}{\partial z_k} \cdot \frac{\partial z_k}{\partial o_j} \cdot \frac{\partial o_j}{\partial z_j}$$

$$= \sum_{k \in Downstream(j)} \delta_k w_{kj} o_j (1 - o_j)$$

Thus, $\delta_k = o_j(1 - o_j) \sum_{k \in Downstream(j)} \delta_k$ we are now in a position to state the Back propagation algorithm formally.

Formal statement of the algorithm:

Stochastic Back propagation (training examples, η, n_i, n_h, n_o) each training example is of the form >where is the input vector and is the target vector. is the learning rate (e.g., .05). n_i, n_h and n_o are the number of input, hidden and output nodes respectively. Input from unit i to unit j is denoted x_{ji} and its weight is denoted by w_{ji}.

- Create a feed-forward network with n_i inputs, n_h hidden units, and n_o output units.
- Initialize all the weights to small random values (e.g., between -.05 and .05)
- Until termination condition is met, Do
  - For each training example > , Do

1. Input the instance and compute the output o_u of every unit.
2. For each output unit k, calculate $\delta_k = o_k(1 - o_k)(t_k -$
3. For each hidden unit h, calculate

$$\delta_h = o_h(1 - o_h) \sum_{k \in downstream(h)} w_i$$

4. Update each network weight w_{ji} as follows:

$w_{ji} \leftarrow w_{ji}$, where Δw_{ji}

Bilateral filtering algorithm:

Consider a shift-invariant low-pass domain filter applied to an image

$$h(x) = k_d^{-1} \iint_{\infty}^{\infty} f(\varepsilon)\, c(\varepsilon - x)\, d\varepsilon$$

The f and h emphasizes the fact that both input and output images may be multi-band. In order to preserve the DC component, it must be

$$k_d = \iint_{\infty}^{\infty} c(\varepsilon)\, d\varepsilon$$

Range filtering is similarly defined:

$$h(x) = k_r^{-1}(x) \iint_{\infty}^{\infty} f(\varepsilon)\, s\big(f(\varepsilon) - f(x)\big)\, d\varepsilon$$

In this case, the kernel measures the *photometric* similarity between pixels. The normalization constant in this case is

$$k_r(x) = \iint_{\infty}^{\infty} f(\varepsilon)\, s\big(f(\varepsilon) - f(x)\big)\, d\varepsilon$$

The spatial distribution of image intensities plays no role in range filtering taken by itself. Combining intensities from the entire image, however, makes little sense, since the distribution of image values far away from **x** ought not to affect the final value at **x**. In addition, one can show that range filtering without domain filtering merely changes the color map of an image, and is therefore of little use. The appropriate solution is to combine domain and range filtering, thereby enforcing both geometric and photometric locality. Combined filtering can be described as follows:

$$h(x) = k^{-1} \iint_{\infty}^{\infty} f(\varepsilon)\, c(\varepsilon - x)\, s\big(f(\varepsilon) - f(x)\big)\, d\varepsilon$$

With the normalization

$$k(x) = \iint_{\infty}^{\infty} c(\varepsilon - x)\, s\big(f(\varepsilon) - f(x)\big)\, d\varepsilon$$

Combined domain and range filtering will be denoted as *bilateral filtering*. It replaces the pixel value at **x** with an average of similar and nearby pixel values. In smooth regions, pixel values in a small neighborhood are similar to each other, and the bilateral filter acts essentially as a standard domain filter, averaging away the small, weakly correlated differences between pixel values caused by noise.

When the bilateral filter is centred, say, on a pixel on the bright side of the boundary, the similarity function *s* assumes values close to one for pixels on the same side, and values close to zero for pixels on the dark side.

The normalization term ***k(x)*** ensures that the weights for all the pixels add up to one. As a result, the filter replaces the bright pixel at the centre by an average of the bright pixels in its vicinity, and essentially ignores the dark pixels. Conversely, when the filter is centred on a dark pixel, the bright pixels are ignored instead. Thus, good filtering behaviour is achieved at the boundaries, thanks to the domain component of the filter, and crisp edges are preserved at the same time, thanks to the range component.

Iterative process:

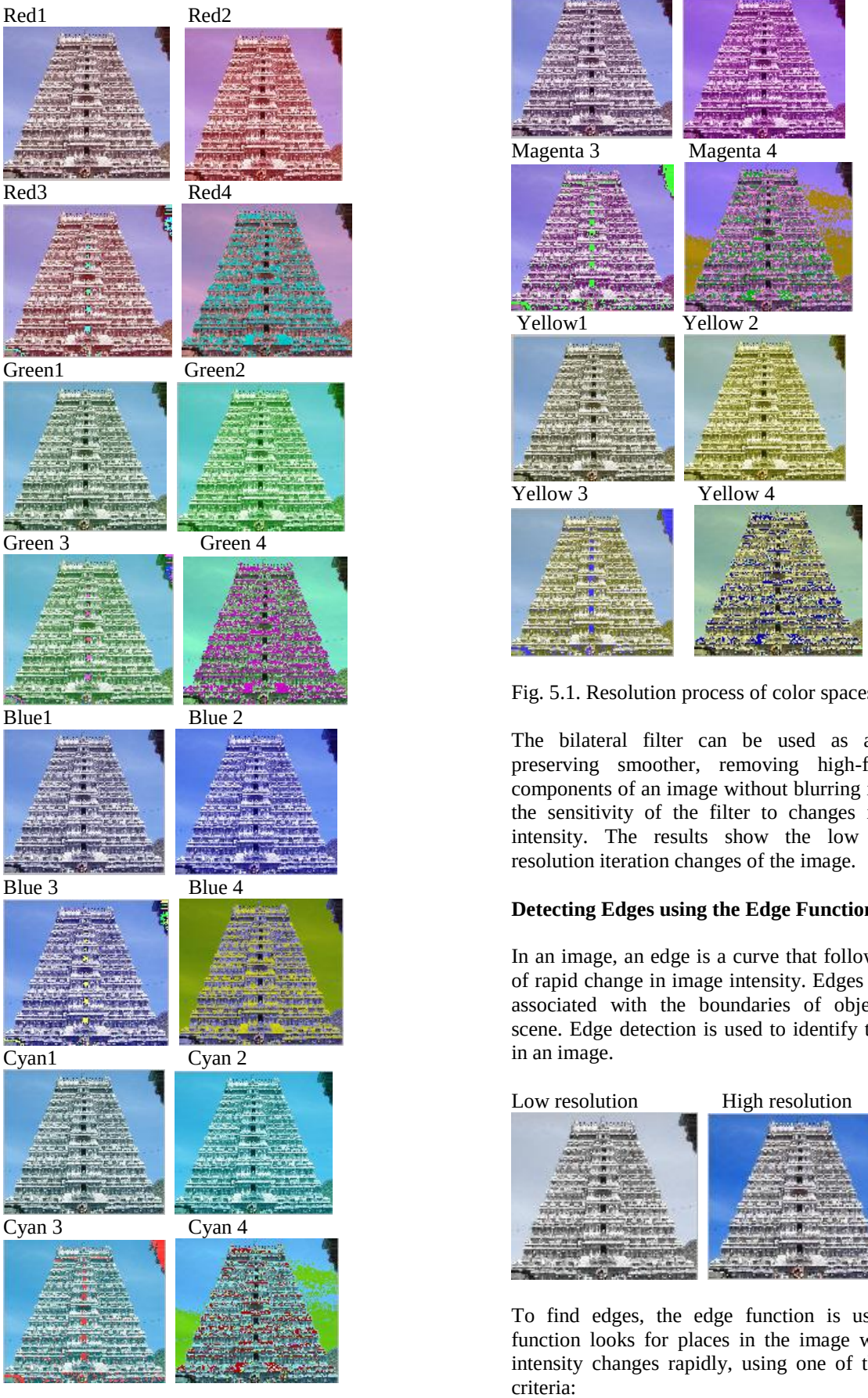

Fig. 5.1. Resolution process of color spaces.

The bilateral filter can be used as an edge-preserving smoother, removing high-frequency components of an image without blurring its edges, the sensitivity of the filter to changes in image intensity. The results show the low to high resolution iteration changes of the image.

Detecting Edges using the Edge Function

In an image, an edge is a curve that follows a path of rapid change in image intensity. Edges are often associated with the boundaries of objects in a scene. Edge detection is used to identify the edges in an image.

To find edges, the edge function is used. This function looks for places in the image where the intensity changes rapidly, using one of these two criteria:

- Places where the first derivative of the intensity is larger in magnitude than some threshold
- Places where the second derivative of the intensity has a zero crossing

Edge provides a number of derivative estimators, each of which implements one of the definitions above. For some of these estimators, we can specify whether the operation should be sensitive to horizontal edges, vertical edges, or both. Edge returns a binary image containing 1's where edges are found and 0's elsewhere.

The most powerful edge-detection method that edge provides is the canny method.

The Canny method differs from the other edge-detection methods in that it uses two different thresholds (to detect strong and weak edges), and includes the weak edges in the output only if they are connected to strong edges. This method is therefore less likely than the others to be fooled by noise, and more likely to detect true weak edges.

The following example illustrates the power of the Canny edge detector by showing the results of applying the Sobel and Canny edge detectors to the same image:

```
I = imread('ayan1.jpg');
J = rgb2gray(I);
figure, imshow(I);title('Original Image of ayan1.jpg');
figure, imshow(J);title('Gray Scale Image of ayan1.jpg');
BW1 = edge(J,'sobel');
figure, imshow(BW1);title('Edge Detection by Sobel Method');
BW2 = edge(J,'canny');
```

Tracing Object Boundaries in an Image

The toolbox includes two functions we can use to find the boundaries of objects in a binary image:
bwtraceboundary
bwboundaries

The bwtraceboundary function returns the row and column coordinates of all the pixels on the border of an object in an image. We must specify the location of a border pixel on the object as the starting point for the trace.

The bwboundaries function returns the row and column coordinates of border pixels of all the objects in an image. For both functions, the nonzero pixels in the binary image belong to an object and pixels with the value 0 (zero) constitute the background.

The following example uses bwtraceboundary to trace the border of an object in a binary image and then uses bwboundaries to trace the borders of all the objects in the image:

```
I = imread('ayan1.jpg');
J = rgb2gray(I);
figure, imshow(J);title('Original Image of ayan1.jpg');
BW = im2bw(J);
imshow(BW);title('Trace Boundary of Image ayan1.jpg'); dim = size(BW)
col = round(dim(2)/2)-90;
row = min(find(BW(:,col)));
boundary = bwtraceboundary(BW,[row, col],'N');
imshow(J)
hold on;
plot(boundary(:,2),boundary(:,1),'b','LineWidth',1);
BW_filled = imfill(BW,'holes');
boundaries = bwboundaries(BW_filled);
for k=1:10
  b = boundaries{k};
  plot(b(:,2),b(:,1),'b','LineWidth',1);
end
```

Sharpening and Blurring an Image

To sharpen a color image, that is needed to make the luma intensity transitions more acute, while preserving the color information of the image. To do this, we convert an R'G'B' image into the Y'CbCr color space and apply a highpass filter to the luma portion of the image only. Then, we transform the image back to the R'G'B' color space to view the results. To blur an image, we apply a lowpass filter to the luma portion of the image. This example shows how to use the 2-D FIR Filter block to sharpen and blur an image. The prime notation indicates that the signals are gamma corrected.

Define an R'G'B' image in the MATLAB workspace. To read in an R'G'B' image from a PNG file and cast it to the double-precision data type, at the MATLAB command prompt, type

1. I= im2double(imread('peppers.png'));
   I is a 384-by-512-by-3 array of double-precision floating-point values. Each plane of this array represents the red, green, or blue color values of the image.
2. To view the image this array represents, at the MATLAB command prompt, type
   imshow(I)
3. Create a new Simulink model, and add to it the blocks shown in the following table.
4. Position the blocks as shown in the following figure.

5. Use the Image from Workspace block to import the R'G'B' image from the MATLAB workspace. Set the parameters as follows:
   Main pane, Value = I
   Main pane, Image signal = Separate color Signals. The block outputs the R', G', and B' planes of the I array at the output ports
6. The first Color Space Conversion block converts color information from the R'G'B' color space to the Y'CbCr color space. Set the Image signal parameter to Separate color signals.
7. Use the 2-D FIR Filter block to filter the luma portion of the image. Set the block parameters as follows:
   Coefficients = fspecial('unsharp')
   Output size = Same as input port I
   Padding options = Symmetric
   Filtering based on = Correlation
   The fspecial('unsharp') command creates two-dimensional highpass filter coefficients suitable for correlation. This highpass filter sharpens the image by removing the low Frequency noise in it.
8. Use the Color Space Conversion1 block to converts the color information from the Y'CbCr color space to the R'G'B' color space. Set the block parameters as follows:
   Conversion = Y'CbCr to R'G'B'
   Image signal = Separate color signals
9. Use the Video Viewer block to automatically display the new, sharper image in the Video Viewer window when we run the model. Set the Image signal parameter to Separate color signals.
10. Connect the blocks as shown in the following figure.
11. Set the configuration parameters. Open the Configuration dialog box by selecting Configuration Parameters from the Simulation menu. Set the parameters as follows:
    - Solver pane, Stop time = 0
    - Solver pane, Type = Fixed-step
    - Solver pane, Solver = discrete (no continuous states)
12. Run the model.
    A sharper version of the original image appears in the Video Viewer window.
    To view the image at its true size, right-click the window and select Set Display To True Size.
13. To blur the image, double-click the 2-D FIR Filter block. Set Coefficients parameter to fspecial('gaussian',[15 15],7) and then click OK.
    The fspecial('gaussian',[15 15],7) command creates two-dimensional Gaussian lowpass filter coefficients. This lowpass filter blurs the image by removing the high frequency noise in it.
14. Run the model.

Conclusion:

The method of the back propagation and bilateral filtering process is being implemented for processing of an image for multiresolution, with image which is trained by the weights. Further, the weights are supplied forming the extraction of the enhanced image of low resolution to high resolution.

It is understood that the back-projection method can minimize the reconstruction error efficiently by an iterative process. And the basic idea underlying bilateral filtering is to do in the range of an image what traditional filters, which provides the bilateral filter that, can be used as an edge-preserving smoother, removing high-frequency components of an image without blurring its edges, the sensitivity of the filter to changes in image intensity. The results show the low to high resolution iteration changes of the image.

Finally the edge is a curve that follows a path of rapid change in image intensity.

References:

- BARASH, D. 2012. A fundamental relationship between bilateral filtering, adaptive smoothing and the nonlinear diffusion equation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 24*, 6.
- DURAND, F., AND DORSEY, J. 2011. Fast bilateral filtering for the display of high-dynamic-range images. *ACM Transactions on Graphics (TOG) 21*, 3, 257.266.
- ELAD, M. 2011. On the bilateral filter and ways to improve it. *IEEE Transactions On Image Processing*.
- TOMASI, C., AND MANDUCHI, R. 2012. Bilateral filtering for gray and color images. In *ICCV*, 839.846.
- T. S. Huang, G. J. Yang, and G. Y. Tang. A fast two-dimensional median filtering algorithm. *IEEE Trans.*, ASSP-27(1):13–18, 2009.
- J. S. Lee. Digital image enhancement and noise filtering by use of local statistics. *IEEE Trans.*, PAMI-2(2):165–168, 2012.

- D.E. Rumelhart, G.E. Hinton, and R.J. Williams, *Learning Internal Representations by Error Propagation*, in D.E.Rumelhart and J.L. McClelland (eds.) Parallel Distributed Processing: Explorations in the microstructure of Cognition (vol. 1), MIT Press, 2013.