Wavelet based Image Reconstruction from Gradient Data

Version 1.0

I.S. Sevcenco[1], P.J. Hampton[2], P. Agathoklis[1]

1) Department of Electrical and Computer Engineering

University of Victoria Victoria, B.C., Canada

2) Cellula Robotics Ltd., New Westminster, BC

October 2014

Contents

Introduction

The **ImageRecH** toolbox can be used to reconstruct an image from gradient data. The main function in the toolbox is **ImageRecH.m**, which can be used to reconstruct images that are either single channel (i.e., grayscale) or multi channel (i.e., colour, in three channel representation, such as RGB), given some gradient measurements and a mean value. The theory behind the toolbox was developed in the papers [1], [2] and [3].

The table below lists the functions in this toolbox and their purposes.

Function name	Purpose
Ad_GWN	Adds Gaussian white noise to a signal
splitRGB	Splits a multichannel image into its component channels and computes corresponding mean values
CombineRGB	Combines three channels of an image into its multichannel representation
getGradientH	Computes discrete approximation of image gradient
getGradient	Computes discrete approximation of image gradient
ConvertGradientHF	Converts between two gradient discretization models (see sec. 2.1 in 1)
CorrectMeanValue	Corrects mean value of an image
Diffmeasure	Compares two signals in terms of different measures
getDiv	Computes divergence of vector
getLaplacian	Computes Laplacian
GradientAnalysis	Analysis step as in [2]
GradientAnalysisH	Analysis step as in [1]
GradientSynthesisH	Synthesis step as in [1]
PoissonSolveExtend	Poisson solver used in synthesis step
OneChannelRec	Reconstructs one channel in an image (called in ImageRecH)
ImageRecH	Reconstructs image from gradient data and mean value
Reflect	Reflects gradient in non-square case as in [2]
ReflectH	Reflects gradient in non-square case as in [1]
ReflectHC	called in ReflectsH

Example 1: Reconstruction of a single channel image (i.e., grayscale) from gradient data

This example illustrates how the software ImageRecH.m can be used to reconstruct a single channel image (i.e., a grayscale image) from a gradient data set. To illustrate how the algorithm works, the ground truth image is considered known, and the reconstructed image will be compared against it.

The goal is to use the gradient and the mean value of the image as inputs to the software (ImageRecH.m), output a reconstructed image and compare it to the original. The output should be identical to the input in the case of clean, noise free input gradient and after the correction of the mean value.

Read grayscale image into workspace

```
I = double(imread('moon.tif'));
```

Compute mean value of the grayscale image

```
meanval = mean(I(:));
```

Compute the gradient of the grayscale image

Ensure boundary conditions are satisfied, i.e. the x-gradient is zero on the last column and the y-gradient is zero on the last row

```
[gxH,gyH] = getGradientH(I,1);
```

Specify whether or not Poisson smoothing is used in the reconstruction

Here, the Poisson solver is **not** used; to indicate this, the variable **PoissonOn** is set to 0.

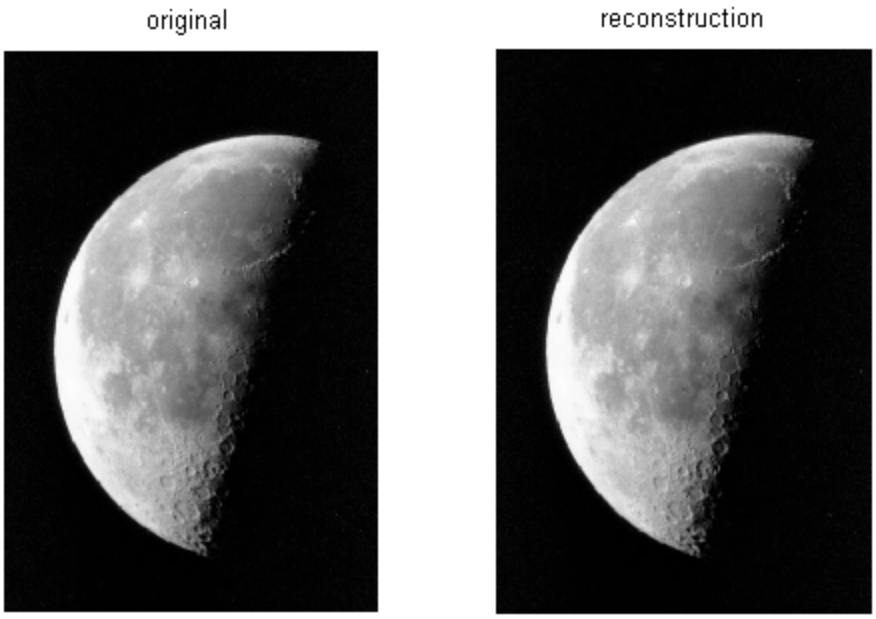
```
PoissonOn = 0;
```

Reconstruct the image using the wavelet based algorithm [1]

```
R = ImageRecH(gxH,gyH,meanval,PoissonOn);
```

Display the original image side by side with the reconstructed

```
figure, subplot(121), imshow(uint8(I)), title('original'),
subplot(122), imshow(uint8(R)), title('reconstruction')
```



Compare reconstruction to ground truth

Returns 1 if they are equal

```
isequal(I,R)
```

```
ans =

     1
```

Example 2: Reconstruction of a multi channel image (i.e., colour image, in RGB) from gradient data

This example illustrates how the software ImageRecH.m can be used to reconstruct a multichannel image (i.e., a colour image, in RGB representation) from a gradient data set. To illustrate how the algorithm works, the ground truth image is considered known, and the reconstructed image will be compared against it.

The goal is to use the gradient and the mean value of each individual color channels as inputs to the software (ImageRecH.m), output a reconstructed image and compare it to the original. The output should be identical to the input in the case of clean, noise free input gradient and after the correction of the mean value.

Read colour image into workspace

```
I = double(imread('gantrycrane.png'));
```

Compute mean value of each channel of the colour image

```
[~,~,~,mr,mg,mb] = splitRGB(I);
```

Store the three values in a vector

```
meanval = [mr,mg,mb];
```

Compute the gradient of the multichannel image

Ensure boundary conditions are satisfied, i.e. the x-gradient is zero on the last column and the y-gradient is zero on the last row

```
[gxH,gyH] = getGradientH(I,1);
```

Specify whether or not Poisson smoothing is used in the reconstruction

Here, the Poisson solver is **not** used; to indicate this, the variable PoissonOn is set to 0.

```
PoissonOn = 0;
```

Reconstruct the image using the wavelet based algorithm [1]

```
R = ImageRecH(gxH,gyH,meanval,PoissonOn);
```

Display the original image side by side with the reconstructed

```
figure, subplot(121), imshow(uint8(I)), title('original'),
subplot(122), imshow(uint8(R)), title('reconstruction')
```

Compare reconstruction to ground truth

Returns 1 if they are equal

```
isequal(I,R)
```

```
ans =

     1
```

Example 3: Reconstruction of a single channel image (i.e., grayscale) from noisy gradient data

This example illustrates how the software ImageRecH.m can be used to reconstruct a single channel image (i.e., a grayscale image) from a **noisy** gradient data set. To illustrate how the algorithm works, the ground truth image is considered known. Then, the approximation of its gradient is computed, noise is added to the gradient data, and reconstruction using the wavelet based technique is attempted. To illustrate the role of the Poisson solver during the synthesis stage, two reconstructions are generated: one without the Poisson solver, and the other one with it. The two reconstructed images are displayed and compared against the original.

Read grayscale image into workspace

```
I = double(imread('coins.png'));
```

Compute mean value of the grayscale image

```
meanval = mean(I(:));
```

Compute the gradient of the grayscale image Ensure boundary conditions are satisfied, i.e. the x-gradient is zero on the last column and the y-gradient is zero on the last row)

```
[gxH,gyH] = getGradientH(I,1);
```

Add Gaussian white noise to gradient.

Specify desired SNR (in dB). In this example, noise was added to generate an SNR of $\sim$ 0dB in each gradient component.

```
dsnr = 0;
```

Specify seed for random number generator, to allow result repeatability

```
sdx = 3;
gxHn = Ad_GWN(gxH,dsnr,sdx);
```

Ensure boundary conditions

```
gxHn(:,end) = 0;
```

Specify another seed for random number generator

```
sdy = 7;
gyHn = Ad_GWN(gyH,dsnr,sdy);
```

Ensure boundary conditions

```
gyHn(end,:) = 0;
```

Reconstruct the image using the wavelet based algorithm [1]

The variable `PoissonOn` should be set to:

- 0 (zero) if the Poisson solver is not used
- 1 (one) if the Poisson solver is used

When the Poisson solver is used, the user can specify a desired number of iterations. The recommended (and default) number of iterations for the Poisson solver is three. If more iterations are needed, the user can set the desired number with the following command:

```
no_iter = 7;
```

As the default number of iterations is predefined, the last input of the function `ImageRecH.m` is not required. If the user prefers to use more than three iterations of the solver per resolution level, then the function `ImageRecH.m` should be called with five input arguments, the last of which should be the desired number of iterations `no_iter` .

```
R = ImageRecH(gxH,gyH,meanval,PoissonOn,no_iter) ;
```

First, reconstruct the image from the noisy gradient data without the Poisson solver

```
PoissonOn = 0;
R1 = ImageRecH(gxHn,gyHn,meanval,PoissonOn);
```

Now, reconstruct with Poisson solver.

In this example, the default configuration will be used (three iterations per resolution level)

```
PoissonOn = 1;
R2 = ImageRecH(gxHn,gyHn,meanval,PoissonOn);
```

Display the original image and the two reconstructions side by side.

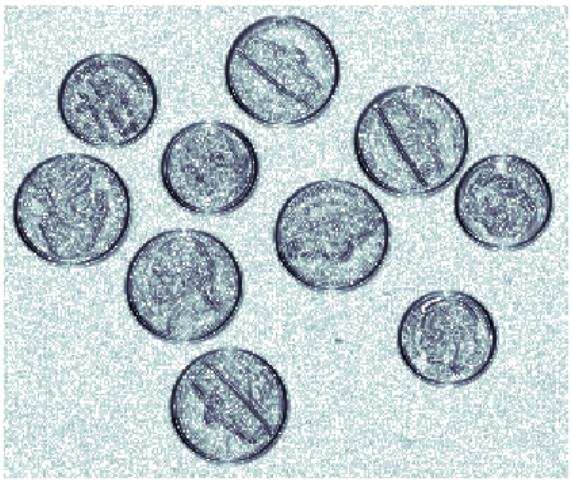
```
figure, imshow(uint8(I)), title('Original image');
figure, imshow(uint8(R1)), title('Reconstructed with Poisson');
xlabel(['Mean squared error: ',num2str(diffmeasure(I,R1,'mse'))]);
figure, imshow(uint8(R2)), title('Reconstruction with Poisson');
xlabel(['Mean squared error: ',num2str(diffmeasure(I,R2,'mse'))]);
```

Compare reconstructions to ground truth in terms of mean squared error

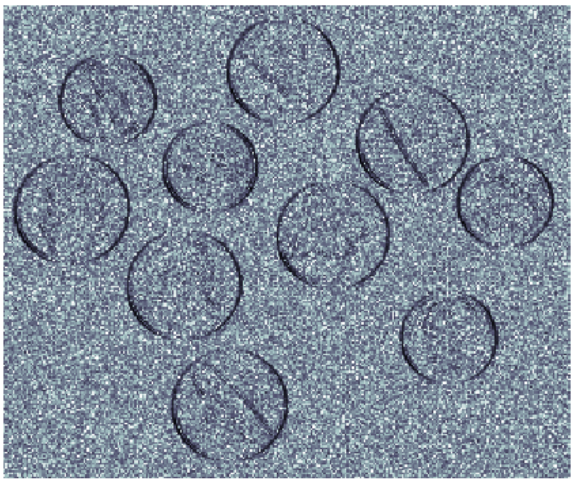
```
diffmeasure(I,R1,'mse')
diffmeasure(I,R2,'mse')

ans =

  106.9800


ans =

   69.9757
```

Original image

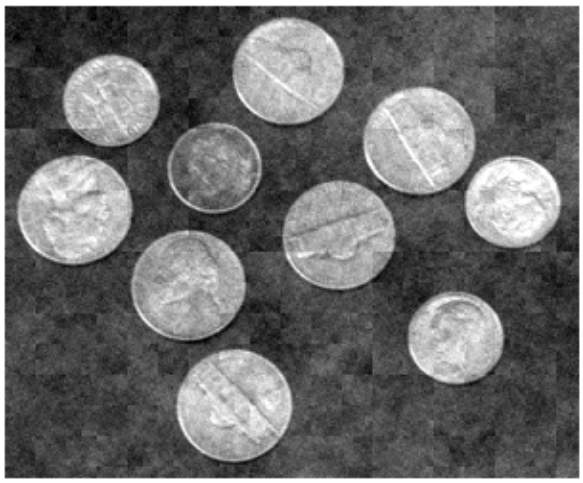
x gradient component of original



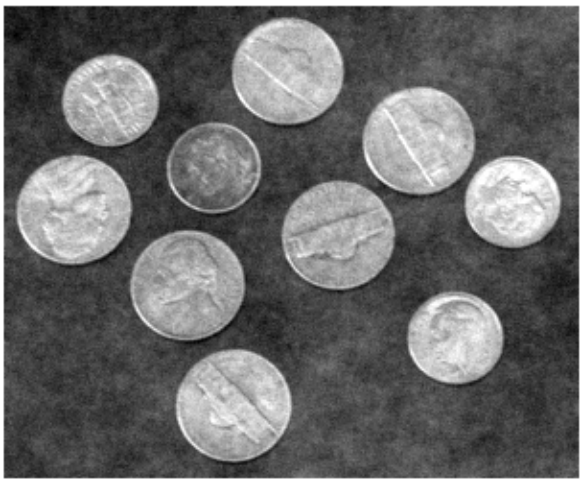
Noisy x gradient component - SNR = 0 dB



Reconstructed without Poisson



Reconstruction with Poisson



Mean squared error: 338.3005

Mean squared error: 221.2825

Note the elimination of the blocky artifacts with the use of the Poisson solver.

Acknowledgements

This document was created using images and publishing capabilities from MATLAB 2014a.

References

[1] I.S. Sevcenco, P.J. Hampton, P. Agathoklis, "A wavelet based method for image reconstruction from gradient data with applications", Multidimensional Systems and Signal Processing, November 2013

Earlier references using Fried gradient discretization (see sec. 2.1. in [1]):

[2] P.J. Hampton, P. Agathoklis, C. Bradley, "A New Wave-Front Reconstruction Method for Adaptive Optics Systems Using Wavelets", IEEE Journal of Selected Topics in Signal Processing, vol. 2, no. 5, October 2008

[3] P.J. Hampton, P. Agathoklis, "Comparison of Haar Wavelet-based and Poisson-based Numerical Integration Techniques", Proceedings of Circuits and Systems (ISCAS), pp.1623-1626, 2010