Image Fusion Using Wavelet Transform

Aggariyo Wanagiri[5419883] and Hans Brouwer[4376625]

Delft University of Technology

Abstract. This paper outlines an image fusion method using the wavelet transform, and a method using a guided filtering approach. The two methods of fusion are explained and are then implemented. The results are then compared against the image fusion function included in Matlab. Then, a series of metrics are used to assess the quality of fusion, followed by a comparison of the execution time of each method.

1 Introduction

Image fusion is a process where multiple images are fused together into one single image that ideally contains all the important characteristics and information from each source image. This is done for many applications, and is particularly useful for combining images from multiple imaging modalities. This report will investigate two methods of image fusion based on two papers: one using the wavelet transform, introduced by H. Li et al. [6], and another using guided filtering, introduced by S. Li et al. [7]. Moreover, a comparison will be made using multiple quality metrics, followed by a discussion of the findings, and concluding with recommendations on the strengths and weaknesses of each image fusion method.

2 Wavelet Transform Fusion

The wavelet algorithm can be summarized into the following steps[6]:

1. Wavelet transform
2. Identifying important features in the coefficients (activity measure)
3. Selecting the most relevant features (binary decision map)
4. Inverse wavelet transform

Wavelet transform The wavelet transform is used because it provides both spatial and frequency information of the image. This way, important features in the image are pronounced and can be selected more easily. Each iteration of the wavelet transform gives four sub images, where each one is a quarter the size of the original image. The four sub images are denoted as T_ϕ, T_ψ^H, T_ψ^V and T_ψ^D[2]. Where T_ϕ is the original image down-sampled two times, and T_ψ^H, T_ψ^V and T_ψ^D the horizontal, vertical and diagonal details coefficients respectively. The resulting coefficients are then joined together into one image. Figure 1 shows

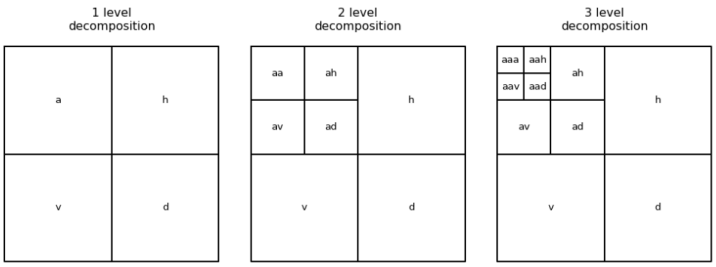

Fig. 1. Representation of three iterations of the wavelet transform[5]

three iterations of the wavelet transform and its resulting coefficients joined together into one image.

For this algorithm, the wavelet transform of two images that needs to be fused are calculated. Multiple iterations are done on T_ϕ until the image reaches a footprint smaller than the kernel size employed.

Activity Measure The activity measure is a way to find the most salient features in a neighborhood of pixels such as edges and lines. This is done by taking a window of the local neighborhood at each pixel, and calculating the absolute maximum associated with the center pixel over the chosen window size. By using the absolute maximum, more prominent parts of one image will be saved over the other when fused. Nevertheless, only using the absolute maximum of an area creates a problem. For example, if the highest pixel value comes from image A while the surrounding pixels consists of pixels from image B, the pixel from image A will be saved and the surrounding B pixels will be discarded despite being more prominent.

Binary Decision Map The binary decision map is used to verify the consistency of the surrounding pixels. The decision map is initialized to contain 0s when image A's activity is higher and 1s when image B's activity is higher. This creates a binary map of the intermediate fused image in the wavelet domain. Then a pair of majority filters is used over the binary map. First the unprocessed binary map is filtered, then negated and filtered again, and finally negated once more. This morphological step ensures that the decision map is smooth and consistent. The final fused image in the wavelet domain is formed the binary decision map. Subsequently, the inverse wavelet transform is used over each details coefficient iteratively until the last one, to obtain the final fused image.

3 Guided Filtering Fusion

The guided filter algorithm can be summarized into the following steps [7]:

1. Two-scale image decomposition (low-pass and high-pass)
2. Weight map construction
3. Guided filtering
4. Two-scale image reconstruction

Two-scale image decomposition Firstly, two source images that need to be fused are decomposed into two layers. The base layer contains the large scale variations in intensity and is obtained through the convolution of the source image with an average filter. The detail layer contains small scale variations in the image; obtained by subtracting the source image with the corresponding base layer.

Weight map construction The weight map is similar to the binary map created in wavelet-based fusion. In this case, it aims to create a binary map by comparing the saliency map of each image. The saliency map is a way to show the prominent details in an image and is obtained in two steps. First, a high-pass image is obtained by using a Laplacian filter on each source image. Then, a Gaussian low-pass filter is used on the absolute value of the resulting high-pass image to obtain its saliency map.

The weight map is then obtained by doing a pixel by pixel comparison of the two saliency maps. For example, if a pixel of saliency map A has a higher value than the corresponding pixel in saliency map B, then a 1 is chosen for that pixel in the weight map of A and a 0 is chosen in the weight map of B. Similar to the binary decision map in wavelet-based fusion, this weight map does not take into account the salience of the surrounding pixels, resulting in noise or unaligned object boundaries. Nevertheless, instead of using a majority filter, guided filtering is used in this implementation of image fusion.

Guided filtering The guided filter uses two inputs, a guidance image I_i and an input image P_i, which in our case are a source image and weight map respectively.

The goal of the filter is to apply a linear transformation to the input image such that its output is as close to the guidance image as possible. This can be done via a linear regression whose coefficients per window can be solved for in closed form:

$$a_k = \frac{\frac{1}{|w|}\sum_{i \in w_k} I_i P_i - \mu_k \overline{P_k}}{\delta_k + \epsilon} \tag{1}$$

$$b_k = \overline{P_k} - a_k \mu_k \tag{2}$$

Here $|w|$ is the number of pixels in the window, w_k. μ_k and δ_k are the mean and variance of I. ϵ is a regularization parameter provided by the user and $\overline{P_k}$ is the mean of P in w_k.

One issue with this linear regression step is that the solution for a given pixel can vary per window. Therefore the linear coefficients must be calculated for all windows and then averaged.

The guided filter is applied to the weight maps from the previous step with the source images as guides. This results in four guided weight maps as an output.

Two-scale image reconstruction The four resulting base and detail layers from the two source images can then be fused together using weighted averaging, resulting in a fused base layer and a fused detail layer. Finally, this can be summed together to form the fused image.

4 Quality Metrics

To assess the quality of fusion, S. Li et al. [7] used a series of metrics. We implemented three of these to assess our results: Q_y, Q_c, and Q_{NMI}.

Q_y is an image fusion metric introduced by Yang et al. [10] that relies on the SSIM [9] between each of the reference images and the fused image. The pixel-wise SSIMs are compared in terms of local variance: when they are similar, the variance weighted average is taken, when they are dissimilar, the maximum is used. This gives an adaptive SSIM score that prioritizes quality in the most structurally varied (and so presumably important) parts of the image.

Q_c was introduced by Cvejic et al. [1]. This applies a very similar weighting scheme based instead on the covariance between the images. Q_c uses the universal image quality index (UIQI) [8] for local image score. The UIQI metric evaluates the amount of distortion present in the image, so Q_c seeks to measure how well the fused image achieves low distortion with respect to its reference images.

Q_{NMI} simply takes the mean of the normalized mutual information [4] between each of the reference images and the fused image. This measures the amount of dependence betweeen the fused image and the reference images (i.e. the amount of information in the fused image belied by observing the reference images).

5 Results

To test the robustness of the algorithm, three sets of images are used shown in Fig. 2

We evaluated the two methods we implemented, `wavelet` and `guided_filter`, as well as compared against the image fusion function included in Matlab, `wfusimg`.

`wfusimg` Matlab's implementation works at a single level in the wavelet hierarchy and gives different options to fuse the results (min, mean, or max for the approximation and detail portions of the coefficients). After some manual, qualitative evaluation, we found that the best results were achieved when fusing Haar wavelets on level 2 with the mean of the approximation coefficients and the max of the detail coefficients. These were the settings benchmarked in all of the follow analysis.

`wavelet` Our implementation of `wavelet` uses the Python package PyWavelets to calculate the 2D wavelet transform of the inputs. This supports a number of different wavelet families. First we evaluated which of the different families

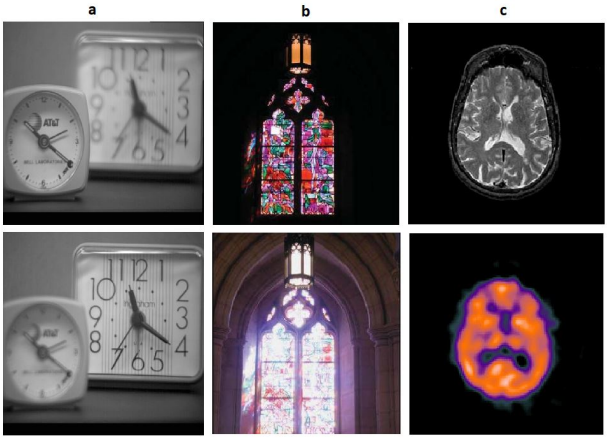

Fig. 2. The three sets of images used to evaluate fusion performance.

and kernel sizes gave the best performance for our wavelet-based fusion. We evaluated all of the supported discrete wavelets (Haar, Daubechies, Symlets, Coiflets, Biorthogonal, Reverse biorthogonal, and discrete Meyer) with kernel sizes varying from 3 to 40. This resulted in about 1300 different configurations to test. We recorded the Q_y, Q_c, and Q_{NMI} for each configuration on our test pairs and selected the best performing wavelets according to each metric. The fused image of the top 5 configurations per metric are shown in Figure 3.

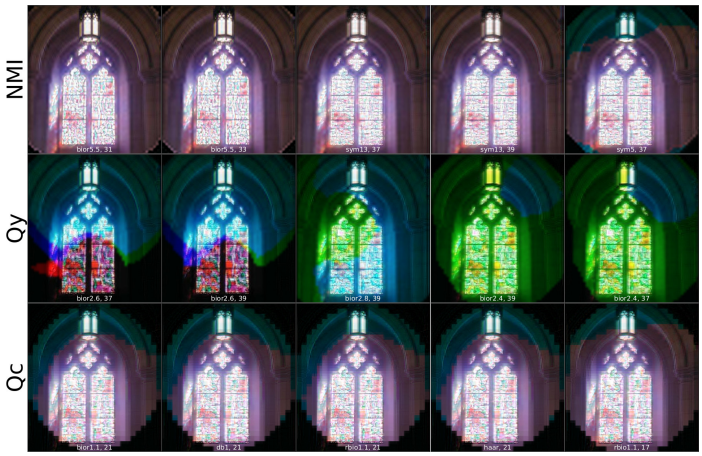

Fig. 3. Results of various wavelet families and kernel sizes. The specific wavelet and kernel size used for each are listed in white on the image. Each row shows the top 5 performing wavelets according to the metric listed on the left.

Immediately apparent is that the metrics used for evaluation do not necessarily correspond to "naturally" fused images. Many fused images have clear vignettes (which are not penalized due to zero variance areas across images leading to divisions by zero in both Q_y and Q_c which are disregarded for the score). Q_y also does not seem to penalize mis-colorizations, presumably due to the underlying SSIM metric being focused on image structure. Q_c does apparently penalize mis-colorization, which makes sense as the underlying UIQI seeks to minimize luminance and contrast distortion with resepct to the reference images. Q_{NMI} seems to be most directly correlated with images that seem to be qualitatively well-fused.

Based on this result we adopt the Symlet13 wavelet with kernel size 37 as the default setting for our `wavelet` method (this is the highest-scoring configuration that introduced no vignette).

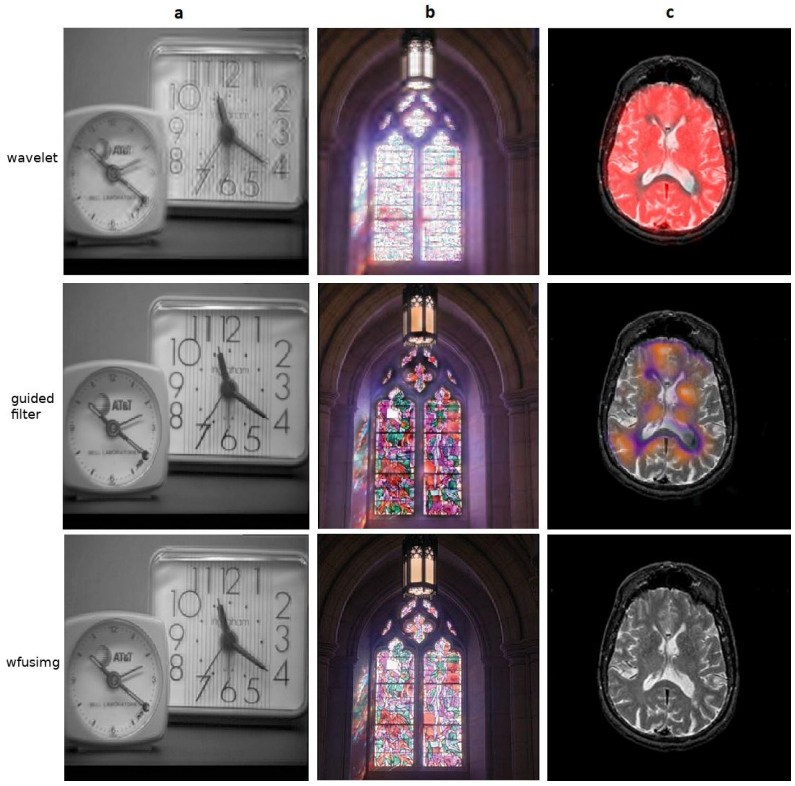

Fig. 4. Results of fusion of each of the three methods on our test pairs.

Qualitative The results of fusion are shown in Figure 4. There does not seem to be a clear overall winner. The `guided_filter` seems to come out ahead in test

pair **a**, just barely clearer than `wfusimg` and significantly better than `wavelet`. However, on test pair **b**, `guided_filter` seems to give an unnaturally fused result while `wfusimg` seems better natural. On the fusion of test pair **c**, none of the methods seems to perform particularly well. `wfusimg` does not support merging images with different numbers of channels and essentially seems to discard the colored part (or at least its effect on the output is hard to discern). `guided_filter` only merges certain portions of the colored signal. `wavelet` seems to fair best of the methods, although it discards the majority of the color channels, favoring just red.

Quantitative Table 1 shows the metric values achieved by each method on the test pairs. `guided_filter` generally achieves the highest scores across all metrics. However, as we found in our earlier analysis of the best wavelet families and as can be seen in the fused images, this does not always translate to achieving the best result.

Table 1. The scores of the three methods on each test pair

Method	Metric	Test pair **a**	Test pair **b**	Test pair **c**
`wavelet`	Q_{NMI}	1.155	1.247	1.094
	Q_c	0.374	0.677	0.275
	Q_y	0.763	0.887	0.878
`guided_filter`	Q_{NMI}	**1.269**	**1.420**	1.085
	Q_c	**0.558**	**0.809**	**0.418**
	Q_y	**0.938**	**0.983**	0.926
`wfusimg`	Q_{NMI}	1.220	1.313	**1.241**
	Q_c	0.473	0.801	0.291
	Q_y	0.798	0.957	**0.955**

SSIM To gain a little more insight into how the each algorithm was performing and why we were seeing these results, we analyzed the SSIMs of test pair **b** for the different algorithms. The results can be seen in Figure 5 below.

The results here clearly show why `guided_filter` scores best on Q_y. The fused image has essentially perfectly reproduced the structure of reference B outside the window (white corresponds to a score of 1, the maximum), while taking the structure from A within the window. The fused pixel-wise SSIM (Q_y) is therefore also very strong.

`wfusimg` seems to compromise between the two SSIMs (apparent in the lighter green in $SSIM_A$ and darker grey in $SSIM_B$). The result is a Q_y which has no strong structural distortions anywhere, but is neither perfect anywhere. Apparently, this quality in the SSIMs leads to a more natural looking result where both references are represented less than perfectly, rather than merging the two parts at any cost as `guided_filter` seems to do.

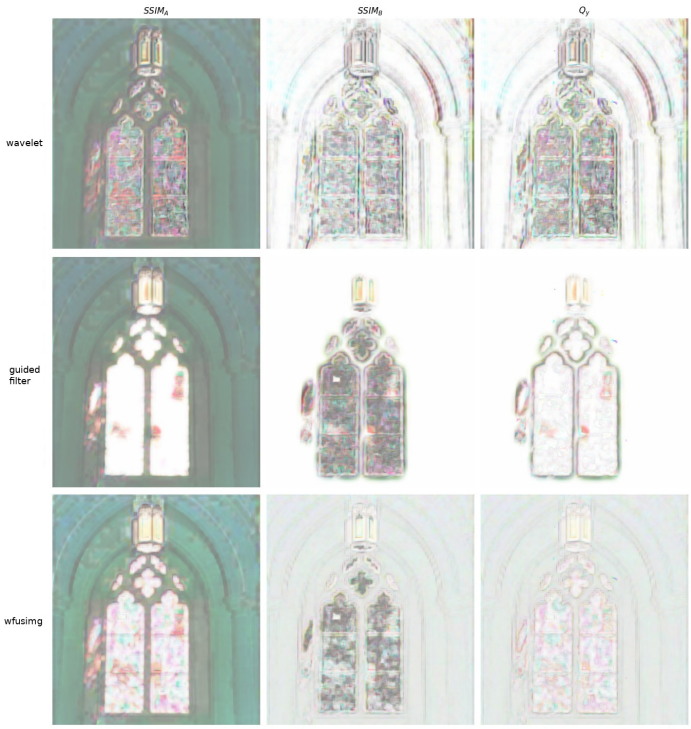

Fig. 5. Pixel-wise SSIMs (between reference A & the fused image (left) and B & the fused image (right)) as well as pixel-wise Q_y for per method on test pair **b**.

`wavelet` has the worst SSIMs and Q_y of the three methods (apperent by its significantly lower score in Table 1). Interestingly, Q_y seems to correspond strongly to $SSIM_B$. Likewise the fused image for `wavelet` is much closer to reference B. It seems that the metric is biased toward this SSIM result because the local SSIM variance is higher. The effect is that Q_y scores the fusion fairly high even though the structure from A within the window is not faithfully reproduced.

Time Finally we investigate the execution time of each algorithm. Figure 6 shows the scaling of execution time with respect to the area of the image. The experiments were performed on a consumer grade laptop with a 1.6 GHz Intel Core i5-8265U CPU and 16 GB of RAM.

The results show that the `wfusimg` scales most favorably with respect to image size. This is logical as it only requires fusing a single level of the wavelet hierarchy, whereas the `wavelet` requires fusing each level from top to bottom. `guided_filter` is even more expensive.

The results for `wavelet` and `guided_filter` are using our final fast implementation. The original naive versions used hand coded for-loop-based convolutions and filters. The naive `guided_filter` required 43 seconds to fuse 256x256

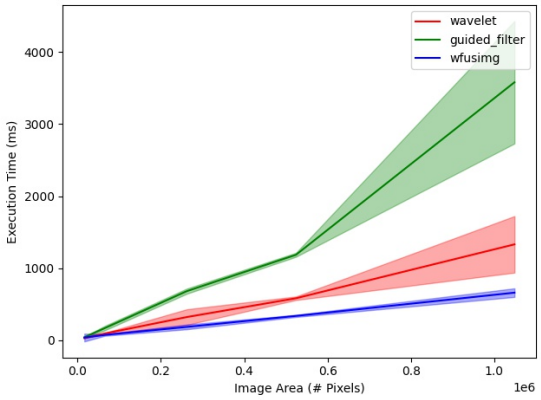

Fig. 6. Execution time in milliseconds of each algorithm when fusing images with varying number of pixels. The shaded region shows the 95% confidence interval of the execution time over 100 trials.

color images and the `wavelet` needed 12 seconds for the same images. Note that we did not bother to implement the wavelet transform with for loops and instead directly chose to use PyWavelets due to the severely degraded performance we saw with the manual approach to `guided_filter`. Our fast version of `guided_filter` uses the linear-time box filter algorithm introduced in the original guided filter paper by He et al. [3] as well as the FFT-based convolution from the Python package Sci-kit Image. The box filter approach and fast convolution proved to be useful for similar primitives in `wavelet` as well as the evaluation metric implementations of Q_y and Q_c.

6 Conclusion

We were able to successfully reproduce many of the results found by Li et al. and Li et al. Our implementations compared favorably to the well-tested implementation from Matlab. No algorithm proved definitively better than the others in terms of our qualitative and quantitative analysis. However, in terms of speed and general performance, Matlab's implementation does seem to be the most favorable overall as long as all fusion targets have the same number of channels. One interesting result is that the quantitative image fusion metrics did not always correspond with what seemed to be the best fusion from a perceptual standpoint. This highlights the need for further research into metrics for image fusion. Another interesting note is the large variance in fusion quality of the `wavelet` algorithm with respect to the wavelet and kernel size. When using this setting, it is important to try multiple configurations to attain the best results. Image fusion has come very far since the 1990s, more modern approaches show clear improvements over their predecessors in terms of performance and general usability.

References

1. Cvejic, N., Loza, A., Bull, D., Canagarajah, N.: A similarity metric for assessment of image fusion algorithms **2**
2. Gonzalez, R.C., Woods, R.E.: Digital image processing fourth edition (2018)
3. He, K., Sun, J., Tang, X.: Guided Image Filtering. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) Computer Vision – ECCV 2010. pp. 1–14. Lecture Notes in Computer Science, Springer. https://doi.org/10.1007/978-3-642-15549-9
4. Hossny, M., Nahavandi, S., Creighton, D.: Comments on 'Information measure for performance of image fusion' . https://doi.org/10.1049/EL:20081754
5. Lee, G.R., Gommers, R., Waselewski, F., Wohlfahrt, K., O'Leary, A.: Pywavelets: A python package for wavelet analysis. Journal of Open Source Software **4**(36), 1237 (2019)
6. Li, H., Manjunath, B., Mitra, S.K.: Multisensor image fusion using the wavelet transform. Graphical models and image processing **57**(3), 235–245 (1995)
7. Li, S., Kang, X., Hu, J.: Image fusion with guided filtering. IEEE Transactions on Image processing **22**(7), 2864–2875 (2013)
8. Wang, Z., Bovik, A.: A universal image quality index **9**(3), 81–84. https://doi.org/10.1109/97.995823
9. Wang, Z., Bovik, A., Sheikh, H., Simoncelli, E.: Image quality assessment: From error visibility to structural similarity **13**(4), 600–612. https://doi.org/10.1109/TIP.2003.819861
10. Yang, C., Zhang, J.Q., Wang, X.R., Liu, X.: A novel similarity based quality metric for image fusion **9**(2), 156–160. https://doi.org/10.1016/j.inffus.2006.09.001, https://www.sciencedirect.com/science/article/pii/S1566253506000704