# Contents

1 Contents ................................. 1
   1.1 Command line tools .................. 1
   1.2 Algorithm description ............... 1

2 Installation ............................ 3
1.1 Command line tools

This project contains the following command-line tools:

1. **fuseimages** - Takes one or more source frames and produces various fused results from them. See the output of `fuseimages --help` for more information.

1.2 Algorithm description

This section provides a brief overview of the DTCWT-based fusion algorithm as used in this project. The input to the algorithm is a set of input images, \( \mathcal{I} \).

1.2.1 Alignment

This step aligns each input image with a single translation to match as well as possible a template image. The central image of \( \mathcal{I} \) is selected as the template image \( T \). For each image \( I \in \mathcal{I} \):

1. Compute the cross-correlation image \( C = (I \cdot w) \ast (T \cdot w) \) where \( w \) is a two-dimensional Hamming window, \( \cdot \) denotes pixel-wise multiplication and \( \ast \) is the cross-correlation operator. Normalise this cross-correlation, \( C \rightarrow C / (w \ast w) \) where \( / \) denotes element-wise division.

2. Find the location of the maximum of \( C \) and compute the corresponding translational shift for that location. The maximum is found ignoring an apron around the edge of the image to avoid over matching of small overlap-regions.

3. Warp \( I \) according to that translation.

Combine all aligned images into the set of aligned images, \( \mathcal{I}_a \).

1.2.2 Registration

This step locally warps each aligned image to best match the same template image as above. For each image \( I \in \mathcal{I}_a \):

1. Compute the local affine warp mapping \( I \) to \( T \) as described in [1, 2].

2. Warp \( I \) according to the registration.

Combine all registered images into the set of registered images, \( \mathcal{I}_r \).
1.2.3 Fusion

This step combines all images in $I_r$ into a single fused image. The fusion is performed in the wavelet domain and is based on the technique in [3]. The overall lowpass image is computed by taking the mean of the lowpass images corresponding to each image in $I_r$. Letting $\theta_d^{(i)}$ correspond to the $j$-th highpass coefficient in direction $d$ at level $\ell$ of the DTCWT transform of the $i$-th image in $I_r$, we can construct the fused wavelet coefficients $\theta_d^{(i)}$ in the following way:

1. Compute $\Theta_{d,\ell,j} = \sum_i \theta_d^{(i)}$ and then $\phi_{d,\ell,j} = \Theta_{d,\ell,j} / |\Theta_{d,\ell,j}|$. These unit-magnitude complex numbers represent the average phase of corresponding wavelet coefficients over all registered images.

2. Form the set $T_{d,\ell,j} = \{ |\theta_d^{(1)}|, |\theta_d^{(2)}|, \ldots, |\theta_d^{(N)}| \}$ for the $N$ images in $I_r$. Select $T_{d,\ell,j}$ from this set using some heuristic. In the current implementation this can be one of: mean value, maximum value or maximum value after 2-sigma outliers are removed. Which strategy is best may depend on input imagery.

3. Compute $\theta_{d,\ell,j} = T_{d,\ell,j} \phi_{d,\ell,j}$.

4. Inverse DTCWT to form the fused image $I_f$.

1.2.4 Shrinkage

The wavelet coefficients of the fused image $I_f$ were selected to maximise sharpness. This may cause noise to be incorrectly preserved in the output image. A wavelet coefficient shrinkage method based on that in [4] is then applied to give the final fused and denoised image.

1.2.5 References


The easiest way to install the library is via `pip`:

```
$ pip install git+https://github.com/rjw57/dtcwtfusion.git
```

Along with support libraries, this will also install the command line tools.