Development of enhanced multi-spot structured illumination microscopy with fluorescence difference

EDWARD NICHOLAS WARD

1. Programming

Background and system requirements.

Throughout the project coding was performed in MATLAB 2017a and 2018b and has been tested in both environments. The code was run on a 64-bit system with an intel i7-7700 processor and 16GB of random access memory. The graphical processor unit used was a Nvidia GTX 1050Ti or GTX 1060. Hardware control was achieved using LabVIEW 2016 with the IMAQ, IMAQdx and DAQmx packages.

Detailed below is the pseudocode for the reconstruction and image processing steps respectively.

Local maxima pseudocode

The local maxima were found as follows:

```
>>>  % To remove high frequency noise, the image is convolved
>>>  % with the detection PSF.
>>>  % The convolution is performed as an element-wise multiplication
>>>  % in frequency space. A threshold is then subtracted to ensure only peaks
>>>  % above a certain intensity are found.
>>>  OTF = Fourier transform PSF
>>>  FOR each frame
>>>          FT = Fourier transform pattern frame;
>>>          FD = FT * OTF;
>>>          Filtered = inverse Fourier transform of FD;
>>>          Filtered = Filtered – threshold;
>>>  % Negative values are then removed.
>>>          Filtered (Filtered<0) = 0;
>>>  % The image is then binarised, i.e. all image values > threshold
>>>  % are set to 1 and all others to 0.
>>>          Filtered (Filtered>0) = 1;
>>>  % The image is then eroded. This takes clusters of positive pixels and
>>>  % converts them to a single positive pixel at the middle of the cluster.
>>>          Eroded = erode filtered image;
>>>  % The coordinates of the excitation PSFs are then extracted as the
>>>  % locations of all the non-zero elements of the eroded image.
>>>          Coordinates = find non-zero elements of Eroded;
>>>  END
```

Joint Richardson Lucy pseudocode

The workflow for JRL is outlined below:

```
>>>   FOR the number of iterations
>>>         FOR each frame
>>>   % The update step is calculated through a single temporary variable.
>>>   % The first three steps generate an expected diffraction limited image.
>>>         temp = pattern frame * estimated structure;
>>>         temp = Fourier transform of temp;
>>>         temp = inverse Fourier transform of temp * detection OTF;
>>>   % This is then compared to the raw data incorporating Gaussian noise
>>>   % with a variance, sigma, into the reconstruction.
>>>         temp = (raw data frame + sigma) / (temp + sigma);
>>>   % This is then convolved with the detection PSF.
>>>         temp = Fourier transform of temp;
>>>         temp = inverse Fourier transform of temp * detection OTF;
>>>   % The update is then formed form the pattern and temp. This is updated
>>>   % for each frame and then applied after all frames have been processed.
>>>         temp = temp * pattern;
>>>         update = update + temp;
>>>         END
>>>   estimate = estimate * update;
>>>   END
>>>   %While holding the information in a constantly changing temporary variable is a
>>>   %convoluted approach, it accelerates performance as the multiple cores are not
>>>   %attempting to access the same memory location simultaneously.
```

Pattern Illuminated Fourier Ptychography pseudocode

The workflow for PIFP is outlined below:

```
>>>  % As the Fourier Transform of the patterns is used throughout and remains
>>>  % constant, this is calculated before the iterations begin
>>>  FD = Fourier transform of patterns
>>>  % Begin the PIFP iterations.
>>>  FOR the number of iterations
>>>      FOR each frame
>>>  % PIFP works best if the order of pattern illuminations is shuffled so a frame and.
>>>  % matching pattern are chosen at random and the predicted response is calculated
>>>      FR = random raw data image;
>>>      PR = matching random pattern;
>>>      response = estimate sample * PR;
>>>  % We move to the Fourier domain to perform intermediate calculations.
>>>      FT = Fourier transform of response;
>>>      FT = FT - OTF * Fourier transform of PR;
>>>      FT = FT + conjugate of OTF * Fourier transform of FD;
>>>  % We then move to real space to calculate the update.
>>>      update = absolute value of the inverse Fourier transform of FT;
>>>      update = update – response;
>>>      update = update * PR / max value of PR²;
>>>  % We then apply the update to the estimate.
>>>      estimate = estimate + update;
>>>      END
>>>  END
>>>  % Looking at the code, the update is applied after every frame rather than after
>>>  % every iteration as in JRL. In addition, the number of variables that must be
>>>  % held simultaneously increases the memory usage. The combination of these
>>>  % two factors means that PIFP is poorly suited to GPU acceleration.
```

Estimating the pattern shift

The workflow for pattern shift estimation is outlined below:

```
>>>  % To build the ideal pattern stack the shift of each grid relative
>>>  % to the previous one is calculated.
>>>  shift = pattern spacing (pixels) / number of shifts;
>>>  FOR every frame
>>>          next frame in stack = shifted previous frame;
>>>  END
>>>  % The global shift of this ideal stack over the image must then
>>>  % be determined by scanning the whole ideal stack over the predicted
>>>  % sample and comparing to the raw data.
>>>  predicted sample = mean of all frames of the raw data;
>>>  FOR every x shift scanned
>>>          FOR every y shift scanned
>>>                  FOR every frame
>>>                          shift the corresponding frame of the ideal pattern…
>>>                          … by the x and y shifts being tested;
>>>                          predicted image = pattern frame * predicted sample;
>>>                          difference = (predicted imaged – raw data)²;
>>>                  END
>>>          END
>>>          error of this shift = sum of all differences;
>>>  END
>>>  global shift = shift with the lowest error value;
>>>  apply the global shift to the ideal pattern;
```

Downloads:

The full code and supporting documentation can be found at:

https://github.com/edward-n-ward

This also includes the programs used for the holographic projection and hardware control. The relevant publications can be found at Ward et al. 2018[1] and 2019[2] for the simulations and holographic projection respectively.

2. References

1. E. N. Ward, F. H. Torkelsen, and R. Pal, "Enhancing multi-spot structured illumination microscopy with fluorescence difference," R. Soc. Open Sci. **5**(3), 171336 (2018).

2. E. N. Ward and R. Pal, "Holographic projection for multispot structured illumination microscopy," J. Microsc. **274**(2), 114–120 (2019).