Monthly report

(08/15/2015-09/11/2015)

Weizhi Li, Algorithm Engineer

Item lists:

1.1 Code bugs in Axosoft

From last month, we have begin to work on the code bug list in Axosoft. The algorithm team has spent more time in this part during this month. This part will be significant part in our algorithm verification and validation part.

12	**Bug #4 Createcolor.m**	**Ready For T...**	**Medium**	**Weizhi Li**	**Testing**	**4 hrs**	**1 hrs**	**0.6 hrs**
13	**Bug# 5 averageFrameMatrixImage.mw**	**Fixed**	**Medium**	**Weizhi Li**	**Closed**	**1 hrs**	**2 hrs**	**0**
14	Bug #6 averageFrameMatrixImage.m	Fixed	Medium	Yang Lu	Closed	1 hrs	2 hrs	0
15	**Bug #7 averageFrameMatrixImage.m**	**Fixed**	**Medium**	**Weizhi Li**	**Closed**	**1 hrs**	**1 hrs**	**0**
16	Bug#8 estimateHRimage12.m	Fixed	Medium	Yang Lu	Closed	1 hrs	1 hrs	0
17	Bug #9 estimateHRimage12.m	Fixed	Medium	Yang Lu	Closed	1.5 hrs	1.5 hrs	0
18	Bug #10 estimateHRimage12.m	Fixed	Medium	Yang Lu	Closed	1 hrs	1 hrs	0
19	Bug# 11 estimateHRimage12.m	Fixed	Medium	Yang Lu	Closed	1 hrs	1 hrs	0
20	**Bug #12 generatePPGfftArrays_deco...**	**Fixed**	**Medium**	**Weizhi Li**	**Closed**	**1 hrs**	**0**	**0**
21	**Bug# 13 generatePPGfftArrays_deco...**	**Fixed**	**Medium**	**Weizhi Li**	**Closed**	**1 hrs**	**1 hrs**	**0**

For the bug lists, I have done the bugs fix below:
(1). Determine the effect of Deconvolution part

(2). Determine the real SNR calculation to the change of our final outputs

(3). Determine the change of real signal integration part in our final outputs.

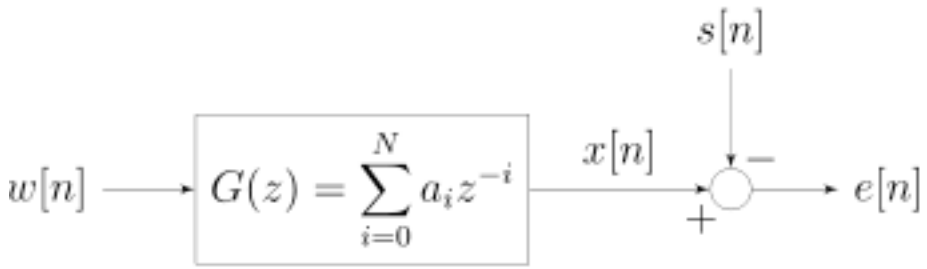
For the first one, the deconvolution part:

From the view of mathematics part:

f*g = h,

f is the target signal, and h is the observered signal. The function g is the transfer function. So, if we know g, then we can recover the target signal f. This is the same conception in Wiener deconvolution, it means that we need to get the Wiener filter through optimization techniques — the least mean squares filter. After getting the inverse system function, we can recover the signal.

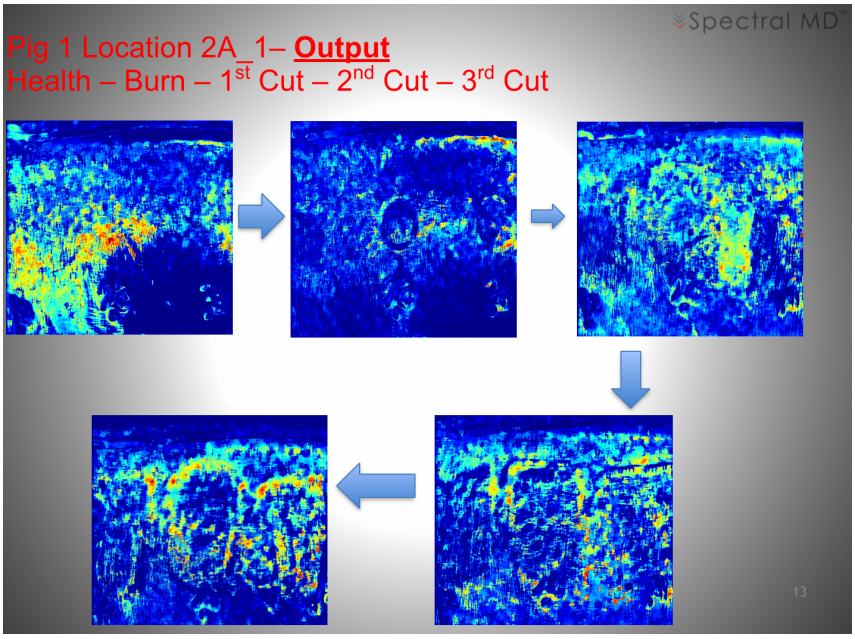
In our algorithm, we have this techniques — Deconvolution conception. And, for our PPG signal is stationary signal and stochastic processing.

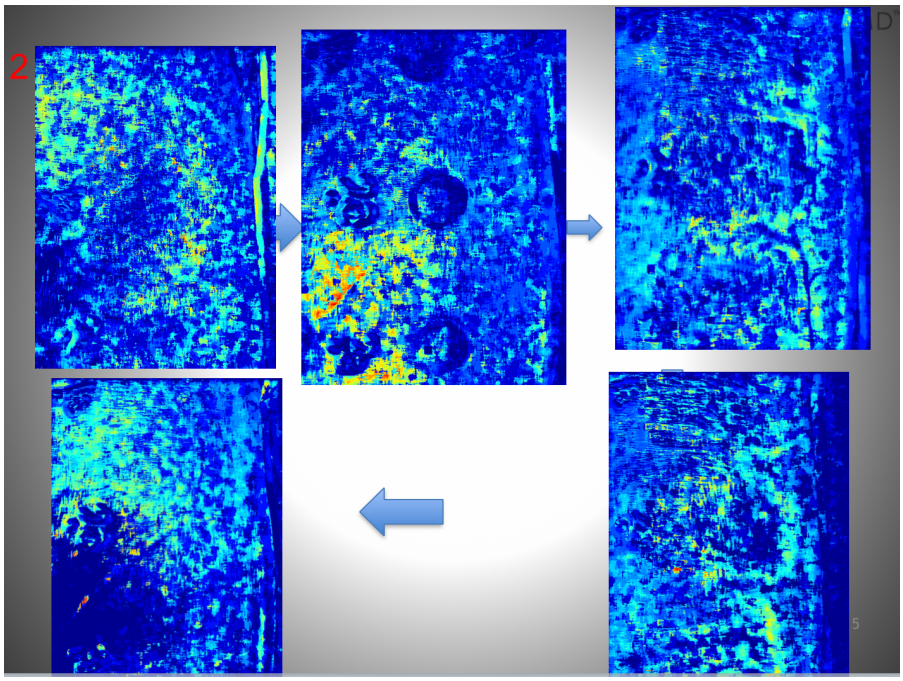


And, we design a low pass FIR filter and extract the frequency information, the cut off frequency is 0.75 Hz, and use the inverse FFT to get the time information, which can be considered as inverse system function. By dividing the function g, we may get the optimal target signal.

However, it may not real deconvolution part in algorithm design view, it looks like normalization part. Because we can use other algorithms, like Hiltert transform to do the envelope. It may from different view to check the system transfer matrix g.

From the discussion above, it is necessary to do the deconvolution testing, which may improve the outputs quality. I have ran 120 test data from this June animal data. In order to prove the outputs, I just show one location in pig one, the output below:

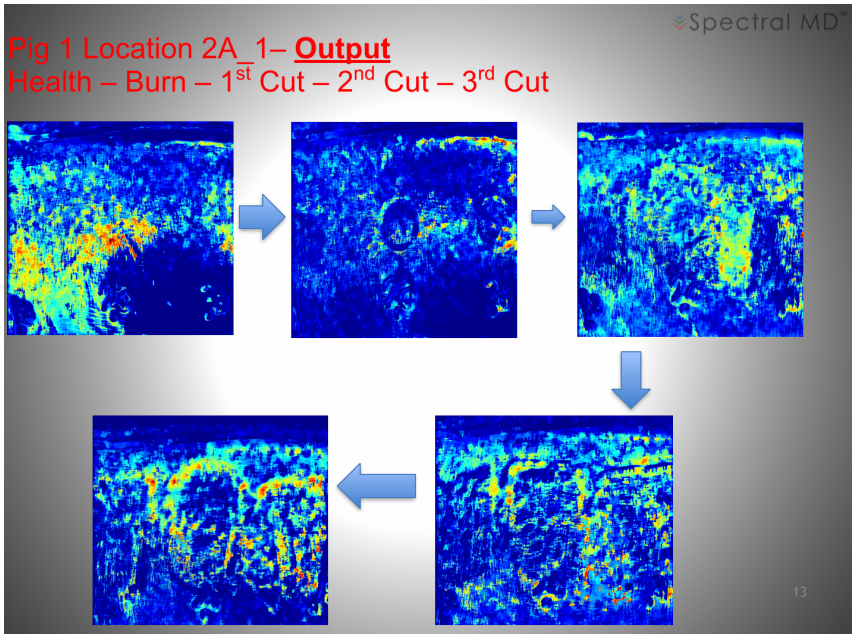


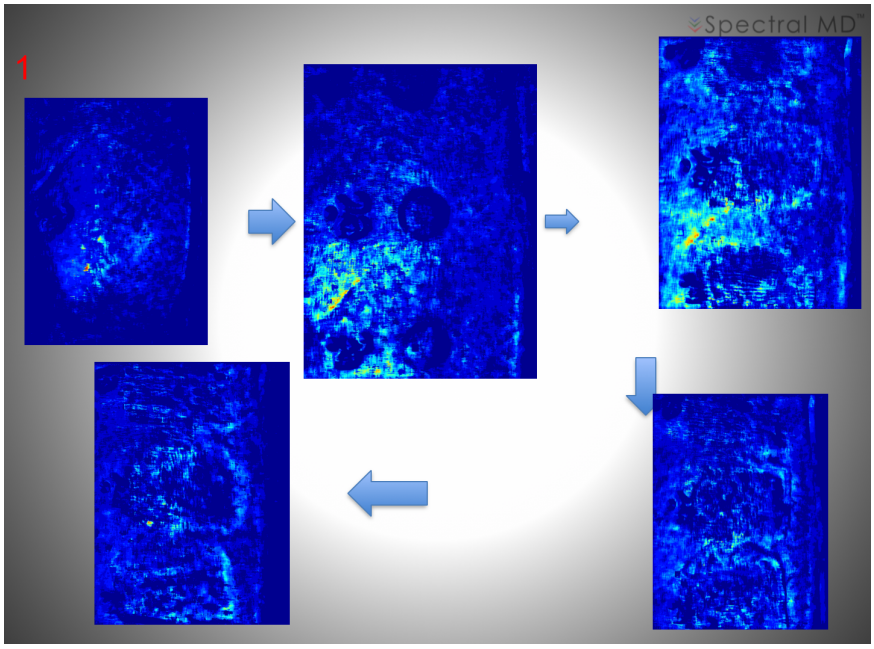


From the algorithm side, we did not get the quantify variable from clinic side, the only one variable we used is the heart rate. From the outputs we got, both of these algorithms can detect correct heart rate. So, in this case, we keep using the deconvolution step.

2. Determine the real SNR calculation to the change of our final outputs

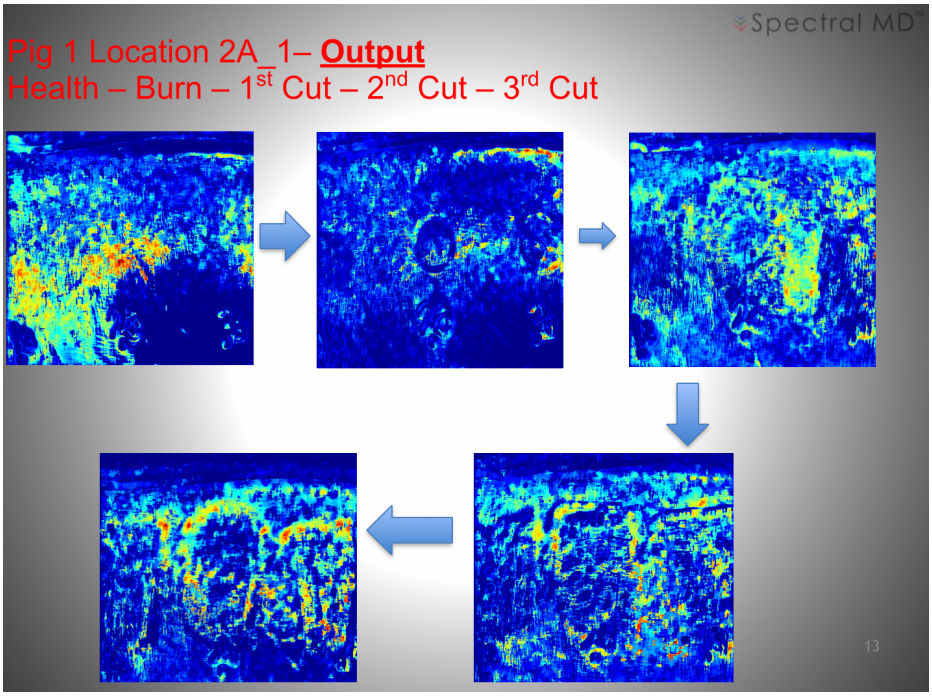
From the code, we found we have found that we did not use real signal energy in our calculation, we use the mean of signal. Please see the outputs below, it seems that we have lose some information, we keep use the mean of signal to present real signal energy. It may show more information by gamma correlation. Because of the limit of time, we did not do it now

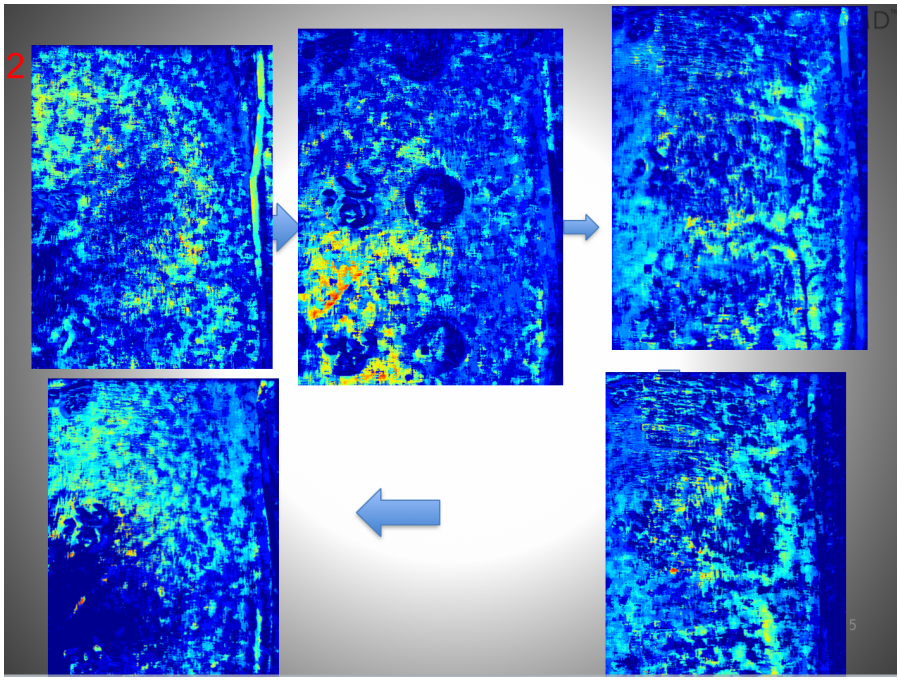




(3). Determine the change of real signal integration part in our final outputs.

In frequency domain, we need to do the signal integration to get the energy power. I have done this test to check the range of signal. The reason is the same as the reason one, we cannot get the any quantify variable from the clinic side. This test is going on. Please check the outputs below:


Spectral MD
Pig 1 Location 2A_1– Output
Health – Burn – 1st Cut – 2nd Cut – 3rd Cut
13


2
5

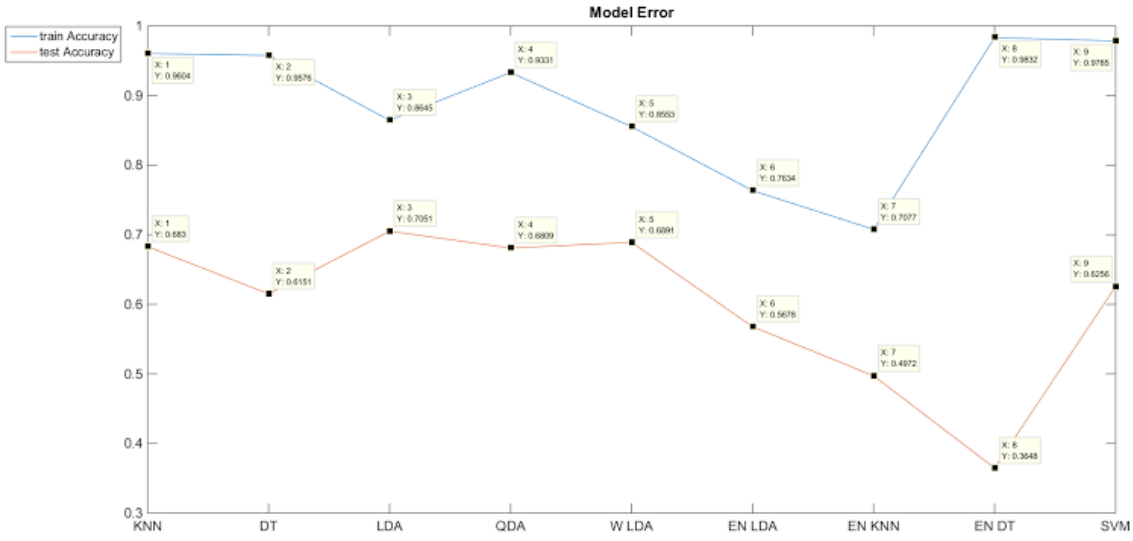
2.1 Code merge

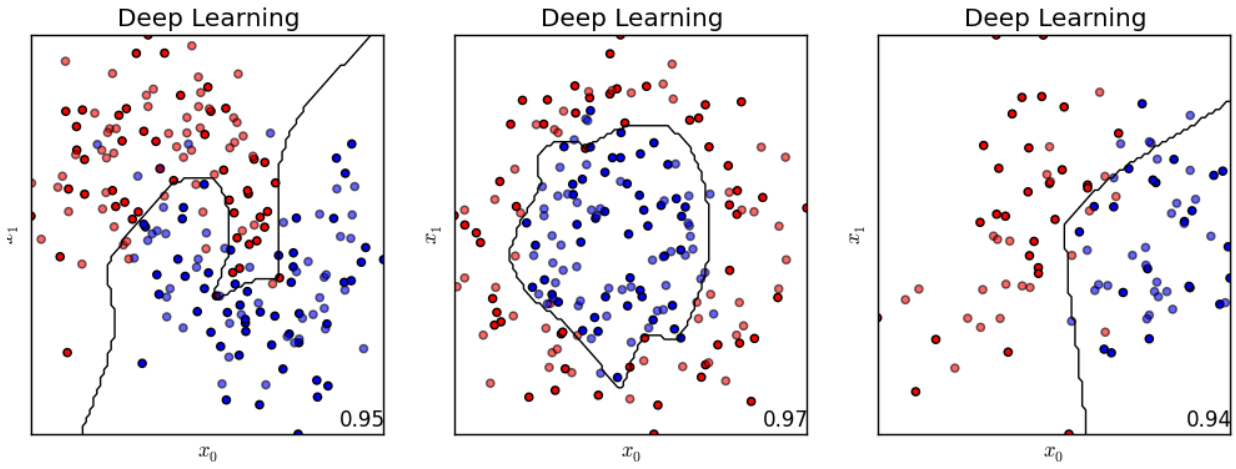
Finish the code merge and generate the new code version 1.3.0. It may be tested in next month during the software integration part.

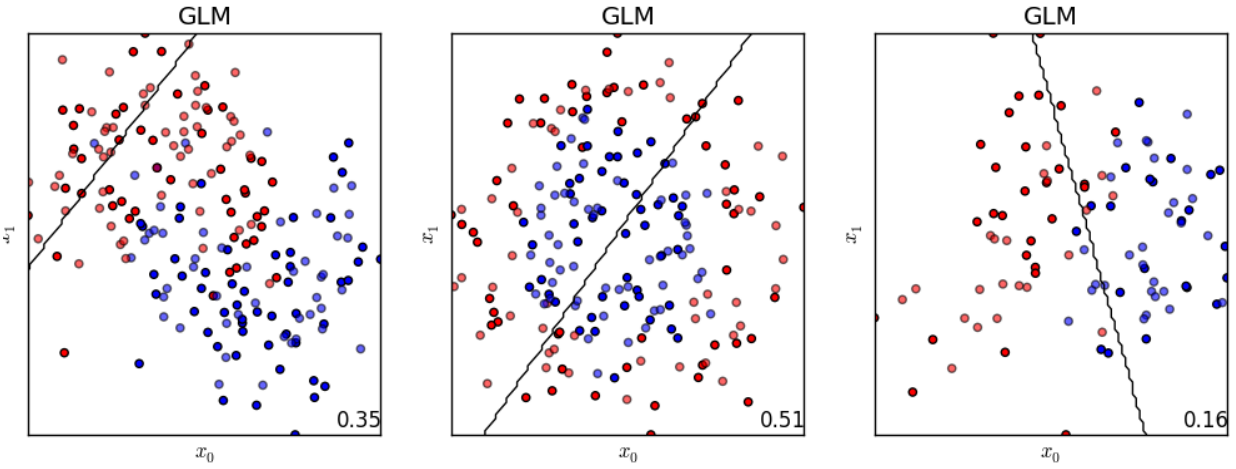
3.1 Model Boundary explanation.

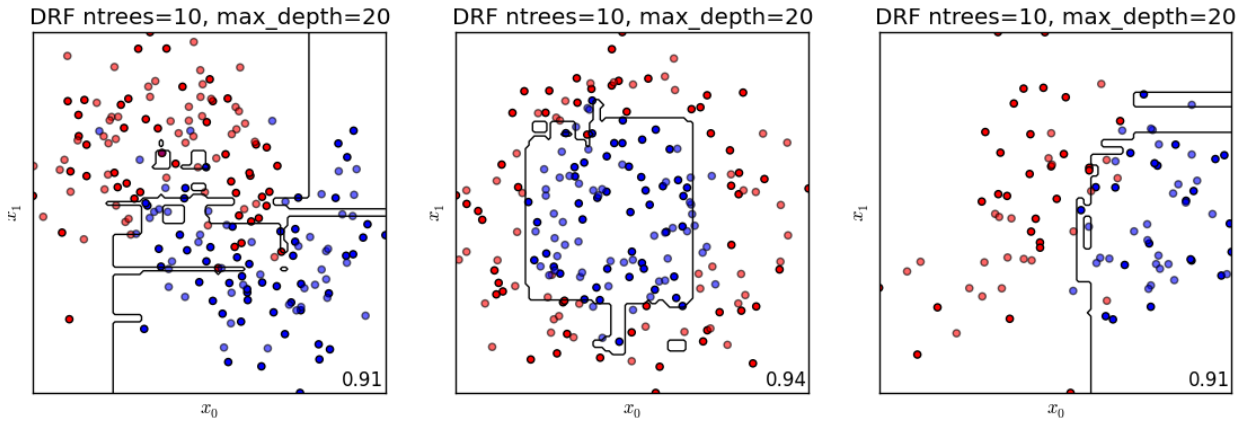
In order to do the model explanation and improve the accuracy of prediction, we have spent more time. In last monthly report, I have used K-nearest neighbors (KNN), decision tree (DT), linear discriminant analysis (LDA), weighted linear discriminant analysis (W-LDA), quadratic discriminant analysis (QDA), ensemble linear discriminant analysis (EN-LDA), ensemble K-nearest neighbors (EN-KNN), ensemble decision tree (EN-DT), and support vector machine (SVM).

The mean test accuracy of the algorithms were KNN 68.3%, DT 61.5%, LDA 70.5%, W-LDA 68.1%, QDA 68.9%, EN-LDA 56.8%, EN-KNN 49.7%, EN-DT 36.5%, and SVM 62.6%. Please see the curve above. LDA had the highest test accuracy (70.5%).
Continuing this part work, I have used the machine learning algorithms to plot the boundary of 2-D features in our data.




Deep Learning
Deep Learning
Deep Learning
0.95
0.97
0.94
x_0
x_1


GLM
GLM
GLM
0.35
0.51
0.16
x_0
x_1


DRF ntrees=10, max_depth=20
DRF ntrees=10, max_depth=20
DRF ntrees=10, max_depth=20
0.91
0.94
0.91
x_0
x_1

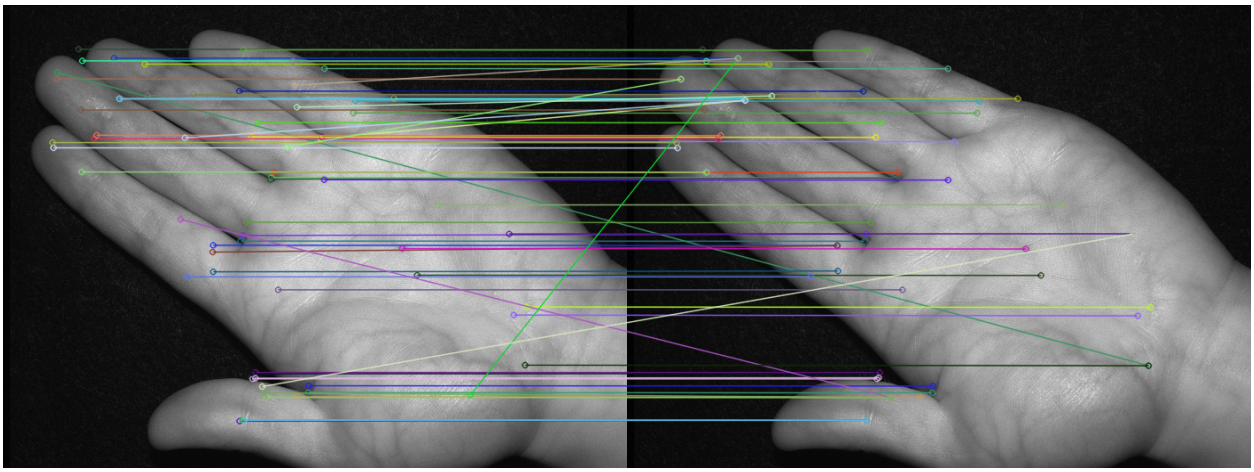
4.1 Data testing

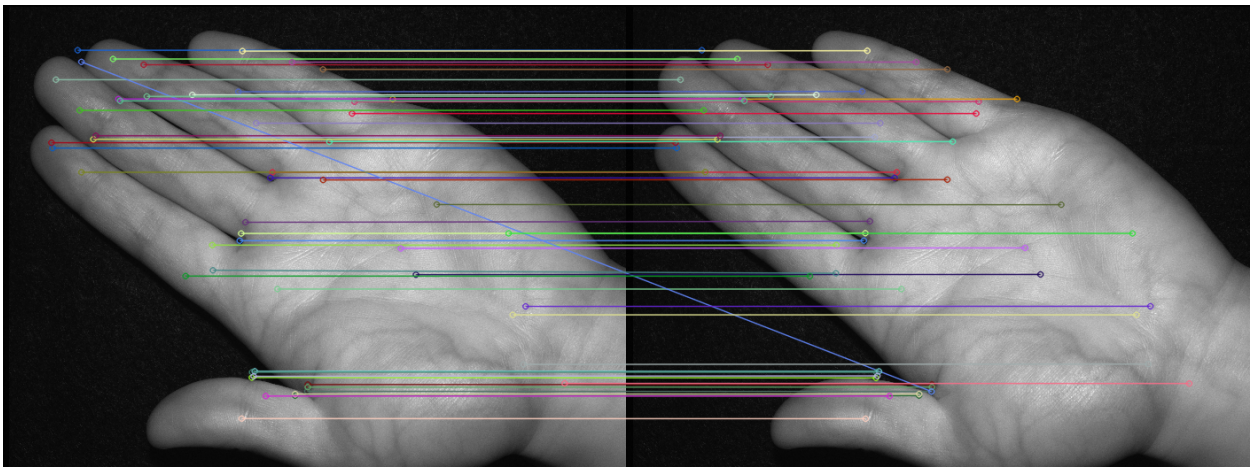
In oder to do the motion detection, we have shown the outputs below and used the feature match algorithms to quantify the motion part, like FLANN (Fast Approximate Nearest Neighbor Search) to finish this part. We used the Open-Source library, OpenCV (Its interface to FLANN library). The code are below:
class flann:: Index_

This class is template with the type of elements for which the index is built. [2 http://docs.opencv.org/modules/flann/doc/flann_fast_approximate_nearest_neighbor_search.html], please see the method below [3 http://docs.opencv.org/modules/features2d/doc/common_interfaces_of_descriptor_matchers.html]:

```
class FlannBasedMatcher : public DescriptorMatcher
{
public:
    FlannBasedMatcher(
      const Ptr<flann::IndexParams>& indexParams=new flann::KDTreeIndexParams(),
      const Ptr<flann::SearchParams>& searchParams=new flann::SearchParams() );

    virtual void add( const vector<Mat>& descriptors );
    virtual void clear();

    virtual void train();
    virtual bool isMaskSupported() const;

    virtual Ptr<DescriptorMatcher> clone( bool emptyTrainData=false ) const;
protected:
    ...
};
```

During the month, I have also tried other match algorithms, like KNNMATH, radius Match. It need more time to do the testing. Because of lots of time are spent in algorithm verification and validation part, this part. I may talk more details and show the comparison in next months.





C:\Users\Algorithm 001\documents\visual studio 2013\Projects\ConsoleApplication2\Debug\Conso...

```
the size of the vectors is:55
the size of matchespoints: 55
the percentce matches is: 1
the matches points distance is: 2.58042
the size of the vectors is:69
the size of matchespoints: 69
the percentce matches is: 1
the matches points distance is: 5.72857
the size of the vectors is:62
the size of matchespoints: 62
the percentce matches is: 1
the matches points distance is: 3.34897
the size of the vectors is:65
the size of matchespoints: 65
the percentce matches is: 1
the matches points distance is: 4.90091
the size of the vectors is:65
the size of matchespoints: 65
the percentce matches is: 1
the matches points distance is: 7.80696
the size of the vectors is:46
the size of matchespoints: 46
the percentce matches is: 1
the matches points distance is: 3.53883
```