**ECE 510: Mathematical Foundations of Machine Learning** **Spring 2018**

# Course Project: Robust Subspace Recovery
Due: June 12, 2018, 5:00PM PT

*Student Names: Scott Jewell, Mrudula Gopal Krishna, Archit Tatwawadi*

## 1 Paper Descriptions

### 1.1 Paper 1

**Paper Title:** Fast, Robust, and Non-convex Subspace Recovery, G. Lerman, T. Maunu [2]
**Student Name:** Scott Jewell

#### 1.1.1 Problem Description & Formulation

Robust subspace recovery (RSR) is a method of separating data will low variances from those with higher variances. This technique has practical uses in denoising, compression, facial recognition and many more. The basic theory involves projecting data from a higher dimension subspace (data with a higher variance) or outlier to a lower dimensional subspace (recovering the lower variance data) or inliers.

Outliers and inliers result from changes over the dimension(s) of change; for instance in a movie, the dimension of change is time. If a pixel changes color often from frame to frame, it is probably an outlier. Because inliers can be expressed with less data than outliers, they can be described using a lower dimension.

Fast Median Subspace (FMS) recovery is a specialized version of RSR which focusing on reducing the computational complexity of applying RSR. The algorithm is non-convex and allows for several control factors to effect the resulting data set; these controls include the desired rank of the output and the robustness of the process.

#### 1.1.2 Algorithm Description

The geometric interpretation of the FMS algorithm involves minimizing the distance between the data vectors and the nearest subspace of user determined dimension. This is achieved by repeatedly calculating the Euclidean distance between the data vector x and L, the d-dimensional subspace determined by PCA. The subspace spanned by this distance become the new L. A standard PCA routine can be used, but using a random PCA scheme can reduce computation time. Once the the user determined conditions are met(robustness, iterations, the result can be inverted from L.

The core of this algorithm involves minimizing the distance between the original data, x, and the desired dimensional subspace, L. This minimization takes the form of an inverse radial problem:

$$y = \frac{x}{max(dist(x, L))^{(2-p)/2}}$$

This should look familiar to any with a familiarity with inverse radial physics problems, such as Newtownian gravity or classic electromagnetics. y then forms the basis for the next iteration of L.Once the new L is calculated the process is repeated until either the desired minimization is achieved or the algorithm times out.

User defined constraints include k iterations, d desired subspace dimension. This dimensionality is poorly defined in literature, although research is ongoing. Instead d is often determined by trial and error or using rules of thumb for the assigned task (i.e. facial recognition often uses d = 9). p robustness power which is related the use of Principal Component Analysis in determining the desired subspaces; setting this to 1 allows us to treat the minimization as a geometric median subspace. Delta is an "abstract global minimizer" and is not well defined, but acts as an alternate minimization to y. Delta in the algorithm is insensitive to it except under special circumstances; previous work by the authors shows that under many conditions delta can be assumed to be zero.

#### 1.1.3 Theoretical Results

The authors determine use six theorems to prove points about the FMS model.

- Local convergence of L results. Under special circumstances it can achieve global convergence, however there is a small probability for convergence to a saddle point.
- Determination of lower dimensional subspaces.
- The algorithm is robust to noisy data, with larger selections of p giving greater robustness.
- The global convergence of one dimensional data.
- Two theorems on the special conditions of convergence.

There seems to be an assumption in all of these that if there is noise present within the data, that it will be relatively small, without specifying how small.

The FMS algorithm was compared with 11 other algorithms. They tested this using both robustness powers of $p = 0.1$ and 1. In general both FMS algorithms beat or were competitive with the others even at higher percentages of higher dimensional data, larger ambient dimensions and small difference in higher and lower dimensions. They also demonstrate that FMS can recover the target data to theoretical limit around one third to one half of the time.

Finally they test the FMS algorithm on real data sets. These include classifying the spectra of galaxies and categorizing them. They also used FMS to cluster data as we have done for other algorithms in class (a classification problem). In the first two cases the authors find that FMS achieves 85 % + accuracy in a small number of iterations ($> 10$). However in the case of facial recognition, FMS converged faster than most of the other algorithms but achieved less accuracy.

#### 1.1.4 Relation to Course Material

The area where this paper most overlaps with the course material is in its use of principal component analysis. However, at least in geometric interpretation I think this algorithm reminds me of gradient descent, in that both involve geometric minimizations; furthermore this strikes me as being similar to the physics of waves travelling within a medium, where they move along the shortest path allowed by their energy. I wonder if there is some want to connect these mathematically. Additionally, the distance maximization routine appears to use regularization exponential parameter similar to those employed in IRLS or ADMM.

Frankly, there is very little I would understand about this paper without the knowledge gained in class. While the idea of of energy minimization is intuitive to someone with a physics background, applying that to subspaces and subspace bases would be well beyond my knowledge and intuition. There are still many things I do not understand about this paper, much of it contained within the extensive theorems. There are many concepts which I am familiar with in the context of linear systems which make much less sense to me in subspaces; what does a stationary point mean in higher dimensional space? Or a "stationary region?" Finally there is a major gap in intuition about the relationship of outliers and noise. Instinctively I would say that outliers will contain noise; defining the dimension of the outliers will act as a limit to how much noise they contain. However the paper seems to treat noise and outliers as two completely separable things.

### 1.2 Paper 2

**Paper Title:** Coherence Pursuit: Fast, Simple, and Robust Principal Component Analysis [3]
**Student Name:** Archit Tatwawadi

### 1.2.1 Problem Description & Formulation

Principal Component Analysis(PCA) a method used when the number of features available are very high and we are required to reduce the dimensions(thus losing few features) due to computational limitations. The features which are not important for a particular type of problem are termed as outliers and the important features are termed as inliers thus the data is termed as a corrupted data. Coherence pursuit(CoP) is an algorithm to implement the Robust Principal Component Analysis(PCA) method. The PCA, a method to reduce the higher dimensional data to lower dimensions, is very sensitive to presence of outliers in the data. This results in completely different high dimensional subspace when we apply the PCA to a corrupted data. Thus, we need to apply some algorithm to separate the outliers from inliers and then apply PCA. Lets assume the data is spread in $\mathbb{R}^m$ subspace and there exists $n_1$ inliers and $n_2$ outliers. The mentioned data then can be modelled as

$$D = [A\ B]T \tag{1}$$

where $D \in \mathbb{R}^{m\times n}$, $A \in \mathbb{R}^{m\times n_1}$, $B \in \mathbb{R}^{m\times n_2}$, T is an arbitrary permutation matrix. The matrix $A$ denotes the inlier data and the matrix $B$ denotes the outlier data. The columns of $A$ lie in an $r$-dimensional subspace $\mathcal{U}$ and columns of $B$ does not completely lie in $\mathcal{U}$.

The authors used a particular term called as mutual coherence to solve this problem. Mutual coherence is the closeness of a particular data point in terms of features with the other data points.The CoP algorithm mentioned in this paper separates the inliers from outliers by calculating the mutual coherences between each data points and gives an orthonormal columns of basis for subspace of the inliers. The methods proposed before the publishing of CoP paper required heavy complex calculations and were iterative in nature. Also most of the algorithms targeted only one model of outlier data and were not robust enough to external additive noise. CoP solved these problems. The CoP involves only a matrix multiplication and is non iterative in nature. It is robust to external additive noise($\varepsilon$) and also supports structured and unstructured outlier data models. The application of this algorithm is in computer vision, any subspace clustering problem, image processing, bioinformatics, etc.

### 1.2.2 Algorithm Description

The CoP algorithm is a very simple way to separate out the inlier data from the corrupted data. This algorithm is non-iterative in nature and involves only a matrix multiplication to compute a gram matrix($G$). The algorithm begins with normalization of data matrix $D$ as given in (1), by its 2-norm value and stored in variable $X \in \mathbb{R}^{m\times n}$. Then, we calculate the mutual coherences between different data points by taking the gram matrix of normalized data points($X$) and store it in a variable $G \in \mathbb{R}^{n\times n}$. The logic behind calculating the coherence is that the inlier data will have more coherence with other inlier data points rather than with outlier data, which authors prove in the theoretical results. Once the mutual coherences are calculated, the columns with more mutual coherences are separated out in another matrix which is of the rank $r$. This step is called the subspace identification step which can be calculated in multiple ways. One way to identify the subspace is to sample the columns of coherence matrix and stop when the rank of the output matrix is $r$. Another way to identify the subspace can be applied when we know the percent of the outliers($n_2$%) present in the data.In this method, $n_2$% of columns from the Coherence matrix with lower coherence value are removed from the data and the remaining columns are that of inliers. One more way to calculate this is by projecting the data matrix onto a random $kr$ - dimensional subspace($X_\phi \in \mathbb{R}^{kr\times m}$) to reduce the computational complexity for some $k > 1$. After projection of the data, the rank of inliers $A$ still remains $r$ whereas the columns of outliers $B$ do not lie in $\mathcal{R}(A)$. Take $r$ iterations and update a matrix $F$ which has orthonormal basis of columns of $X_\phi$ for maximum values of vector $p$ in every iteration. The columns of projected matrix $X_\phi$ are updated every iteration as

$$X_\phi = X_\phi - FF^T X_\phi \tag{2}$$

A matrix $Y$ is constructed at the end of all the iterations by combining the columns of $X$ which corresponds to columns of $F$. The columns of $Y$ are a basis for inlier data subspace.

The basic intuition for CoP is that the outlier data will have more coherence with other outlier data and in most cases will form a cluster in higher dimensional subspace whereas the inlier data will form a cluster in the lower dimensional subspace($r$) and would have very less coherence with other outlier data. This separates the outliers from the inliers. This algorithm uses a PCA method for dimensionality reduction and for identification of the subspace in which the inlier data lies. This algorithm performs well for both structured outlier model(few linearly dependent columns) and unstructured outlier data model(linearly dependent columns or form clusters). Also the subspace recovery is guaranteed with this algorithm even when the data is corrupted with high amount of unstructured outliers and when the $\frac{n_1}{n_2}$ which is inlier to outlier data ratio approaches zero provided that $\frac{n_1}{n_2}\frac{m}{r}$ is enough large. The overall complexity of the algorithm with the third way used for subspace identification is $\mathcal{O}(mn^2 + r^3n)$

### 1.2.3 Theoretical Results

The authors assumed that the inliers are drawn from intersection of subspace $\mathcal{U}$ which is a random $r$ dimensional subspace in $\mathbb{R}^m$ and $l_2$-unit norm sphere in $\mathbb{R}^m$.The columns of outliers are drawn at random from $l_2$-norm sphere of $\mathbb{R}^m$. The permutation matrix($T$) is just assumed to be identity matrix for simplicity of understanding. Using expectation operator the authors were able to show that the inliers have larger coherence values as compared to outliers when the $m \gg r$.With these assumptions, the authors showed that even when the $\frac{n_1}{n_2} = 0.01$, the CoP algorithm was able to clearly separate the inliers and outliers. The authors proved that CoP was very robust to external additive noise. The authors were able to separate the inliers from the outliers even when the signal to noise ratio is very low(around 0.5 or less). They proved this by using a parameter called $\tau$ which is a ratio of expectation operator of noise to expectation operator of signal. Since CoP gives a global view of the data, it works good even with structured outlier columns. Authors put forth few algorithms for subspace identification which works even if the data is clustered. The overall complexity of this algorithm is $\mathcal{O}(mn^2 + r^3n)$.

### 1.2.4 Relation to Course Material

In this paper, the authors used few of the topics which were covered in the class. The main topic being the normalization and mutual coherence matrix. The data was 2-normalized and then gram matrix of the normalized data was used to create a mutual coherence matrix. This coherence matrix played the main role for subspace clustering. As we learned in the k-means spectral clustering demo, the data was divided into different subspaces on the basis of distance, whereas in this paper the decision was based on the mutual coherence of the data.The algorithm also used p-norm of coherence matrix to feed it to subspace identification algorithm. The PCA was used in the robust sense to identify the subspace of inlier and for dimensionality reduction of the given data matrix just like we did in the multidimensional scaling topic where the input was a distance matrix whereas in CoP it was a coherence matrix. For subspace identification, the data was projected on a $kr$-dimensional subspace as we learned in the class. I would not have understood this paper on a mathematical level as I understand it now if I had not taken the ECE 510 course. This course helped me to understand and look at research papers from a mathematical perspective. The concepts like PCA and norm which were the basic foundation of this paper were understood only because of ECE 510 course. Few of the mathematical concepts which were not taught in the class revolve around the Expectation operator and its use to prove the use of coherence matrix for separation of inliers and outliers. This concept of expectation operator was beyond the scope of ECE 510.

## 1.3 Paper 3

**Paper Title:** RANSAC Algorithms for Subspace Recovery and Subspace Clustering. [1]
**Student Name:** Mrudula Gopalkrishna

### 1.3.1 Problem Description & Formulation

A number of applications require estimation of model parameters from data. A general strategy to go about doing this is to use least squares regression but most times, the observed data is contaminated with noise or contains points that do not conform to the model that is being estimated. Least squares estimation being very sensitive to outliers can greatly skew the result. The Random Sample Consensus Algorithm(RANSAC) is an iterative method used to estimate reasonable functions to fit the data containing outliers.
This algorithm is of importance because sometimes the outliers in the data are due to errors in measurement or noise and need to be rejected. The need for rejection is that In these cases outliers do not contain any useful information of the model. The way RANSAC deals with is, it iterates selecting the smallest subset of data required to define a model and follows a hypothesize and test framework.
This paper is narrowed down to applying RANSAC to solve subspace clustering and subspace recovery problems. RANSAC is well known to be computationally expensive and this paper explores this problem in the context of subspace recovery and subspace clustering by deriving performance and computational complexity of these two problems. They are defined as follows:

*Subspace Recovery*: The problem of subspace recovery is defined under a noiseless setting. The observed data is considered to have $n$ points in dimension $p$ represented as $x_1, x_2...x_n \in \mathbb{R}^p$. $m$ points out of the observed data is assumed to lie on a $d$ dimensional linear subspace $L$. Points that lie on subspace $L$ are considered inliers and the rest are deemed outliers. A $q$ tuple of points from the observed data is assumed to be linearly independent if it includes a minimum of $(d+1)$ points from $L$. Subspace recovery is the act of obtaining $L$. This can also be perceived as a separation of outliers from the inliers.

*Subspace Clustering*: The problem of subspace clustering is also considered under a noiseless setting. It is assumed to have $K$ subspaces in total. The observed data is considered to have n points in dimension $p$ represented as $x_1, x_2...x_n \in \mathbb{R}^p$. $m_k$ points out of the observed data is assumed to lie on a $d_k$ dimensional linear subspace $L_k$ where $k = 1, 2 \ldots K$. In this case, inliers are defined to be the points on one of the K subspaces deeming all other points as outliers. Subspace clustering is hence grouping $m_k$ points with respective $L_k$ for all values of $k$

### 1.3.2 Algorithm Description

RANSAC is a simple iterative algorithm. It follows a hypothesize and test framework. It starts off by picking the minimum number of datapoints required to evaluate a model to fit the data, hypothesizes a model and checks with rest of the datapoints. The datapoints that lie within the error tolerance of the estimated model form the consensus set. These consensus sets are ranked based on the the cardinality of inliers. The algorithm iterates until probability of the discovering a consensus set with a certain number of inliers below a threshold. This algorithm should work as the samples are chosen such that they are both random and the smallest subset of the observed data that is required to estimate a model to fit our data and there is constant verification of an estimated model. RANSAC is essentially an exhaustive search for a mathematical model that best describes the observed data.

This paper describes three versions of the RANSAC algorithm which happen to be the original or canonical form as described by Fischler and Bolles(1961), its adaptation by Hardt-Moitra(2013) and the Chen-Lerman(2006) adaptation which is also referred to as The Spectral Curvature Clustering(SCC) algorithm.

For Robust subspace recovery, since the minimum number of parameters required to describe the d dimensional subspace according to the assumption made is $d+1$, the canonical RANSAC randomly chooses a tuple of size $(d+1)$ and examines against the condition to form a d- dimensional linear subspace. The condition for a subset $S$ of vector space $V$ forms a linear space iff S is closed under vector addition and scalar multiplication. If the condition is met, the subspace is recovered and the algorithm is terminated. Failing the condition causes further iterates until the chosen tuple meets the linear dependence condition. A set of

vectors are said to be linearly dependent iff no linear combination of vectors is zero, except when all the coefficients are zero. Although the canonical RANSAC algorithm gives accurate results, the time taken to reach those results is very long.
Hardt-Moitra adaptation of RANSAC tackles this problem. It is designed for cases where the number of data points $N$ is greater than the dimension $p$ wherein it lies. The algorithm samples tuples of size $p$ and checks for linear dependence. If the condition is met, all the points in that tuple are considered to be inliers and the subspace they lie on is hence recovered. Otherwise, the algorithm iterates until a tuple which is linearly dependent is sampled. This algorithm does not require the the dimension of the underlying subspace in our observed data to be predetermined. Both the above adaptation of RANSAC assumes a noiseless setting. For cases where our input data points are contaminated with noise, Chen and Learmans SCC algorithm is adopted. The basic idea of this algorithm is to remove the outliers before fitting a model. This is done by building a weight matrix that expresses the probability of the chosen points lying on a $d$ dimensional subspace based on an affinity matrix A whose jk entry measures the similarity between points $j$ and $k$. The degrees of the data points, which is the sum of weights in the corresponding row is calculated. The points with a comparatively low degree are considered to be outliers.

The subspace clustering problem can be seen as a case of union of multiple subspaces. Canonical RANSAC is used to solve this problem under the assumption that the all subspaces in the observed data are of equal dimension $d$. RANSAC is applied to the dataset as in case of subspace recovery, once a subspace is recovered, all the inliers in the current data are removed and the algorithm is run over the data again. The iteratives terminate only after all the subspaces are recovered. SImilarly, Hardt-Moitra algorithm is applied to recover one subspace at a time as described for in the subspace recovery problem for cases where the ambient dimension p is greater than the difference between total number of data points and sum of points lying on K sumber of linear subspaces present in the observed data. As the dimensions of the observed data increases, the performance of the above two algorithm worsens. For subspace clustering problems involving high-dimensional data, Clen- Lermans SCC is used. The algorithm is run for one subspace at a time as described for the subspace recovery problem and iterated until all the $K$ subspaces are recovered.

Addressing the subspace recovery problem,the paper reports results from numerical experiments comparing RANSAC, Hardt-Moitra(HM) and Geometric Mean Subspace(GMS).
The set of parameters varied for each experiment are the dimension of the subspace $d$, ambient dimension $p$, number of inliers $m$ and number of outliers $m_0$. The average time taken by each algorithm were recorded over 1000 iterations. It was found that as the dimensionality of the problem increases, RANSACs complexity becomes quickly impractical.

For subspace clustering, numerical experiments comparing RANSAC,Sparse Subspace Clustering(SSC) and Thresholding-based Subspace Clustering(TSC) were studied.
In addition to the parameters used for subspace recover another parameter number of subspaces was varied over 500 iterations. The algorithms were compared with respect to the average system time and rand index. Rand index is a measure of accuracy of data clustering. Since RANSAC is exact, its rand index is 1 at all times. It was observed that RANSAC was competitive with other methods in cases where the intrinsic dimensionality was not too large and there were not too many underlying subspaces.

### 1.3.3 Theoretical Results

The most important assumption to note is that problems that use RANSAC assume that the outliers in the data contain no useful information of the model being estimated. RANSAC also requires a number of parameters to be predetermined namely, the minimum size of sample that can accurately define model parameters,error tolerance threshold,minimum consensus threshold and number of iterations.
In addition to this, it is important to estimate the proportion of inliers in the data set is another aspect to consider to efficiently use RANSAC since the number of iterations depend on it which in turn dictates the time required for the algorithm to converge.

The number of iterations by the Canonical RANSAC is of the order $O(pd^2)$ where p is the dimension where our datapoints lie and $d$ is the dimension of the linear subspaces. Thus, the iterations exponentially increases if the dimension of the subspace is high leading to slower convergence. The number of iterations in case of the Hardt-Moitra(HM) algorithm is given by $O(p^3)$. Here, time taken by the algorithm to converge depends on the dimension of the data and not the underlying linear subspace. It is also shown that HM algorithm has a constant number of iterations no matter the dimension of the data as long as it satisfies the condition $m/n > d/p$ and $n > p$.
RANSAC being a randomized algorithm, does not guarantee to find the optimal model parameters to fit the observed data with respect to inliers. But this can be controlled by tweaking the parameters like error tolerance and minimum consensus threshold.

#### 1.3.4 Relation to Course Material

A range of topics that are described in this paper was used in class. Some of them being, the concept of linear independence, Definition of subspace and the concept of linear subspace. Rank of a matrix was emphasized while describing the theoretical results of the three adaptations of the RANSAC algorithm. Least squares came up as a reference to the general technique to estimate mathematical models to describe observed data. Understanding the limitations of least squares was crucial to understanding why RANSAC is necessary in the first place. As a student who was never used to see data in terms of linear algebra, this paper would probably have taken be ten times more the effort it took to decipher and understand the essence of RANSAC. The concept of sparse regression, IRLS and ADMM are still pretty vague to me. This might be because seeing things from a math perspective somehow still overwhelms me. I was introduced to machine learning through online classes and conferences which almost always dismiss the linear algebra in them so I was mostly used to not seeing machine learning problems this way. Having said that, I am glad to have taken this class. My eyes do not glaze over math formulae at machine learning classes anymore and it is rewarding to actually be able to understand the nitty gritty details of machine learning algorithms and go beyond just using them out of a predefined library.

## 2 Comparison of Algorithms

Both RANSAC and FMS are fairly easy to interpret geometrically, being based on the minimization of distance between data vectors and desired subspace. Coherence pursuit, on the other hand, tries to match coherence between data columns. CoP is the simplest in implement, relying only on Gram matrix operations. FMS is somewhat simple, needing mostly geometric distance calculations but requires a complicated randomized PCA algorithm to achieve best computational performance.

FMS has multiple theoretical guarantees to convergence, robustness and speed, including complete recovery of single dimension data. This is offset by the dependence on tuning factors which are poorly defined and may need to be set by trial and error and robustness for only a "small" amount of noise. RANSAC is robust to outliers when compared to it commonly used techniques for parameter estimation like least squares,M-estimators and least median of squares. It works particularly well for small dimensional and a low percentage of outliers. Coherence pursuit guarantees robustness to large percentage of noise and outliers and low computational cost (only one matrix operation).

Both FMS and Coherence pursuit tout the speed and inexpensive computation as their benefits. RANSAC although exact is explicitly stated to be slower than competing algorithms.

FMS is able to outperform many algorithms put against it when processing a very large dataset; however in facial recognition, while it runs faster than many other algorithms, it produces less accurate results.

The authors compared the CoP algorithm[3] with robust-PCA algorithms including FMS[2], GMS, R1-PCA, OP and SPCA algorithms. The CoP gave the least error compared to above mentioned algorithms. It gave a stable performance even when the $\frac{n_2}{n_1}$ was varied while performance by other algorithms varied a lot. The same algorithms were also tested for clustering error vs iteration number. The CoP(using the third way for subspace identification) gave the least clustering error(less than 0.05) among the other algorithms.

The authors tested CoP and FMS algorithms on MSRA Salient Object Database and it performed slightly better than FMS algorithm. The authors also tested the algorithm on waving tree video file and was able to identify the outlier frames in the video. For Hopkins155 data set where the inlier lie in multiple subspaces due to different motions, the CoP was able to outperform the above mentioned algorithms. The average clustering error(ACE) for CoP was between 0.1-0.01 whereas for FMS the ACE was between 0.22-0.18. From this we can conclude that the CoP was able to guarantee the subspace clustering in extreme conditions. As CoP clusters subspace based on the mutual coherences of data points, the algorithm gives a global look at the data.

## 3 Algorithm Implementation & Testing

We chose to implement coherence pursuit. In order to determine which algorithm, we each tried to implement our individual algorithms and see which produced the best results. Coherence pursuit was completed before RANSAC and matched its authors results better than FMS.

### 3.1 Results on Synthetic Data

In this section, we have included a plot of performance on synthetic data. The plot is of subspace clustering error vs number of outliers in the data. We generated synthetic data as shown inWe timed the run of synthetic test and the results are:

- Time required for authors function(Coherence_pursuit) : 3.941s
- Time required for our function(CoP) : 4.581s

The authors error vs outlier plot is attached below:

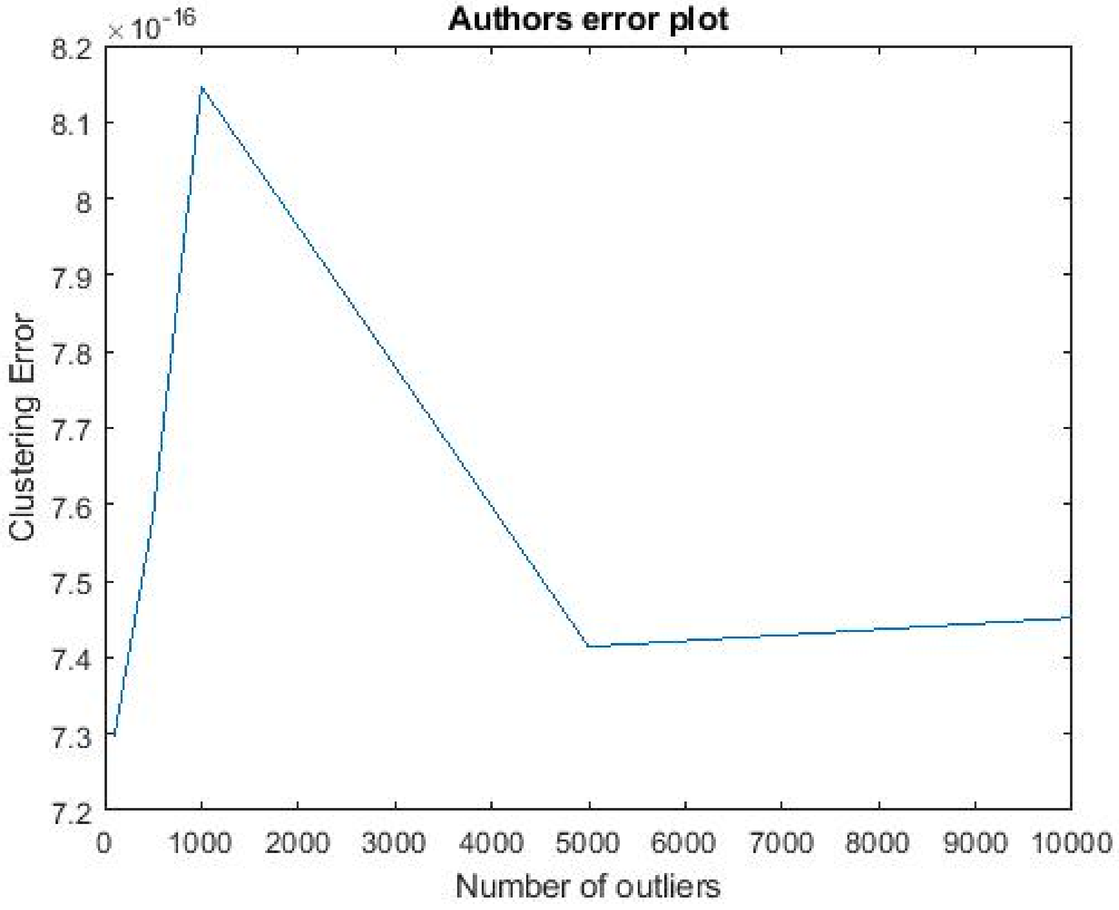

Figure 1: Clustering Error vs Number of Outliers for Coherence_pursuit function.

The error vs outlier plot for our code is attached below:

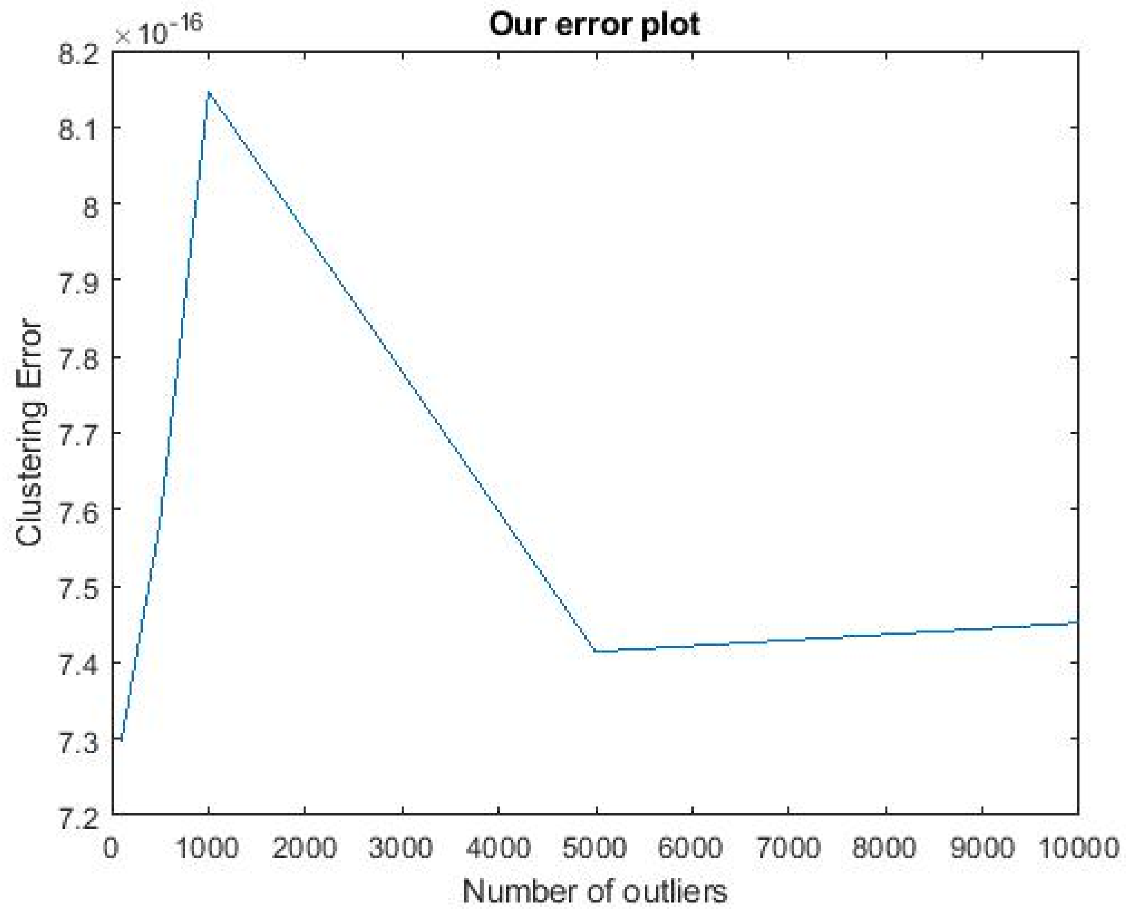

Figure 2: Clustering Error vs Number of Outliers for CoP function.

As we can see from above plots, the code that we implemented works on par with the authors code. It lags behind in terms of computational time.

## 3.2 Results on Benchmark Data

In this section, we have included a plot of separated data on waving trees datasetThe algorithm was able to identify the outliers perfectly when the 2-norm was used. We timed the run of benchmark dataset and the results are:

- Time required for authors function(Coherence_pursuit) : 12.886084s
- Time required for our function(CoP) : 13.052168s

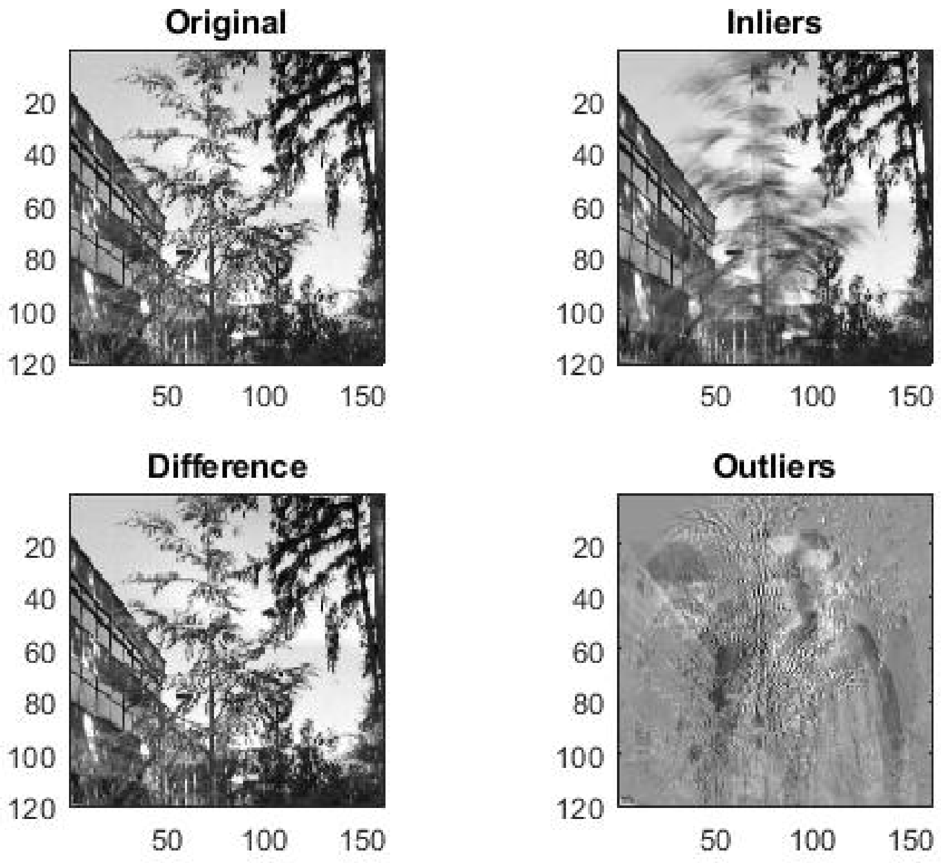

Figure 3: Plot for benchmark testing using authors code with rank equal to the number of images.

The plot of data along with inlier and outliers for our code is attached below:

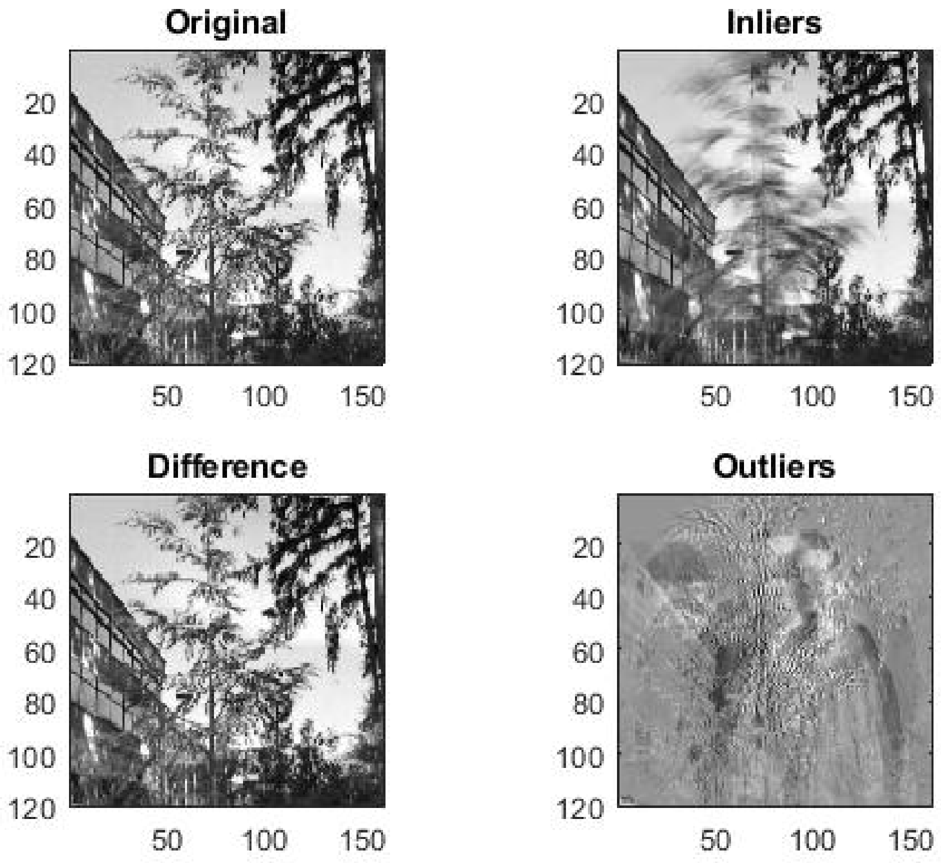

Figure 4: Plot for benchmark testing using our code with rank equal to the number of images.

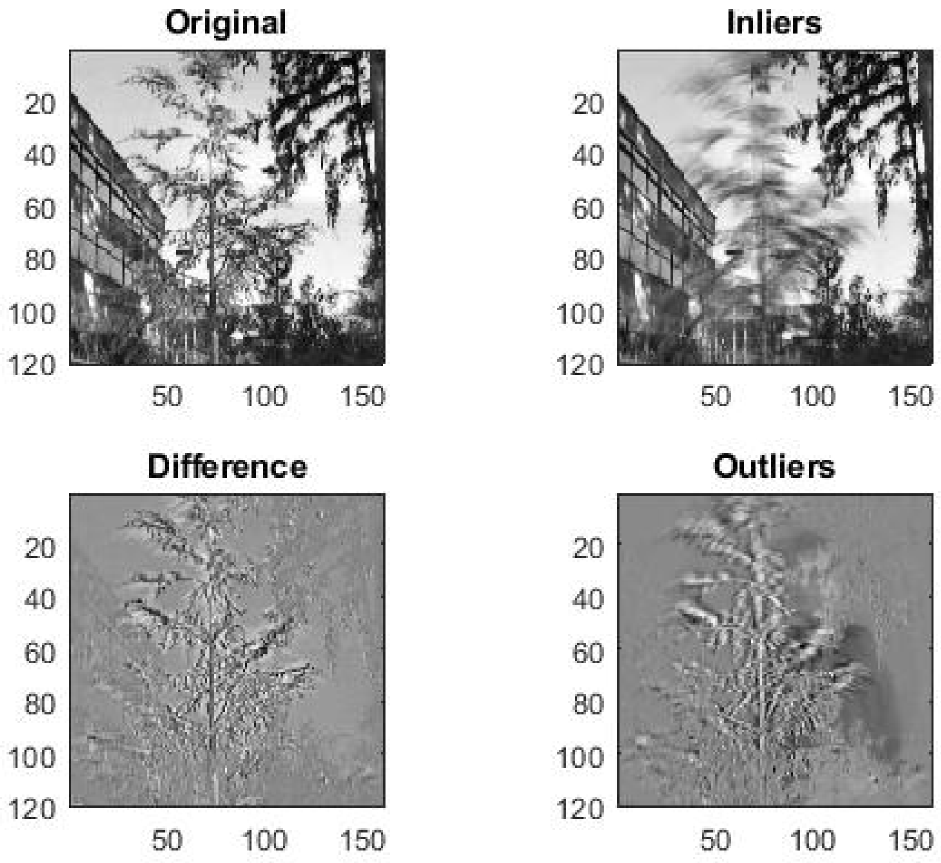

Figure 5: Plot for benchmark testing using authors code rank = 3.

The plot of data along with inlier and outliers for our code is attached below:

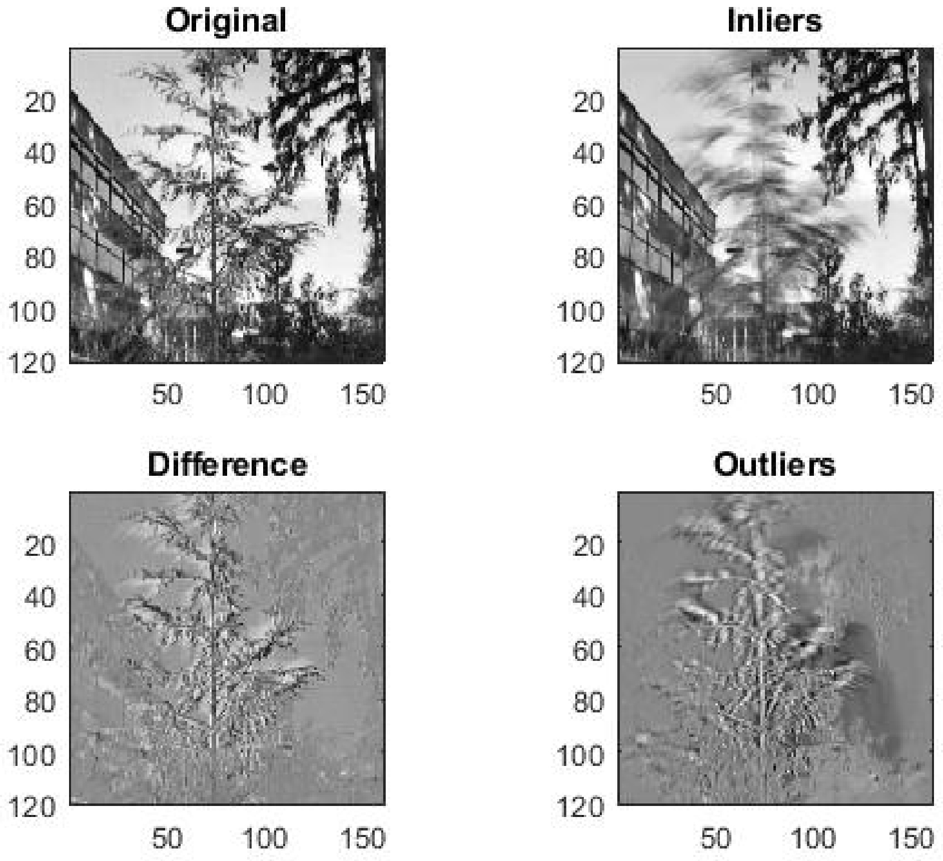

Figure 6: Plot for benchmark testing using our code at rank = 3.

We ran both our version of coherence pursuit and the authors on a set of images, set in time like a movie. The images included a tree shaking in the breeze and a man walking past. We varied the p parameter and found that p = 3 seemed to provide the best results (fig 5-6). As we can see, the code separates out the background inliers and the movement of the tree and person in the outliers. For benchmark testing we set $m = 2, r = 3, n =$ number of images.

This dataset is available at: https://www.microsoft.com/en-us/research/project/test-images-for-wallflower-paper/

## References

[1] E. Arias-Castro, J. Wang "RANSAC Algoritms for Subspace Recovery and Subspace Clustering" math ST 30 Nov 2017 Available online: `arXiv:1711.11220v1`

[2] G. Lerman, T. Maunu "Fast, Robust, and Non-Convex Subspace Recovery" *Elsevier* 9 Jun, 2016 Available online: `arXiv:1406.6145v2`

[3] M. Rahmani, G. K. Atia "Coherence Pursuit: Fast, Simple, and Robust Principal Component Analysis in *IEEE Transactions on Signal Processing* Vol. 65, No. 23, December 1, 2017

## A Appendix

### A.1 Coherence Pursuit Matlab code

```
function U = CoP(D, r,n,m)
%D = data matrix
%r = dimension of desired matrix
%n = size of recovered subspace
%m = order of norm

r = fix(r);
n = fix(n);

X = D./sum(D.^2).^0.5 ;
[N1,~] = size(X);

G = X'*X ;
G = G - diag(diag(G)) ;
p = sum(abs(G).^m,1);

[~,p1] = sort(p,'descend');
y = zeros(N1,n);

% for i=1:N1
%     y(:,i) = X(:,p1(i));
% end
y = X(:,p1(1:n));

[Ur,~,~] = svd(y,'econ');

U = Ur(: , 1:r);
```

### A.2 Synthetic Data Code

```
clc ; close all ; clear all ;

N1 = 200 ;  % The dimension of ambient space
n1 = 100 ;  % The number of inliers
n2 = 10000;  % The number of outliers
r = 5 ;     % The rank of low rank matrix
U = randn(N1,r) ;
iterations = [100 500 1000 5000 10000];

for iter=1:length(iterations)
    A = U*randn(r,n1) ; U = orth(U);
    B = randn(N1,iterations(iter)) ;
    D= [A  B] ;    % Given data

    n = 10*3 ;     % Number of data points sampled by CoP algorithm
                   % to form the recovered subspace

    Uh = Coherence_pursuit(D , n, r) ;
    U_our = CoP(D,r,n,2);
    V = Uh - U*U'*Uh;
    V_our = U_our - U*U'*U_our;
    recovery_error1(iter) = norm(V(:),2)/norm(U(:),2);  % Recovery error
    recovery_error2(iter) = norm(V_our(:),2)/norm(U(:),2);  % Recovery error
end
plot(iterations,recovery_error1);
title('Authors error plot');
xlabel('Number of outliers');
ylabel('Clustering Error');
figure;
plot(iterations,recovery_error2);
title('Our error plot');
xlabel('Number of outliers');
ylabel('Clustering Error');
```

## A.3 Benchmark Code

```
clear

r = 2;
m = 2;

olddir = cd;
[path] = uigetdir;
cd(path)
list = dir('*.bmp');

N = length(list)-1;
U1 = zeros(19200,N);
U2 = zeros(19200,N);
U3 = zeros(19200,N);

for n = 1:N;

    A = imread(list(n).name);
%     image(A)
%     drawnow
    U1(:,n) = reshape(double(A(:,:,1)),19200,1);
    U2(:,n) = reshape(double(A(:,:,2)),19200,1);
    U3(:,n) = reshape(double(A(:,:,3)),19200,1);

end

cd(olddir)
[a,b,c] = size(A);

% Utrain = CoP(U1(:,1:t),t,t,m);
% V = Utrain - U1*U1'*Utrain;
% ut = reshape(V,a,b,t)/norm(V);
% U11 = reshape(U1(:,1:t),a,b,t);
tic;
```

```
Utrain = CoP(U1,3,N,m);
V = Utrain - U1*U1'*Utrain;
ut = reshape(V,a,b,3)/norm(V);
U11 = reshape(U1,a,b,N);
toc;
for n = 1:N-1;

    %Utrain = CoP(U1,r,2,n);
    %V = Utrain - U1*U1'*Utrain;
    %ut = reshape(V,a,b,m)/norm(V);
    %U11 = reshape(U1,a,b,N);
    subplot(2,2,1)
    imagesc(U11(:,:,n))
    colormap(gray)
    title('Original')
    pbaspect([1 1 1])
    subplot(2,2,2)
    imagesc(ut(:,:,1))
    title('Inliers')
    colormap(gray)
    pbaspect([1 1 1])
    subplot(2,2,3)
    imagesc(U11(:,:,n) - ut(:,:,1)*norm(U11(:,:,n)))
    title('Difference')
    colormap(gray)
    pbaspect([1 1 1])
    subplot(2,2,4)
    imagesc(ut(:,:,3))
    title('Outliers')
    colormap(gray)
    pbaspect([1 1 1])
    drawnow
    pause(.1)
end
```