Sparse Subspace Clustering Experiment I with Images from Elephants Dream

August 17, 2017

Introduction:

In this experiment, I implemented a sparse subspace clustering algorithm and performed experiments with it. This algorithm can be found in the paper *"Sparse Subspace Clustering: Algorithm, Theory, and Applications"* by Rene Vidal and Ehsan Elhamifar. I have obtained movie frames from an open-source movie called Elephants Dream. Then, I ran these images in my program to see how each one of these images can be expressed as a linear combination of the others in a sparse way. In this way, the program was able to identify which images are similar and actually coming from the same parts of the movie. The images that you will see in the upcoming pages represent symmetric coefficient matrices. The more yellow a pixel is, the larger the coefficient in that entry is. Therefore, when you look at row #N or column #N, the most yellow entries on that row or column demonstrate the images(vectors) that can be used to write vector #N as a linear combination.

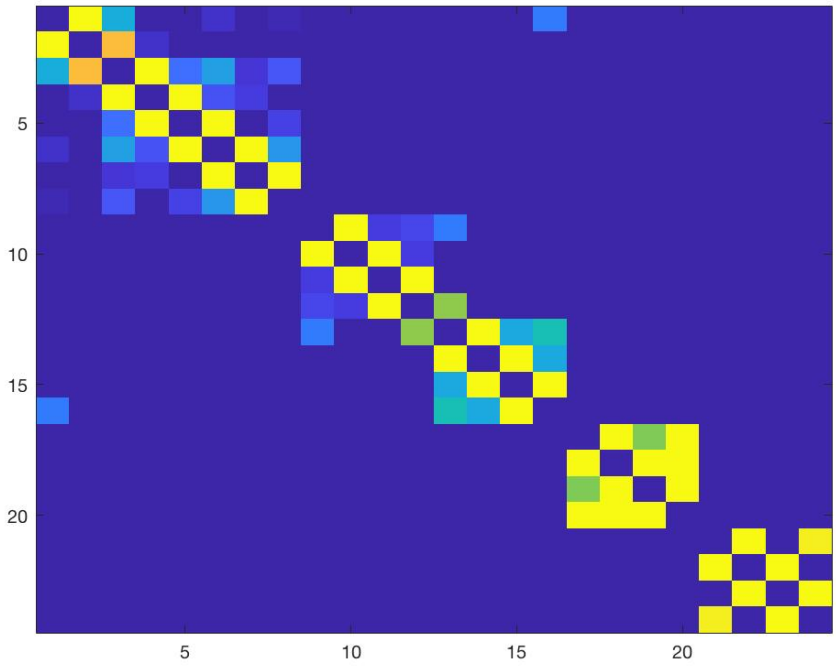
Experiment 1: (24 images, 3 clusters)

Cluster 1: *00143, 00144, 00151, 00156, 00157, 00161, 00165, 00167*

Cluster 2: *00341, 00342, 00343, 00344, 00363, 00374, 00377, 00389*

Cluster 3: *09708, 09711, 09713, 09717, 09725, 09734, 09738, 09743*

Result·



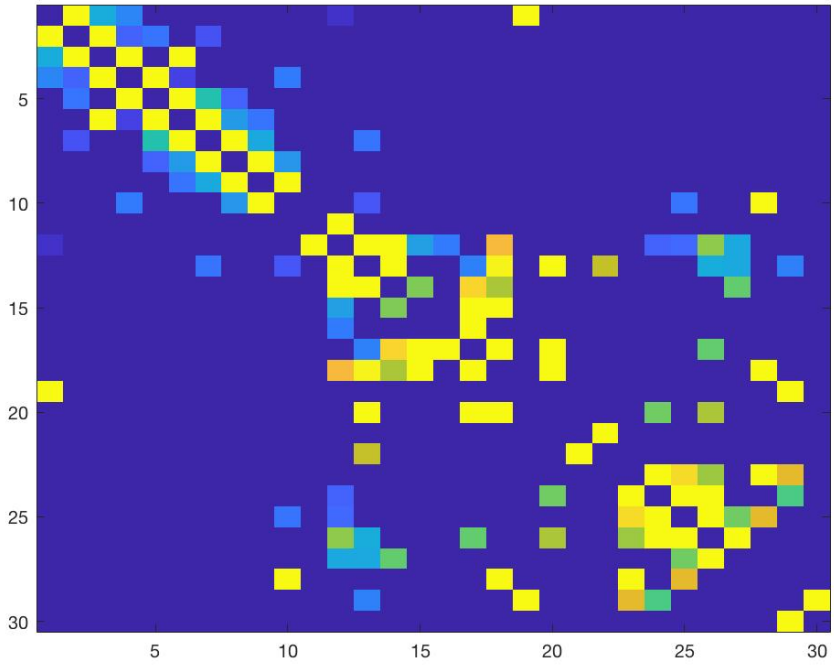
Experiment 2: (30 images, 3 clusters)

Cluster 1: *00900, 00925, 00950, 00975, 01000, 01025, 01050, 01075, 01100, 01125*

Cluster 2: *08000, 08025, 08050, 08075, 08100, 08125, 08150, 08175, 08200, 08225*

Cluster 3: *12000, 12025, 12050, 12075, 12100, 12125, 12150, 12175, 12200, 12225*

Result:



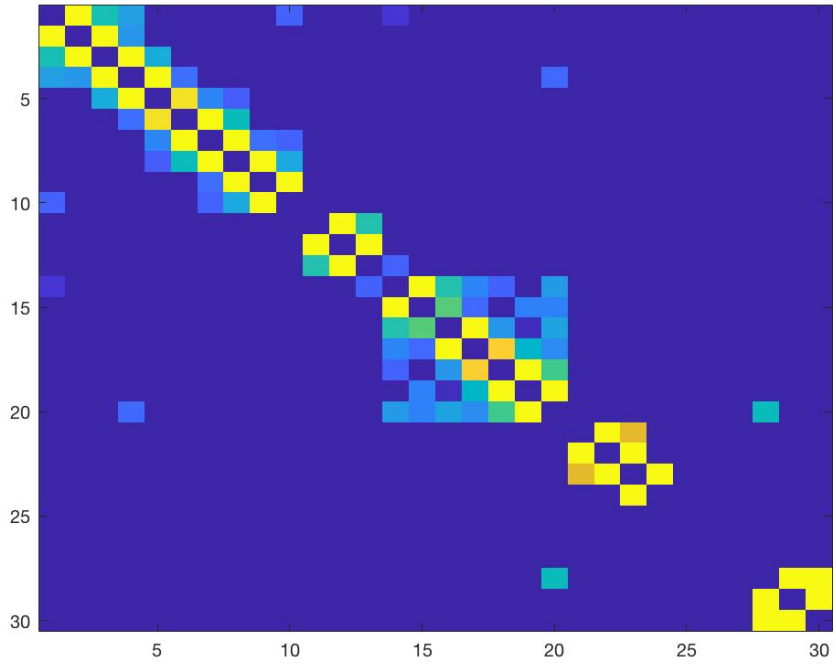
Experiment 3: (30 images, 3 clusters)

Cluster 1: *00300, 00305, 00315, 00330, 00350, 00375, 00380, 00390, 00405, 00425*

Cluster 2: *01500, 01505, 01515, 01530, 01550, 01575, 01580, 01590, 01605, 01625*

Cluster 3: *11000, 11005, 11015, 11030, 11050, 11075, 11080, 11090, 11105, 11125*

Result:



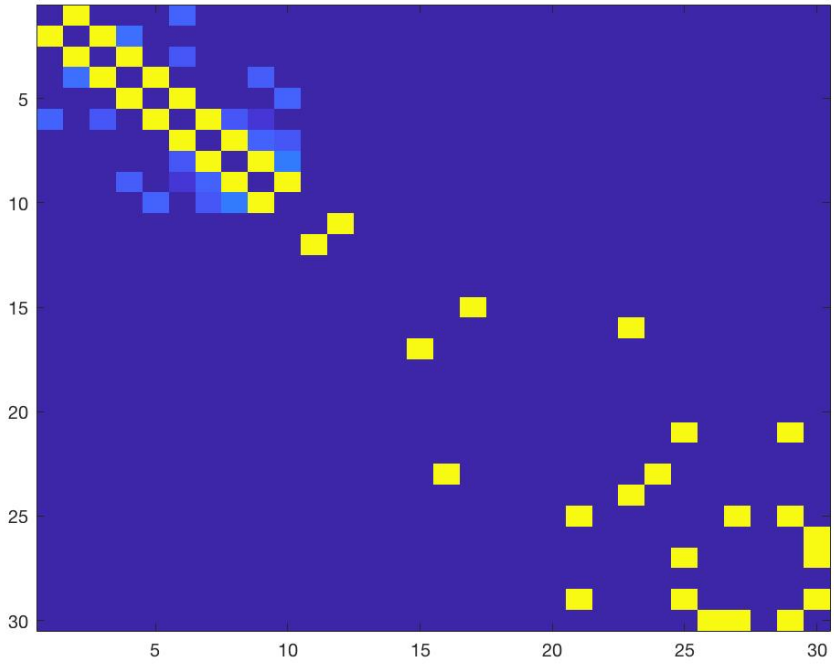
Experiment 4: (30 images, 3 clusters)

Cluster 1: *02000, 02010, 02020, 02030, 02040, 02050, 02060, 02070, 02080, 02090*

Cluster 2: *03000, 03020, 03040, 03060, 03080, 03100, 03120, 03140, 03160, 03180*

Cluster 3: *04000, 04050, 04100, 04150, 04200, 04250, 04300, 04350, 04400, 04450*

Result:



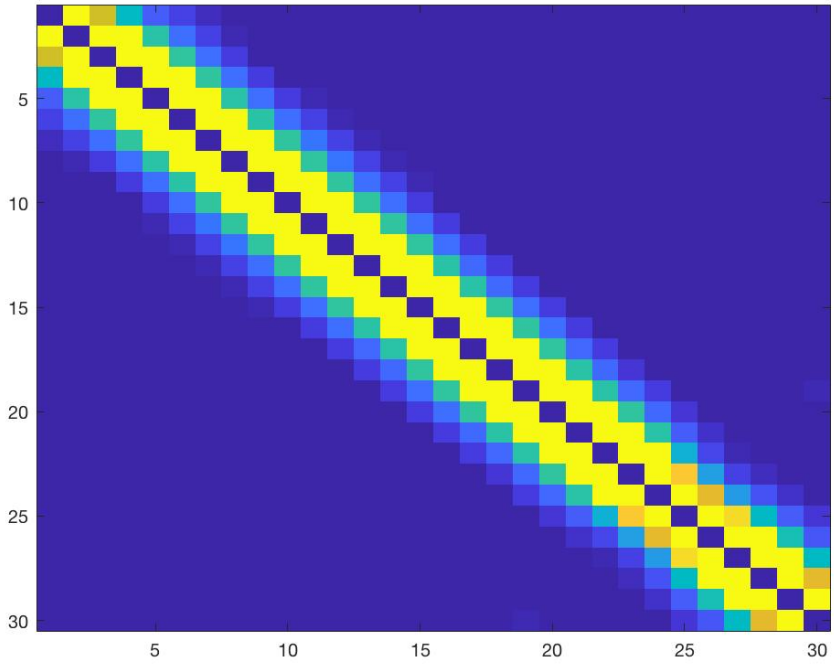
Experiment 5: (30 images, 3 clusters)

Cluster 1: *05000, 05001, 05002, 05003, 05004, 05005, 05006, 05007, 05008, 05009*

Cluster 2: *05010, 05011, 05012, 05013, 05014, 05015, 05016, 05017, 05018, 05019*

Cluster 3: *05020, 05021, 05022, 05023, 05024, 05025, 05026, 05027, 05028, 05029*

Result:



Experiment 6: (21 images, 3 clusters) ~ Fibonacci

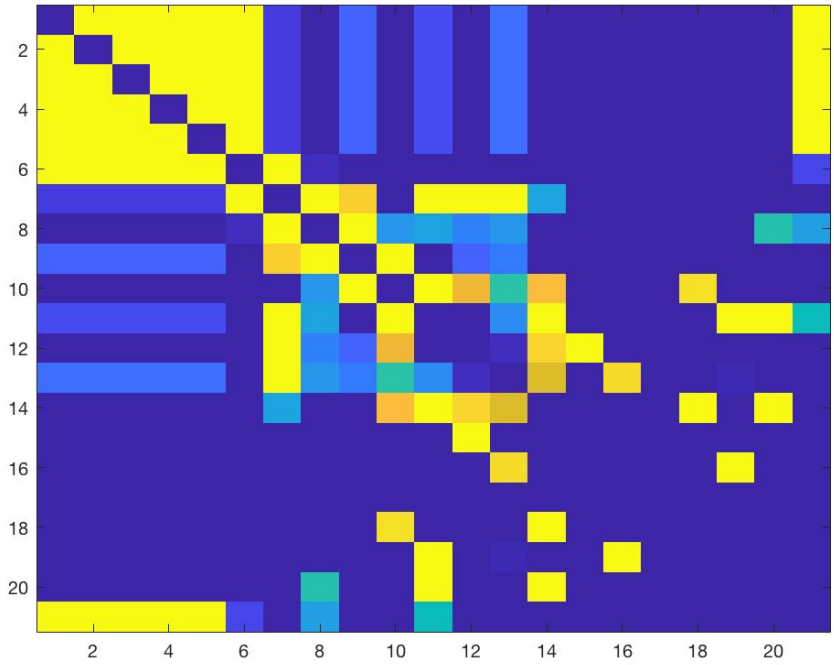
Cluster 1: *00001, 00002, 00003, 00005, 00008, 00013, 00021*

Cluster 2: *00034, 00055, 00089, 00144, 00233, 00377, 00610*

Cluster 3: *00987, 01597, 02584, 04181, 06765, 10946, 15691**

**Not Fibonacci*

Result:



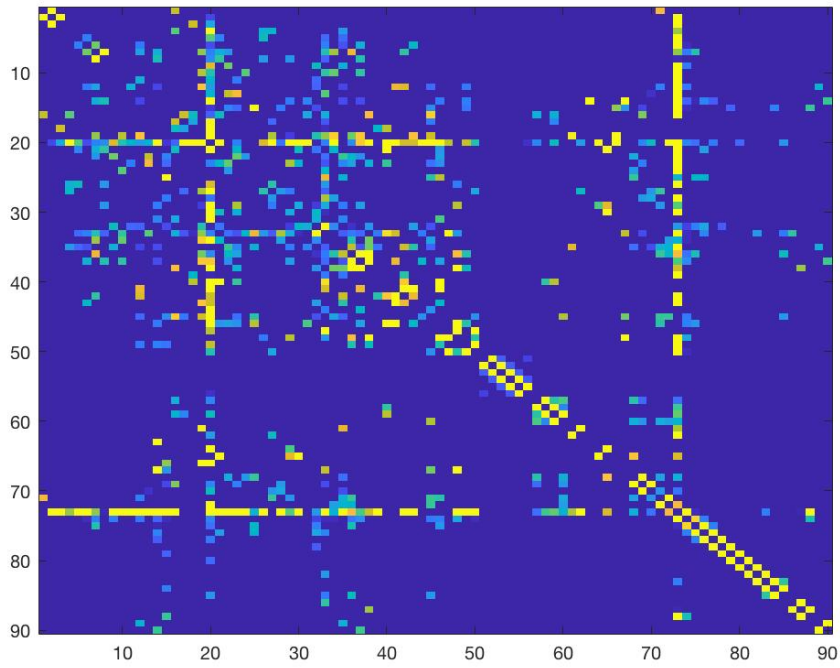
Experiment 7: (90 images, 3 clusters)

Cluster 1: *06000, 06024, 06048, 06072, 06096, 06120, 06144, 06168, 06192, 06216, 06240, 06264, 06288, 06312, 06336, 06360, 06384, 06408, 06432, 06456, 06480, 06504, 06528, 06552, 06576, 06600, 06624, 06648, 06672, 06696*

Cluster 2: *08000, 08024, 08048, 08072, 08096, 08120, 08144, 08168, 08192, 08216, 08240, 08264, 08288, 08312, 08336, 08360, 08384, 08408, 08432, 08456, 08480, 08504, 08528, 08552, 08576, 08600, 08624, 08648, 08672, 08696*

Cluster 3: *10000, 10024, 10048, 10072, 10096, 10120, 10144, 10168, 10192, 10216, 10240, 10264, 10288, 10312, 10336, 10360, 10384, 10408, 10432, 10456, 10480, 10504, 10528, 10552, 10576, 10600, 10624, 10648, 10672, 10696*

Result:



Experiment 8: (100 images, 5 clusters)

Cluster 1: *00005, 00010, 00015, 00020, 00025, 00030, 00035, 00040, 00045, 00050, 00055, 00060, 00065, 00070, 00075, 00080, 00085, 00090, 00095, 00100*

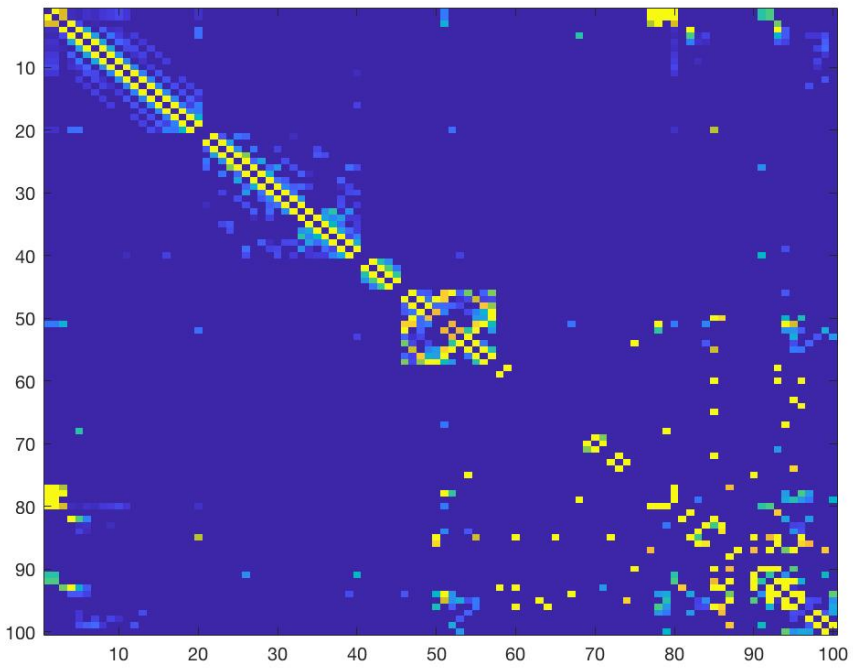
Cluster 2: *01010, 01020, 01030, 01040, 01050, 01060, 01070, 01080, 01090, 01100, 01110, 01120, 01130, 01140, 01150, 01160, 01170, 01180, 01190, 01200*

Cluster 3: *02020, 02040, 02060, 02080, 02100, 02120, 02140, 02160, 02180, 02200, 02220, 02240, 02260, 02280, 02300, 02320, 02340, 02360, 02380, 02400*

Cluster 4: *03030, 03060, 03090, 03120, 03150, 03180, 03210, 03240, 03270, 03300, 03330, 03360, 03390, 03420, 03450, 03480, 03510, 03540, 03570, 03600*

Cluster 5: *04040, 04080, 04120, 04160, 04200, 04240, 04280, 04320, 04360, 04400, 04440, 04480, 04520, 04560, 04600, 04640, 04680, 04720, 04760, 04800*

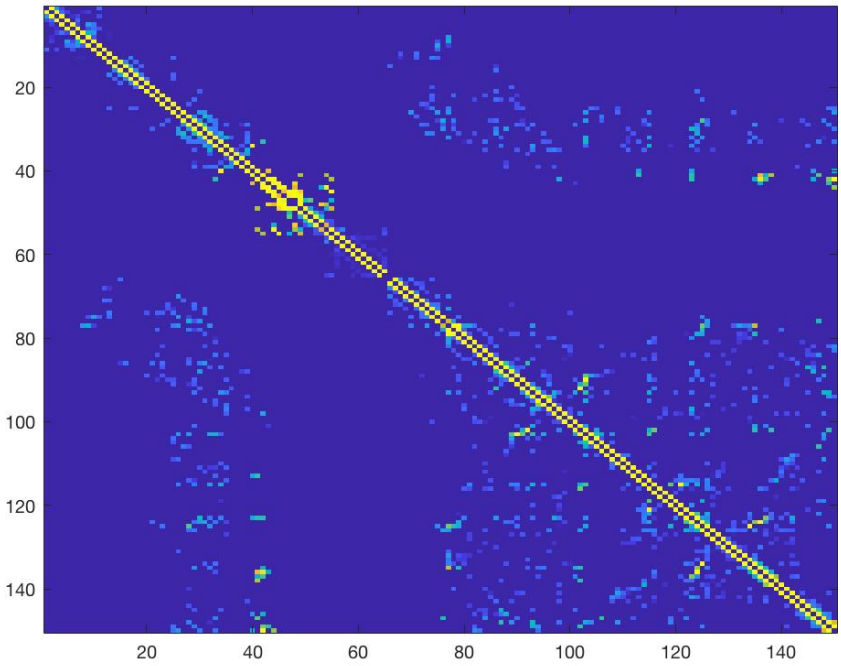
Result:



Experiment 9: (150 images, 1 cluster)

Cluster 1: *06400, 06401, 06402, 06403, 06404, 06405, 06406, 06407, 06408, 06409, 06410, 06411, 06412, 06413, 06414, 06415, 06416, 06417, 06418, 06419, 06420, 06421, 06422, 06423, 06424, 06425, 06426, 06427, 06428, 06429, 06430, 06431, 06432, 06433, 06434, 06435, 06436, 06437, 06438, 06439, 06440, 06441, 06442, 06443, 06444, 06445, 06446, 06447, 06448, 06449, 06450, 06451, 06452, 06453, 06454, 06455, 06456, 06457, 06458, 06459, 06460, 06461, 06462, 06463, 06464, 06465, 06466, 06467, 06468, 06469, 06470, 06471, 06472, 06473, 06474, 06475, 06476, 06477, 06478, 06479, 06480, 06481, 06482, 06483, 06484, 06485, 06486, 06487, 06488, 06489, 06490, 06491, 06492, 06493, 06494, 06495, 06496, 06497, 06498, 06499, 06500, 06501, 06502, 06503, 06504, 06505, 06506, 06507, 06508, 06509, 06510, 06511, 06512, 06513, 06514, 06515, 06516, 06517, 06518, 06519, 06520, 06521, 06522, 06523, 06524, 06525, 06526, 06527, 06528, 06529, 06530, 06531, 06532, 06533, 06534, 06535, 06536, 06537, 06538, 06539, 06540, 06541, 06542, 06543, 06544, 06545, 06546, 06547, 06548, 06549*

Result:



Summary:

It seems to me that as long as the difference between each image in the same cluster is between 20 and 30 frames, the algorithm does a good job of separating the different sets of similar images. I think that this observation is heavily linked to the fact that one second of the movie corresponds to 24 frames, which is in the range given above. In addition, it would make sense to say that the far the consecutive images in a cluster are apart from each other, the less similar they will be to one another.

It is also worth to note that the algorithm clusters the images better when the frames are selected from the specific parts of the movie where the scene does not include a lot of dynamic components and the film does not keep cutting between distinct scenes in a frequent way. As can be seen in Experiment 7, all of the images were selected one second apart from each other. However, the algorithm was able to detect the third cluster better than the first two. This tells us that the images that were in the third cluster were from a somewhat static scene(s) and **did not** have a lot of cuts to other scenes.