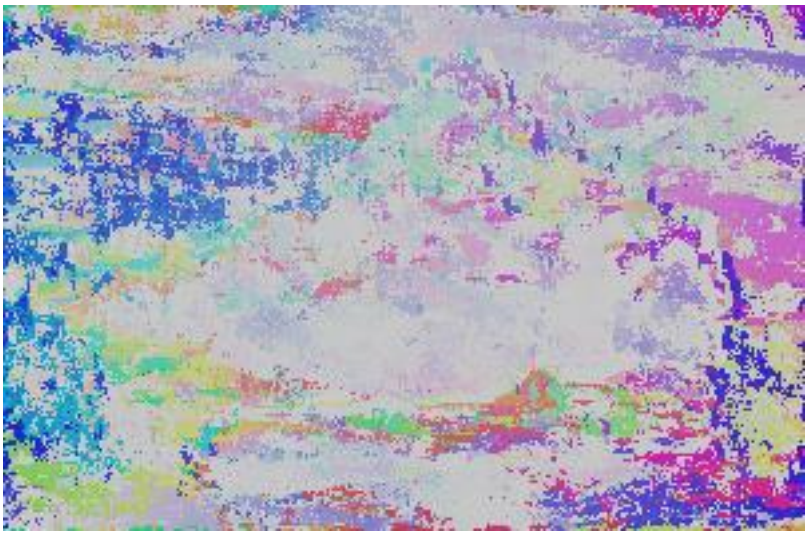
Report
Part 1:
Experiment 1:
Parameters:

```
------------------------------------------------------------------
PatchMatch parameters:
        Initial NNF:            Generated internally
        Iterations:             4
        Patch size:             7
        NLM h:                  25.0
        Alpha:                  0.5
        W:                      300
        K:                      5
        Run NLM algorithm:      True
        Propagation enabled:    True
        Random search enabled:  True
Output path and base filename:  ../results/report/report
Visualization parameters:
        Output files:           correspondences, color nnf, rec'd source
        NNF subsampling:        100
        NNF line width:         0.5
        NNF line color:         r
        Matplotlib server mode: False
        Tmp directory:          ./
------------------------------------------------------------------
```

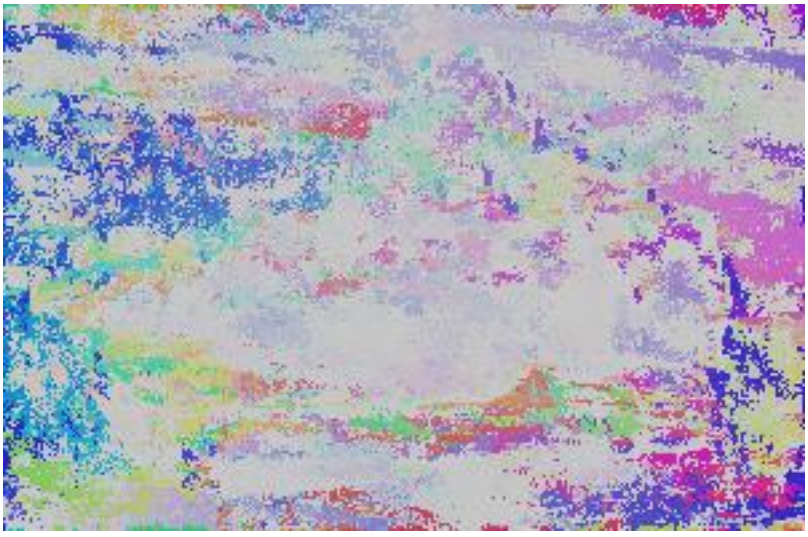
Last Iteration:
NLM :

order=0
NNF:

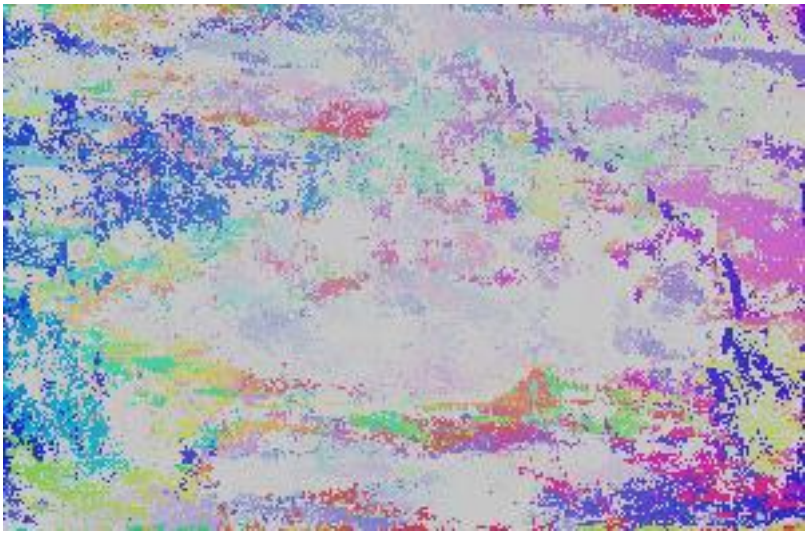
order=1
NNF:



order=3
NNF:



order=4
NNF:



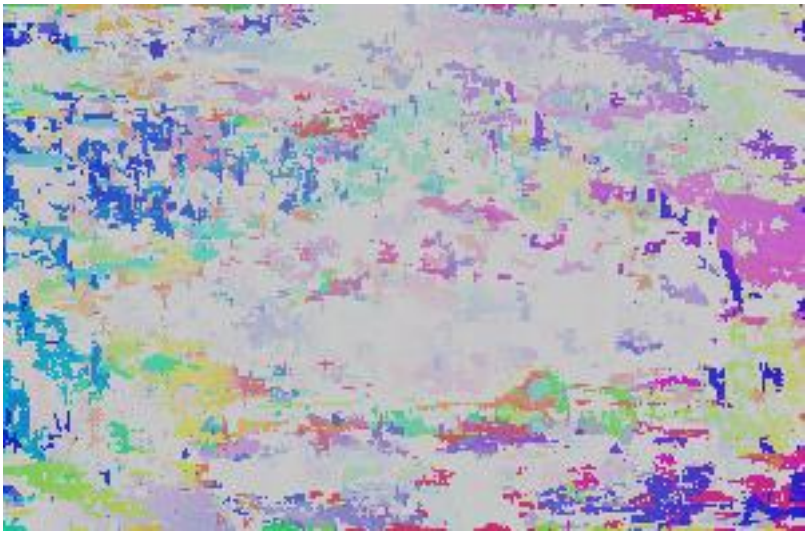
Experiment 2:
Parameters:

```
----------------------------------------------------------------
PatchMatch parameters:
        Initial NNF:            Generated internally
        Iterations:             4
        Patch size:             7
        NLM h:                  25.0
        Alpha:                  0.5
        W:                      300
        K:                      3
        Run NLM algorithm:      True
        Propagation enabled:    True
        Random search enabled:  True
Output path and base filename:  ../results/report1/report1
Visualization parameters:
        Output files:           correspondences, color nnf, rec'd source
        NNF subsampling:        100
        NNF line width:         0.5
        NNF line color:         r
        Matplotlib server mode: False
        Tmp directory:          ./
----------------------------------------------------------------
```

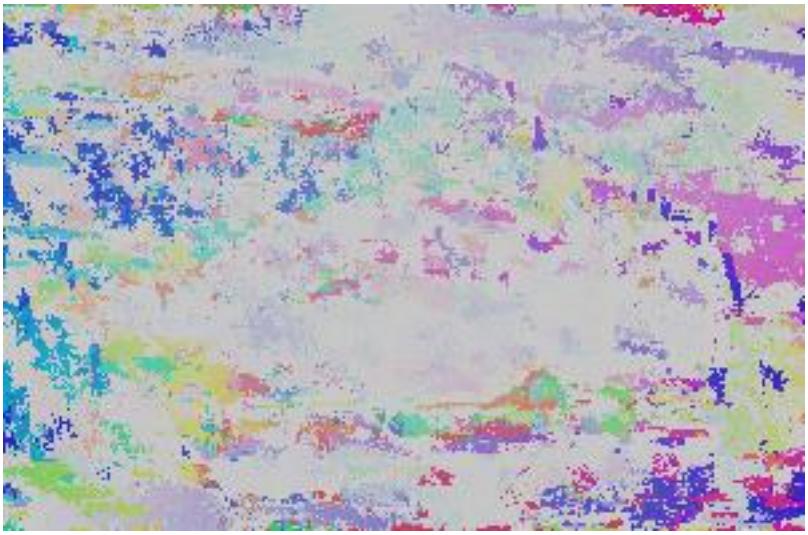
Iteration4:
NLM :

order=0
NNF:

order=1
NNF:



order=2
NNF:



Compare Experiment 1 and Experiment 2. We can see that NLM in 1 is better than 2. Since order k in Experiment 1 is larger than Experiment 2, the proportion for K best NNF images are more balance. When the algorithm reconstructs a NLM image, for a noise pixel, we now have k similar pixels found on the image to reconstruct. For those samples, we use weight to balance each other. Thus, a noise pixel will be reduced by consider more k NNF.

Experiment 3:
Parameters:

```
------------------------------------------------------------------
PatchMatch parameters:
        Initial NNF:            Generated internally
        Iterations:             4
        Patch size:             19
        NLM h:                  25.0
        Alpha:                  0.5
        W:                      300
        K:                      5
        Run NLM algorithm:      True
        Propagation enabled:    True
        Random search enabled:  True
Output path and base filename:  ../results/report3/report3
Visualization parameters:
        Output files:           correspondences, color nnf, rec'd source
        NNF subsampling:        100
        NNF line width:         0.5
        NNF line color:         r
        Matplotlib server mode: False
        Tmp directory:          ./
------------------------------------------------------------------
```

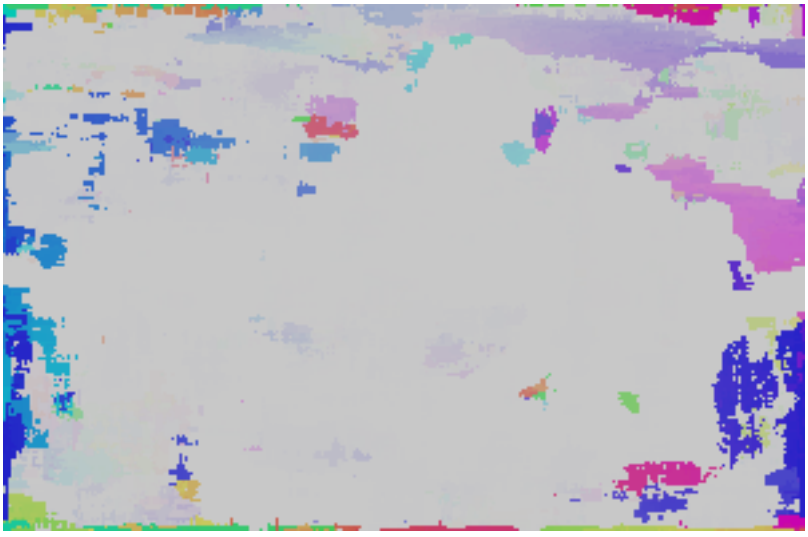
Last Iteration
NLM :

order=0

NNF:

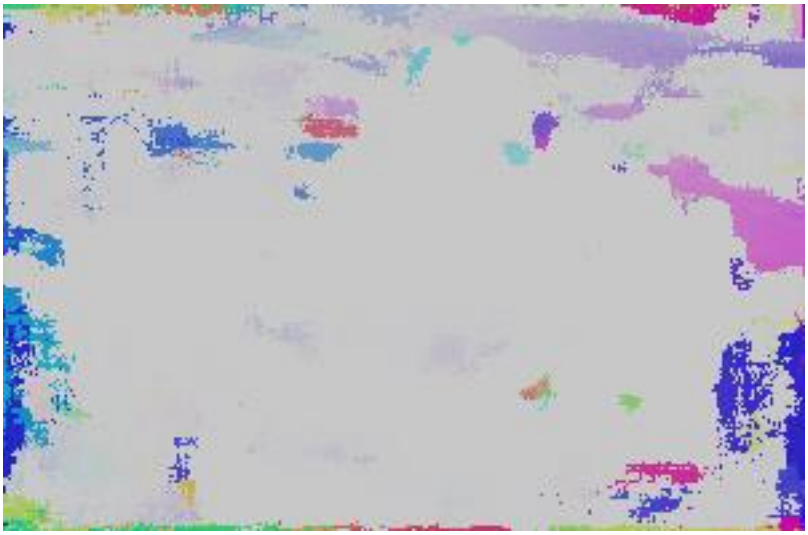
order=1

NNF:



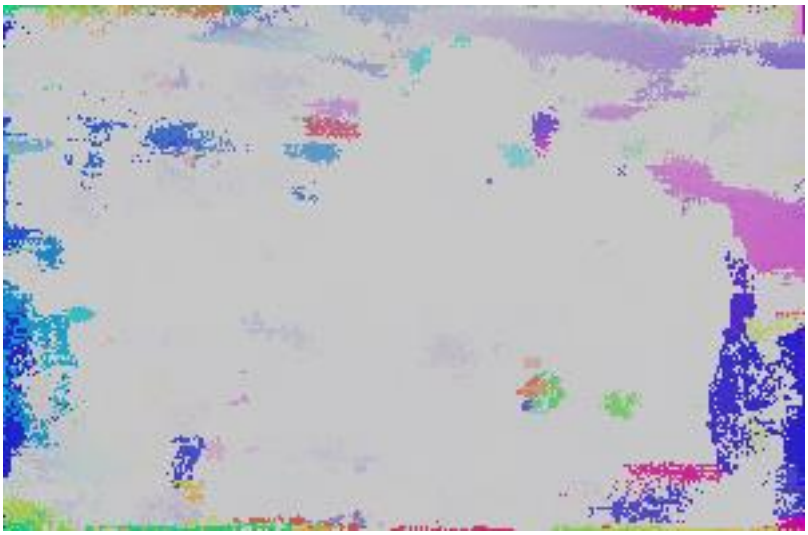
order=2

NNF:



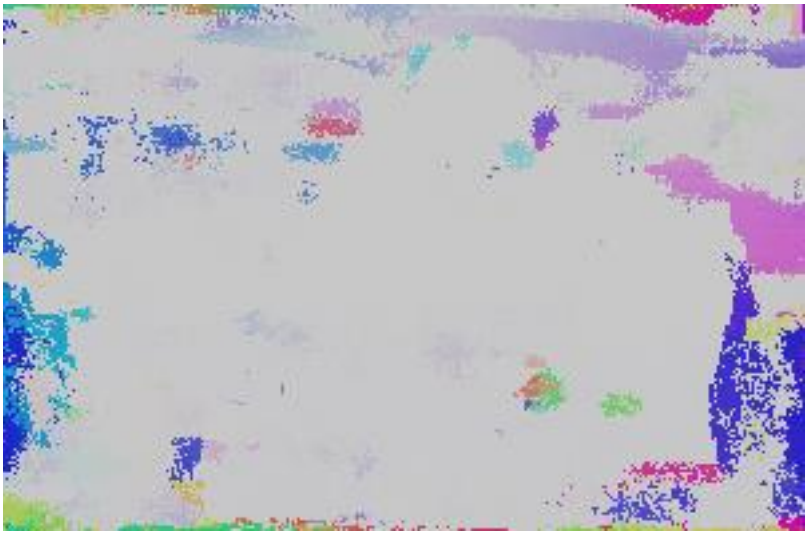
order=3

NNF:



order=4

NNF:



In Experiment 1 and 3, for each iteration, NNFs is unlikely to change when patch size is bigger. Because patch size determines the score of similarity, and if patch size is big, at

propagation, for each pixel, the nearby patches have the same rating, so offset is unlikely to change at every iteration. As a result, the NLM image twists.

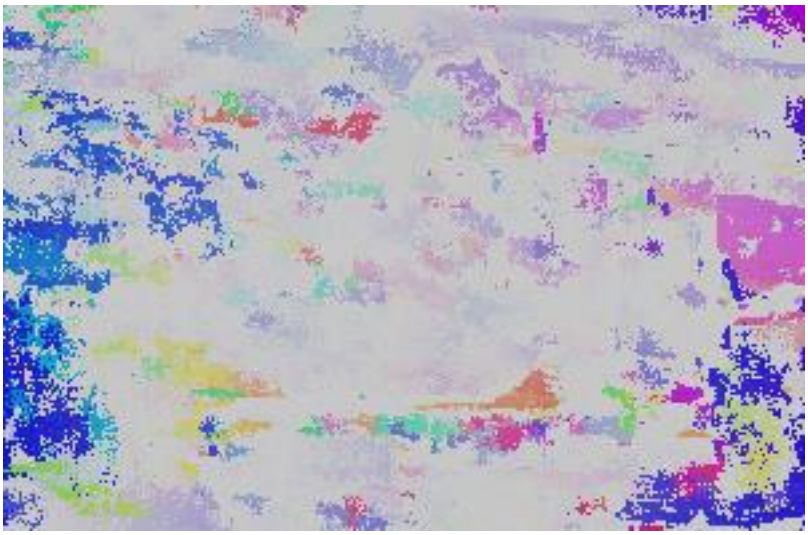
Experiment 4:
Parameters:

```
----------------------------------------------------------------
PatchMatch parameters:
        Initial NNF:            Generated internally
        Iterations:             4
        Patch size:             7
        NLM h:                  25.0
        Alpha:                  0.5
        W:                      15
        K:                      5
        Run NLM algorithm:      True
        Propagation enabled:    True
        Random search enabled:  True
Output path and base filename:  ../results/report2/report2
Visualization parameters:
        Output files:           correspondences, color nnf, rec'd source
        NNF subsampling:        100
        NNF line width:         0.5
        NNF line color:         r
        Matplotlib server mode: False
        Tmp directory:          ./
----------------------------------------------------------------
```

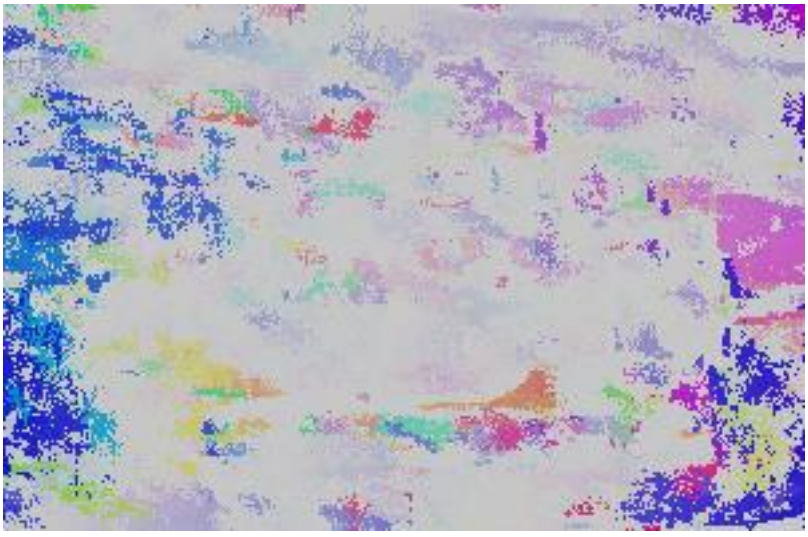
Iteration 4:
NLM :

order=0
NNF:

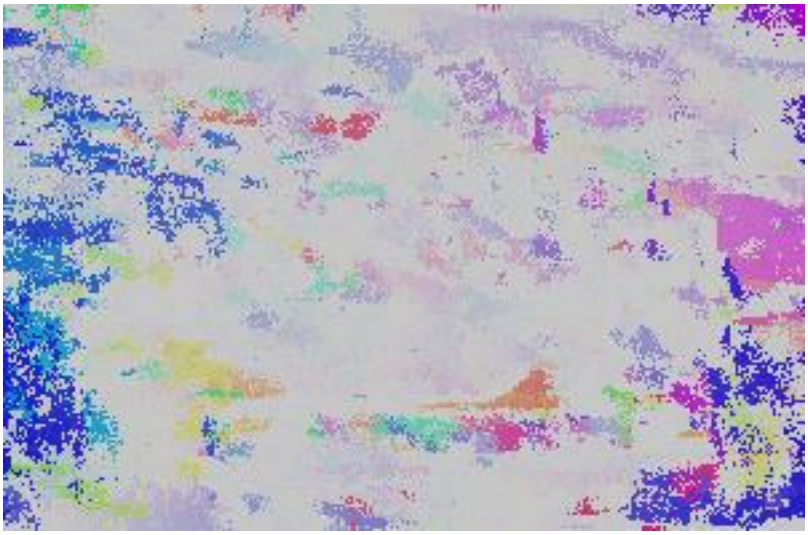
order=2
NNF:



order=3
NNF:



order=4
NNF:



In this Experiment, the random search patch radius is significant smaller than the Experiment 1. When the radius is small, NNF pictures tend to grey, because the random search size is small and the algorithm has fewer sample to check. Thus, many pixels may trap in a local min area, which leads NNF pictures to grey.

Generalized PatchMatch on the jaguar2 image pair for k = 3:
(source.png and target.png)
Iteration 1:
order=0
NNF:

NNF-vector:

Rec-source:

order=1

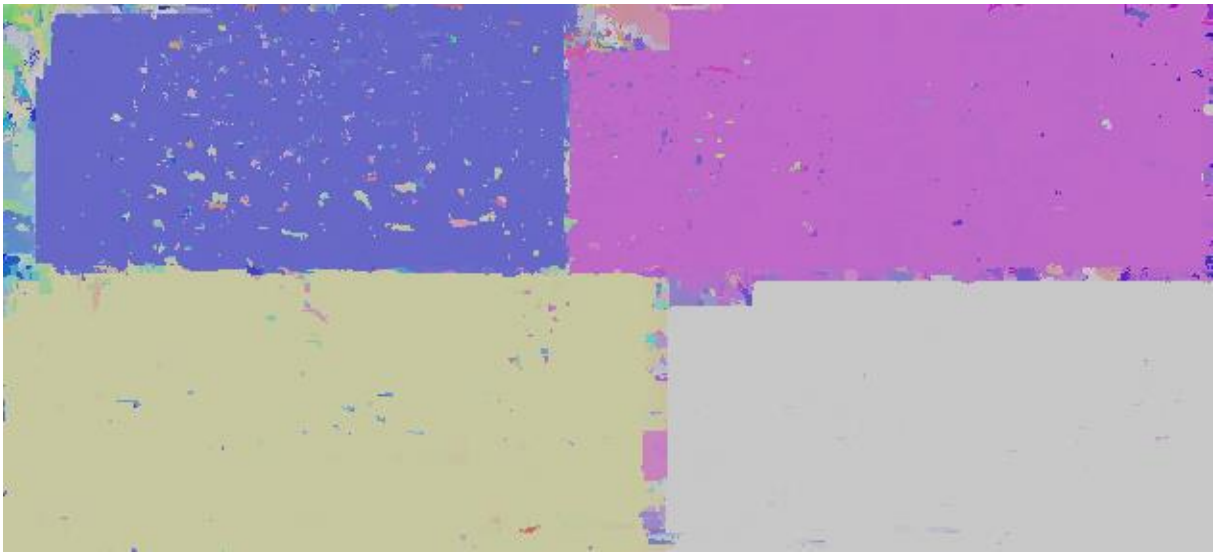
NNF:

NNF-vector:

Rec-source:

order=2

NNF:



NNF-vector:

Rec-source:

Iteration 2:

order=0

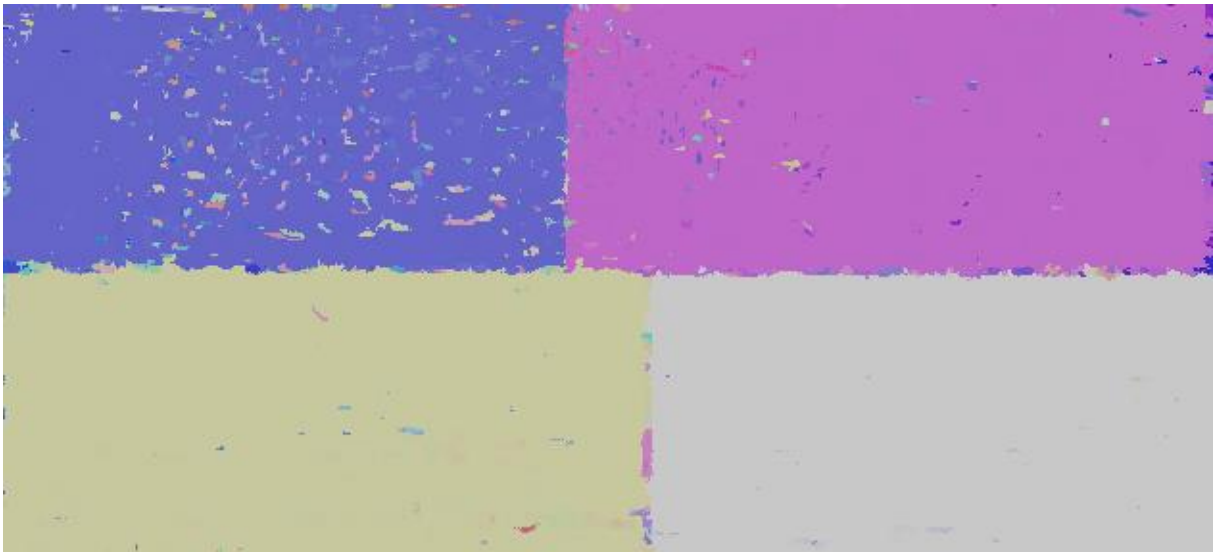
NNF:

NNF-vector:

Rec-source:

order=1

NNF:

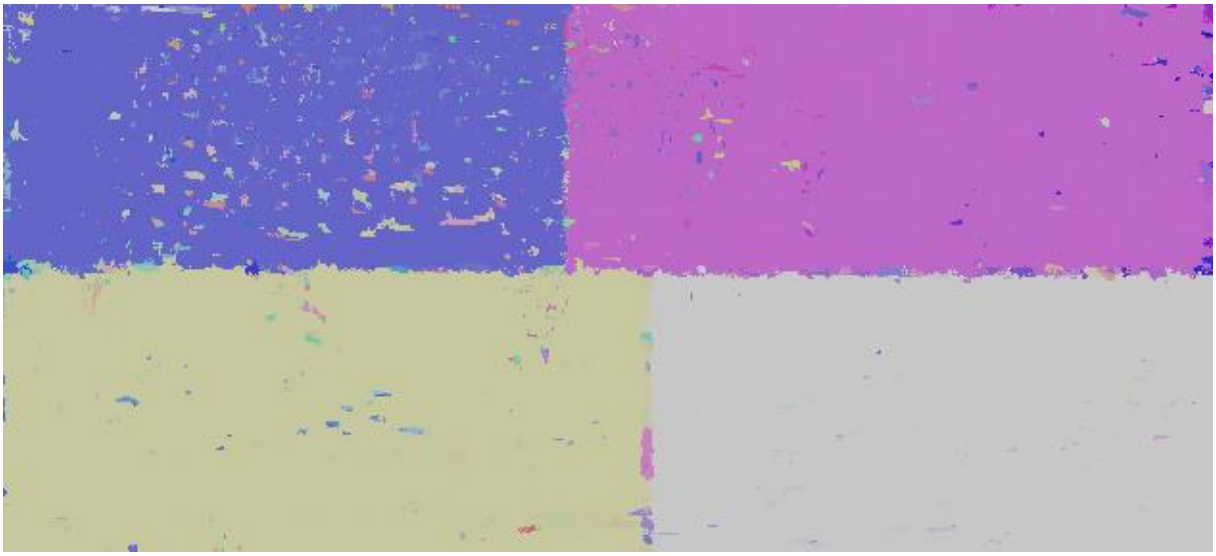


NNF-vector:

Rec-source:

order=2

NNF:



NNF-vector:

Rec-source:

Iteration 3:

order=0

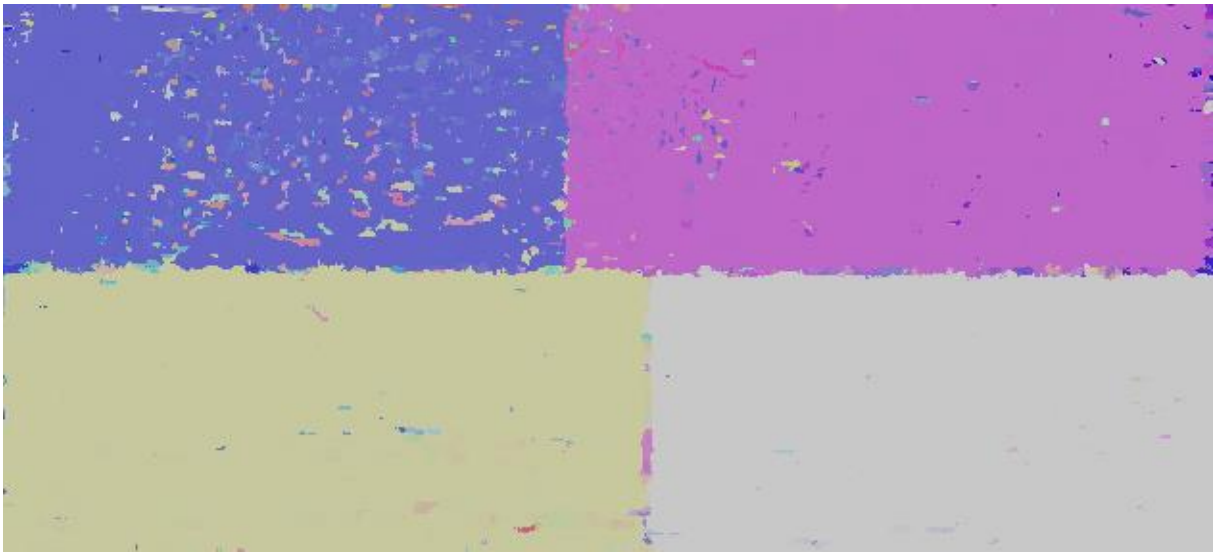
NNF:

NNF-vector:

Rec-source:

order=1

NNF:



NNF-vector:

Rec-source:

order=2

NNF:

NNF-vector:

Rec-source:

Iteration 4:

order=0

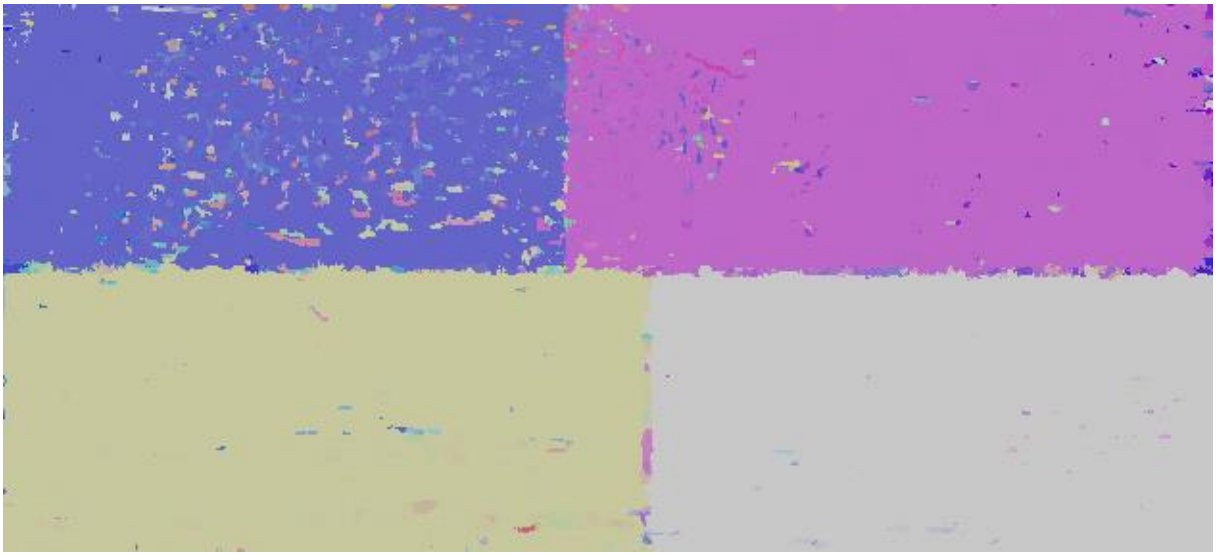
NNF:

NNF-vector:

Rec-source:

order=1

NNF:

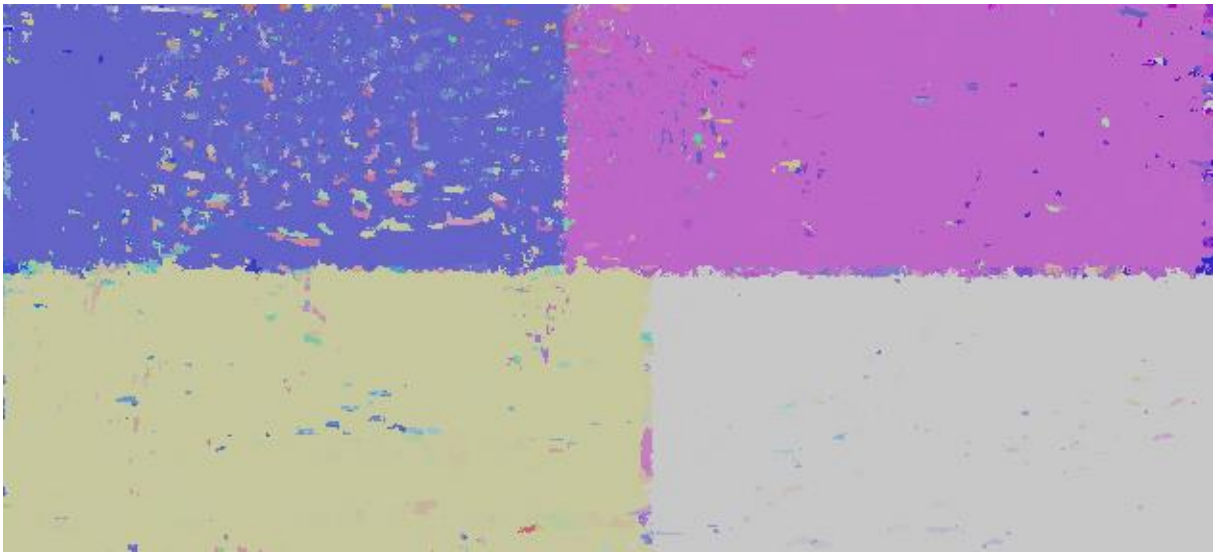


NNF-vector:

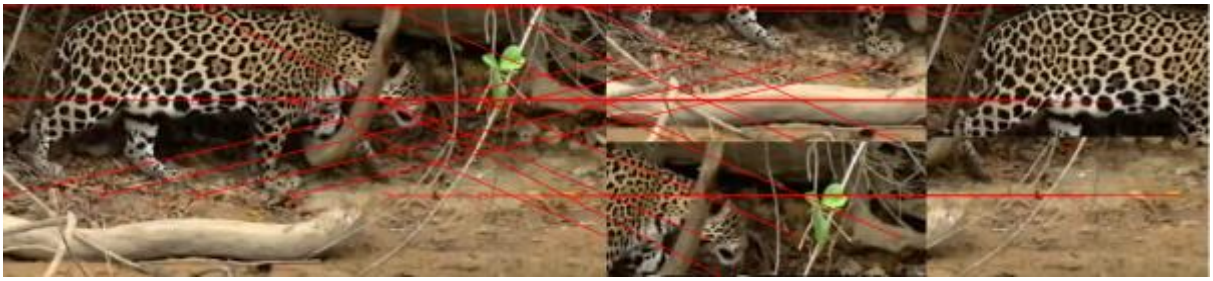
Rec-source:

order=2

NNF:



NNF-vector:



Rec-source: