

TERM PROJECT SIGNALS & SYSTEMS

2015005187 최철훈
2015005241 허재석

FLOW CHART

1. Download & Cut Image
2. HPF
3. 2D DTFS
4. Choose the Values among the Coefficients
5. Normalize Vector
6. K-mean Clustering
7. Adjust Centroid
8. Execute Test

DOWNLOAD & CUT

Google audi a8 grille

All Images Videos News More Settings Tools View saved SafeSearch ▾

GLOBALAUTOSHOP.COM

ITEK

GLOBALAUTOSHOP.COM

GLOBALAUTOSHOP.COM

OEM Audi W12

Made In Germany

DOWNLOAD & CUT

1. K5
2. Starex
3. Genesis G80
4. Avante AD
5. Audi A8

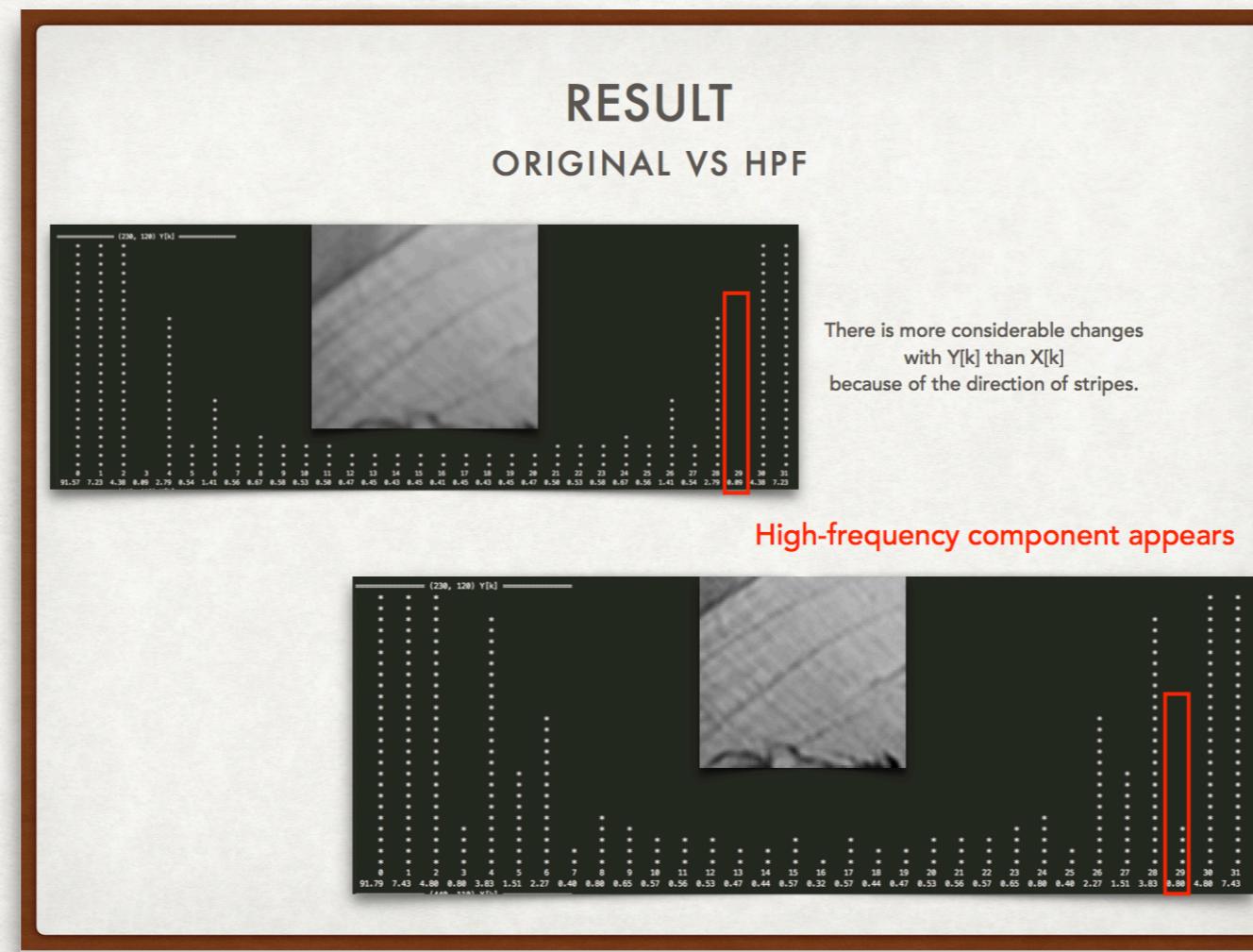
}

x 50 images = 250 images



HPF

- make the boundary of an image clear



2D DTFS

$$c_{k_1, k_2} = \frac{1}{N_1 N_2} \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} \tilde{x}[n_1, n_2] \exp\left(-jk_1 \frac{2\pi}{N_1} n_1 - jk_2 \frac{2\pi}{N_2} n_2\right)$$

$$k_1 = 0..N_1-1, \quad k_2 = 0..N_2-1$$

Due to the **symmetry**, we can do it **half** as much as the original.

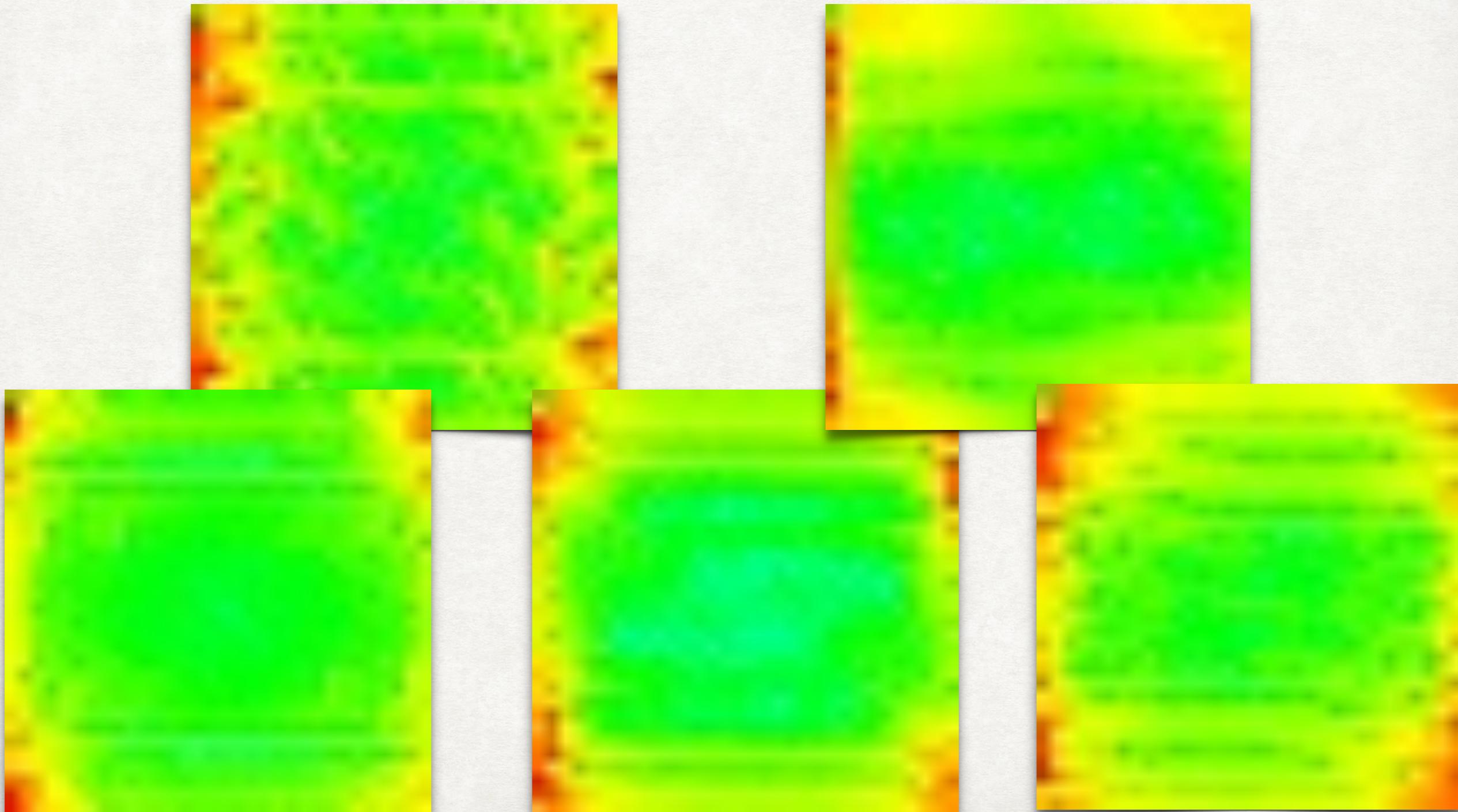
CHOOSE THE VALUES

LINEAR SCALE



CHOOSE THE VALUES

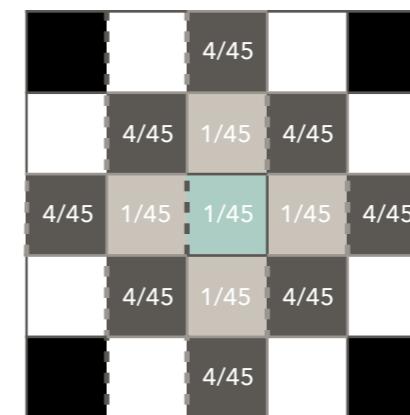
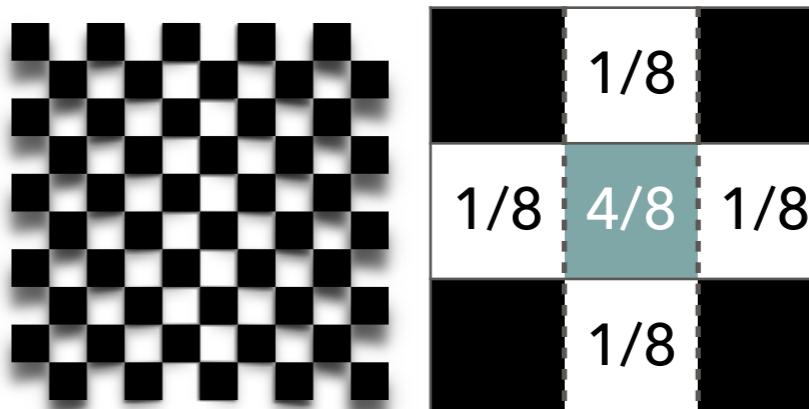
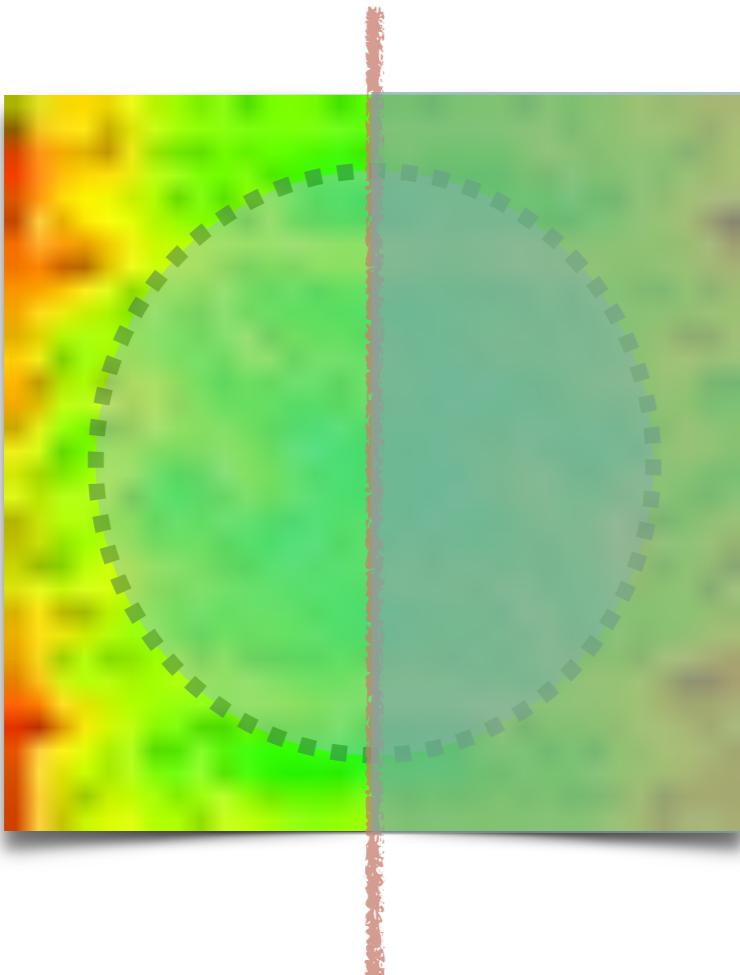
LOGARITHMIC SCALE



CHOOSE THE VALUES

LOGARITHMIC SCALE

- CUT OFF the right half
- Ignore DC component
- Choose 1/2 in the rectangular outside the circle
- Choose 1/8 inside the circle



Finally, about 200 points

NORMALIZE VECTOR

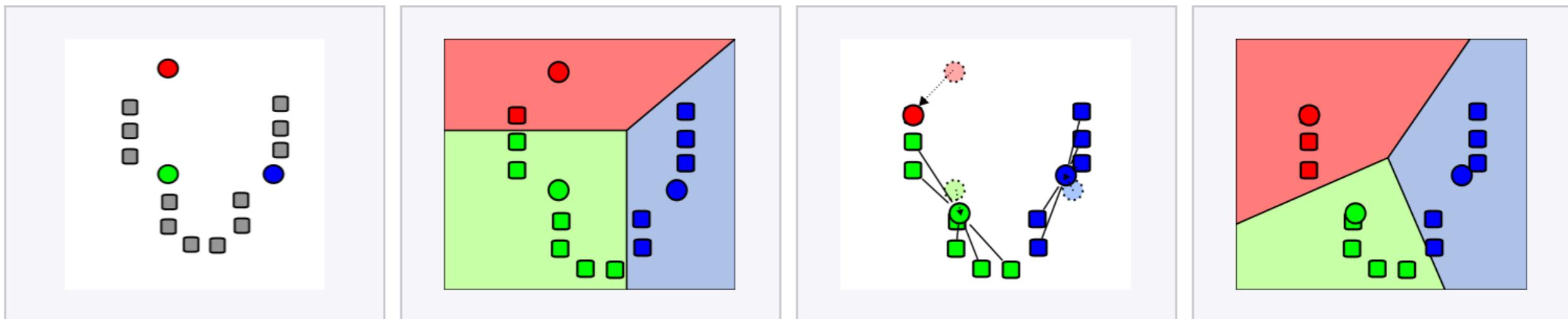
- Make all the vectors have same LENGTH
- To reduce the relative (truncate) error, make the floating-point variable nearby 1.0

$$\begin{aligned} d(\mathbf{p}, \mathbf{q}) = d(\mathbf{q}, \mathbf{p}) &= \sqrt{\underbrace{(q_1 - p_1)^2}_{\text{extremely small}} + \underbrace{(q_2 - p_2)^2}_{\text{extremely small}} + \cdots + \underbrace{(q_n - p_n)^2}_{\text{extremely small}}} \\ &= \sqrt{\underbrace{\sum_{i=1}^n (q_i - p_i)^2}_{\text{extremely small}}}. \end{aligned}$$

- Length = 1.0 * (dimension)

K-MEAN CLUSTERING

Demonstration of the standard algorithm



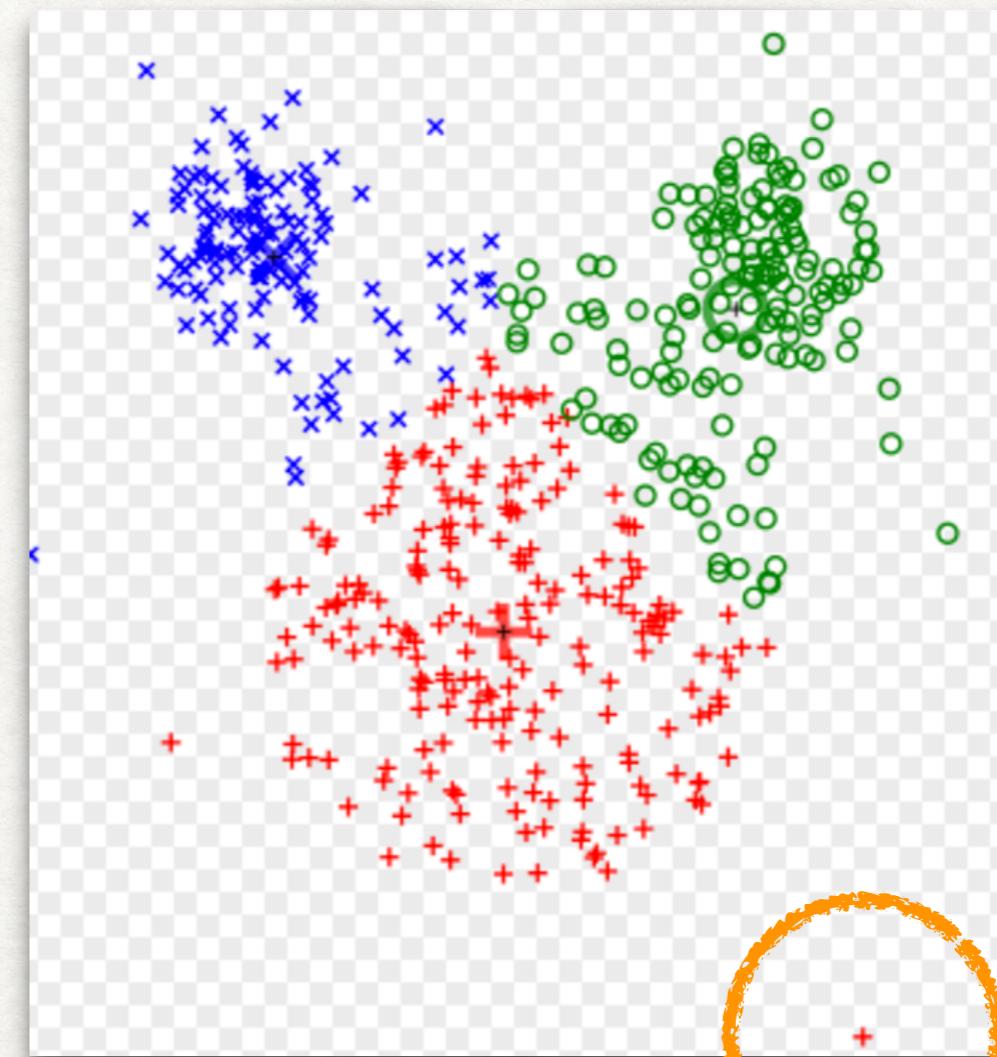
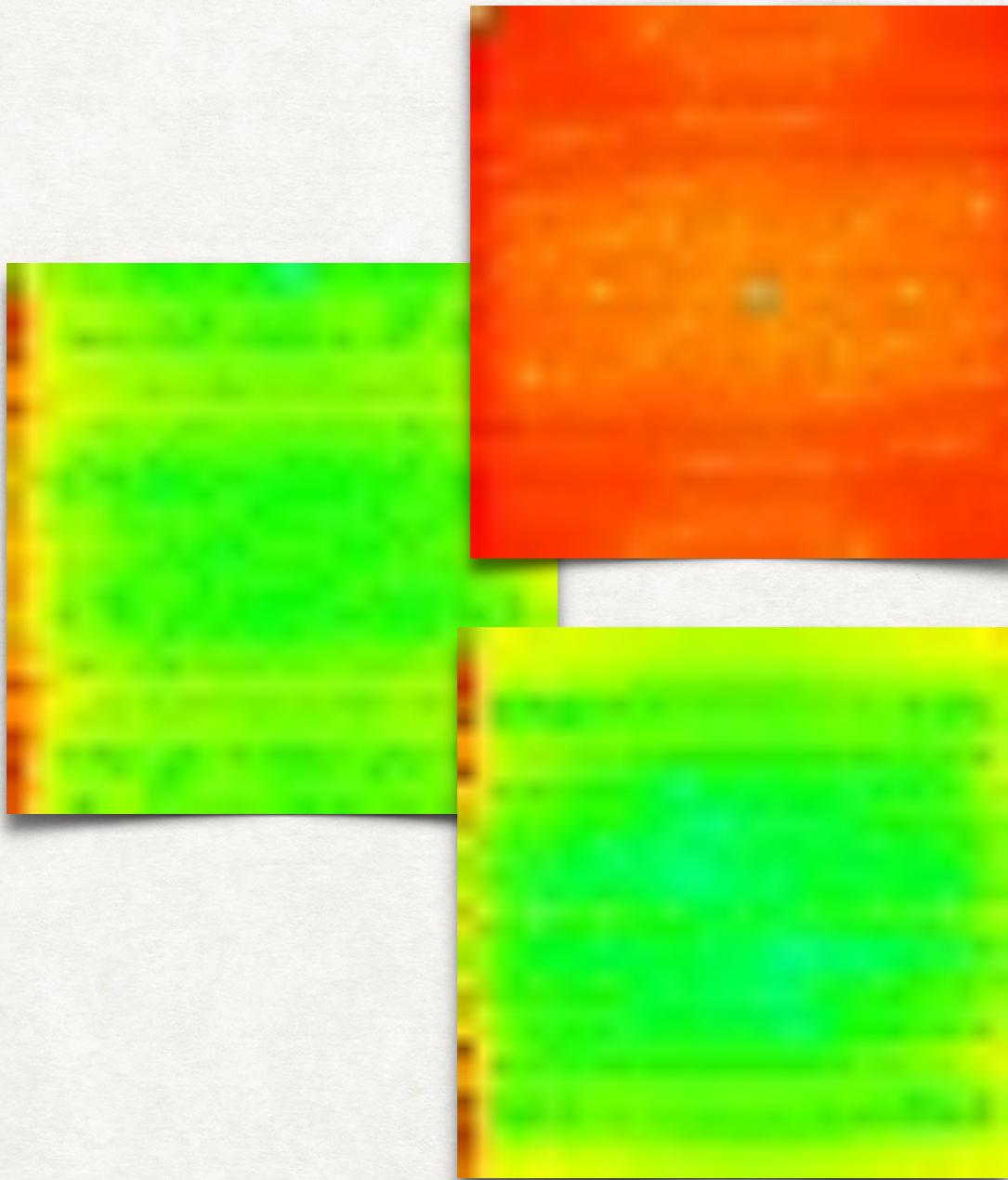
1. k initial "means" (in this case $k=3$) are randomly generated within the data domain (shown in color).

2. k clusters are created by associating every observation with the nearest mean. The partitions here represent the [Voronoi diagram](#) generated by the means.

3. The [centroid](#) of each of the k clusters becomes the new mean.

4. Steps 2 and 3 are repeated until convergence has been reached.

ADJUST CENTROID



EXECUTE TEST

WITHOUT HPF, NORMALIZATION

OpenCV Version : 3.0.0

[CPU Time]= 8.970104 sec. Analysing 224 Images Completed.

[CPU Time]= 8.970214 sec. Clustering Completed.

[CPU Time]= 9.689396 sec. Classifying 26 Images Completed.

Type = 01, Success Rate = 40.00% (2 / 5)

Type = 02, Success Rate = 66.67% (4 / 6)

Type = 03, Success Rate = 25.00% (1 / 4)

Type = 04, Success Rate = 80.00% (4 / 5)

Type = 05, Success Rate = 0.00% (0 / 6)

Total Success Rate = **42.31%** (11 / 26)

EXECUTE TEST WITH HPF

OpenCV Version : 3.0.0

[CPU Time]= 8.334871 sec. Analysing 224 Images Completed.

[CPU Time]= 8.334976 sec. Clustering Completed.

[CPU Time]= 9.071938 sec. Classifying 26 Images Completed.

Type = 01, Success Rate = 20.00% (1 / 5)

Type = 02, Success Rate = 83.33% (5 / 6)

Type = 03, Success Rate = 75.00% (3 / 4)

Type = 04, Success Rate = 0.00% (0 / 5)

Type = 05, Success Rate = 66.67% (4 / 6)

Total Success Rate = **50.00%** (13 / 26)

improved 7.69 %p

EXECUTE TEST WITH HPF, NORMALIZATION

OpenCV Version : 3.0.0

[CPU Time]= 8.739225 sec. Analysing 224 Images Completed.

[CPU Time]= 8.739335 sec. Clustering Completed.

[CPU Time]= 9.482151 sec. Classifying 26 Images Completed.

Type = 01, Success Rate = 40.00% (2 / 5)

Type = 02, Success Rate = 83.33% (5 / 6)

Type = 03, Success Rate = 75.00% (3 / 4)

Type = 04, Success Rate = 80.00% (4 / 5)

Type = 05, Success Rate = 50.00% (3 / 6)

Total Success Rate = **65.38%** (17 / 26)

improved 23.07 %p

EXECUTE TEST WITH HPF, NORMALIZATION, ADJUSTING CENTROID

OpenCV Version : 3.0.0

[CPU Time]= 8.399481 sec. Analysing 224 Images Completed.

[CPU Time]= 8.399612 sec. Clustering Completed.

[CPU Time]= 9.175937 sec. Classifying 26 Images Completed.

Type = 01, Success Rate = 80.00% (4 / 5)

Type = 02, Success Rate = 83.33% (5 / 6)

Type = 03, Success Rate = 75.00% (3 / 4)

Type = 04, Success Rate = 100.00% (5 / 5)

Type = 05, Success Rate = 83.33% (5 / 6)

Total Success Rate = **84.62%** (22 / 26)

improved 42.31 %p !

EXECUTE TEST WITH LESS SAMPLES

OpenCV Version : 3.0.0

[CPU Time]= 5.701056 sec. Analysing 199 Images Completed.

[CPU Time]= 5.701179 sec. Clustering Completed.

[CPU Time]= 7.128032 sec. Classifying 51 Images Completed.

Type = 01, Success Rate = 20.00% (2 / 10)

Type = 02, Success Rate = 100.00% (10 / 10)

Type = 03, Success Rate = 71.43% (5 / 7)

Type = 04, Success Rate = 80.00% (8 / 10)

Type = 05, Success Rate = 0.00% (0 / 14)

Total Success Rate = **49.02%** (25 / 51)

DISCUSS

- To improve the accuracy,
 1. choose more values from the coefficients
 2. use more precise floating-point system
 3. collect much more samples
- To improve the speed,
 1. use FFT algorithm ($N \lg N$)
 2. use pre-computed trigonometric value

You could get the full source code at <https://github.com/iriszero>

