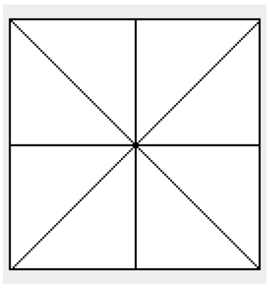
Analysis Questions for part 1 – submit as a pdf or word document

1. Let's try an experiment where s (scale factor) remains constant and n (number of lines) is allowed to vary. Comment on your results as to now aliasing is affected by using various constant values of s for changing n in each case. You may attach results, plot charts etc. to quantify your results.

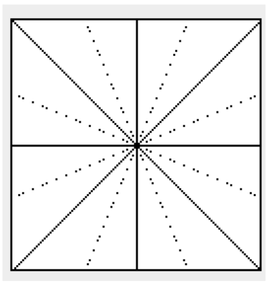
Aliasing is affected when I use the specific constant values of s for changing n value. Let's say s is 4.0.

1. multiples of eight for n values (8,16,24,32) is quite accurate. That is because the angle is composed of 45, 90, 135, 180 degrees which means the slope is 0 or 1.

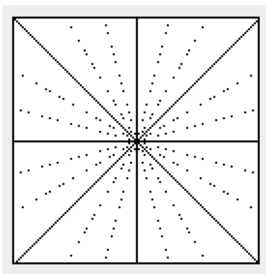
s: 4.0, n: 8



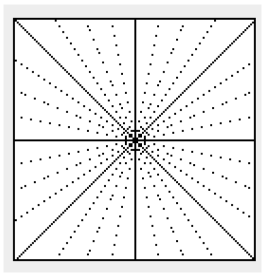
s: 4.0, n: 16



s: 4.0, n: 32

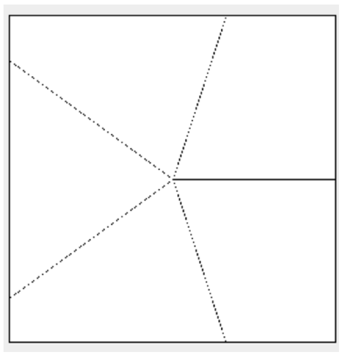


s: 4.0, n: 64

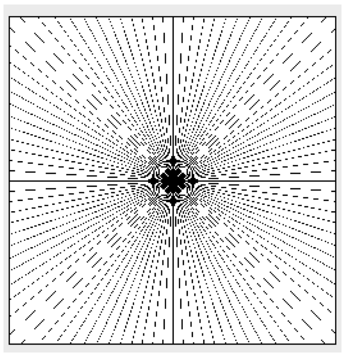


2. If the slope is not integer, we can easily recognize aliasing when scaling down. That is because float values of slope are converted into integer which means the lose of data. Let's say s value is 2.0.

s: 2.0, n: 5



s: 2.0, n: 100

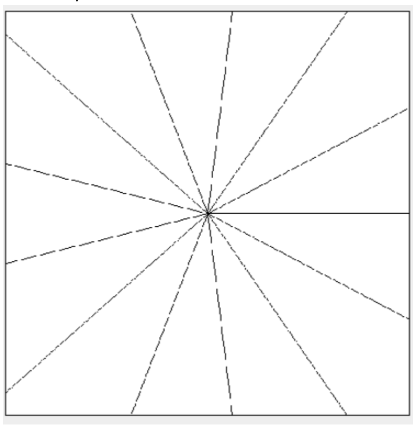


2. Let’s try another experiment, this time keep n (number of lines) constant and varying s (scale factor). Comment on your results as to now aliasing is affected by using various constant values of n for changing s. You may attach results, plot charts etc. to quantify your results.

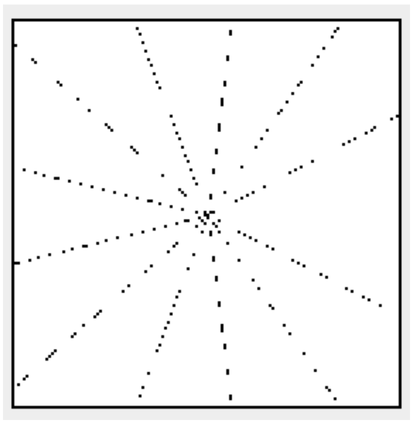
Aliasing is affected when I use the specific constant values of n for changing s value. Let’s say n is 13.

1. size of s value affects image quality. The bigger an image size is, the more the image can get quality. That is because, the image can get more samples from the original image.

n: 13, s: 1.2

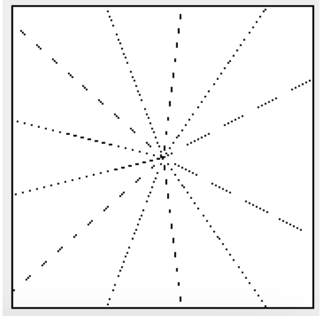


n: 13, s: 3.7

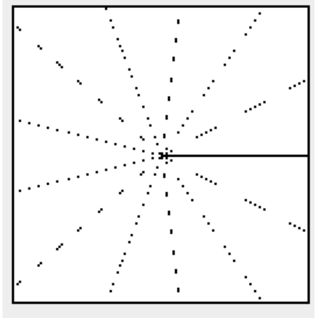


1. Even if the image does not have enough samples, factors of 512 for s values (2,4,8,16...) may improve image quality. That is because, when sampling down, there are no loss of data for converting float value into integer.

n: 13, s: 3.0



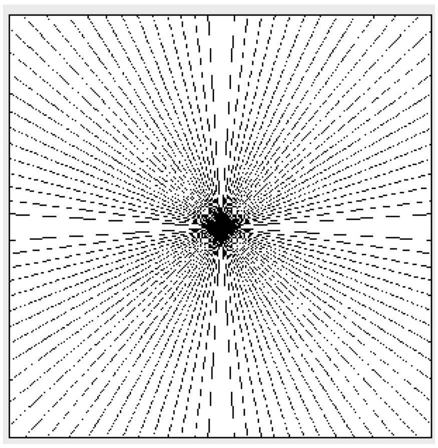
n: 13, s: 4.0



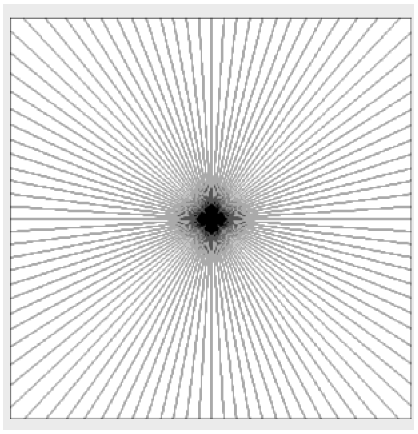
3. Anti-Aliasing

I get quite accurate result of anti-aliasing. Copy an average value from the original image. The pixel range is 3x3 for calculating average value. I dealt with marginal pixel for average value. For example, (0,0) pixel has 3 adjacent pixels (1,0)(0,1)(1,1) so gave a weight value 1/4 for each pixels. Also, (1,1) pixel has 8 adjacent pixels (0,0)(0,1)(0,2)(1,0)(1,1)(1,2)(2,0)(2,1)(2,2) so gave a weight value 1/9 for each pixels. The pixel color of the result is 9 grayscales which means it contains light-gray, dark-gray, black, white and so on.

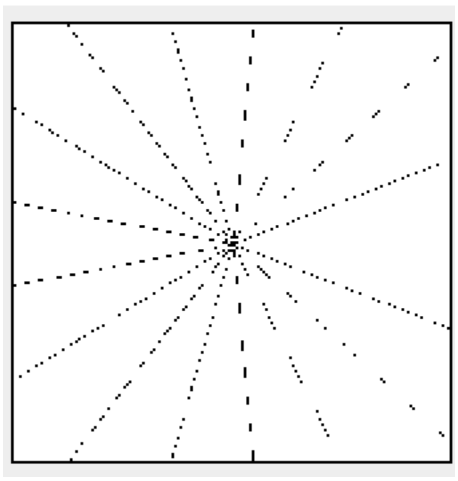
n: 100 s: 1.7



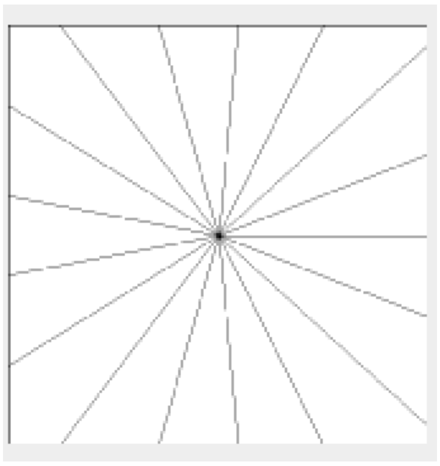
n: 100 s :1.7, Anti-Aliasing



n: 17, s :3.1



n: 17, s: 3.1, Anti-Aliasing

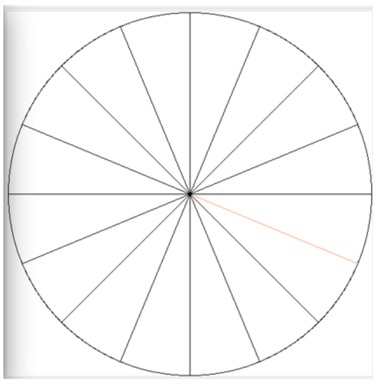


Analysis Questions for part 2 – submit as a pdf or word document

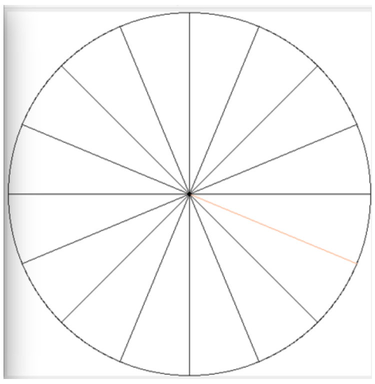
Let's try an experiment where s (speed of rotation) remains constant and fps is allowed to vary. Study the value of the os (observed speed of rotation), especially when there is temporal aliasing.

Let's say **s = 1** rotations per second. If the image is displayed every one second (**fps = 1**), people think the image is being stopped even if the image is rotating.

1 second



2 second



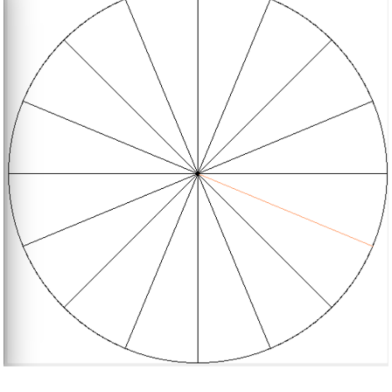
There is a temporal aliasing. We can simply know os is 0

Let's think about other case, **s = 1** rotations per second and the image is displayed every 0.5 seconds (**fps = 2**). There are two possible cases in this example.

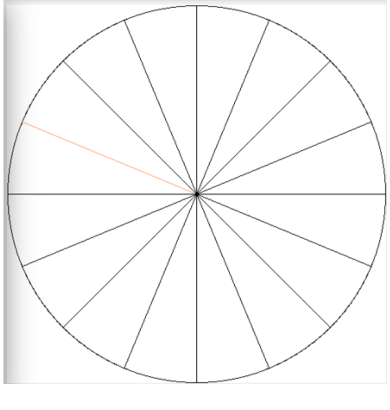
(1) number of lines are even number

People think this image is being stopped because the image is shown every 180 degrees.

0.5 second



1 second



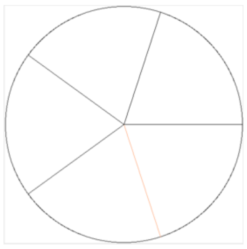
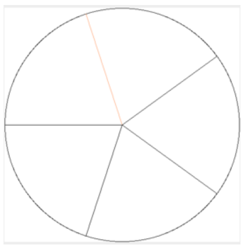
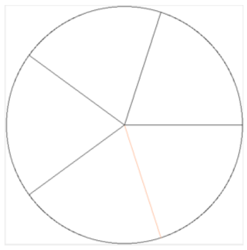
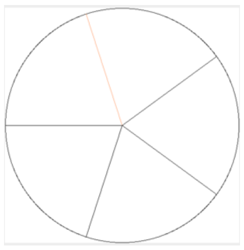
(2) number of lines are odd number

People think this image is back and forth between some specific area (seems that image shakes) because the image is changed only 2 times every second.

0, 1, 2, 3, 4, 5... second (0,360,720... => one same image),

0.5, 1.5 ,2,5, 3,5, 4.5... second (180,540,900... =>another same image)

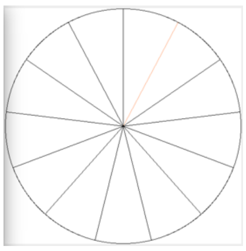
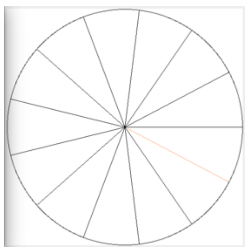
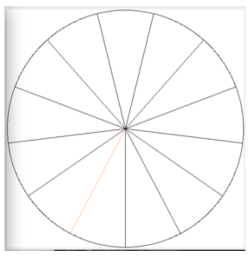
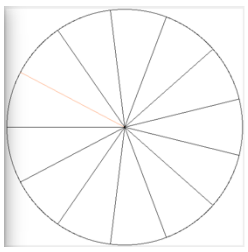
0.5 second 1 second 1.5 second 2 seconds

There is a temporal aliasing. When I calculate os, I will ignore the exceptional cases (number of lines).

Let’s think about other case, **s = 1** rotations per second and the image is displayed every 0.25 seconds (**fps = 4**).

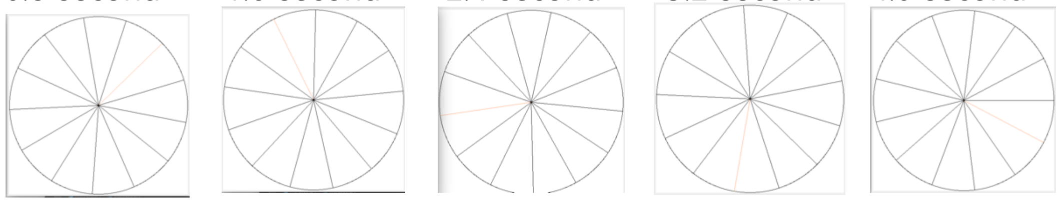
0.25 second 0.5 second 0.75 second 1 second

There is no temporal aliasing. People can think the image is rotating clockwise.

Let's think about other case, **s = 1** rotations per second and the image is displayed every 0.8 seconds (**fps = 1.25**).

0.8 second 1.6 second 2.4 second 3.2 second 4.0 second



There is a temporal aliasing. People can think the image is rotating counter-clockwise.

P.S: The original image's fps (left side image) = 10.

1. Can you design a formula relating s, fps and os.

Aside from exceptional cases (I explained above such as number of lines), let's think about designing a formula.

(1) we want to know rotation/frame which means how much rotating the image is per frame (displaying image).

Rotation/frame = s / fps

(2) Visual Perceptions

Convert[Rotation/frame]

People think 1,2,3,4,5,6... rotations per frame is same as 1 rotation per frame.

1) Rotation/frame >= 1.0 use only decimal point.

Ex) Rotation/frame = 25.9

Convert[Rotation/frame] = 0.9

2) 0.5 < Convert[Rotation/frame] or Rotation/frame < 1.0

=> temporal aliasing. It seems that images are rotating counter-clockwise.

CounterConvert[Rotation/frame] = -|1-0.9| = -0.1

3) 0 < Convert[Rotation/frame] or Rotation/frame <= 0.5

Rotation/frame = Convert[Rotation/frame]

(3) What is the observed speed?

1) **Os = Convert [s / fps] * fps** **// clockwise**

2) **Os = CounterConvert [s / fps] * fps** **// counter-clockwise**

Evaluate if your formula works for certain values of s and fps. If s = 10 rotations per second,

2. What is the observed speed os for an fps of 25?

Rotation/frame = s/fps = 0.4 rotation/frame

Convert [0.4 rotation/frame] = 0.4 rotation/frame

(Since there is no aliasing, os should be same as s)

os = 0.4 rotation/frame * 25 frame/second = **10 rotations/second. (for clockwise)**

P.S) If you really want to think about number of lines, the formula is S*(spokes/rotation) / fps

For example, Q1: 0.4 rotation/frame * n spokes/rotation = 0.4n spokes/frame

If n (=number of lines) is 10, then 4 spokes/frame. In this case, people think the image is being stopped which means os = 0 rotations/second.

It is very interesting. If you want to test it, just put number of line=10, speed of rotation=10, fps=25 into my program. You can see the image is stopped.

3. What is the observed speed os for an fps of 16?

NOTE THAT there is an aliasing because Rotation/frame is more than 0.5. In this case, the image seems rotating counter-clockwise.

Rotation/frame = s/fps = 0.625 rotation/frame

ConvertCounter [0.625 rotation/frame] = -|1 – 0.625| = -0.375 rotation/frame

Os = -0.375 rotation/frame * 16 frame/second = -**6 rotations/second. (for counter-clockwise)**

P.S) If you really want to calculate clockwise observed speed, there is a way to ignore aliasing.

Rotation/frame = s/fps = 0.625 rotation/frame

Convert [0.625 rotation/frame] = 0.625 rotation/frame

os = 0.625 rotation/frame * 16 frame/second = 10 rotations/second. (for clockwise)

4. What is the observed speed os for an fps of 10?

Rotation/frame = s/fps = 1 rotation/frame

Convert [1 rotation/frame] = 0 rotation/frame

os = 0 rotation/frame * 10 frame/second = **0 rotations/second. (for clockwise)**

5. What is the observed speed os for an fps of 8?

Rotation/frame = s/fps = 1.25 rotation/frame

Convert [1.25 rotation/frame] = 0.25 rotation/frame

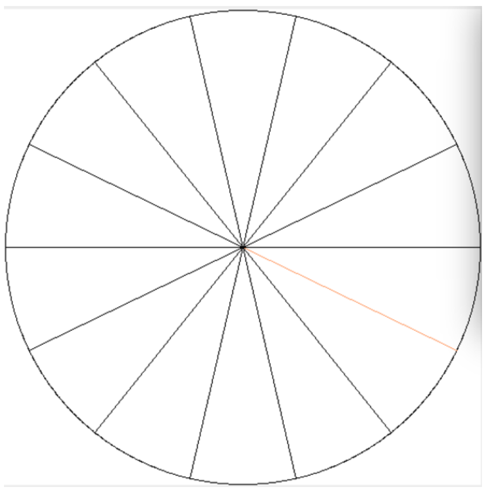
os = 0.25 rotation/frame * 8 frame/second = **2 rotations/second. (for clockwise)**

Analysis Questions for part 3 – Explain down-scaling, temporal-anti-aliasing and spatial-anti-aliasing

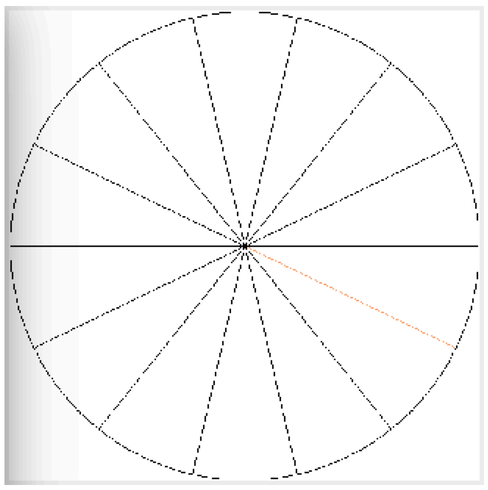
(1) down-scaling

Similar with Question for part 1. Dealing with scaling the temporal signal.

Original, n: 14



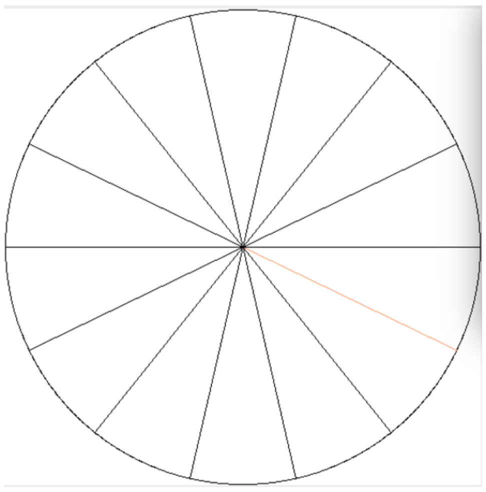
n: 14, s: 1.6, Aliasing



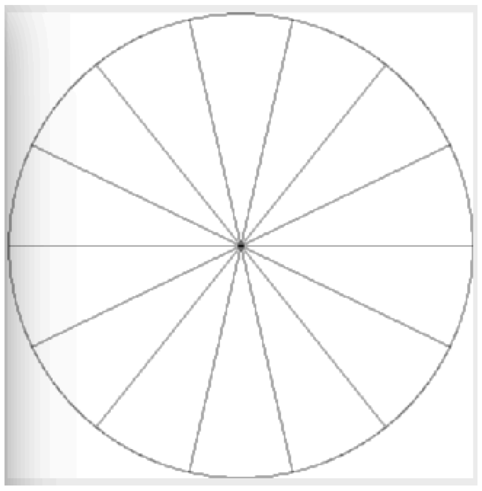
(2) Spatial-anti-aliasing

Similar with Question for part 1.

Original, n: 14



n: 14, s: 1.6, Spatial-anti-aliasing



(3) Temporal-anti-aliasing

Aliasing occurs when the frequency content of the signal exceeds the Nyquist frequency. Once aliasing has occurred, there is no way to get back the original signal. Therefore, before aliasing occurs, I have to deal with the aliasing. The one way is to increase sampling rate(fps). However, it is difficult to sample fast enough due to hardware problems. (For example, 25fps = 100fps = 10000fps). Thus, I implemented anti-aliasing filter which cutoffs the high frequencies if it is not satisfied with Nyquist's Sampling Theorem.

Let's think about s=6 revolutions per second, fps=10 frame/second. The Nyquist factor is 12.0 and alias frequency is 2.0. Cutoff alias frequency so Nyquist factor should be less than 12.0-2.0=10.0 which means s should be less than 5.