UNIVERSITY OF MORATUWA

FACULTY OF ENGINEERING


# EN4353 - RADAR AND NAVIGATION

## ASSIGNMENT 1 TARGET AND CLOUD DETECTION BY AMPLITUDE PROCESSING

DATE OF SUBMISSION: MARCH 9, 2022


| **Name** | **Index Number** |
|---|---|
| Sampath M.K.T | 170543G |


THIS REPORT IS SUBMITTED AS A PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE MODULE MN4150 PROJECT MANAGEMENT


DEPARTMENT OF ELECTRONICS & TELECOMMUNICATIONS ENGINEERING

# Contents

# 1. Introduction

In this assignment, the task is development an algorithm to,

1. Detect possible targets
2. Mark clouds

In given radar data samples. This report describes the approach carried out to do those tasks, assumptions, and methods. Assignment has three data sets.

# 2. Methodology

## 2.1 Data Preparation

### 2.1.1 Constants

1. N = 30 : Number of range bins
2. fr = 1800 : Pulse repetition frequency
3. F = 1080000 : Original sampling frequency
4. K = 2.0 : Constant k in equation $Vth = Vn_avg + k.sigma$
5. SIGMA = 1 : Constant sigma in equation $Vth = Vn_avg + k.sigma$
6. RPM = 12 : RPM of the radar
7. $Thita_H = 2$ : Horizontal Beam Width
8. $GuardBand = 4$
9. $WindowSize = 21$
10. k = 0.5 : Constant of non-coherent binary integration target selecting
11. $C = 3 * 10^8$ speed of light

### 2.1.2 Initial Calculations

Calculation are done in python and only final answers are show here without substituting to equations.
Sampling time $Ts = 1/(N.fr)$
$fs = int(1/Ts) = 54000Hz$

Time taken to collect given data
$T_{sampled} = OriginalSampleCount/F$

Angle rotated with in collecting data
$Thita_{rotated} = (T_{sampled} * (RPM/60) * 360) = 1.2deg$

Number of echoes per echos per sweep $n = (thita_H * fr)/(6 * N)$
$n = int((Thita_H * fr)/(6 * RPM)) = 50$

Number of echos expected for rotated angle
$n = int((n/Thita_H) * Thita_{rotated}) = 30$

Threshold of accepting bin as a possible tagert bin,
$obj_acceptingthreshold = int(n * k)$

### 2.1.3 Sampling

Hence original signal is sampled in $1080khz$ while sampling frequency required for given number of bins is $54kHz$. The sampling factor is 20. Therefore given raw signal is sampled by this sampling factor starting from $0^{th}$ sample.

## 2.2 Target Detection

### 2.2.1 Calculate $(V_n)_{avg}$

Using equation,

$(V_n)_{avg} = \frac{\sum_j^m(pastvideo)+\sum_j^m(futurevideo)}{12}$
with m=10 and j=4, for every sample in sampled signal A array of $(V_n)_{avg}$ is generated.

### 2.2.2 Calculate $V_{TH}$

Using equation,
$V_{TH} = (V_n)_{avg} + k \cdot \sigma_n$
with k = 2.0 and $\sigma_n$ = 1 for every sample in sampled signal A array of $(V_n)_{avg}$ is generated.

### 2.2.3 Thresholding

Using equation,
$If\ V_k > V_{TH}\ then\ V_k = 1\ otherwise\ 0$
Then resultant array of samples is reshaped to matrix of size $PulseCount*NumberOfBins$.

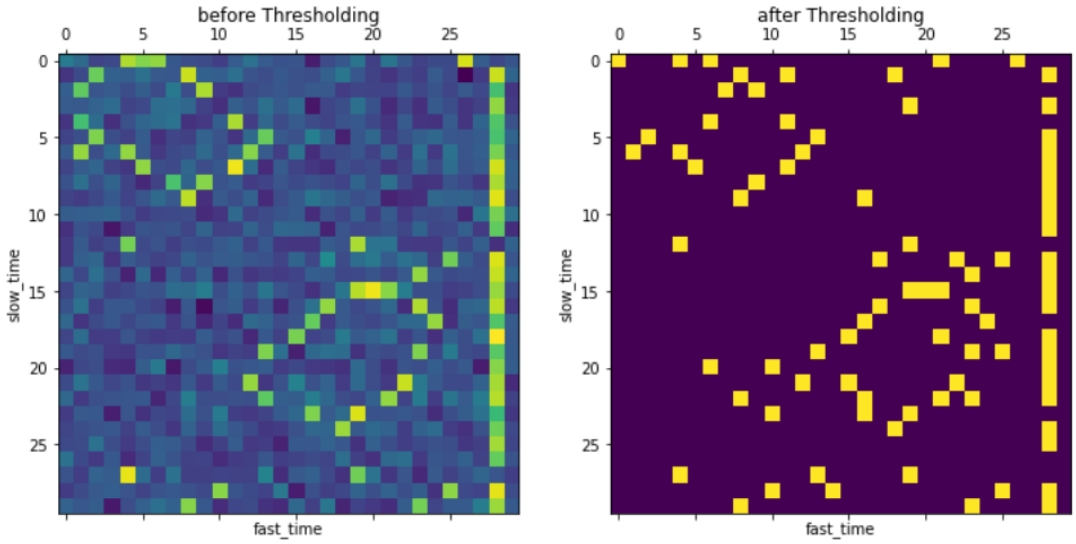


Figure 2.1

### 2.2.4 Detect Possible Targets

Non Coherent Binary Integration is used to detect targets. Hence to using condition,
**If count of 1 for given bin along all pulses > 0.5 [Expected echoes per target per scan]**
Bins with possible targets are selected and stored.

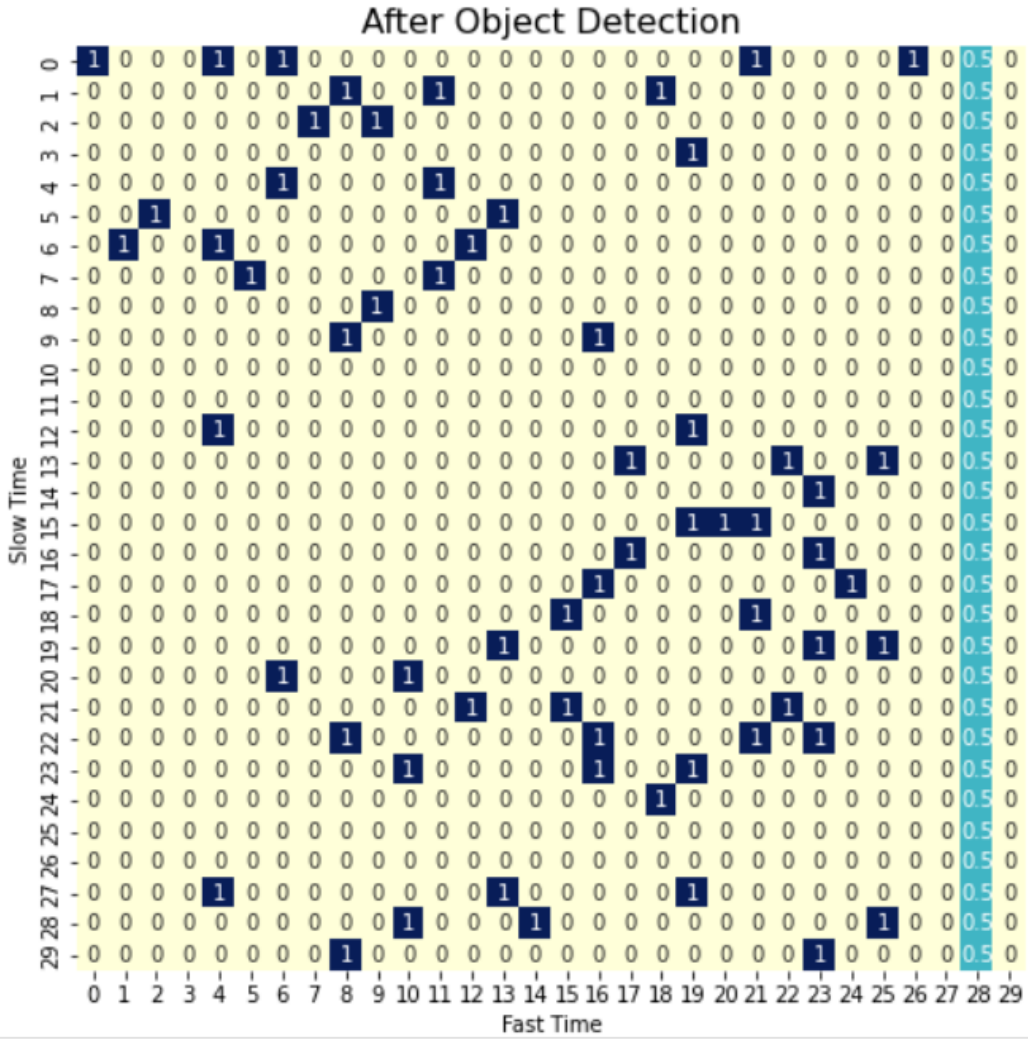


Figure 2.2: Possible Target bin is masked with 0.5

### 2.2.5 Calculate Asimuth Angle and Range of Targets

Using equation,
$AzimuthAngle = (n+(m-n)/2)*(RadarRotationSpeed/PulseRepititionFrequency)$
Angle is calculated per each target.

Then to find range, first calculate R
$R = (CTs)/2 = 310^8(1/54000) = 2778m$
Then Range for given range $n^{th}$ bin is n*R

For Third data set (raw data3),
Target 1 : azimuth angle = 0.64° | Range = 80.56km

## 2.3 Cloud Detection

Since data set has noise before detecting clouds noise filtering should be done. Then Target bins are cleared our for sake of easiness (later targets will be added).

### 2.3.1 Pre-Processing

Following steps are followed to pre-process data to noise filtering.

1. Threshold a fresh data matrix using previously calculated $V_{TH}$.
2. Threshold previous matrix with tuneable threshold.
3. Map thresholded matrix in to binary matrix.
4. Remove possible targets. Make all previously detected target bins zero.

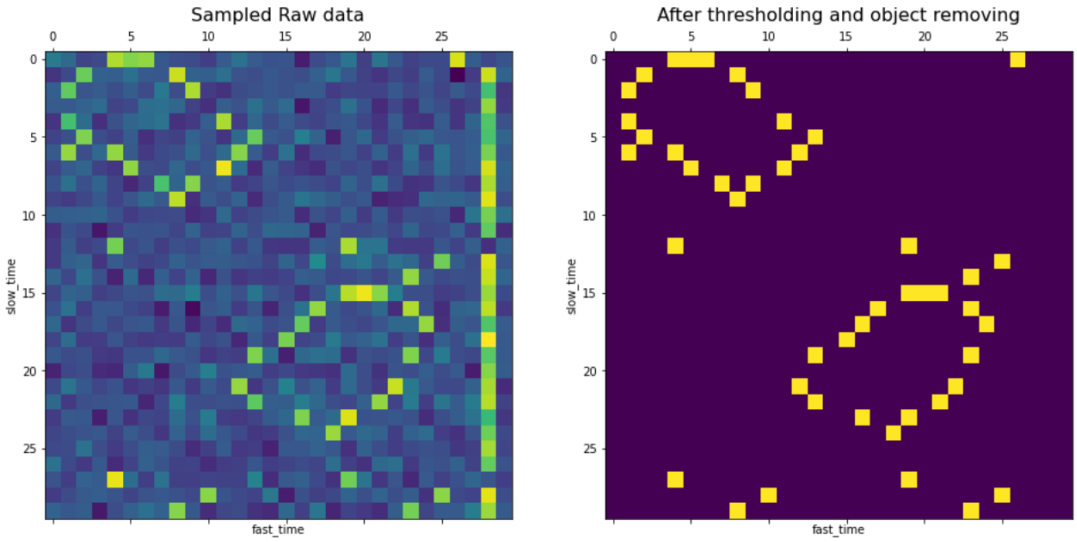


Figure 2.3: After Pre-Processing

### 2.3.2 Noise Filtering

To filter out noise to custom filters are used.

#### Filter 1

This filter returns a possible noise point as a matrix.
Inputs

1. Data matrix
2. Kernal size

3. Threshold

Output

1. Matrix of random noise points (same size as the input matrix)

Method

1. Run a square Kernal on the input matrix
2. Adds up values under kernel
3. If the sum is greater than threshold ‘which means its data, not random, alone noise point‘ erase (set to zero) whole area under the kernel
4. Return the resultant matrix

Then subtract noise from data.

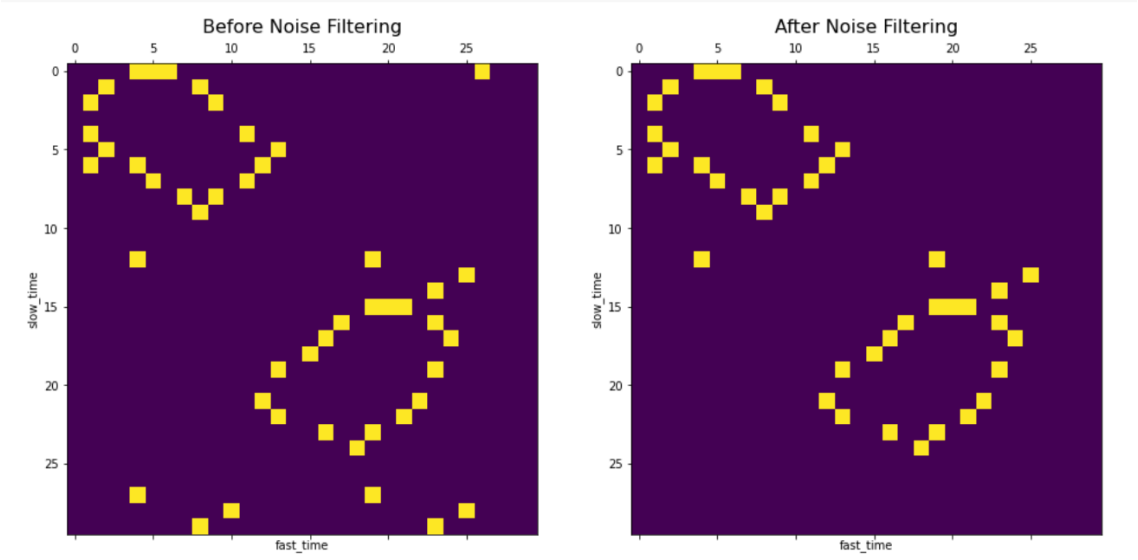


Figure 2.4

### Filter 2

This filter returns a possible noise point as a matrix.
Inputs

1. Data matrix
2. Kernal size
3. Threshold

Output

1. Matrix of noise removed (same size as the input matrix)

Method

1. Run a square Kernal on the input matrix
2. Adds up values under kernel
3. If the sum is less than threshold ‘which means its random noise point or spike‘ erase (set to zero) only center of the kernel
4. Return the resultant matrix

Then subtract noise from data.

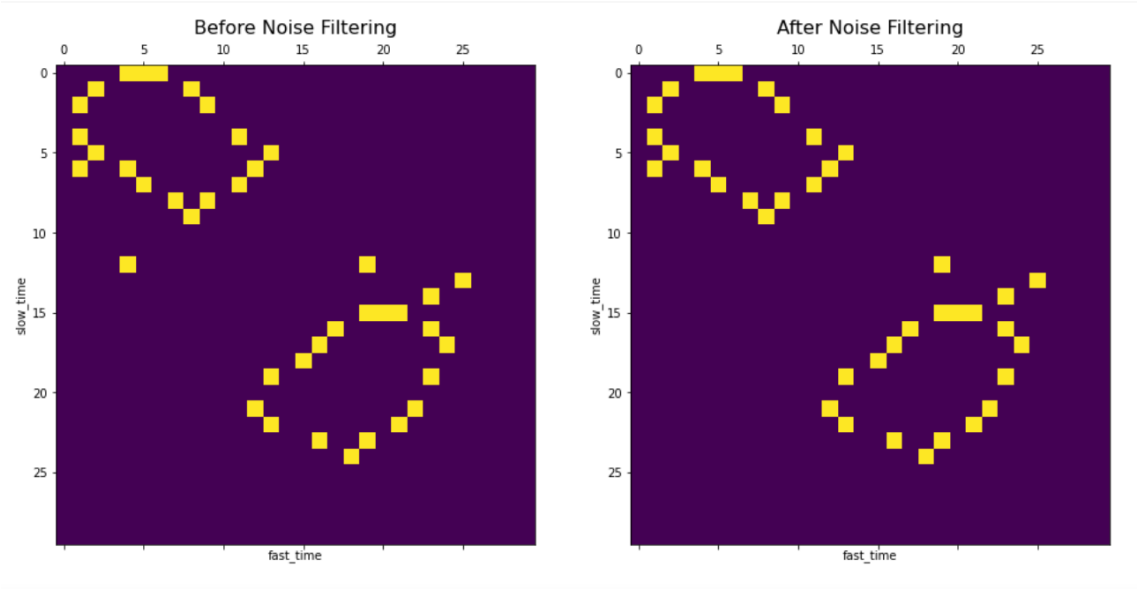


Figure 2.5

### 2.3.3 Filling Gaps

To fill up cloud contours, cloud edges need to be gap free (continuous). Hence, that gaps need to be filled. for that two algorithms are used. One for fill two slot gaps and one for fill single slot gaps.

#### connect two pixel gaps(mat)

This function connect ‘two slots gaps‘. Following figure describe algorithm.

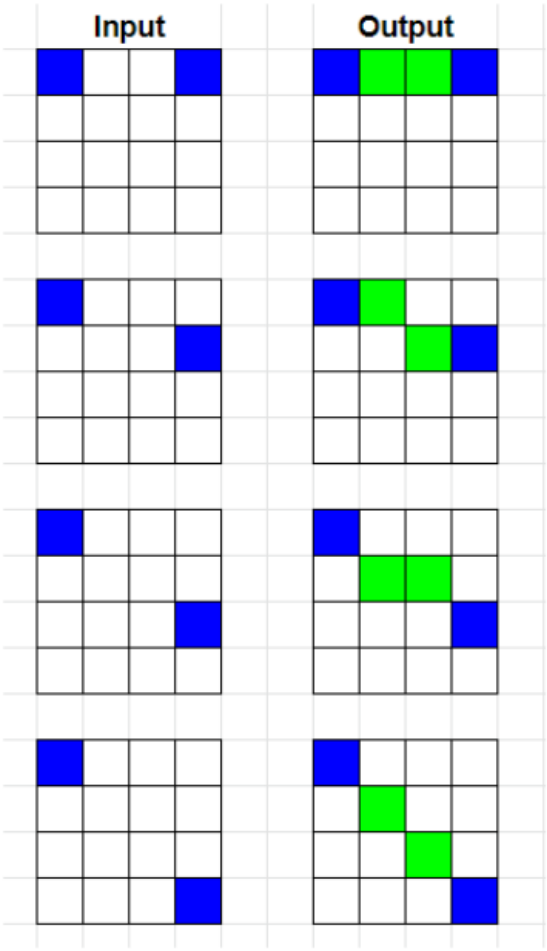


Figure 2.6

**connect one pixel gaps(mat)**

This function connect ‘one slot gaps‘. Following figure describe algorithm.

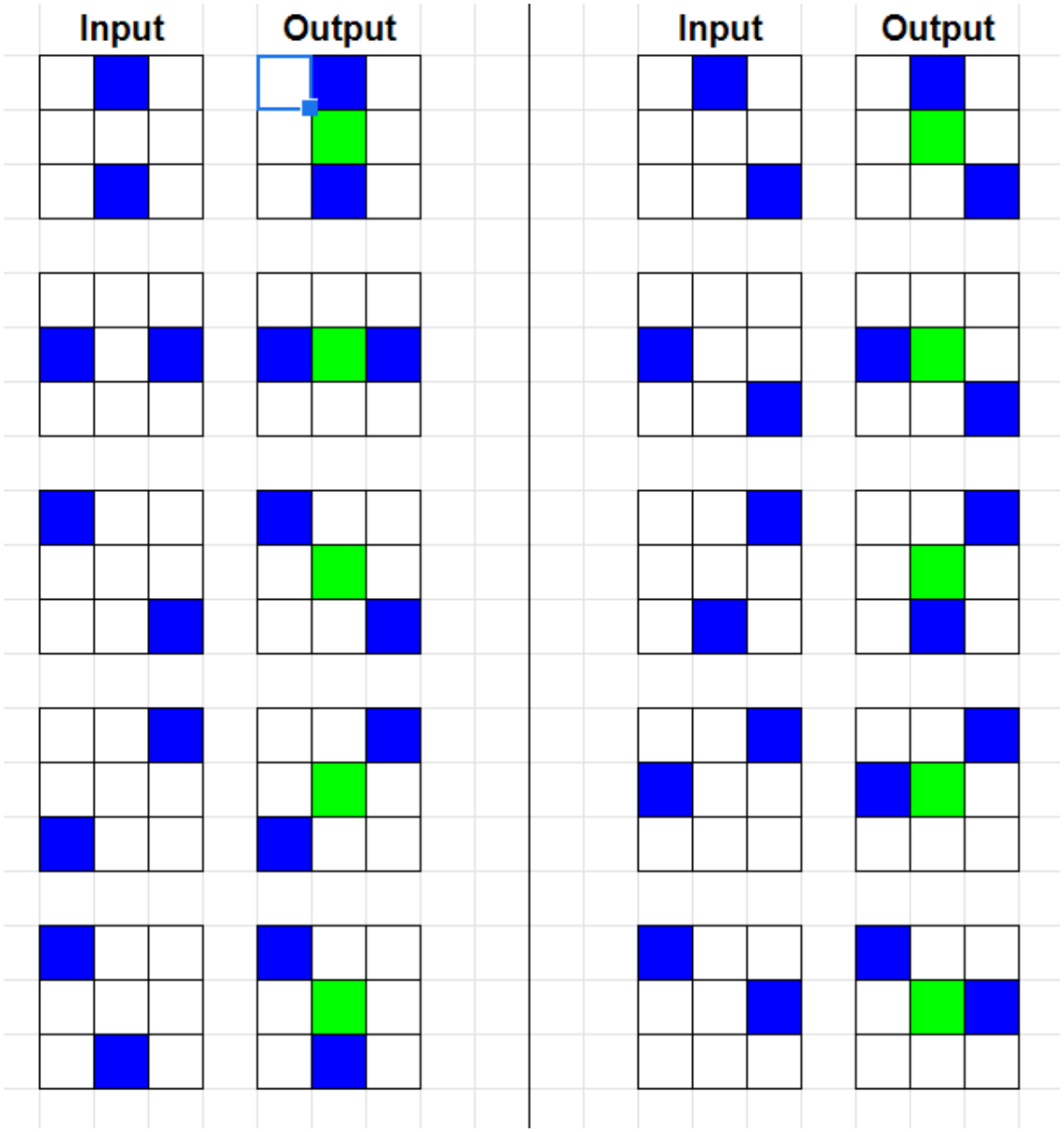


Figure 2.7

Then This two Gap filling algorithms are apply on noise removed data matrix. When applying, each algorithm is applied three times, which are,

1. original matrix
2. after flipping original matrix
3. after transposing flipped matrix.

This will ensure gaps in every directions are filled.
After this three iteration, resultant matrix is send through noise filter to remove any noises generated from the algorithm.

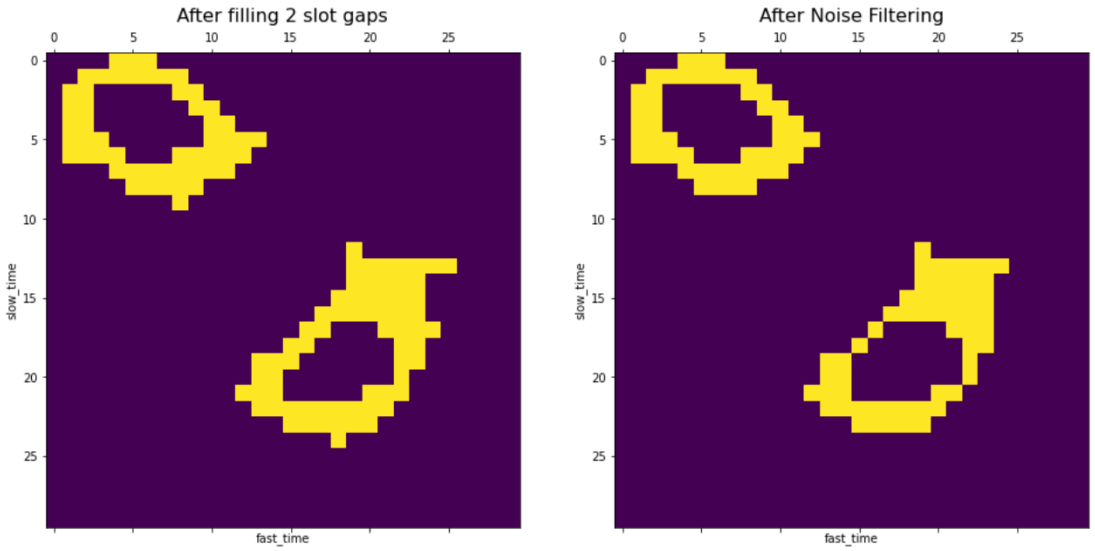


Figure 2.8: Two gap filling and filtering

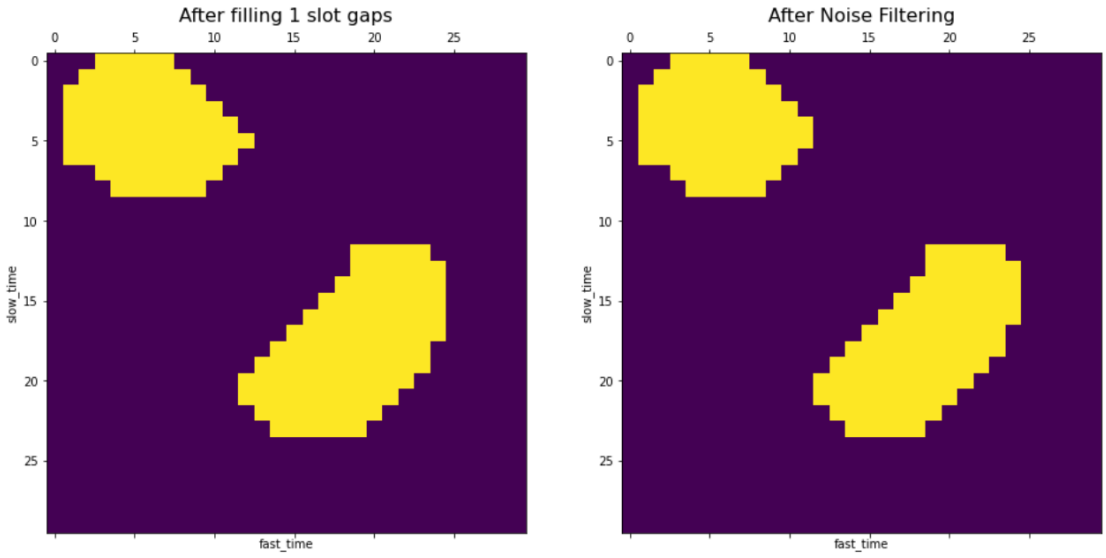


Figure 2.9: One gap filling and filtering

Even though resultant matrix clouds are filled. This cannot ensure to be the case for any input. hence flood fill is still required.

### 2.3.4 Flood Filling

To fill out cloud contours, following two steps are performed.

1. Fill the whole area outside the cloud contours with -1's to mask out non-cloud area.
2. All slots with -1's are set to 0s and other slots are filled with 1's

Flood fill algorithm requires a position to start with (seed). this function searches first non-cloud slot and return. **find seed point** function returns a possible seed point from non-cloud area.

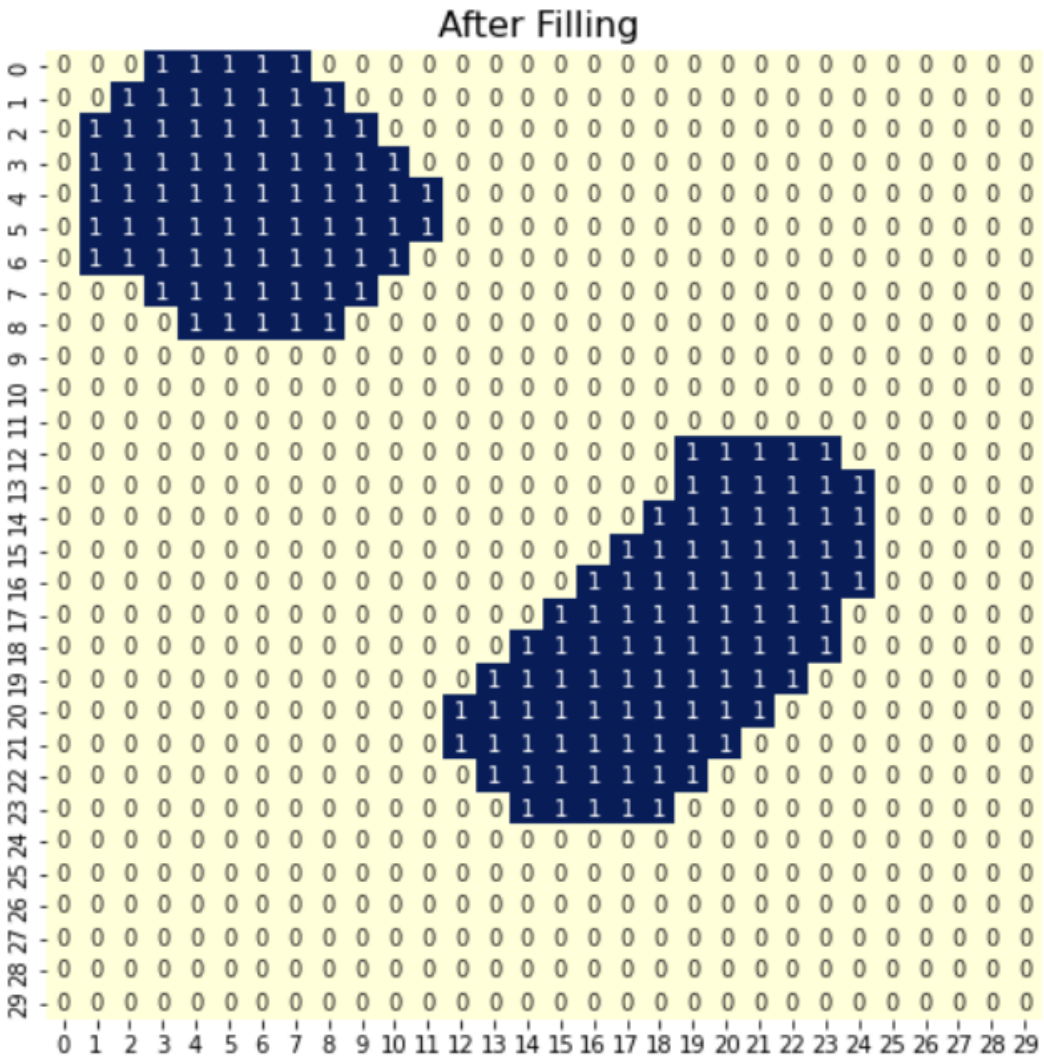


Figure 2.10: After fill cloud contours

### 2.3.5 Adding Targets to Finalize Matrix

After noise filtering and cloud filling, previously identified targets are encoded in to resultant matrix to finalize the task.

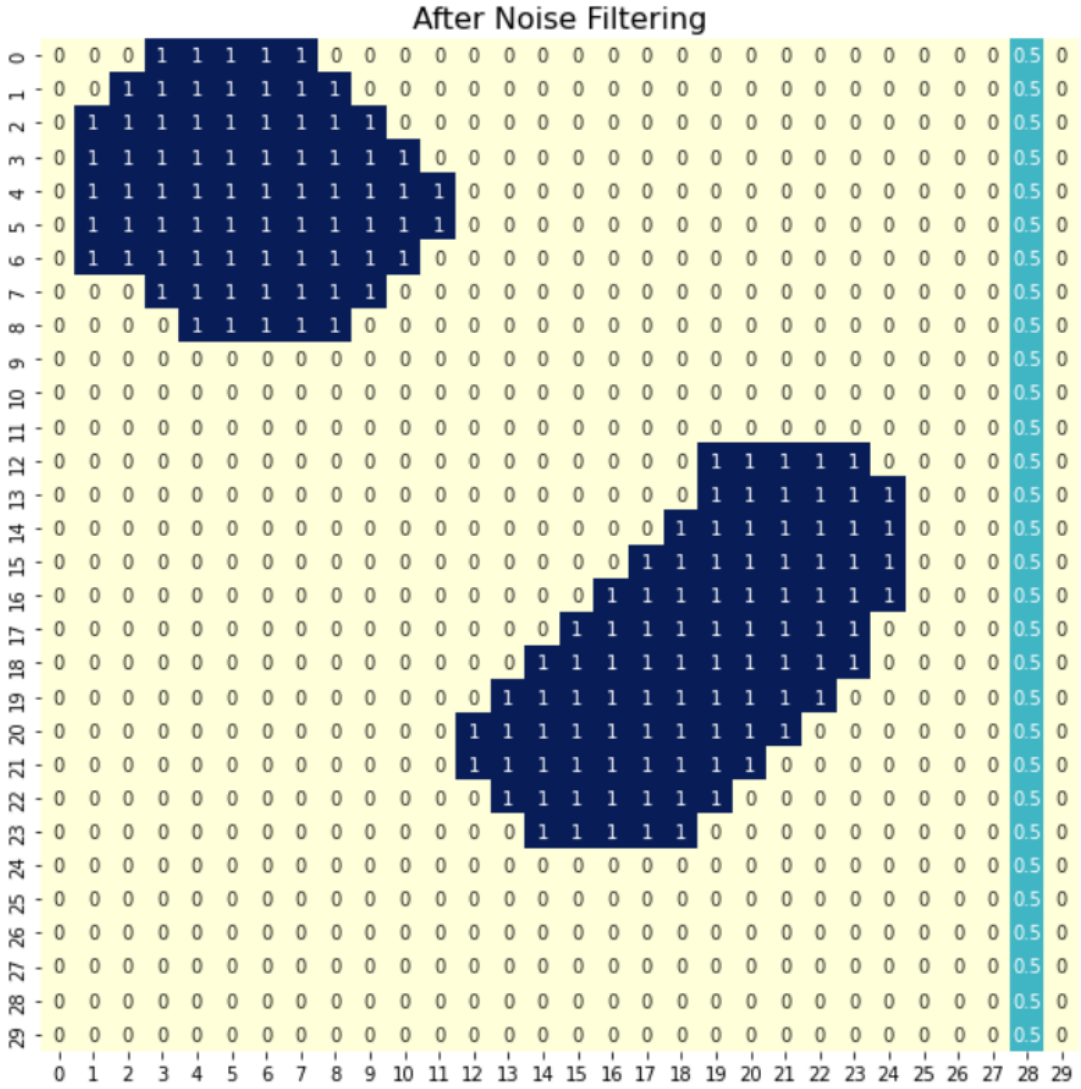


Figure 2.11: Final Result after filtering, contour completion and encoding

# 3. Conclusion

This series of algorithm can be used to detect rough contour depicting the cloud, presence of a targets, range and the azimuth angle of targets. More advance noise filtering algorithm can be used to get more accurate results. Since pulse count of the given data set is even less than expected pulse count of $\theta_H$, Changes of azimuth angle is very small and is not accurate.
Following figures show final result of the algorithm for given three data sets.

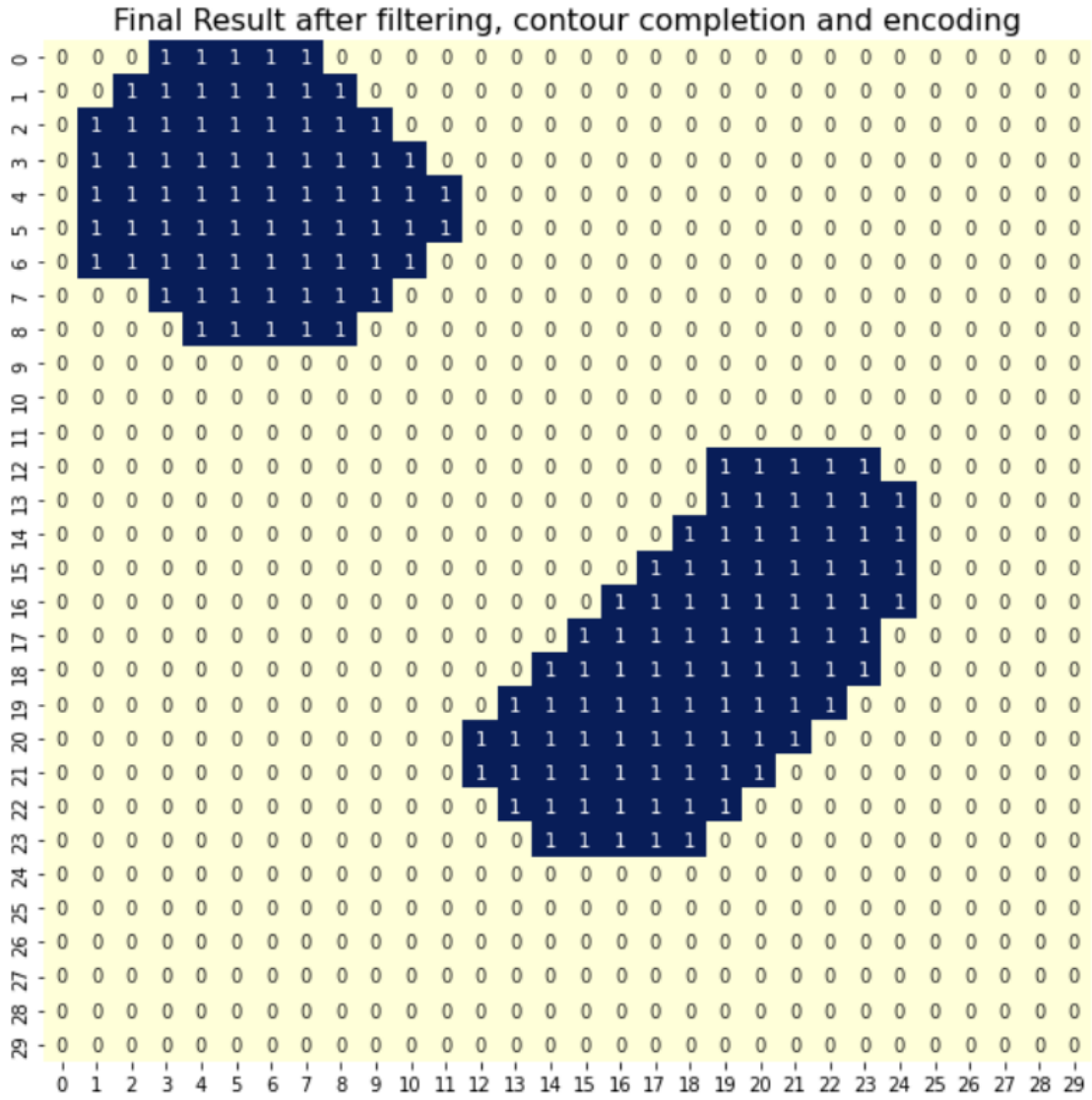


Figure 3.1: Final Result after filtering, contour completion and encoding for Data Set 1

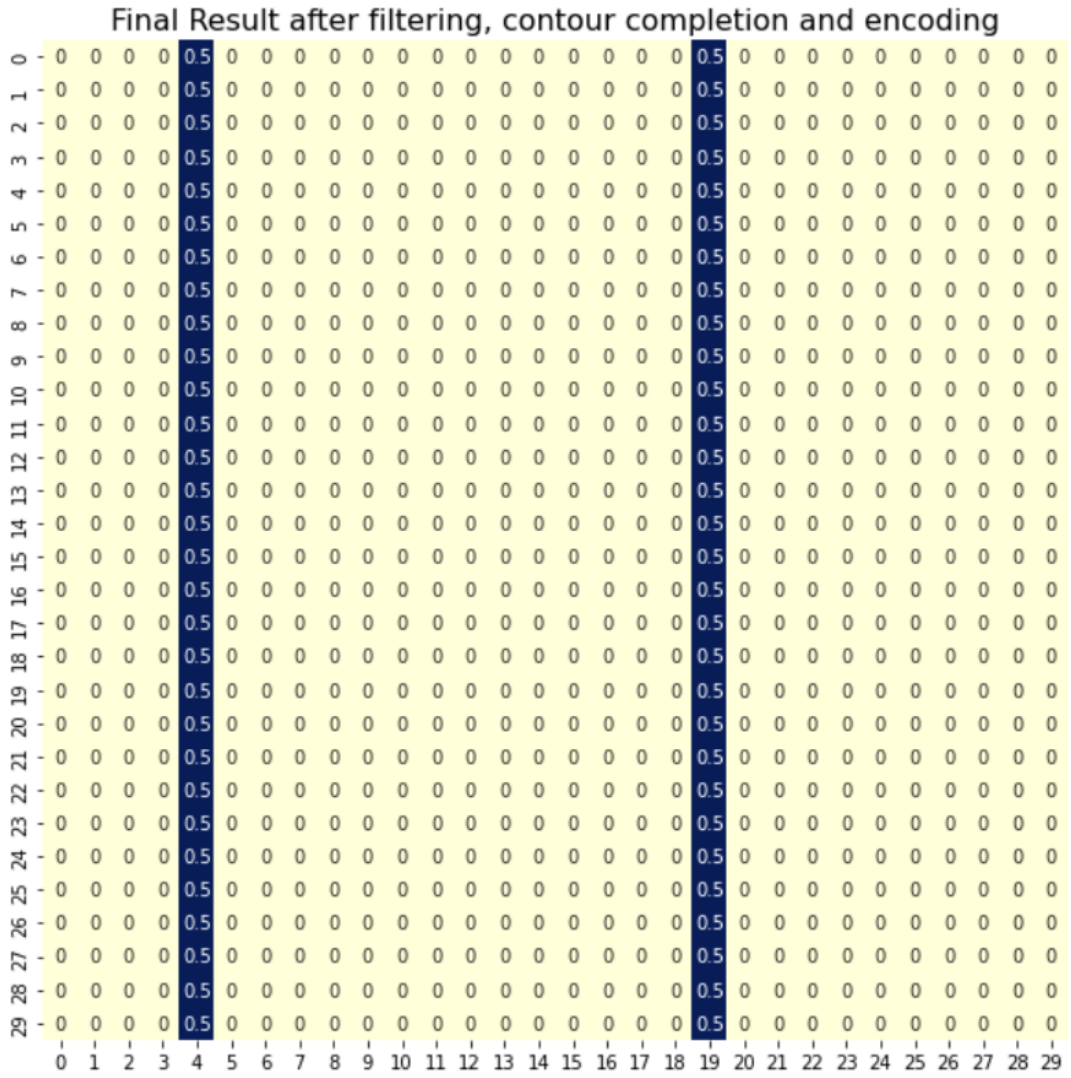


Figure 3.2: Final Result after filtering, contour completion and encoding for Data Set 2

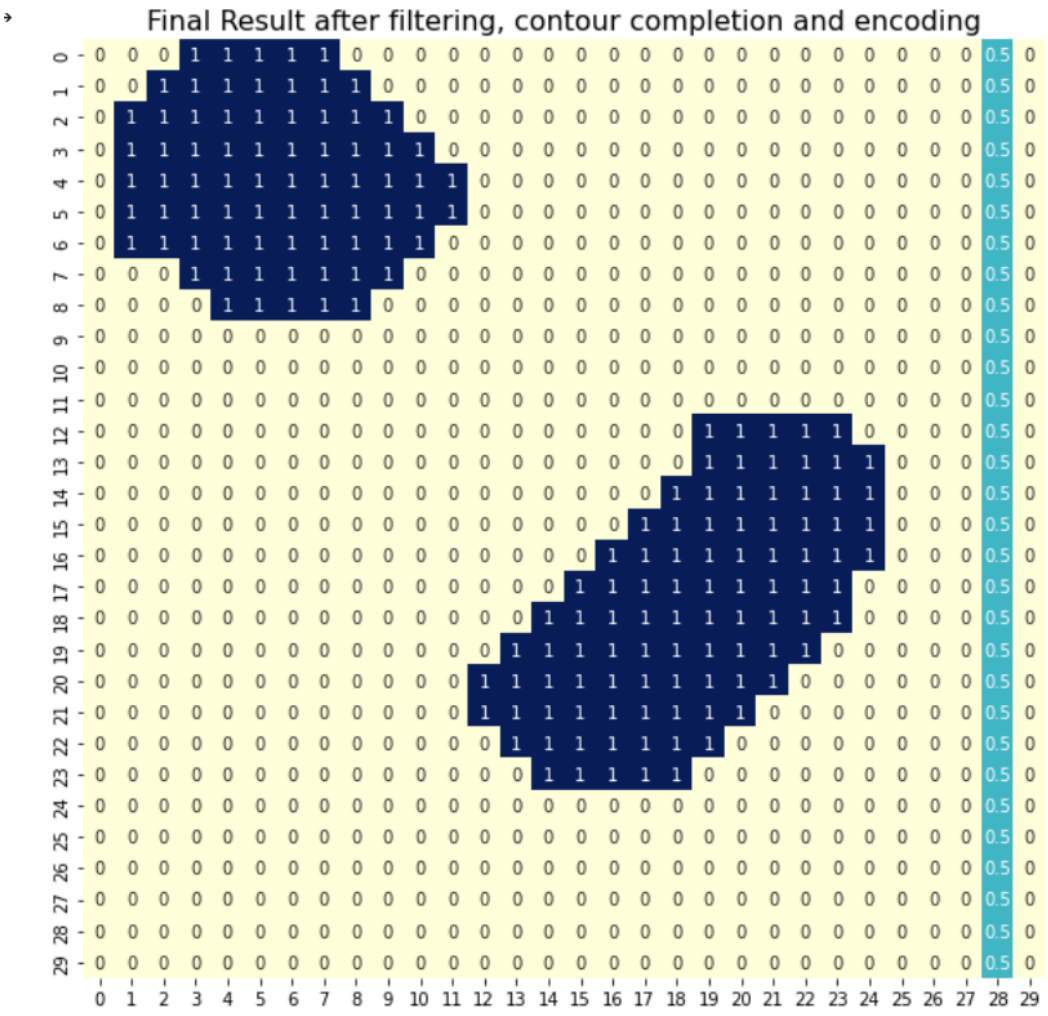


Figure 3.3: Final Result after filtering, contour completion and encoding for Data Set 3