How to extract the 5 signs

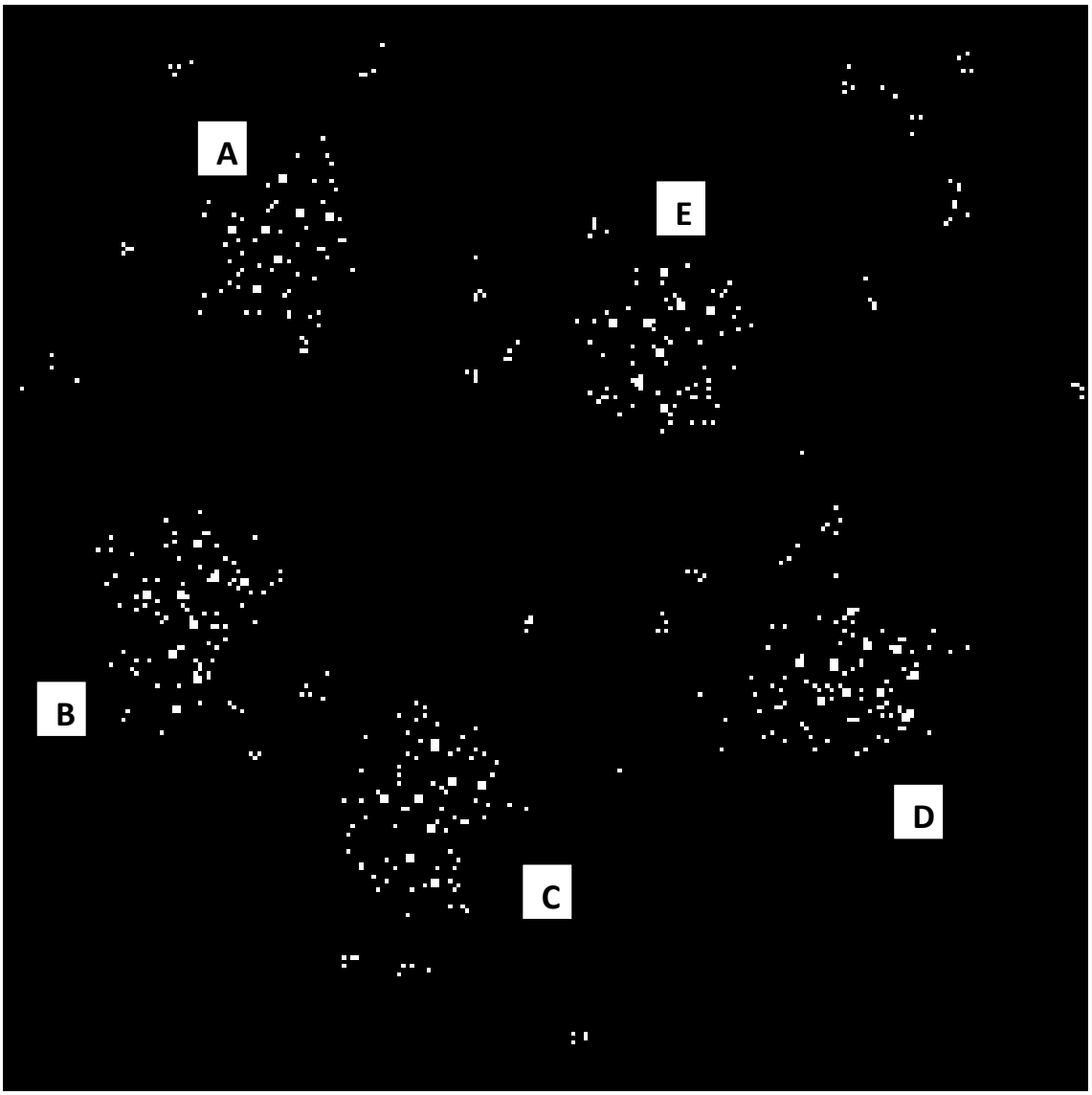
Discord user: Iokie .iokie

Presentation: Nov. 22, 2023

The message

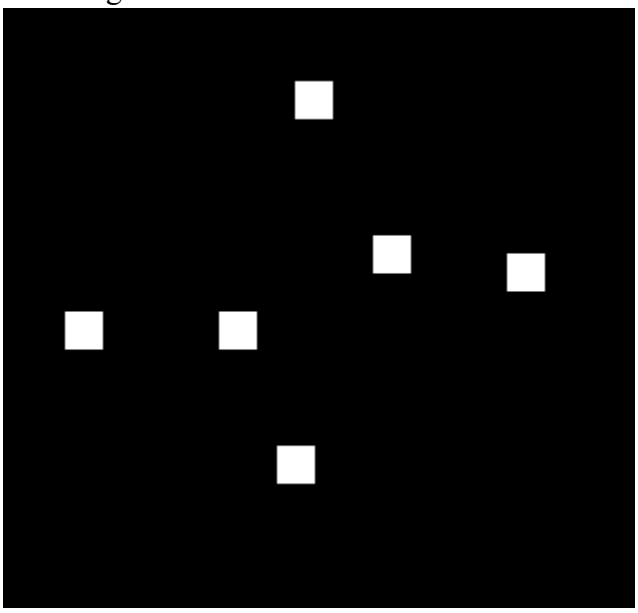
The message has been perceived as having a header, a message, and a footer. Each containing different data. The header consists of the first 80 bit. The message contains 65536 bit. The footer contains 80 bits.

This paper address only the 65536 bits of the message section. After many attempts to further interpret the data17.bin, the most coherent for the message section still appears to be the 256x256 2d image representation.



The user community has from right after the initial decode of the radio/telemetry transmission noted that there are 5 clusters in the 2d image. They were initially labelled A, B, C, D and E. A was the upper left most cluster. B cluster is the is the one immediately below the A cluster. The rest are in order proceeding counter clockwise ending with E being in the upper right quadrant of the 2d image. These are being referred to as the 'signs.'

The 5 signs also have 6 2x2 blocks with the exact same relative position in each sign.



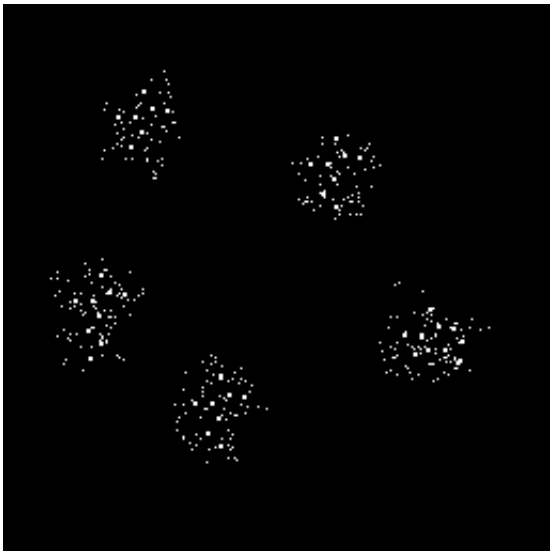
Many on Discord have speculated that this is represents a common element in all 5 signs. One of the interpretation of the header is bytes 3 and 4 has the value 40 and 65. Assuming zero based counting Row 40, Column 65 is also the coordinate of the first 2x2 (topmost) block in sign A. This has led many of us to believe that this further makes the likelihood that there are 5 separate signs and using the 6 common 2x2 blocks we can determine the relative position of each sign in the 2d image.

The problem

One of the problems is deciding which bits belong to which signs. It also brings up some questions.

1.) Are all the bits significant in the message? (i.e., are all the bits deliberate to the message or are some of the bits noise?).
2.) If they are all significant then the expectation is then every bit would belong to a sign but which one (could they be equivalent to the dot over an i or an ~ over a letter)? Showing the peripheral bits and the distance they are from various signs they are closest to brings the third question.
3.) Given the lengths a sign may be spread over then is it likely that some of the bits close to a given sign may actually be part of a different sign?
4.) The 5 signs have relative positions make a kernel or other transforms larger than 2x2 unworkable on the 5 signs without separating the signs into separate images. How do you separate the 5 signs so they all have the same alignment?

These images are from the Doug Arterburn and illustrate these questions, see https://discord.com/channels/1066055437457297469/1111404329694400542/1176554992300212425



This is one approach of many

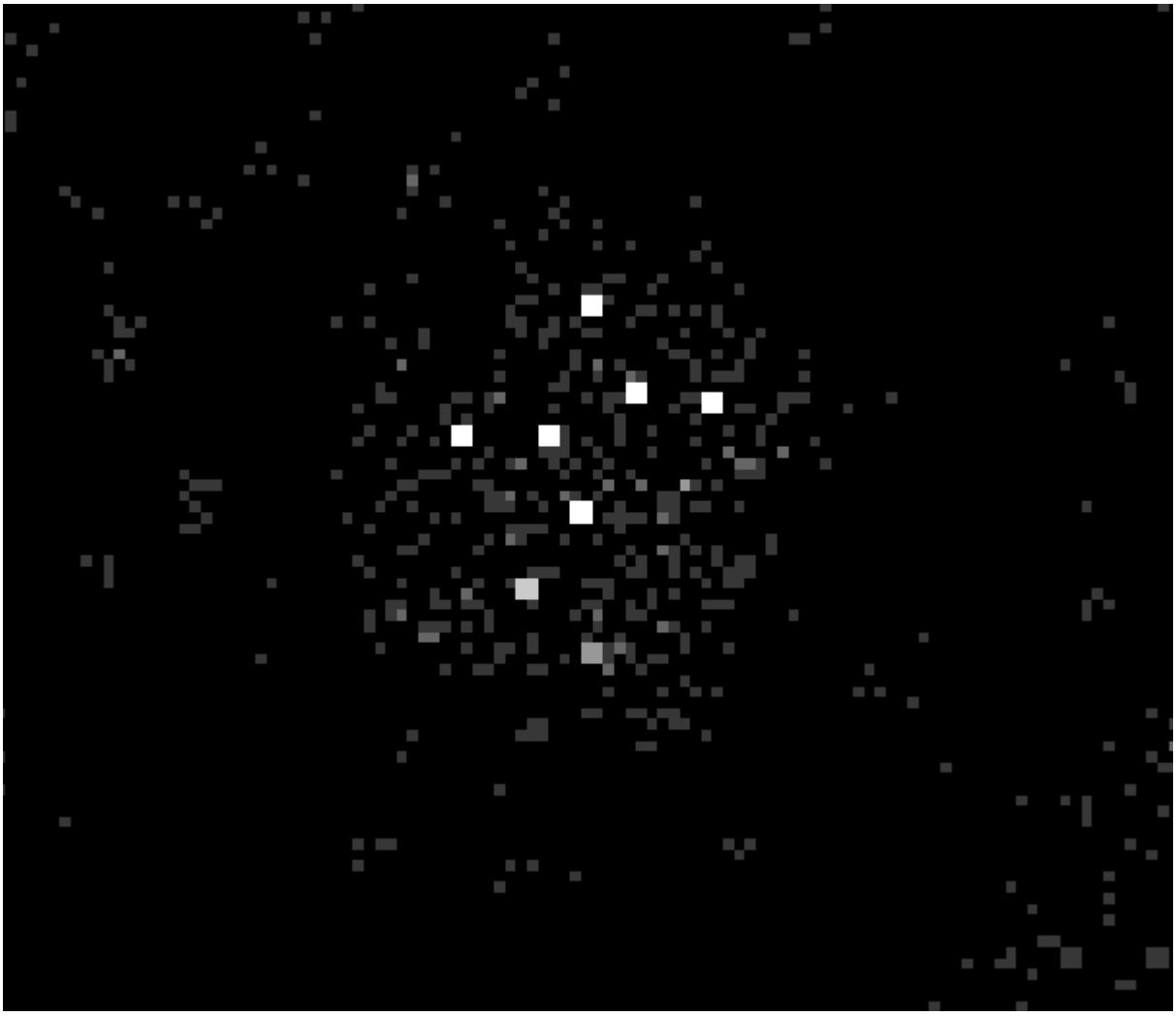
We know from the A Sign In Space team that Data17.bin is the complete message.

The following is just one of many approaches which can be followed in deciphering the message.

a.) Based the on this interpretation of the 2d image, one possibility is the 5 signs were transformed before being added into the 2d image.
b.) The signs were added to the 2d image at specific locations such that there was not an overlap of bits in the final 2d image.
c.) Each of the 5 signs may not be on the same grid alignment in the 2D image relative to the original transform.
d.) Some bits from one sign overlap the space of another sign in the 2d image.
e.) Using the 40V(vertical),65H(horizontal) coordinate of the topmost 2x2 block in sign A we could derive the relative position of each sign from the absolute position of these 2x2 blocks in the other signs. These are (zero based with the origin in the upper left corner of the image):

Sign A, 40V,65H	x,y notation 65,40
Sign B, 126V,45H	x,y notation 45,126
Sign C, 174V,101H	x,y notation 101,174
Sign D, 153V,199H	x,y notation 199,153
Sign E, 73V,155H	x,y notation 155,73

If the 5 signs are isolated, aligned and summed together, one can see that there only a few common bits between the signs. This implies that each sign is effectively different from each other. In this image the darkest grey pixels are the unique bits in each sign.



This makes the supposition that the 5 signs are unique other than the 6 common 2x2 block in each sign.

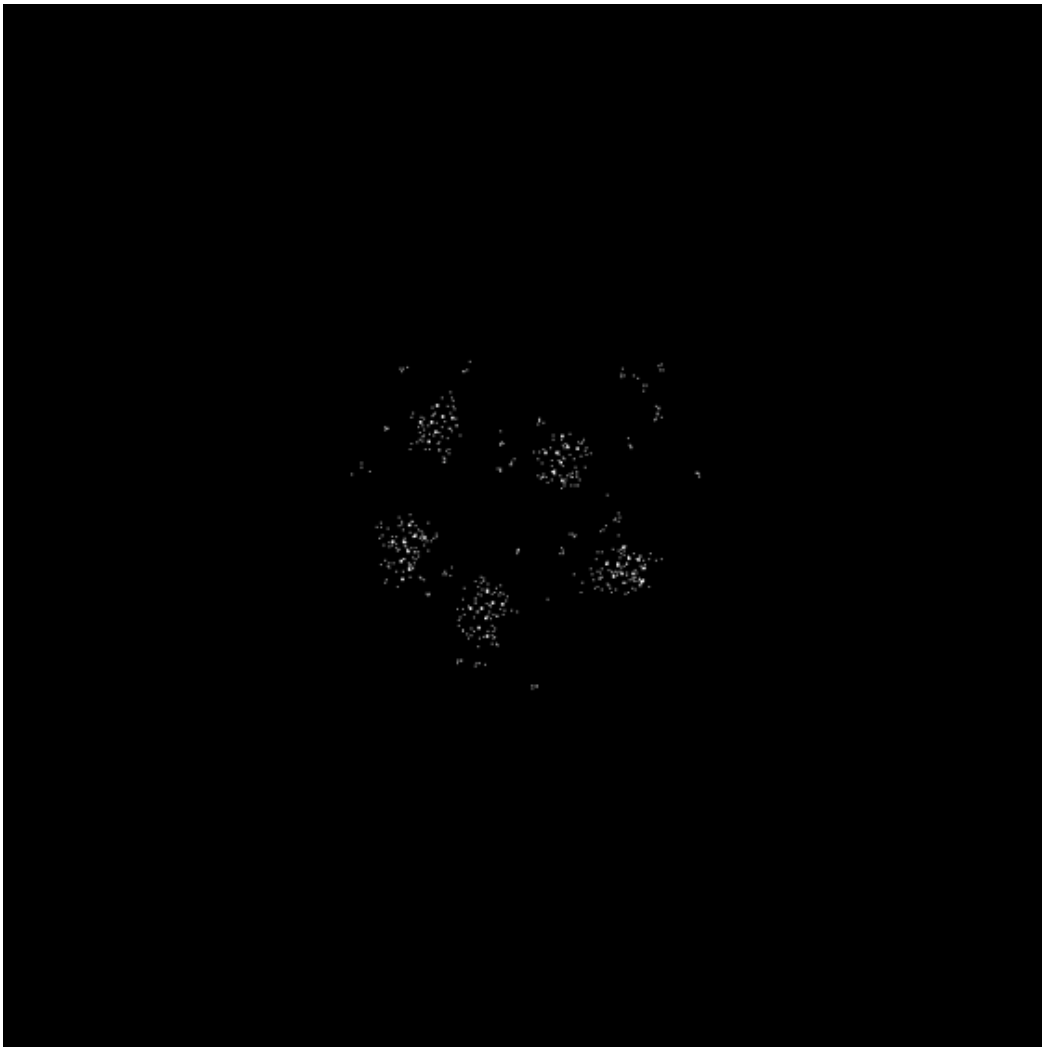
The approaches of how to solve the problems

Problem 1,2 is deciding which bits belong to which sign.
Solution: Assume any bit in the image could belong to a specific sign. This means an aligned Sign image must contain all the original bits in the image. Solving the image transformation for one sign leaves all the other sign bits as noise in the that image. Which would then be removed.

Problem 3,4 is how to separate the 5 signs so that they have a common alignment and satisfy the solution to problem 1,2. Two possible solutions should be considered.

Solution A: This assumes that the original 2d image is complete and were not wrapped on itself when adding the signs into the 2d image (solution 2.2).

Extend the original in all directions padded with 0s. So, the new image is 768x768 in size with the original image is in the center.

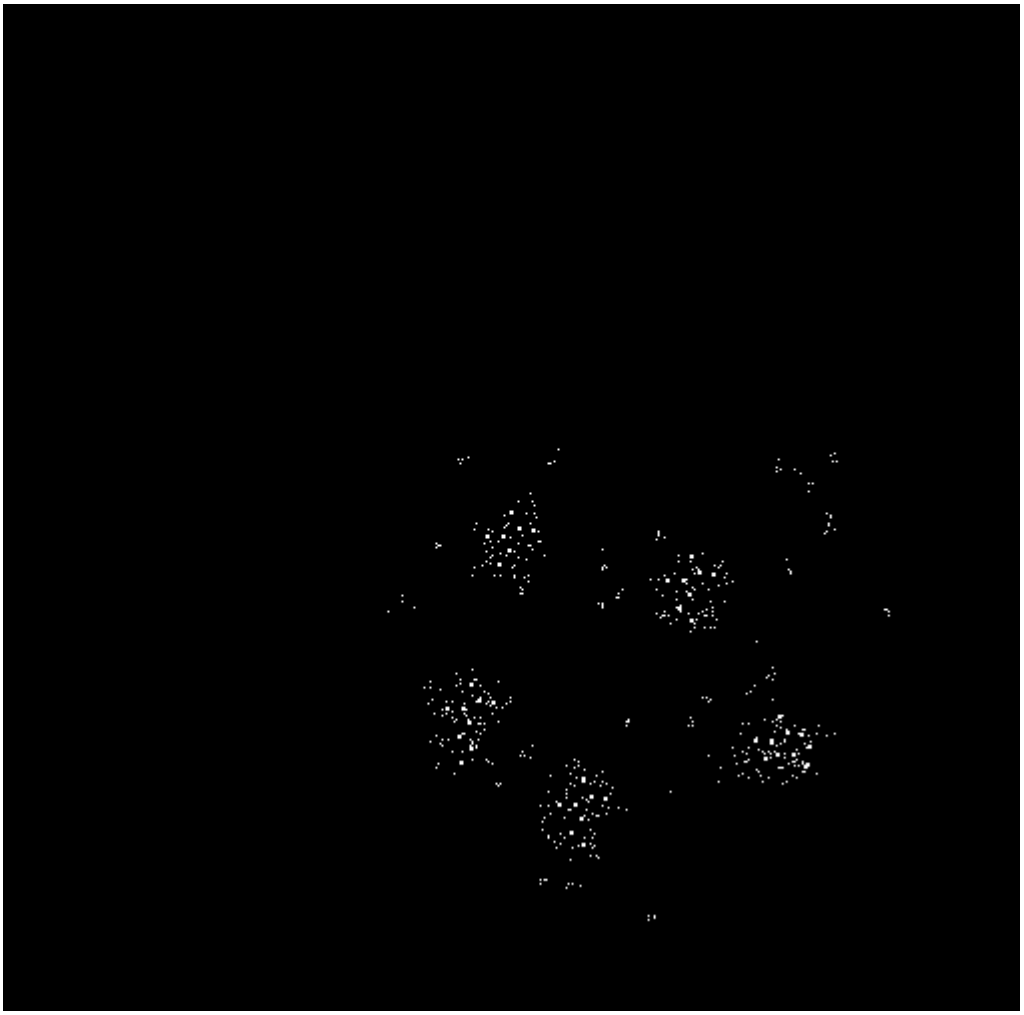


This allows a smaller 512x512 image to be extracted and centered on a specific sign without any bit loss. In the extraction process one must decide what the center alignment of a sign is used. A couple of alignments to try are (using x,y zero based notation):

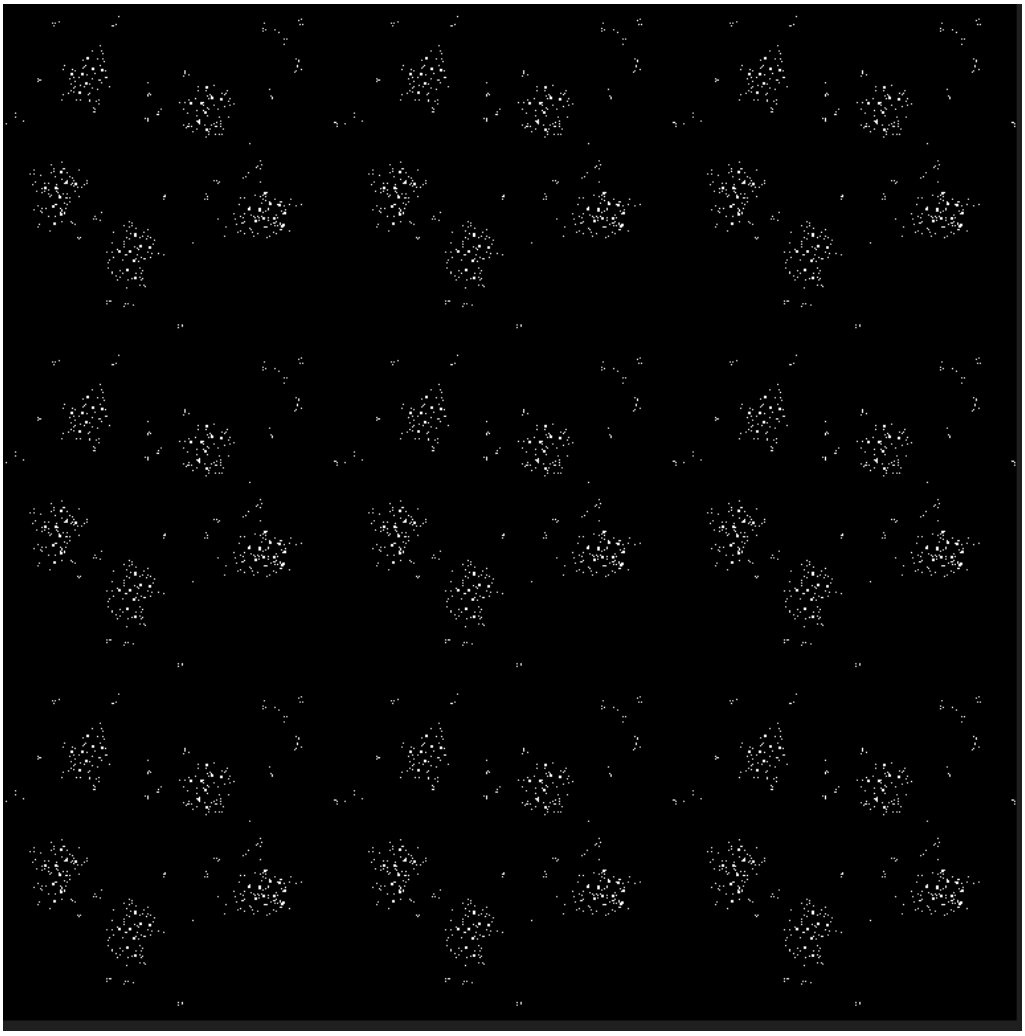
Sign	Left topmost corner of first 2x2 block in a sign	Center of the common 6 2x2 blocks in a sign
A	65,40	65,51
B	45,126	45,137

C	101,174	101,185
D	199,153	199,164
E	155,73	155,84

Example for sign A using Left topmost corner of the first 2x2 block in a sign.



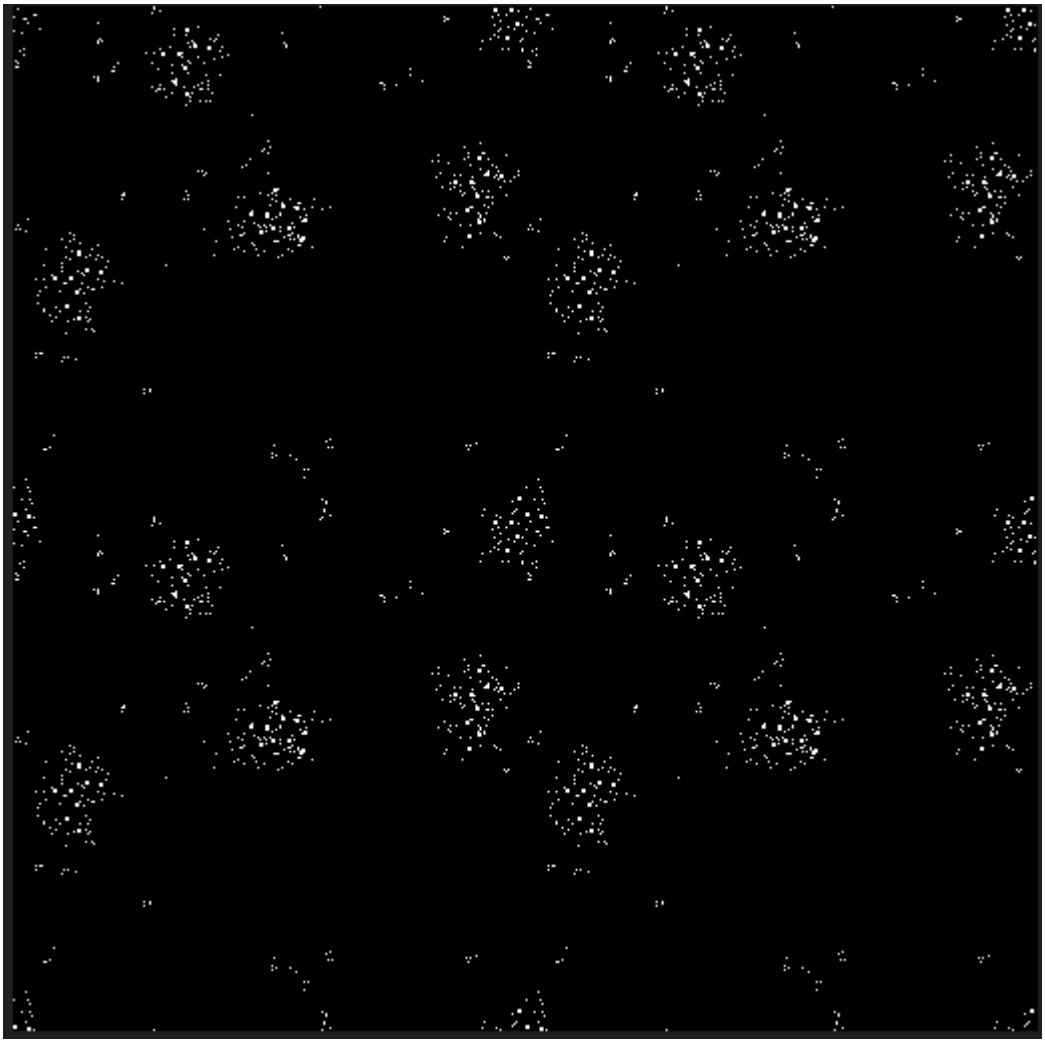
Solution B: This assumes that the original 2d image wrapped on itself when adding the signs into the 2d image. This requires appending the 2d image to itself 3x time horizontal and 3x vertically so the new image is 768x768. The original image is still in the center.



This allows a smaller 512x512 image to be extracted and centered on a specific sign without any bit loss. The bit position of a wrapped bit from a sign in the original 2d image is now in its original position before being wrapped. In the extraction process one must decide what the center alignment of a sign is used. A couple of alignments to try are (using x,y zero based notation):

Sign	Left topmost corner of first 2x2 block in a sign	Center of the common 6 2x2 blocks in a sign
A	65,40	65,51
B	45,126	45,137
C	101,174	101,185
D	199,153	199,164
E	155,73	155,84

Example for sign A using the center of the common 6 2x2 blocks in a sign



One could also reduce this to a 256x256 image but it would require that the original unwrapped sign not be larger than 12x128.

The results

The resulting images for each sign would have a common alignment with the other signs. If the 6 common blocks do represent a common element in the original signs, then it is also likely that the same transform was used on all 5 signs. This means the inverse transform that works on one sign should work on the other signs. For the purpose of testing a transform, it means one could focus on solving just one of the signs instead of all of them simultaneously. It also means that initially one could try various alignments of just one sign against a potential inverse transform.

Questions to resolve

What to select the origin in a sign to use as the center of the image. This is straightforward because it is easy to generate a variety of alignments.

What transform to apply to the image. This is the hard part. Currently many different transforms have been tried unsuccessfully. Many believe that the correct transform/rules is somehow defined or embedded in the header, footer, or the combination of the header and footer.

What if any information is also to be derived from the relative position of the 5 signs, Is it pertinent to the message?

Actions or requests from the community:
I would be happy to respond to anyone in the community to discuss this further. I can provide data files in whatever format someone would like within reason (no FITS files). Application executable for Windows and source is also available on GitHub.