# Probabilistic Coordinate Reconstruction (PCR) of Linearly Decoupled Pixel Density Regions

**Matthew Fonken**

Research and Development, Electronics Engrg., Marbl Limited, Fort Collins, CO 80521, mfonken@marbl.co

**Abstract:** This paper describes a blob structure tracking method for minimal image processing systems with severely restricted resources. Such systems can be characterized as having data throughput much smaller than that of the bandwidth of the imaging element. Through lossy hyper-compression the camera output of width $W$ and height $H$ are immediately dimensionally decoupled as to reduce the total information size from $W \times H$ to $W + H$. Significant features of the hyper-compressed image are exposed by linear filters and tracked through probabilistic analysis and Markovian prediction. The proposed method is characterized by near-linear computation and space complexity at standard image resolutions.

**Keywords:** image processing, signal processing, compression, Markov chain, battery powered

## 1. Introduction

Despite the wide range within the modern family of blob tracking algorithms, there is an implicit baseline for the minimum resource requirements due to most modern mobile processors having significant throughput. However, many hand-held devices severely restricted in size and cost can implement processors much smaller —described as "hyper-minimal" in this paper— than average image processing devices. A hyper-minimal system can be characterized as operating with less than 1mWh or occupying less than 1 sq. inch of computing and sensing logic as required by smart pens, peripheral controllers, and wearables generally much smaller than standard mobile electronics such as cellular phones.

A tenant of image processing is the relationship between space and time requirements: The minimum data bus is constrained by one or both. In other words, image entropy is more-or-less preserved along at least one dimension. Take for an image processing analogy of boxes on a conveyor belt representing frames from a camera. A particularly long box (i.e. image with significantly more information than others) can either be transported at a higher speed or turned sideways as to require a wider track. A third option is to transfer the contents of the box to smaller boxes and transfer them separately, however, the worst case in which the box is full provides no advantage. Each of these three cases are fully explored by the majority of blob tracking algorithms through bus and computation improvements, and compression and decompression techniques. Due to the nature of dramatically taxing worst cases in higher and higher dimensional information processing, even the most efficient implementations of the best modern tracking algorithms tend to spend the majority of it’s computation on limit checking if not accessing memory.

The purpose of this paper is to propose a fourth case with solution where the data buses used are significantly wider than the throughput of the processor. Within the analogy above, the technicians are recording inventory of box contents, thus transferring each down a conveyor belt can be replaced by verbal descriptions across the warehouse redefining the data bus from the belt to simply the open space between the technicians: Severe compression before transfer. Practically, this case is fully focused on reaching the most minimal entropy state of an information whether a box or an image as immediately as possible and communicating only specific-selected features that lend to successful reconstruction or understanding without transferring the information itself or any recognizably compressed version. Such lossy compression not only requires an order less memory but and order of more less computation.

## 2. Method

The proposed method follows the following cycle with five major elements, one input (Capture), and one output (Host):

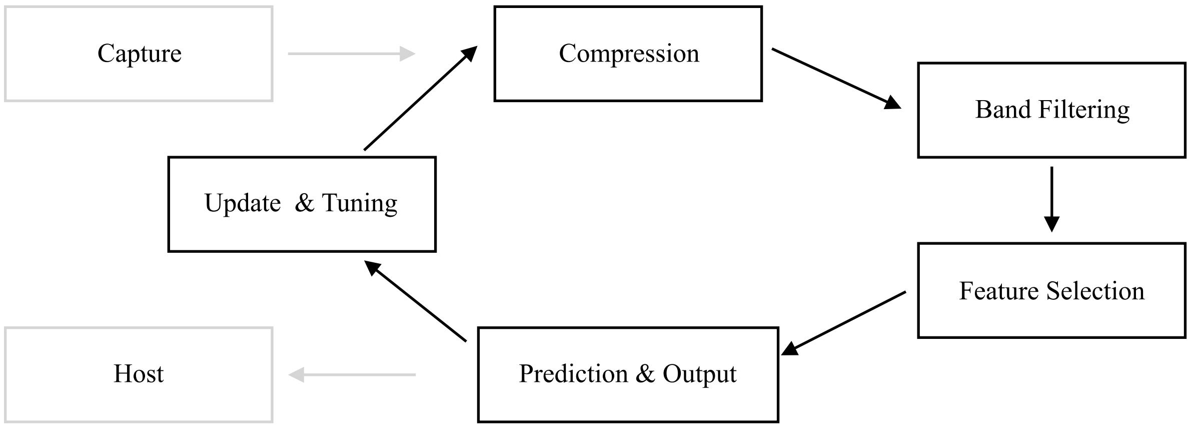

Figure 1

Due to the importance of the compression step being as physically close to the capture (i.e. camera) output possible, the system is designed to piggy-back the capture system and operate independently of the host system. Furthermore, the system is Markovian and thus probabilistic and memory-less. Because the compression element is comparatively unique to the other elements it will be described first as a background to others, although it is important to keep in mind it is the engine behind the dramatic decrease in space and computational complex in relation to common blob tracking algorithms.

### 2.1. Compression and Image Decoupling

The compression technique described here is commonly known as separation and is an implementation of dimensional decoupling. It is the severely lossy compression process in which

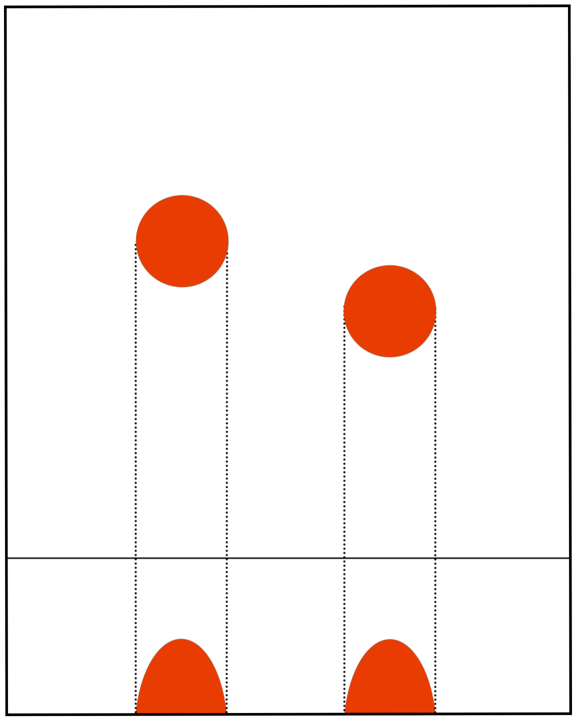

Figure 2

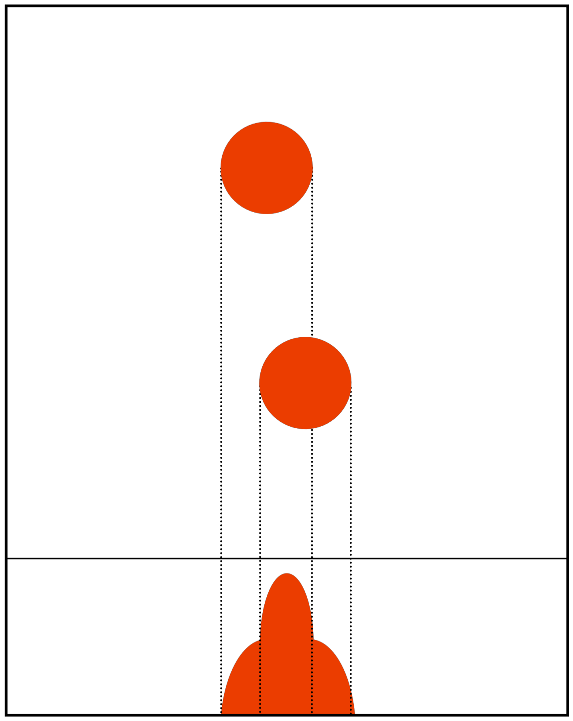

Figure 3

weighted data values (i.e. pixels in images) are projected perpendicularly on both axes generating two loosely-coupled (i.e. dissimilar yet describing the same source), single-dimensional arrays equal to the width and height of the original image as shown here:

Note: For illustration purposes, projection is demonstrated for only one axis and is equivalent for the second except for being in the perpendicular direction.

The benefit of decoupling is the remarkable decrease from quadratic to linear complexity and storage due to the trade of strong dimensional relationships (i.e. unique pixel coordinates) for weak redundancy (i.e. multiple projections of the same dataset). Unlike traditional processing methods and even other lossy compressions, decoupled data is practically irreversible: All positional relationships of features not parallel to either axis are lost initially but can be inferred using probabilistic tracking and key-features.

Note, the projections or “shadows” of image features are generally distinguishable even when represented in a single dimension and more significantly, when overlapping. Unlike physical shadows where inline objects produce a single shadow, the inline image features are stacked as to not block each other due the cumulative nature of the projection.

### 2.1.1. Density Map Generation

A common threshold filter can be used to quickly produce an accurate density map for high contrast images. Given an image of width $W$ and height $H$. For $\rho_{(x,y)}$ being the thresholded image density at coordinate $(x, y)$:

$$\mathbf{D}^X = \left\{ \sum_{x=1}^{W} \rho_{x,y} : \forall y \in 1 \leq y \leq H \right\} \tag{1}$$

$$\mathbf{D}^Y = \left\{ \sum_{y=1}^{H} \rho_{x,y} : \forall x \in 1 \leq x \leq W \right\} \tag{2}$$

where $\rho_{x,y}$ is the image density at pixel $\mathbf{p}_{x,y}$

2.1.2. Quadrant Density Calculation

This compression technique introduces significant ambiguity into the solution. Because all coupling of dimensional components is lost, a perfect solution of a system of $N$ targets without disambiguating information will produce $N!$ valid solutions. The most simple method of overcoming ambiguity without loosing the complexity reduction benefits introduced above is through the use of sectors. Similar to image blurring, sectors describe the total density of pixels in a given area with the sum of all sectors adding to the total density. Centroids can be used to define dynamic sectors stored in a matrix $S$ described generically as:

$$C^{X,Y} = C^D = \mathbf{minsort} \left\{ \begin{array}{c} i = Nm + n \wedge C_i^D = \frac{B_m^D + B_n^D}{2} : \\ m \in \{1,2,\ldots,N-1\} \wedge n \in \{m+1,\ldots,N-1\} \end{array} \right\} \tag{3}$$

$$S = \left\{ \begin{array}{c} S_{a,b} = \sum_{x=1}^{W} \sum_{y=1}^{H} \rho_{x,y} : \\ \forall a \in \{1,2,\ldots,N\} : \forall x \in \begin{cases} 1 \leq x < C_a^X, \, a = 1 \\ C_a^X \leq x < W, \, a = N \\ C_a^X \leq x < C_{a+1}^X, \, \mathbf{otherwise} \end{cases} \\ \wedge \\ \forall b \in \{1,2,\ldots,N\} : \forall y \in \begin{cases} 1 \leq y < C_b^Y, \, b = 1 \\ C_b^Y \leq y < H, \, b = N \\ C_b^Y \leq y < C_{b+1}^Y, \, \mathbf{otherwise} \end{cases} \end{array} \right\} \tag{4}$$

Here $C_i^D$ is the $i$th centroid of $\sum_{i=1}^{N} i$ centroids calculated from all pairs of $N$ blobs($B$)'s values on dimension $D$. Fortunately, few targets are required in most applications such as orientation correction or presence detection thus the following description will assume $N = 2$, creating four sectors differentiated as **quadrants** defined by:

$$Q = \left\{ \begin{array}{c} Q_{a,b} = \sum_{x=1}^{W} \sum_{y=1}^{H} \rho_{x,y} : \\ a = \begin{cases} \mathbf{left}, \, 1 \leq x < C_n^Y \\ \mathbf{right}, \, C_n^Y \leq x < H \end{cases} \wedge b = \begin{cases} \mathbf{top}, \, 1 \leq y < C_n^X \\ \mathbf{bottom}, \, C_n^X \leq y < W \end{cases} \end{array} \right\} \tag{5}$$

Quadrant data can be efficiently calculated during density map generation by incrementing a selected centroid by index and using basic boundary checks and counters to increment that index. The addition of tracking quadrant data for standard image resolutions minimally (less than 10% in tested cases) increases the complexity for only active pixels, thus outweighing the cost of unavoidable 50% ambiguity the quadrants solve. Analytically, a well-filtered binary image with two blobs will have around 5% or fewer active pixels —even test cases with dimensions 1280x800px have shown less than 1% for controlled environments— thus will increase complexity by:

$$100.5\,\% = 110\% \times 5\%_{active} + 95\,\%_{inactive} \tag{6}$$
$$\text{or a minimal } +0.5\,\%$$

Even a horrible case of a taxing quadrant calculation of twice the operations over an extremely noisy, 50% active image will increase complexity by:

$$150\,\% = 200\% \times 50\%_{active} + 50\,\%_{inactive} \tag{7}$$
$$\text{or } +50\,\%$$

This concludes the compression description producing projected density maps of both image dimensions as well as a quadrant density matrix. The following is a technique for reconstruction the two-dimensional coordinate data of blobs in the original image from these maps.

## 2.2. Band and Noise Filtering

A significant characteristic of image decoupling is that not only is target data essentially duplicated but so is noise thus requiring much greater inference and environment-specific tuning than more common tracking techniques. In order to extract and reconstruct target data in the following description, noise is subcategorized into three families: 1) static or environmental noise, 2) dynamic or animate noise, and most unique to decoupling 3) shadowing noise. Because a significant amount of information is lost during compression, rebinding is inherently instantaneously obscured or shadowed. For a two point system, the best case where only two points appear in the frame on a diagonal is a single rectangle where two of four points or 50% are accurate.

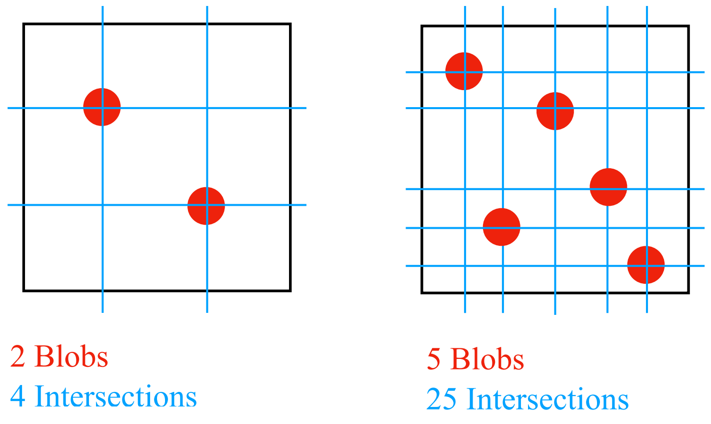

Figure 4

Generic best accuracy for $N$ blobs is:

$$\frac{N}{N^2} = \frac{100}{N}\% \tag{8}$$

Fortunately these inaccuracies are instantaneous and can be overcome using time-based probabilistic filters. Furthermore, because the camera is physical and mounted on a moving structure, filters such as the Kalman filter that can implement physical models such as velocity and acceleration can be implemented to accurately predict proper values even in worst case frames.

Similarly, most environmental light noise areas translate predictably across the frame as the camera or light source moves and can be filtered separate of unpredictable or dynamic noise. Individual noise sources can be distinguished by characteristics under small time-scales and minimal motion. Changes in static sources are negligible compared to that of dynamic ones.

### 2.2.1. Static background subtraction

Image elements can be described as instantaneous binary layers that tend to retain superposition through image compression. For high frame rates where changes in environmental noise become time-insignificant, noise can be filtered statically through binary subtraction. More significantly, because noise is retained through density compression, independent noise snapshots can be created and subtracted post-compression at $O(N)$ cost rather than the $O(N^2)$ cost of standard processing.

Effective background images require environment noise-isolated reference images. This can be achieved through direct electronic control of the target sources where the targets are quickly powered off and on synchronously with the image processing source either by trigger or timer period. During this "backgrounding" event while the target sources are off, an environmental noise snapshot can be generated and stored through normal density compression. While the environmental noise is not completely isolated, the configuration of high-rate backgrounding — in other words instantaneously low-significance backgrounds— and an environment of low-probability dynamic, significant external noise events allows this method to be highly resilient.

Static background subtraction is essential for balancing the density distribution of the image by 1) minimizing large density areas that shadow target areas, and 2) eliminating target-like noise areas. Practical application of this method should allow for a large radius range between the target source(s) and the camera and therefore a large range of visualized target coverage. Because the physical filters described below search for most significant areas converging on tallest and oldest peaks and thus large noise areas are always rewarded. Therefore the goal of the described background subtraction is to minimize noise density coverage to a level less than target coverage.

### 2.2.2. Variance band filtering

Connected regions or blobs within density maps can be described as density given a specific location on a single axis, a collection of connected densities on that axis, and time-persistence across frames. Successive static background subtraction and threshold tuning removes unwanted elements while leaving the target data as described in these three dimensions unchanged. The result is peaks in the density maps whose centroids hold the final target coordinate data. A physical filter is used to extract the centroid data by creating a variance confidence band using the vertical information of the peaks. This band can be used to filter two major sources of error: 1) dynamic environmental noise —noise that changes faster than the backgrounding filter described above— and 2) target shadowing which occurs regardless of noise. The following variance band method presents a solution through 1) peak finding, 2) peak filtering, 3) weight redistribution, 4) centroid calculation, 5) chaos assessment, and 6) coordinate prediction.

The density map peaks can be described generally as physical growth and decay and thus easily tracked using a basic Kalman filter. While pixel densities produced from visualized light sources do not following physical constraints and can be switched on and off, the physical nature of the camera carrier however can be expected to. In an environment of generally static light sources, a Kalman filter embedded with a velocity model can not only tracking the peak with a confidence value, but also produce a reliable prediction for the next frame. These two key values: the confidence value and prediction, create a second thresholding filter mirrored over the peak value creating band.

$$\Delta K = \left| K_T - K_A \right| \tag{9}$$

$B_n = B_{\varnothing}^{+}$, where $B_{\varnothing}$ is the minimum or "calm" band

$$\gamma = \frac{P_{n|n-1}}{P_{n-1|n-1}}, \tag{10}$$

$$B_n = B_{\varnothing}\left(1 + \gamma \Delta K\right) \tag{11}$$

### 2.2.3. Peak finding

For the purposes of this method, only a segmented maximum finder is implemented for peak finding. Whereas additional filters such as smoothers or differentials are often added for peak finding, the variance band filter has been shown to be resilient against ambiguity by itself.

In a system detecting $N$ targets, density maps as split into $N$ segments each with a single maximum. Segments are generated using post-filter density distributions during the prediction stage in the X and Y dimensions in valleys between peaks splitting the density map into localized regions. Local maximums in each region grow and shrink independently, representing the presence of the significant object in that region. Again, this maximum doesn't describe absolute horizontal location only presence as it lies within the range of the significant object of that object. Because the density map represents the cumulative density in a single dimension, peaks detected from wide objects —density distribution parallel the compression axis— tend to poorly describe the absolute centroid of the target object directly. The peak, however, does suggest total density of the object and thus describes the time-consistency of the object which then contributes to detection confidence.

### 2.2.4. Physical peak filtering and path variance

Confidence is an essential element of physical filtering, optimization, and convergence. Physical filters such as a Kalman filter not only produce a prediction but also time-based probability tuning defined by behavior expectations. Filtered systems that follow the expected behaviors will produce higher confidence then ones that are sporadic even though both may produce the same value. In this method, confidence is used to further minimize operated data by performing a modified thresholding process called a variance band threshold.

Consistent systems are either ones with minimal noise and movement and thus a high confidence or ones that are changing but have minimal acceleration and thus a growing confidence. For this description, variance describes the inverse confidence of a system's physical filter and is linearly scaled by static tuning values. The variance band is a density value range typically symmetric above and below the peak value with a scaled variance defined width. As a threshold, it ignores values both above and below the band. As confidence grows, the band and thus data to process shrinks.

Figures 5 through 8 illustrate how the variance band isolates data in steps. Figure 5 shows two identical targets separated with peak filter in a nearly recovered state —this causes the offset from the true peak— and Figure 6 shows the same targets entering an overlapping shadow state where the peak filter still identifies the two targets despite forming a single observed peak.

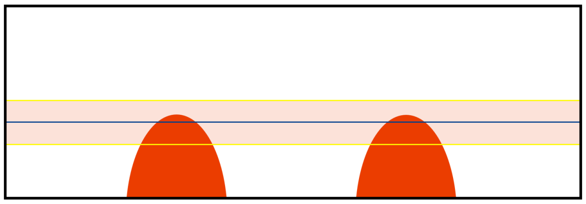
Figure 5

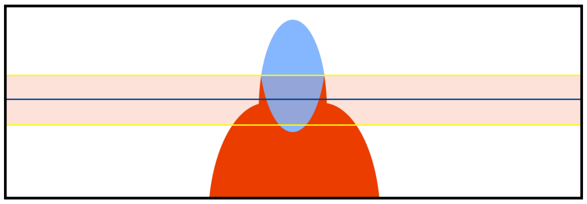
Figure 6

In Figures 6, the light blue area is the density over the band center flipped vertically and removed. The double-sided nature of the variance band allows for quick shadow handling. Shadowing events where independent objects suddenly overlap cause peaks to grow more rapidly than the filter predicts and thus can be generically characterized as significant density values above the variance band. These overflow values are used to punish that portion of data in the band through subtraction. A significant punishment will generate a valley lower than the lower bound of the band, shown in purple, thus split that density peak in two, however only until the physical filter recovers or the shadowing event ends when the observed objects separate on that axis.

It is important to note that the shadowing event may persist longer than the filter takes to recover thus the filter will falsely assume one large object instead of two. This is the most likely worst case for reconstructing two-dimensional data from density map compression and has an error related to the size of the visualized target objects, however, is still convergent to a location within the width of the visualized target. In order to reduce error caused by this case, a typical application will minimize the target object size and orient the targets diagonally from the camera’s axes.

## 2.3. Blob analysis and data extraction

Once the variance band is generated, the density areas in the band are processed as blobs whose centroids can be found using physical density equations. The goal of blob analysis is to generate both centroid values and corresponding probabilities based on an expected target coverage area. In an *N*-target system the *N* most significant blob will be selected for prediction filtering while all others will be accumulated together as a single *alternate* blob and probability.

The alternate blob can represent a combination of many elements in the system: 1) target-like noise that may to be able to become the target prediction in a small number of frames, 2) proper threshold tuning where the physical peak filter is simply recovering and banding incorrectly for a small number of frames, and 3) threshold tuning undershoot that will indefinitely restrict confidence. Because the first two are handled by the filtering method described above, the last one is best handled by system state tuning and is described in detail in the section 3.3. Bayesian State Control.

### 2.3.1. Band centroid detection

While blob centroid detection is a well-established area of image-processing especially in engineering, the density map variance band method eliminates a significant amount of processing required even from popular techniques due to the band being processed in a single pass. This is possible due to the single-dimensional image compression forcing all insignificant values to be eliminated from the band leaving only valid density values. This allows high confidence or thinner bands to perform at true O(*N*) complexity. Figures 7 and 8 illustrate how the blobs generated from the variance band and peak filter are processed, show in red, to find centroids, show as dark blue vertical lines.

A typical centroid detector accumulates perpendicular density values of a blob and averages each

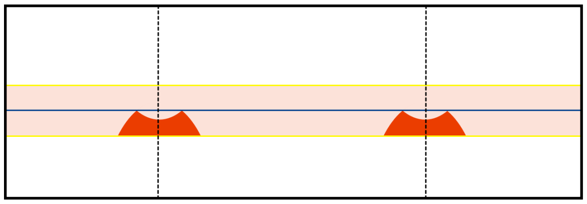
Figure 7

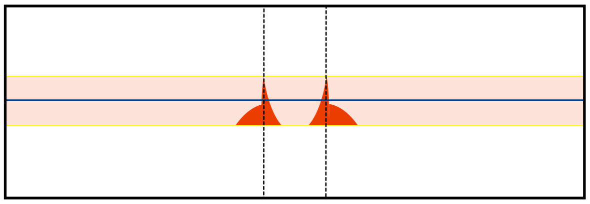
Figure 8

by parallel location. Because the image compression already accumulates the density data both horizontally and vertically as density maps, only averaging must be performed within the band to produce centroids. Furthermore, because the split density maps loosely describe the same objects, the resulting centroids describe horizontal and vertical location values of the original objects yet can be and are already commonly processed independently.

A common density map centroid processor will be configured with a blob gap value that describes the minimum distance between blobs. The detector scans the banded density map once and calculates the centroid using a common moving average function from which maximum and total density can be stored along with the final centroid values of each blob in the band. The most significant are typically the most dense.

2.3.2. Cycle Density Map

Once the band is selected, the density map must only be cycled through once to detect and identify connected density regions or blobs by significant features such as mass, width, centroid or moment, and target correlation or offset. Given a unfiltered density map $D$ and filtering band bounded between $B^{-}$ and $B^{+}$ inclusive with center $B^{C}$, the proposed blob detection technique is as follows:

*Filter/prepare density map*

The raw density map received from compression may require scaling, filtering, or background removal before processing. An example is a basic element-wise is

$$D' = \left\{D_x - B_x : \forall x \in 0 \leq x < L\right\}, \text{ where } L \text{ is the length of } D \tag{12}$$

*Correct overflow*

As described above, overflow above the upper band is inverted as to create a notched blob. Significant overflow generates notches deeper than the band height and thus splitting a single blob into two.

$$D^{B} = \begin{cases} B^{+} - \left(D' - B^{+}\right) = 2B^{+} - D',\ D' > B^{+} \\ D', \textbf{otherwise} \end{cases} \tag{13}$$

*Accumulate connected regions*

Both blob width and separation or consecutive positive or zero banded density map values can be calculated using a single binary switch and counter where consecutive positive counts are notated as $C_{+}$ and zero counts as $C_{\emptyset}$. To account for sparse sets as well as subsampling, blob separation is defined by a gap threshold, $\mathbb{G}$. Cycle variables are blob width $w = C_{+}$, cumulative density $M_n^0$, cumulative moment $M_n^1$, global maximum $P^{max}$, and blob list:

$$\mathfrak{B} = \left\{\mathfrak{B}_1, \ldots, \mathfrak{B}_N\right\}, \text{ where}$$
$$\mathfrak{B}_i = \begin{bmatrix} \textbf{density} = \rho & \textbf{width} = W \\ \textbf{centroid} = C & \textbf{offset} = \Delta P \end{bmatrix}_i \tag{14}$$

A detection is concluded by:

$$\left(C_{\emptyset} > \mathbb{G} \textbf{ or } x = L\right) \textbf{ \& } C_{+} > 0 \tag{15}$$

or a significant gap or end of cycle combined with at least of detection. On a detection event, the current blob is pushed into the blob list, the index $i$ is incremented, and $C$, $w$, $M^0$, and $M^1$ are reset to 0. All other positive banded density map values increment the current blob's width:

(16)

$$w_x = w_{x-1} + 1, \tag{25}$$

Step the density and moment using standard cumulative average equations:

$$M_x^0 = M_{x-1}^0 + \frac{\Delta^B D_x^B - M_{x-1}^0}{w_x} \tag{17}$$

$$M_x^1 = M_{x-1}^1 + \frac{\Delta^B D_x^B x - M_{x-1}^1}{w_x}, \tag{18}$$

And the global maximum is increased if exceeded:

Blob values are then calculated as

$$\mathfrak{B}_i = \begin{bmatrix} \rho = M_n^0 & W = w \\ C = \frac{M_n^1}{M_n^0} & \Delta P = B^C - D^B(C) \end{bmatrix} \tag{19}$$

From here on superscript notation is used to identify blob parameters, for example: $\mathfrak{B}_2^\rho$ represents *the density of the second blob*.

## 2.3.3. Calculate Total Density

The total density is calculated as:

$$\rho_n = \sum_{i=1}^{N} \mathfrak{B}_i^\rho, \text{ where } N \text{ is the number of detected blobs} \tag{20}$$

## 2.3.4. Score Blobs

Before blobs are selected, they are scored and sorted. Blobs with poor scores above the chaos boundary are reprocessed with a tighter band. The chaos boundary $\epsilon$ is calculated as:

$$\epsilon = \frac{\left|\rho_n - \rho_{n-1}\right|}{\rho_n} \tag{21}$$

Blob score $\mathfrak{B}^S$ is calculated for each blob as:

$$\Delta\mathfrak{B}_i^\rho = \frac{\mathfrak{B}_i^\rho}{\rho_n} - 50\,\% \tag{22}$$

$$\mathfrak{B}_i^S = \sqrt{\left(\mathfrak{B}_i^{\Delta P}\right)^2 + \left(\Delta\mathfrak{B}_i^\rho\right)^2} \tag{23}$$

Blobs with a score above chaos boundary

$$s_i > \epsilon \tag{24}$$

Rerun CMA across the blob's approximate width with raised lower band

$$B^- = P_i - \Delta P_i$$
$$\mathfrak{B}_i^C - \frac{w}{2} \le x \le \mathfrak{B}_i^C + \frac{w}{2} \tag{26}$$

All new blobs are rescored without retry. The blob at the current index is updated while new blobs are appended to end of list. After resorting the blob list by score, poor ones can be cut and the number of possible targets $\nu$ on each dimension $D$ can be estimated number:

$$\nu_D = \lambda_{\%}\rho_{\%}N_B \tag{27}$$

2.3.5. Match Blobs with Kalmans

1. Each live Kalman within the filter match vector is incremented to a stale a priori state using the previous velocity.

$$\mathbb{K}(\Delta t,\ \mathbb{K}_V)_{n-1|n-1} \to \mathbb{K}_{n|n-1}, \text{ where } \mathbb{K}_V \text{ is the Kalman's velocity value} \tag{28}$$

2. A blob-filter match matrix is generated using these predictions as:

$$\forall\ \textbf{odd}\ m \in 1 \leq m < \left|\mathbb{K}\right|\ \textbf{and}\ \forall\ \textbf{odd}\ n \in 1 \leq n < \left|\mathfrak{B}\right|: \tag{29}$$

$$\Delta\mathfrak{B}_{m,n} = \begin{bmatrix} \left(\mathbb{K}_{2m} - \mathfrak{B}^C_{2n}\right)^2 & \left(\mathbb{K}_{2m} - \mathfrak{B}^C_{2n+1}\right)^2 \\ \left(\mathbb{K}_{2m+1} - \mathfrak{B}^C_{2n}\right| & \left(\mathbb{K}_{2m+1} - \mathfrak{B}^C_{2n+1}\right)^2 \end{bmatrix} \tag{30}$$

$$\mathbb{K}_{\textbf{update}} = \left\{ \begin{matrix} \mathbb{K}_{2n}(a) \wedge \mathbb{K}_{2n+1}\left(b\right)\ : \\ \forall a \wedge \forall b \in \begin{cases} a = B^C_{2n} \wedge b = B^C_{2n+1}, \left|\Delta\mathfrak{B}_{m,n}\right| \leq 0 \\ a = B^C_{2n+1} \wedge b = B^C_{2n}, \left|\Delta\mathfrak{B}_{m,n}\right| > 0 \end{cases} \end{matrix} \right\} \tag{31}$$

3. There are two post-update cases:
    1. More live Kalman filters than blobs detected:

$$\left\{\textbf{punish}(\mathbb{K}_i) : \forall i \in n+1 \leq i < \left|\mathbb{K}\right|\right\} \tag{32}$$

    2. More blobs detected than live Kalman filters:

$$\left\{\textbf{add}(\mathbb{K}_i) : \forall i \in m+1 \leq i < \textbf{min}\left(\left|\mathfrak{B}\right|, \left|\mathbb{K}\right|_{max}\right)\right\} \tag{33}$$

4. Finally the two highest scored Kalman filter values are averaged to calculated the estimated target centroid $C$ on dimension $D$:

$$C^D_n = \frac{\mathbb{K}_1 + \mathbb{K}_2}{2} \tag{34}$$

## 2.4. Selection

The goal of the prediction processor is to recouple centroid values into likely pairs. As described in the density compression description, this step is crucial for allowing the significant performance increase made possible by three key processes: 1) Density quadrant cumulation and redistribution, 2) a second phase of physical filtering, and 3) probabilistic states. The third is described below and is the convergent feedback loop that tracks the discrete system state based on the probabilities described above. The first two are described in the next two sections requiring a preliminary description of quadrant density tracking.

Figure 10 illustrates how centroid data creates ambiguity. The red circles represent the original density areas while the blue circles represent completely valid alternate solutions for re-coupling. The yellow lines represent the vertical and horizontal frame centroid values of the previous frame.

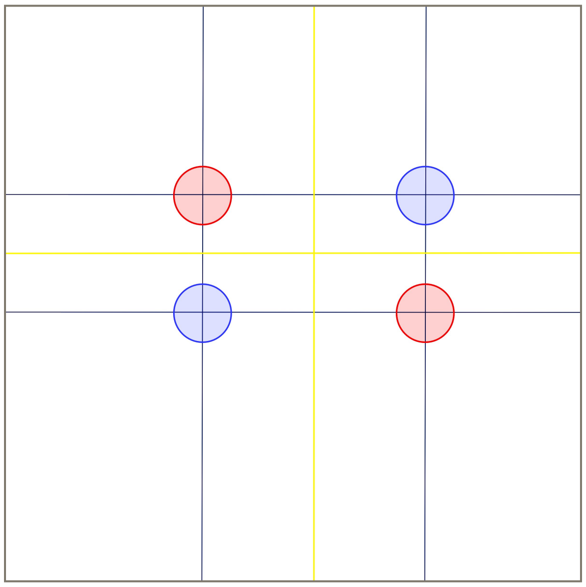
Figure 9

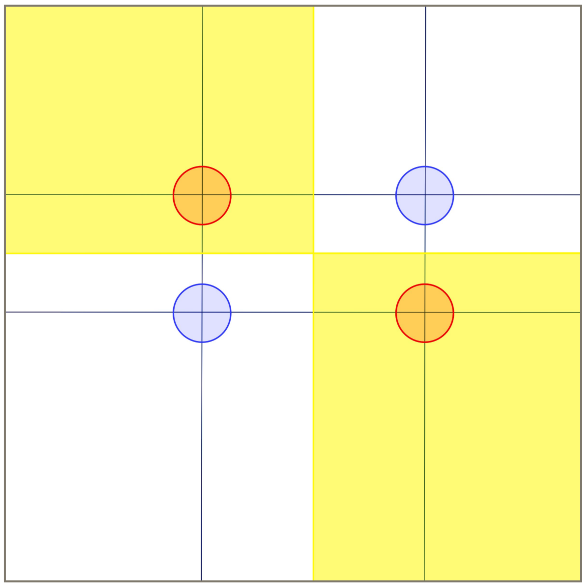
Figure 10

### 2.4.1. Quadrant-Based Ambiguity Compensation

Without further data, 50% ambiguity is the absolute maximum of density map re-coupling which is unusable. To minimize this, quadrant data is gathered during compression. Quadrants can be described by single vertical and horizontal splits of the original frame based on the density center of the previous frame. This density center can be the average of all detected blobs or simply the most significant blobs based on the expected behavior and environment. Because shadowing causes even higher ambiguity, tracking even simple two-dimensional information of the original image during the initial compression —i.e. cumulate density of each quadrant— can significantly improve accuracy and speed during re-coupling. Figure 9 illustrates how even simple quadrant values shown as the intensity of the yellow allows for total confidence during re-coupling.

### 2.4.2. Quadrant Redistribution

This is perfect case without background noise demonstrating 100% confidence. The quadrant values are calculated from the raw image before filtering and thus must be redistributed to account for filtered values. Because quadrant data is inherently designed to be a minimal description, redistribution can be performed efficiently by generating roughly re-coupled background quadrant information using uncoupled background density maps after filtering and subtracting them from the observed raw densities. Figures 10-17 illustrate how quadrant data of static backgrounds are redistributed. The background quadrant information can be generated similar to the two-dimensional generation by splitting the backgrounds by their corresponding combined centroid value and calculating the cumulative densities on either side. Matrix multiplying the two X and two Y densities generates roughly coupled quadrant information of the background.

Figure 11 illustrates simple quadrant data from a normally processed static background. The center cross of the background is the frame centroid at the instant the background is generated and is saved with its quadrant data. Figure 12 illustrates observed quadrant data from a normal frame with the static noise area present. Because the background's centroid is offset from the current centroid and the quadrant values are not location specific, the values must be refactored or "redistributed" before being subtracted.

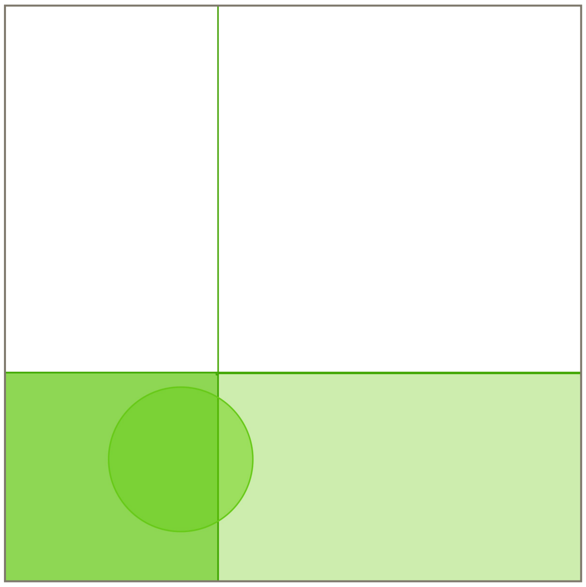
Figure 11

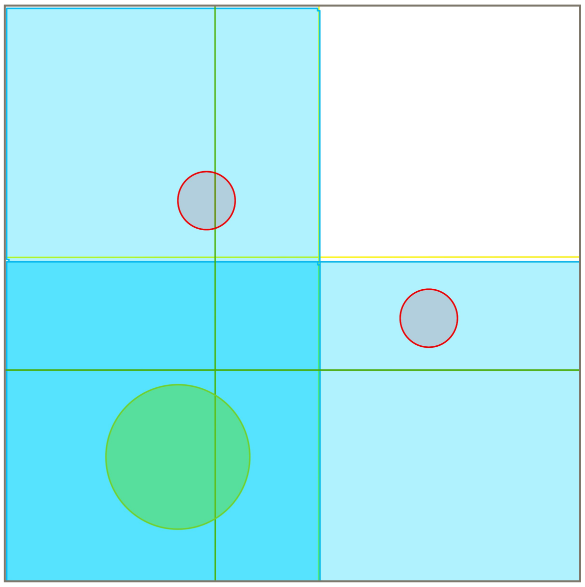
Figure 12

Figures 13 visualize the sub-area's caused by the offset. Similarly Figures 14-16 further highlight the three significant overlaps between the background and the current frame's quadrant data. Overlap is computed using the centroid differences and frame dimensions. For example, the green background quadrant in overlaps about 70% more of the blue current frame quadrant in Figure 16 than it does in Figure 15 thus this background quadrant value will be redistributed 30:70 across the axis of overlap. Note this example is designed for minimal overlap for convenience, however, the method of linear redistribution can be easily performed for each quadrant. See Appendix C for a detailed mathematical description.

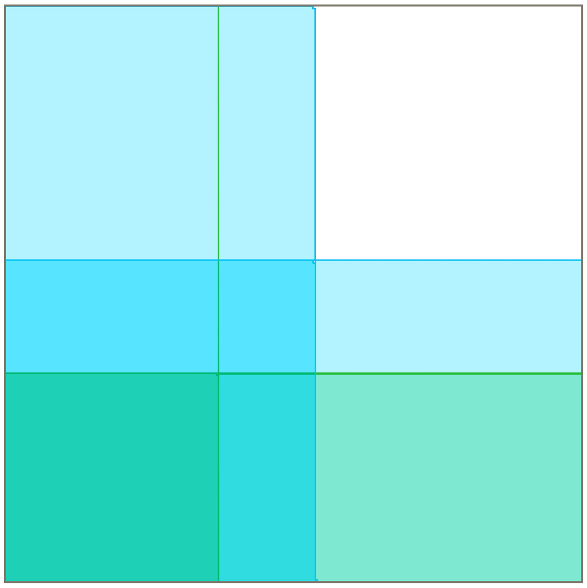
Figure 13

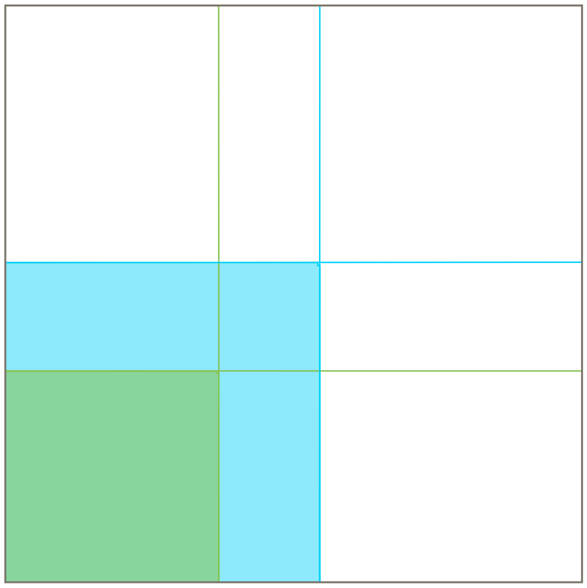
Figure 14

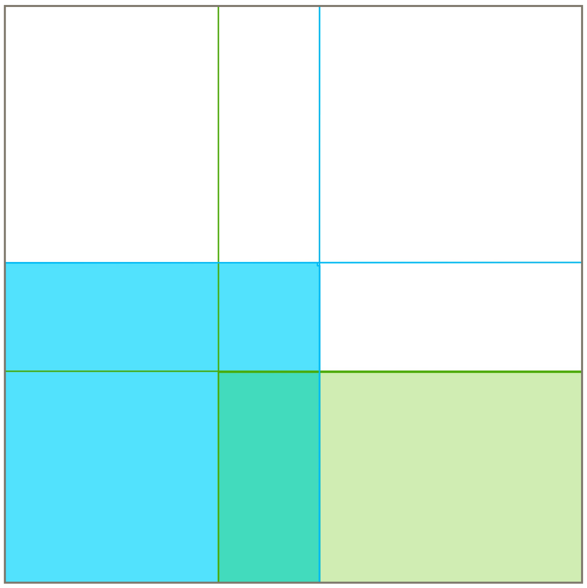
Figure 15

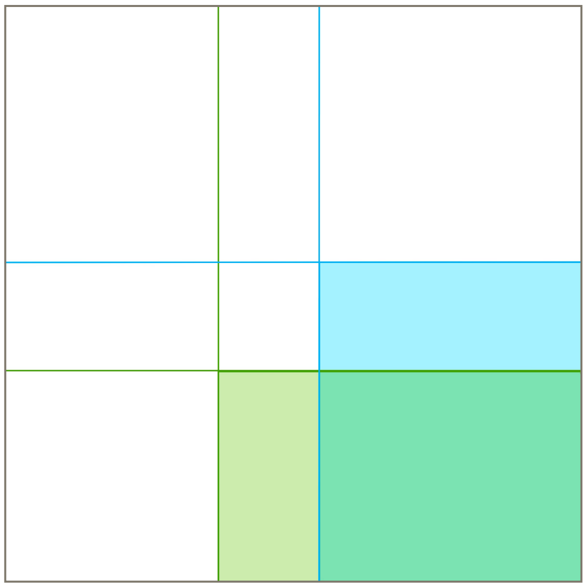
Figure 16

Redistribution using the background and relative centroid offsets generate the quadrant values as shown in Figure 17. Like the clean frame in Figure 10, re-coupling can be performed easily as shown in Figure 18. Note that a faster background refresh period causes less offset between the background centroid and current frame centroid. While increasing the load, this means less redistribution and therefore less ambiguity during re-coupling.

O($N$) complexity is preserved as this method requires only a single pass through the vertical and horizontal background maps during the backgrounding event and the calculations of the nine overlap areas during normal prediction processing. While normal re-coupling is unlikely to regain 100% accuracy due to the lossy nature of the density compression, quadrant redistribution forces the remaining ambiguity of the method described above to converge consistently on at least one two-dimensional density area.

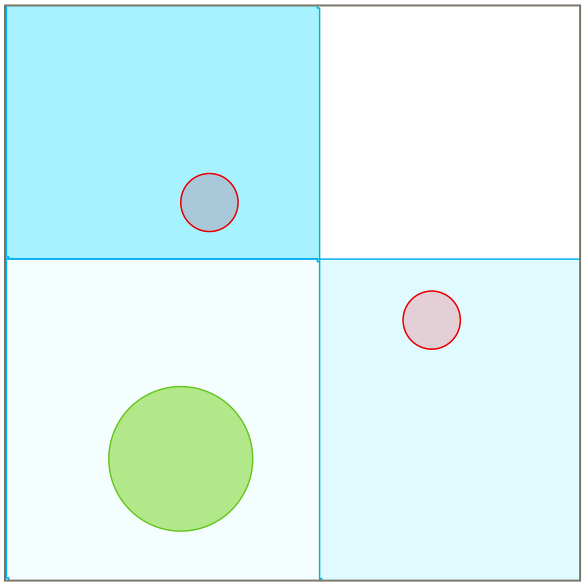
Figure 17

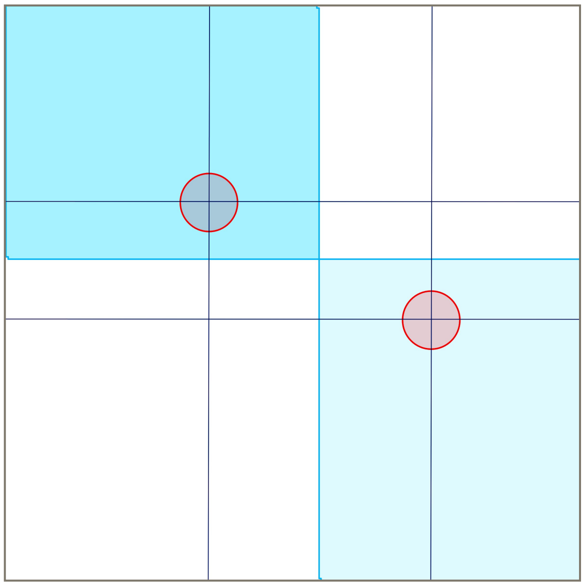
Figure 18

### 2.4.3. Physical location filter matching

While quadrant redistribution aids instantaneous filtering, physical filters are implemented to perform time-based filtering to produce consistent predictions with highest probability across multiple frames. Each of the two axes of each target object is assigned a physical filter to produce motion-based predictions. The Kalman filter is a common and efficient option for such a filter best configured with a simple position and velocity model.

Because the axes are not only decoupled, but the detected centroids are also sorted only by significance and not location and each must be reassigned to its corresponding filter. Reassignment is accomplished in two steps: 1) configuration assessment and 2) extrapolation. Furthermore, each centroid pair before re-coupling defines four points with total ambiguity. The redistributed quadrant density information described above provides two-dimensional suggestions for pairing centroids according to two configurations per object pair: 1) offset or 2) stacked.

This configurations are determined relative to the previous frame's centroid value by which the redistributed quadrants are determined. Offset pairs are both horizontally and vertically opposite each other while stacked pairs are either horizontally or vertically similar. While pairs can share the same quadrant —being both horizontally and vertically similar— this is worst case and will usually be correct by the next frame due to this case forcing the frame centroid and thus the next frame's quadrant splits to be in the center of the ambiguity of the current frame.

Configuration determination is essential for sorting the detected centroids by their axes for filter assignment via extrapolation. By using each filter's previous state and velocity to generate a basic prediction, the detected centroids can be matched by closest proximity. Matching in this case means selecting and updating the filter with a probable centroid value.

The offset configuration —meaning the frame is properly filtered and the density centroid lies in the center of the target objects on both axes— is this method's overall convergence goal. When offset, the predictions on each axis can be re-coupled with little ambiguity. When stacked, the first few frames will hold the most error due assignment ambiguity on the non-stacked axis; however, because frame centroids tend to converge to split objects, this error will diminish quickly as the offset configuration is found.

The final step of re-coupling is to update the physical filters by their matched centroid values and generate a coupling identifier distinguishing AB/CD or AC/BD coupling. The updated filter values are averaged to calculate this frame's centroid to be used to generate quadrant splits in the next frame. The final step of the image processing loop is to update and tune the greater system state machine using the probabilities determined above during blob analysis.

## 2.5. Update

Whereas the goal of prediction step above is post-compression reconstruction, the goal of the update step is loop back and tune compression parameters based on negative features of the current frame. The core tunable parameters are the image compression threshold, target density gain, and band elasticity. Adjustments are closely-tied to doubt accumulated throughout many frames driven by the current system state and variance therein.

State is calculated by traversing the Markovian state network in which the credibility model is updated given features of the current frame, specifically the current confidence in number of valid targets $\nu$ being observed and previously calculated per dimension. Given observations on individual dimensions are partial where one may be more clear than the other, the frame's actual $\nu$ is most likely the highest of both dimensions.

$$\nu_{actual} = \mathbf{max}(\nu_X, \nu_Y) \tag{35}$$

### 2.5.1. Generate state probability vector $S^{\mathbb{P}}$

Because the overall density of each frame is superimposed and dimensionally separated, the already low-confidence dramatically falls when attempting to identify individual blobs far beyond the target count. Because density overshoot is assumed instantaneous and removed as quickly as possible, tuning parameters are tightened independent how many blobs above target are observed, let alone, static sources of density overshoot are already compensated though background subtraction. Thus a cap is set relatively evenly above the target around

$$\left|\mathfrak{B}\right|_{max} = 2 \times \left|\mathfrak{B}\right|_{target} \tag{36}$$

Given this bound, the relatively small values, and continuous estimation of discrete values i.e. number of blobs, state updated information can be extracted use a cumulative Kumaraswamy distribution [1]. The probability distribution function is defined as:

$$\mathfrak{K}_{\alpha,\beta}(x) = \alpha\beta x^{\alpha-1}\left(1 - x^{\alpha}\right)^{\beta-1} \tag{37}$$

Shape parameters $\alpha$ and $\beta$ are defined as $\nu_F$ and $\left|\mathfrak{B}\right|_{max}$ respectively describing actual and expected maximum. Note $\alpha$ may exceed $\beta$, representing instantaneous chaos and useful for triggering harsh tuning. Due to the computational complexity of the PDF as shown in Figure 19, the much more friendly cumulative distribution function is used as shown in Figure 20, define as:

$$\mathscr{K}_{\alpha,\beta}(x) = \int_0^x \mathfrak{K}_{\alpha,\beta}(\mu)d\mu = 1 - \left(1 - x^\alpha\right)^\beta \tag{38}$$

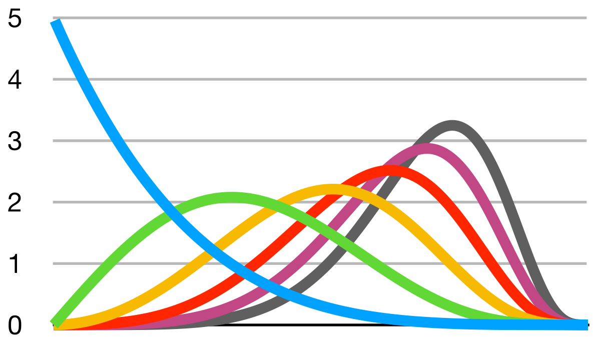

Figure 19

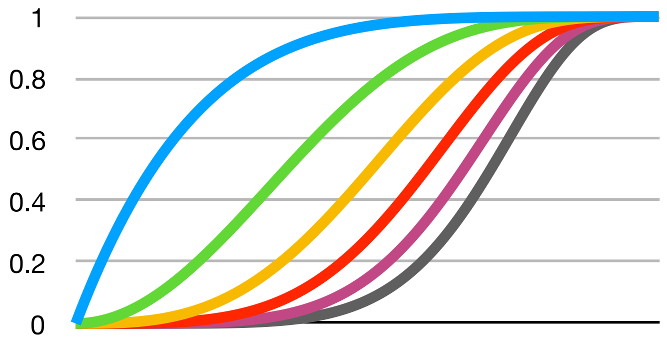

Figure 20

In the context of observed blobs, the cumulative distribution represents the likelihood a probability of a specific blob count is observed. Similarly the state network represents the likelihood of a specific number of blobs given many frames. For evenly distributed states, the probability update for each state is defined by:

$$\mathbb{P}_S = \left\{ \mathscr{K}\left(\frac{i}{N_S}\right) - \mathscr{K}\left(\frac{i-1}{N_S}\right) : 1 \leq i < N_s \textbf{ and } i \in \mathbb{Z} \right\} \tag{39}$$

$$S^{\mathbb{P}} = \left[\mathbb{P}_S(1) \quad \mathbb{P}_S(2) \quad \ldots \quad \mathbb{P}_S(N_S - 1)\right] \tag{40}$$

These state probabilities are inherently similar to the original probability function when rescaled by the estimated maximum $\left|\mathfrak{B}\right|_{max}$ as shown by:

Before updating the state model, the target overshoot $\Delta\mathfrak{K}$ is calculated as the difference of maximums of the target distribution and the actual distribution as represented by the probability density function:

$$\mathfrak{K}^{max} = \mathfrak{K}_{\alpha,\beta}\left(\sqrt[\alpha]{\frac{\alpha - 1}{\alpha\beta - 1}}\right) \approx \alpha\,\mathbf{max}\left(\mathbb{P}_S\right) \tag{41}$$

$$\Delta\mathfrak{K} = \left|\mathfrak{K}^{max}_{target} - \mathfrak{K}^{max}_{actual}\right| \tag{42}$$

2.5.2. Bayesian state control

The feedback loop is closed by implementing a method of state recognition and appropriate value tuning. The following is an example implementation of a Markovian state machine feedback method as shown in Figure 21. The machine implies states traverse by probability where all paths leaving a state hold a probability the sum of all totaling 100%. The states are distinguished by stability status (either stable-S or unstable-U) and number of objects currently being tracked (either none-0, one-1, two-2, or many/more than two-M).

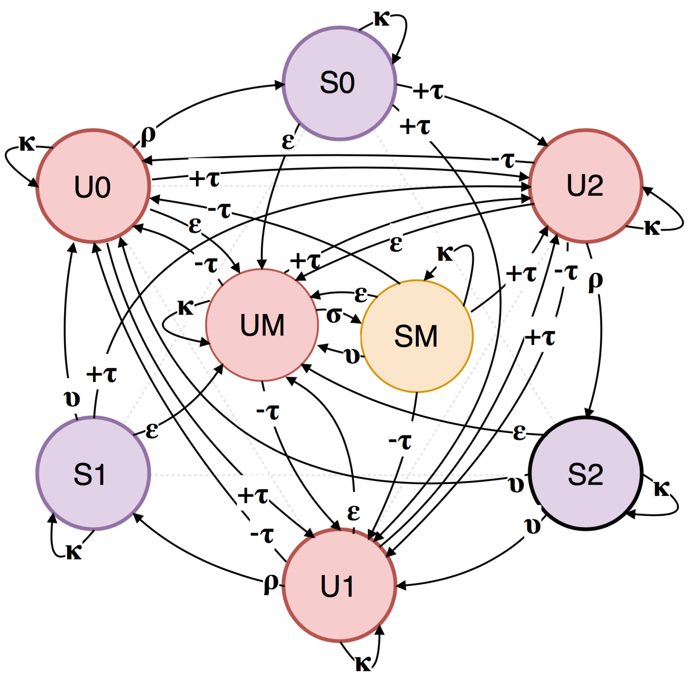

Figure 21

The primary probabilities and their corresponding actions are stabilization($\boldsymbol{\rho}$ and $\boldsymbol{\sigma}$), de-stabilization($\boldsymbol{\upsilon}$), tuning($\boldsymbol{\tau}$). A stable state can only be reach through stabilization from the corresponding unstable state. Similarly, generic stabilization($\boldsymbol{\sigma}$) is only used to describe stabilization of more objects than should be tracked and therefore a negative step. This is distinguished from preferred stabilization where the stabilization is positive for the system. Any stable state can de-stabilize into any unstable state with fewer objects. This characterizes the occurrence of a tracked object being lost. The preferred method of rising in the machine is through tuning($\boldsymbol{\tau}$) where positive results are can be assumed to be caused by alterations in the system. All states may follow either self-reinforcement($\boldsymbol{\kappa}$) and chaos($\boldsymbol{\varepsilon}$) where self-reinforcement is assumed until a significant event occurs and chaos is an unpredictable and significant rise in noise such as a lens-flare or temporary system failure encouraging a state change to state UM.

In order to preserve memory, such a state machine can be implemented as memory-less, popularly referred to as a Markov chain, where state updates require only the current state as shown in Figure 23 where $S_n$ is the current state and $S_{n+1}$ is the next state.

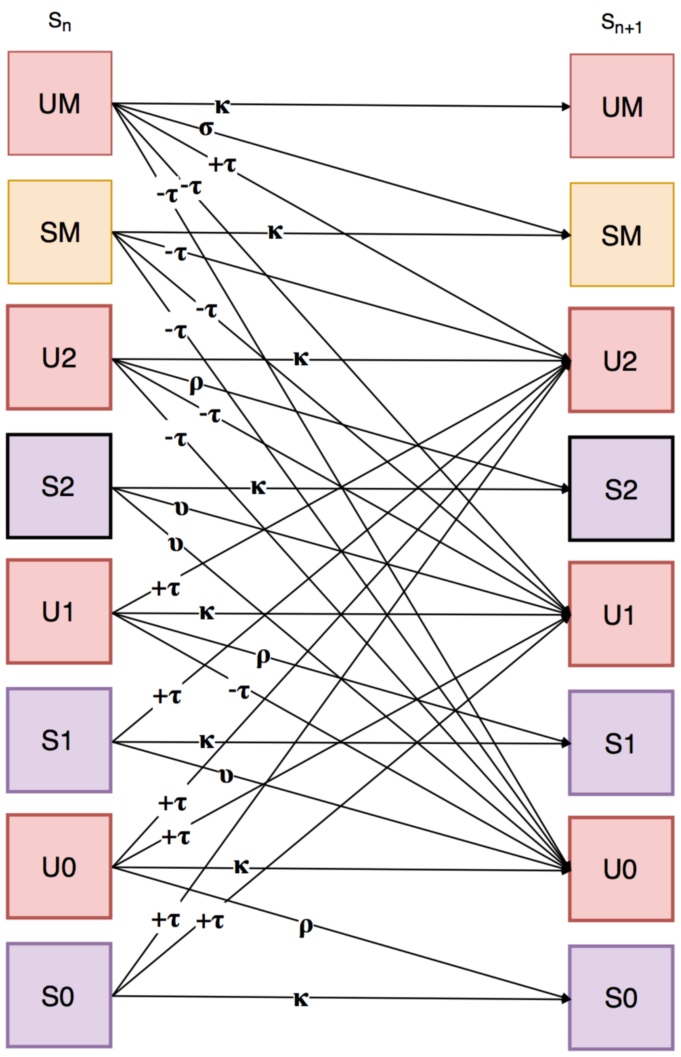

Figure 22

### 2.5.3. Tuning & Feedback

The initial binary thresholding of each frame before compression and more importantly the threshold value itself defines the maximum accuracy for reconstruction during later pieces. Updating the threshold value requires two independent factors: 1) static background noise coverage and 2) the current tracking state. These can be treated as hard and soft tuning respectively. Ideally, the static background noise would be reduced to zero where the target sources are brighter than the background. All noise would then be filtered immediately during binary thresholding with a correctly tuned threshold value. Selecting that value involves only iteratively stepping the threshold up or down until a target background noise coverage value is achieved. This is called hard-tuning because it can be implemented aggressively if necessary as to only require one or two cycles to reach the target threshold.

While hard-tuning is effective for filtering all noise less intense than the target sources, values must be filtered after compression during quadrant redistribution and background subtraction.

Soft-tuning handles the fine selection when background noise areas are as intense or brighter than target sources based on the current state and its stability. Stable states will tune slower than unstable ones as well as lower states will tune towards the target state faster than higher ones.

These two tuning methods should be implemented independently for cases where they oppose each other which are completely valid. Because the threshold updates are predictions on a linear scale the true value can be tracked using a Kalman filter. This encourages the threshold to variate naturally and converge exponentially toward the target value.

### 2.5.4. Threshold Update

Finally, the feedback loop is completed by updating the compression parameter PID filter where the current frame's density mismatch $\rho\,\%$ is checked against the target state overshoot $\boldsymbol{\Delta\mathfrak{K}}$ defined as:

$$\rho\,\% = \frac{\rho_n}{\rho^T} \tag{43}$$

$$\mathbf{PID}\left(\Delta\mathfrak{K}, \rho\,\%\right) \tag{44}$$

The direct output of the PID filter is an updated threshold value for the next image compression. Furthermore, values required by the next frame include targets centroid for each dimension, the current densities both raw and unbanded, maximum peak density, and the quadrant vector. This concludes the compression, reconstruction, and feedback of a single cycle producing the estimated coordinates of the most significant blobs.

## 3. Results

*In progress…*

Current implementation (See Appendix B)

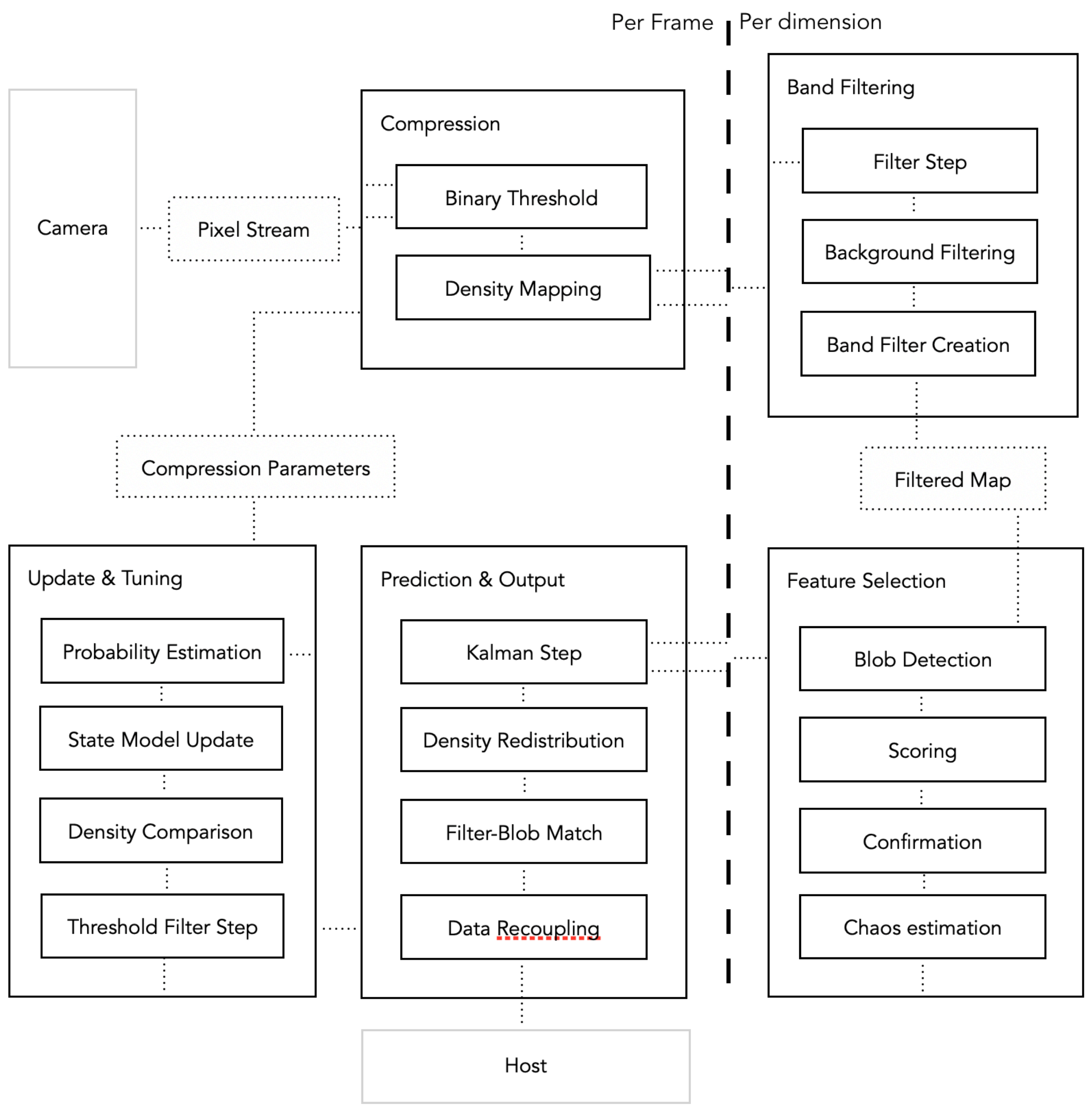

Figure 23

Performance comparison against many common algorithms:

| Algorithm | Performance (ms) | Comparative to PCR |
|---|---|---|
| PCR | **4.766** | **1.0** |
| MEDIANFLOW | **5.047** | **1.1** |
| Basic Centroid | **8.079** | **1.7** |
| BlobDetection | **11.24** | **2.4** |
| KCF | **13.36** | **2.8** |
| MIL | **37.84** | **7.9** |
| BOOSTING | **54.93** | **11.5** |

Table 1

Complexity:

Below are visualizations of $\Omega(N^2)$, $\Omega(N)$ , $\Omega(\mathbf{PCR}) = \Omega(0.005N^2)$, and $\Theta(\mathbf{PCR}) = \Theta(0.1N^2)$

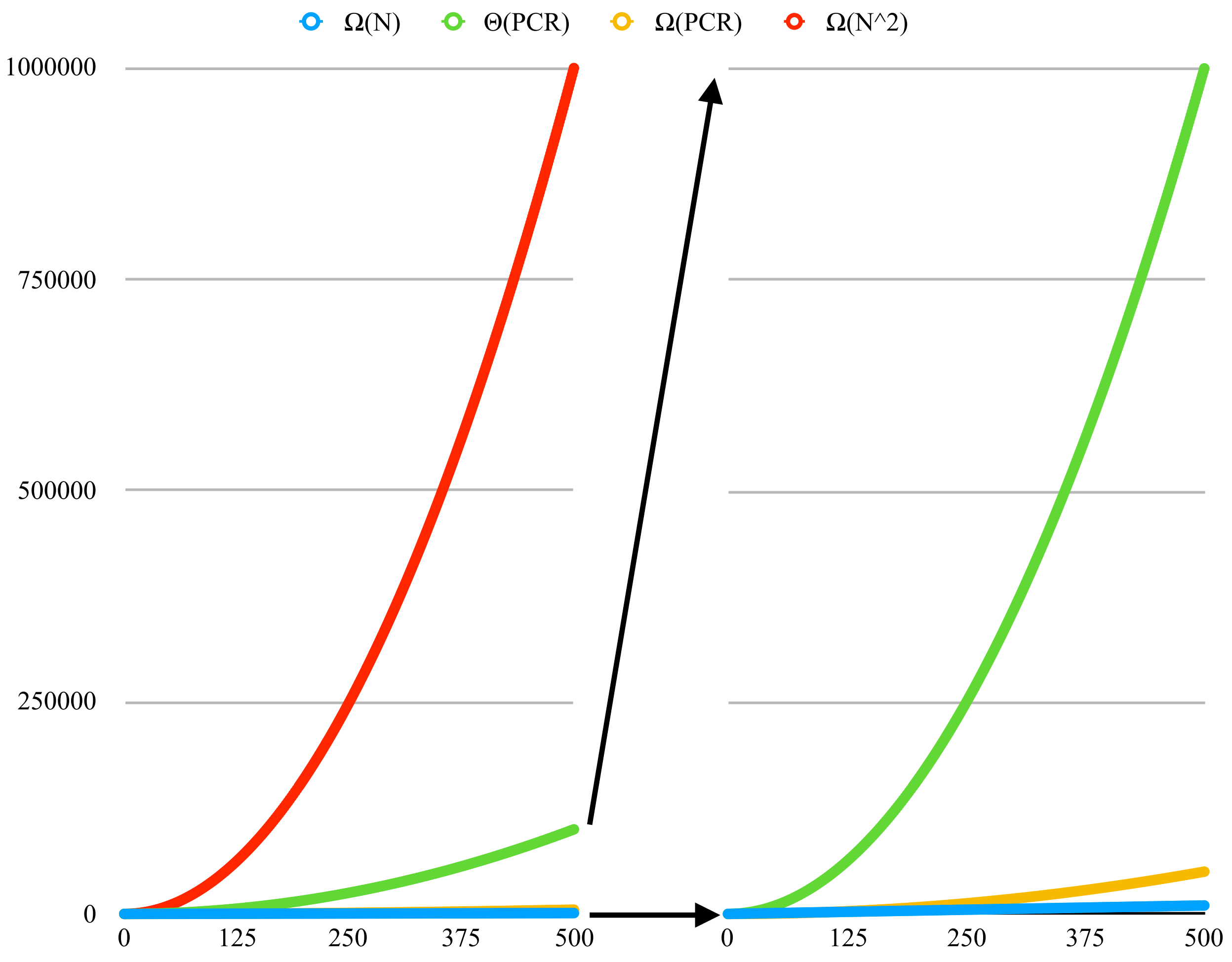

Figure 25

## 4. Discussion

The reconstruction method proposed above is one of many approaches to processing hyper-compressed image data. While the practicality of this compression and the use of density maps is relatively unique to a small family of processors and devices, the development of such algorithms is essential to the exploration of modern imaging and miniaturization of popular techniques. Especially with the rise of augmented and virtual reality technologies and algorithms, the popular focus has been to lay foundations without power and size restrictions. However, it is clear the next generation is shifting focus to mobilizing human-computer interaction in all three spacial dimensions.

While processing technology will inevitably advance in power and reduce in size, an important focus in development must be optimization: Distillation of core concepts over amalgamation. The goal of this paper is ultimately a proposition of possibilities of image processing far below the resources assumed by common ones. By replacing memory requirements with probabilistic estimation and predictive keys, this technique provides and alternative for minimal processors to handle the dimensional complexity of images commonly accounted for by addition of resources in larger devices.

## 5. References

*In progress…*

1. Kumaraswamy

(1) M.C. Jones, Kumaraswamy's distribution: A beta-type distribution with some tractability advantages, Statistical Methodology 6 (2009) 70–81

2. Kalman Filters
3. Thresholding
4. PID/feedback
5. Compression/Projection
6. Markov chain
7. Pixel density
8. Mobile image processing
9. FPGA vs DSP vs MCU power consumption
10. Optimization
11. Centroid/Physics

## 1. Appendix A - Compression architecture

Threshold to FIFO block buffer:

$$\mathbf{B}^{R}_{Wy+x} = \begin{cases} \square, \mathbf{p}_{x,y} \geq \mathbf{t} \\ ■, \textbf{otherwise} \end{cases} \tag{45}$$

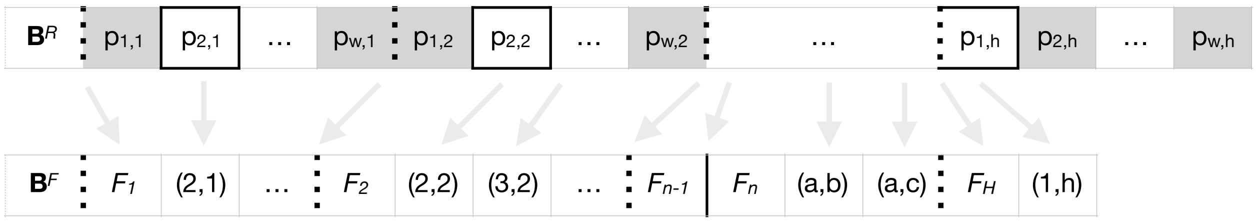

Figure 26

$$\mathbf{D}^X[y_i] = \mathbf{B}^F_{F_{y_i+1}} - \mathbf{B}^F_{F_{y_i}} - 1, \textbf{ where } \mathbf{B}^F_{F_n} \textbf{ is the index of frame delimiter } F_n \textbf{ in } \mathbf{B}^F \tag{46}$$

$$\mathbf{D}^Y[x_i] = \left| \forall i \in \mathbf{B}^F \right|, \textbf{ where } \mathbf{B}^F[i] = (x_i, y_i) \tag{47}$$

## 2. Appendix B - Example Hardware Implementation

This can be performed directly to the camera's out

Consider the following example application implementing this thresholding:

- OV9712 Camera
  - Bayer RG-GB
  - Resolution 1280x800 pixels
  - 30 frames-per-second or 30Hz,
- STM32L432 DSP
  - 64Kb RAM
  - 80Mhz Cortex-M4F core
  - Dual DMA

### Image Compression

The minimum processing speed is then:

$$\frac{1280 \times 800\ \mathbf{pixels}^2}{\mathbf{frame}} @ 30\mathbf{Hz} = 33.3\frac{\mathbf{ms}}{\mathbf{frame}} = 32.6\frac{\mathbf{ns}}{\mathbf{pixel}} \tag{48}$$

Because every other pixel is green and therefore discarded, the maximum processing:

$$32.6\frac{\mathbf{ns}}{\mathbf{pixel}} \times 2 = 65.2\frac{\mathbf{ns}}{\mathbf{pixel}}\mathbf{,\ requires} \approx 15.4\mathbf{MHz} \tag{49}$$

A minimal pixel threshold check can be performed with a single memory load and compare followed by two increment and stores for positive pixels for a total of 6:2 positive-negative cycles per pixel ratio. In a buffered system with hardware triggered memory transfers, 15.4Mhz/ pixel is the average limit for processing an entire frame where given enough negative or non-significant pixels. For example, give an 80Mhz processor pixels must be processed and a maximum of 80/15.4 or 5.19 system ticks per pixel or 86.5:13.5 positive-negative distributed frame. Per-row delays that delimited rows in the processed memory force this ratio lower, however, it remains generally higher than most environments.

A typical good case has shown to lie within the single digit percentages. Given even an unoptimized implementation nearly halving 86.5:13.5 to 50:50 frame balance limit and a significantly larger than suggested target of 10% target coverage, this still leave 40% coverage for noise. Figure 21 and 22 illustrate a typical frame and a coverage limit frame of such an unoptimized implementation respectively. Note how the coverage limit of an unoptimized system is *significantly* far beyond a typical good case.

## 2.1. Appendix C - Static Background Density Redistribution

The following description provides an algorithmic approach to redistributing low refresh rate background and static noise quadrant data to high refresh rate frame quadrant data for proper static noise filtering. Quadrant densities are treated as homogeneously distributed throughout the corresponding quadrant and split linearly by centroids. Figure XX illustrates how the centroids of a background frame overlaid on a normal frame form nine rectangular regions.

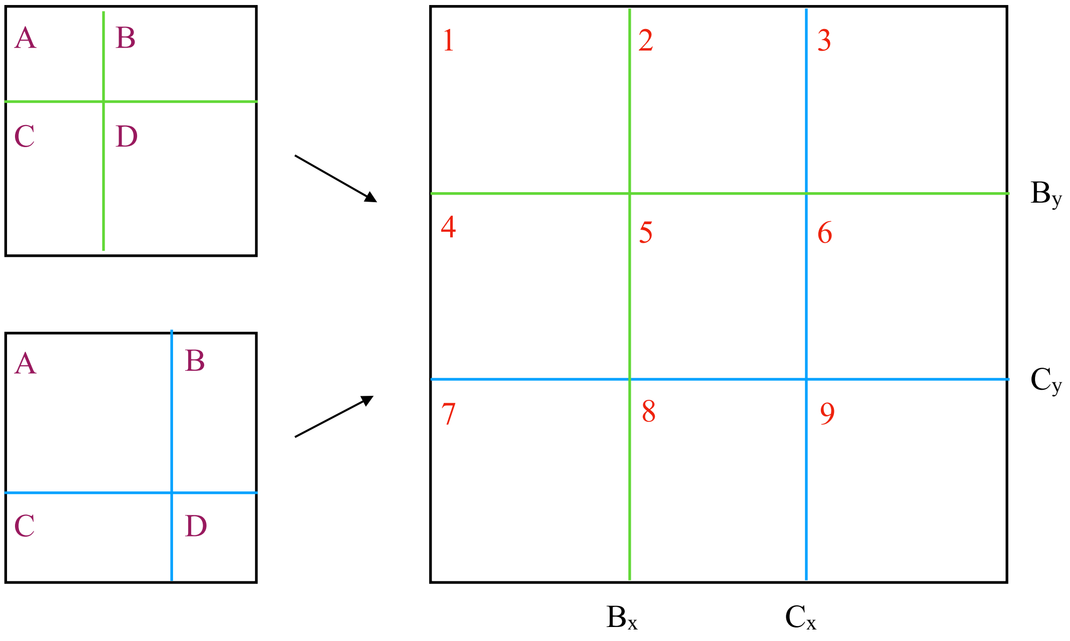

Figure 27

These are the four possible configurations for background and current frame overlap:

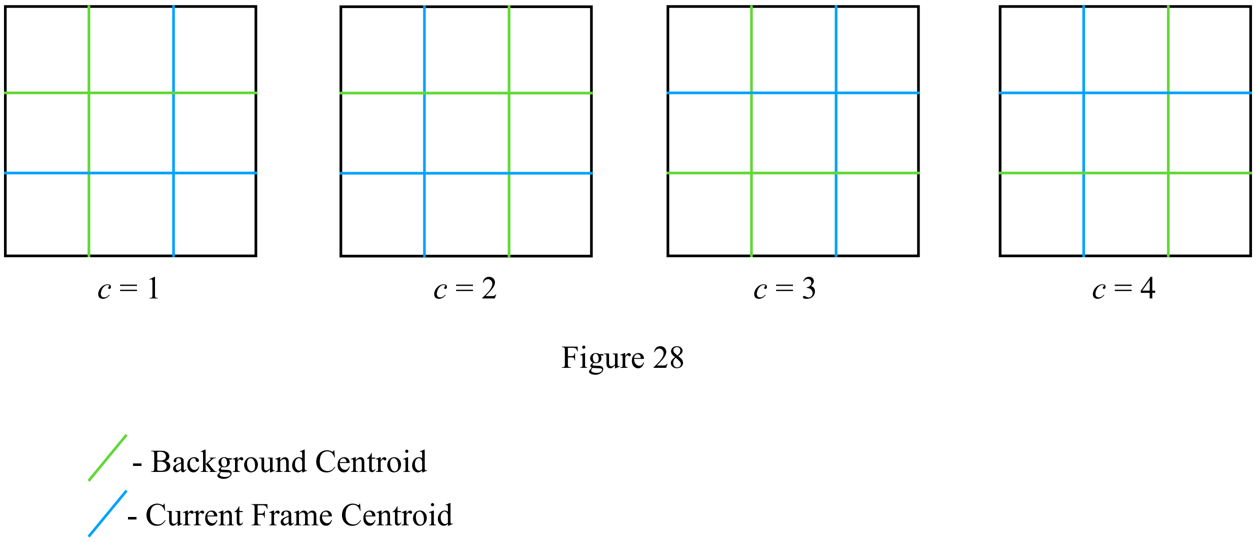

Figure 28

Redistribution Lookup Table

| | c | A | B | C | D | c | A | B | C | D |
|---|---|---|---|---|---|---|---|---|---|---|
| $B_{AL}$ | 1 | 1, 2, 4, 5 | 3, 6 | 7, 8 | 9 | 4 | 1 | 2, 3 | 4, 7 | 5, 6, 8, 9 |
| $C_{AL}$ | | 1 | 2, 3 | 4, 7 | 5, 6, 8, 9 | | 1, 2, 4, 5 | 3, 6 | 7, 8 | 9 |
| $F_L$ | | 1, 2, 3, 4 | 2, 4 | 3, 4 | 4 | | 1 | 1, 2 | 1, 3 | 1, 2, 3, 4 |
| $l$ | | 4 | 2 | 2 | 1 | | 1 | 2 | 2 | 4 |
| $B_{AL}$ | 2 | 1, 4 | 2, 3, 5, 6 | 7 | 8, 9 | 3 | 1, 2, 4, 5 | 3, 6 | 7, 8 | 9 |
| $C_{AL}$ | | 1, 2 | 3 | 4, 5, 7, 8 | 6, 9 | | 1, 2 | 3 | 4, 5, 7, 8 | 6, 9 |
| $F_L$ | | 1, 3 | 1, 2, 3, 4 | 3 | 3, 4 | | 1, 2 | 2 | 1, 2, 3, 4 | 2, 4 |
| $l$ | | 2 | 4 | 1 | 2 | | 2 | 1 | 4 | 2 |

Table 2

Redistribution of current frame's quadrant values $\rho_C \rightarrow \rho'_C$ using lookup:

$$for\ q\mathbf{: A\ to\ D}$$

$$\rho'_C[q] = \rho_C[q] - \sum_{i=1}^{l[q]} \frac{C_{AL}[q][i] \times \rho_B[q]}{\sum_{j=1}^{l[4-q]} B_{AL}\left[F_L[q][i]\right][j]} \tag{50}$$