TAU EXTENDED - FORMAL DESCRIPTION

1. Abstract

Tau+ is an image processing algorithm for object tracking. The objective of this algorithm is to produce pixel coordinate information of interesting objects within an image. Further goals are convergent analysis, self-optimization, high-frequency scaling, and strict and minimal memory implementability. Unlike more traditional methods of object tracking operating in $O(n^2)$ time, Tau+ emphasizes probabilistic analysis of coupled one-dimensional data created by compressing two-dimensional image data allowing for best case O(n) time. Use of image compression in Tau+ is appropriate for implementations emphasizing analog filtering, i.e. NoIR or tinted film filters, due to the relatively low amount of two-dimensionally dependent encoded information. Unlike algorithms such as facial recognition that require unique, minimally compressed, two-dimensional pixel information of faces, Tau+ analyzes information such as movement and persistence of generic dense pixel areas (i.e. blobs) that can be separated into single dimensions. Tau+ has three main elements: 1) density-based image compression and analysis currently described as Rho, 2) probabilistic object tracking and filtering, and 3) memoriless bayesian state handling for tuning and self-optimization.

2. Introduction

Tau+ is designed for object tracking using a camera in a hardware limited system such as embedded micro-electronics and battery powered handheld devices. The specific version described below is designed and tested to track specifically two points who's coordinates can be used later for triangulation in a three-dimensional coordinate reconstruction algorithm.

Coordinate reconstruction is crucial in many trending technologies such as VR (virtual reality) and AR (augmented reality) in which real world objects are tracked in three-dimensions and converted into points, shapes, and actions on a screen. While this technology is progressing rapidly, it is still in its infant stages and hasn't yet broken into a extraordinarily lucrative market of handheld and mobile devices. The most significant problem contributing to this is the computation cost and complexity of almost all popular object tracking algorithms. Small devices reliant on battery power do not have the power both computationally and electrically, the memory, or resources to process data at the speeds required for three-dimensional object tracking. This is the primary goal of Tau+. Paired with other strict hardware conscious algorithms, position tracking can be achieved on FPGA or microcontroller systems with sub-megabyte memory and sub-GHz processing speed.

3. Method

Most popular implementations of object tracking are purely software based and not optimized for specific hardwares. Most implementations such as the camera is virtually independent of the

system processing the camera's data. This forces every pixel of every frame to be processed at least twice: Once to read and save —usually by the camera's processor— and at least once to process. This method may seem unavoidable; however, the follow description details how minimal filtering and processing can be implemented on the camera's processor to drastically reduce computation cost with strategic data loss. By using three forms of probabilistic filters downstream, critical data can be reconstructed. This allows for not only a tunable accuracy and speed, but also a significant drop of required memory space and operations.

Traditional methods of object tracking are dependent on the assumption each frame will be stored and commonly require non-linear access meaning when a pixel or area is read it may be required later and thus cannot be deleted immediately. The alternative presented by Tau+ and probabilistic reconstruction is immediate image compression as the camera's pixels are read into coupled one-dimensional data sets. These sets are coupled in that one can loosely describe the other(s) but cannot recreate the original image. This form of image compression is appropriate due to the relatively low amount of two-dimensionally dependent encoded information.

The data the Tau+ extracts is position data which is naturally encoded as superimposed pixel densities. In other words, an object that is moving and is visualized in a camera will appear as groups of pixels moving a certain amount in X dimension and a certain amount in the Y dimension. Once processed, such an algorithm will produce a set of X coordinates and a set of Y coordinates that are unpaired. The assumptions embedded in Tau+ then are: 1) trackable objects can be relatively cheaply translated in to density data where pixels not populated by the object have little to no density and the those that are have higher density —this is expounded in section 3.1—, 2) the camera processing hardware is capable of minimal analysis of every pixel as they are read from the camera, and 3) the implemented system can pair the resulting unpaired coordinate data produced into useful information.

The follow sections will describe Tau+'s three main elements in greater detail 1) image capturing, compression, and analysis 2) probabilistic object tracking and filtering, and 3) bayesian state handling. The specific implementation described was designed and tested using a camera with a RG-GB Bayer array pattern, "Blue Congo" IR high-pass filtering film, an Altera Max 10 FPGA as the camera processor, infrared LEDs as the objects of interest to be tracked, and **<INSERT MICROCONTROLLER USED>** to perform the probabilistic analysis.

3.1. Image Processing

The combination of infrared LEDs and a RG-GB patterned camera array are ideal for a handheld implementation due to not only both being low-cost and popular but also a processing short-cut they create. A generic low-cost camera already typically used in handheld devices visualize infrared light as violet in the absence of a physical IR blocking filter as shown in **<Figure 1>**.

Because visualized IR has no green, all green information can be discarded from the camera's output. In the case of the RG-GB camera, see **<Figure 2>**, this means half of all pixel

information can be discarded giving each red and blue pixel an extra pixel clock cycle to be processed. This factor is key to following description of image capturing and compression.

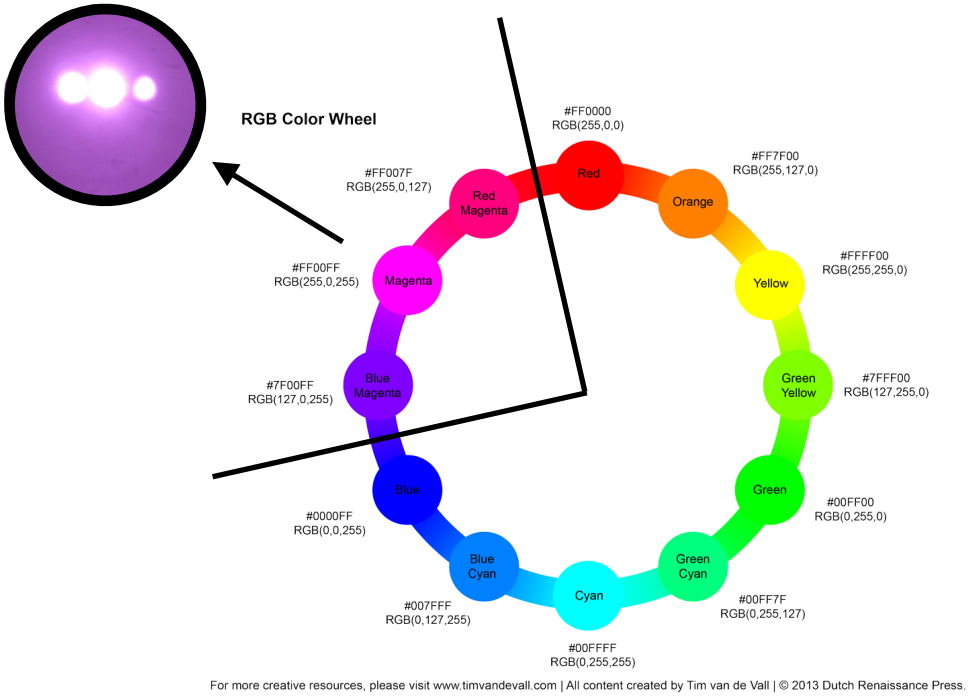

Figure 1 - Camera Visualization of Infrared Light on a Color Wheel

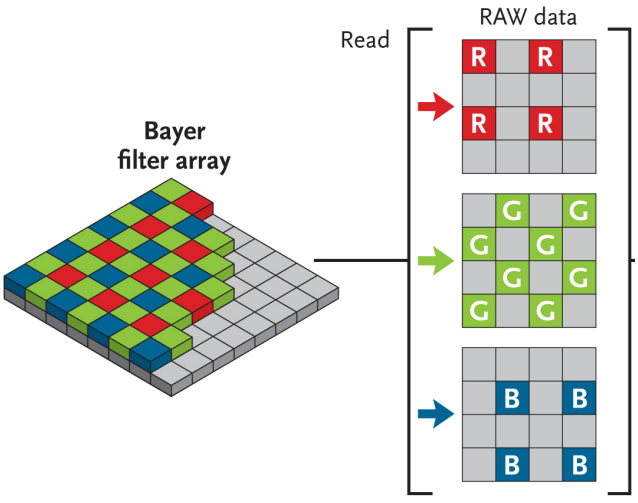

Figure 2 - RG-GB Bayer Array

3.1.1. Image Capturing and Compression

The compression technique implement in Tau+ is called density mapping where the image is binarized or similarly converted so that each pixel describes a one-dimensional density value of the object being tracked and then summed along the X and Y axes independently. Density information describes the presence of an object at a specific location, i.e. violet pixels represent the presence of infrared light. Furthermore, in a binary thresholded image of an infrared LED, each violet pixel would represent a single density count of that LED.

This step forms two one-dimensional snap shots of the image along these axes. While these snap shot sets are irreversible and do not describe the image fully, both loosely describe each other and include identical amounts total density. A spike in density in one will most likely be paired with a density spike in the other, however, the location of one spike will not describe it location on the other axis.

Density mapping significantly reduces the amount of memory required for further analysis of an image of size MxN to M+N and thus an entirely different family of complexity and cost from $O(n^2)$ to $O(n)$. The compression preserves the location information of the objects being tracked, only it is now irreversibly separated into two dimensions. This separation can be quickly recovered using motion data and position hypotheses commonly produced by position reconstruction algorithms.

3.1.2.Density Map Filtering

A threshold filter can quickly produce an accurate density map for high contrast images i.e. infrared LEDs through a NoIR filter. As described above, visualized infrared carries no green thus thresholding can be achieved through adding each pixel's red and blue factors and checking against a given threshold value:

$p^{RED}_{(x,y)} + p^{BLUE}_{(x,y)} > v_{threshold}$, where $p_{(x,y)}$ represent a RGB color value at coordinate (x, y).

or

$$\left(p^{RED}_{(x,y)} + p^{BLUE}_{(x,y)}\right) - v_{threshold} = \rho^{*}_{(x,y)}$$

$$\rho_{(x,y)} = \begin{cases} \rho^{*}_{(x,y)}, & \textbf{for } \rho^{*}_{(x,y)} > 0 \\ 0, & \textbf{for } \rho^{*}_{(x,y)} < 0 \end{cases}$$, where $\rho_{(x,y)}$ is the image density at coordinate (x, y).

Given an image of width W and height H.

$$\rho_x[W] = \sum_{x=1}^{W} \sum_{y=1}^{H} \rho_{(x,y)}$$, where ρ_x is the image's X-axis density set of length W

$$\rho_y[H] = \sum_{y=1}^{H} \sum_{x=1}^{W} \rho_{(x,y)}$$, where ρ_y is the image's Y-axis density set of length H

NOTE: From here, processing of the X and Y information is identical and thus the subscripts are not included.

In a single frame, these density maps hold one-dimensional position information of objects in

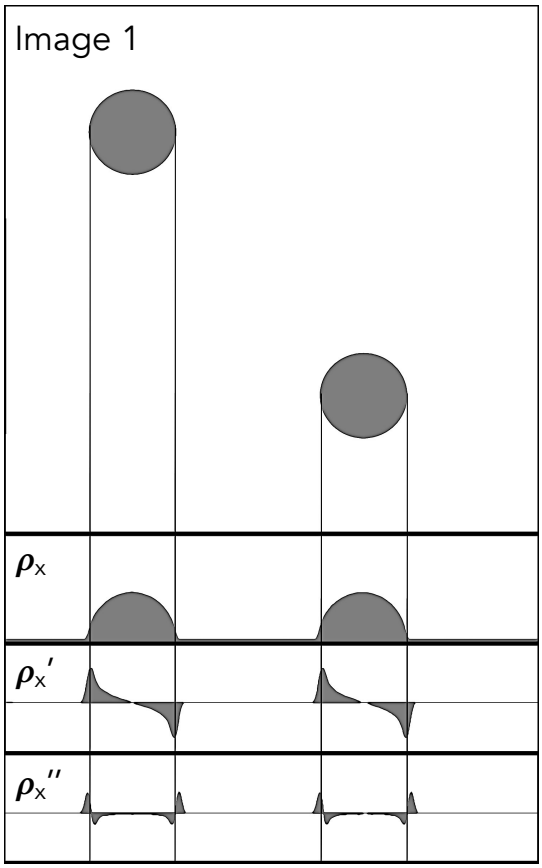

Figure 3 - Example Analysis of Image 1

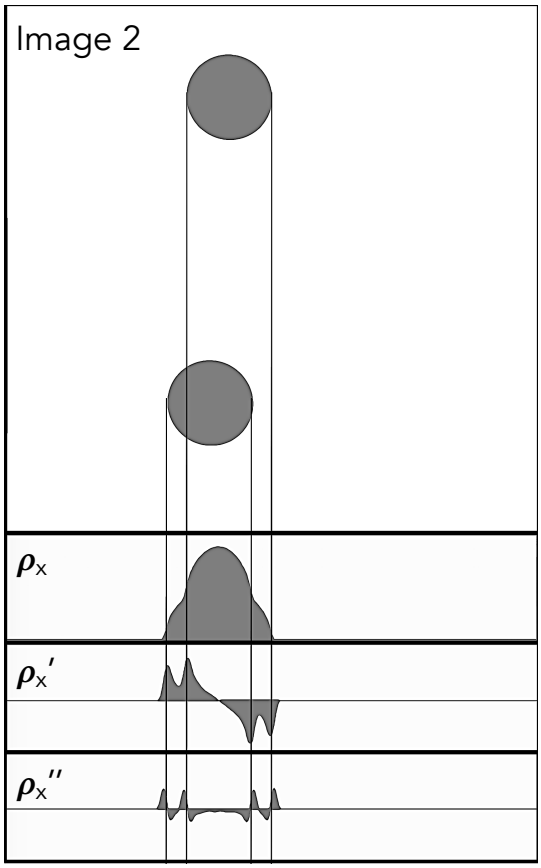

Figure 4 - Example Analysis of Image 2

the image. A set a frames hold persistence and movement information of these objects. This persistence and movement can be tracked and analyzed to Most significant points of interest (or POIs) in these density sets are represented by spikes and peaks. Positions of peaks can be easily found by calculating the first derivative of the density set.

$$\rho' = \frac{dx}{dp}\rho \text{ where } zero(\rho') = \textbf{POI}$$

As showing in Figure 4 where region B overlaps region A on the X-axis, the two densities are combined and indistinguishable on the velocity map. An initial solution to combat this loss of information was to check for close densities peaks merging and growing significantly. While effective in controlled tests with only two objects, the solution is time-dependent and requires many frames to produce a single estimate and thus error and initial false assumptions compounded for more complex images. In order to detect individual objects crossing each

others paths or shadows, each object must be identified independently first causing cases where the objects begin overlap to not only fail for the first frame but for dozens of frames until the initial assumption is overridden.

A more precise shadow handling technique can be achieved by calculating the second derivative of the initial density set:

$$\rho'' = \frac{dx^2}{d^2p}\rho \text{ where } zero(\rho'') = \textbf{POI}$$

Whereas the POI's of the first derivative show locations of density regions, the zeros of the second derivate show edges of density regions. More importantly, points where zeros of ρ'' and $\rho' > 0$ show the starting edges of a region while similar points where $\rho' < 0$ show the ending edge of a region. By pairing starting and ending edges a center point can be approximated. More symmetric objects will hold better approximations. As shown in **<Figure 4>**, overlapping objects can be detected independently.

NOTE: This is true for images with a relatively small amount of noise.

Once a POI is detected, descriptive information such as location and density is saved and processed later in the object tracking and filtering section (3.2). Tau+ is designed to be segmented in such a way as to allow for variation of communication of the resulting list of POIs from density map filtering from implementation to implementation. This is due to the difference in system speeds required to process camera output and process POI output. A typical application may dedicate a high speed controller such an FPGA to perform the less complex density map filtering and a separate low speed controller to perform the more complex object tracking requiring a communication bus between the two.

The minimum bandwidth for this bus is dependent on the data type used for the POIs, compression, protocol, and maximum number to be transferred. For multi-variable verification of objects across multiple frames, it is recommended to describe POIs in at least two dimensions typically location and density, however a single dimension implementation will increase speed with sacrifice of accuracy.

3.2. Object Tracking and Filtering

The goal of the second section of Tau+ is to identify the true coordinates of object(s) such as LEDs. The core assumption is that one or more of the POIs directly hold information about these true coordinates. The two sets of POIs in the X and Y dimensions generated by density map filtering can be processed using a combination of tools such as Kalman filtering, feedback, and coupling analysis for multi-object tracking. In order to encourage convergence, Tau+ uses a dynamic matrix of Kalman filters to not only track individual points but cross-check each point against all others allowing for incorrect assumptions such as deceptive noise or multi-point overlaps to be instantaneously corrected. This is crucial in order to overcome the normally slow corrective nature of feedback and probabilistic systems. Such a dynamic Kalman matrix filter has 4 major steps: Adding, Updating, Selecting, and Removing. When more than one objects are being tracked, a Coupling step can be added.

3.2.1. Kalman Matrix

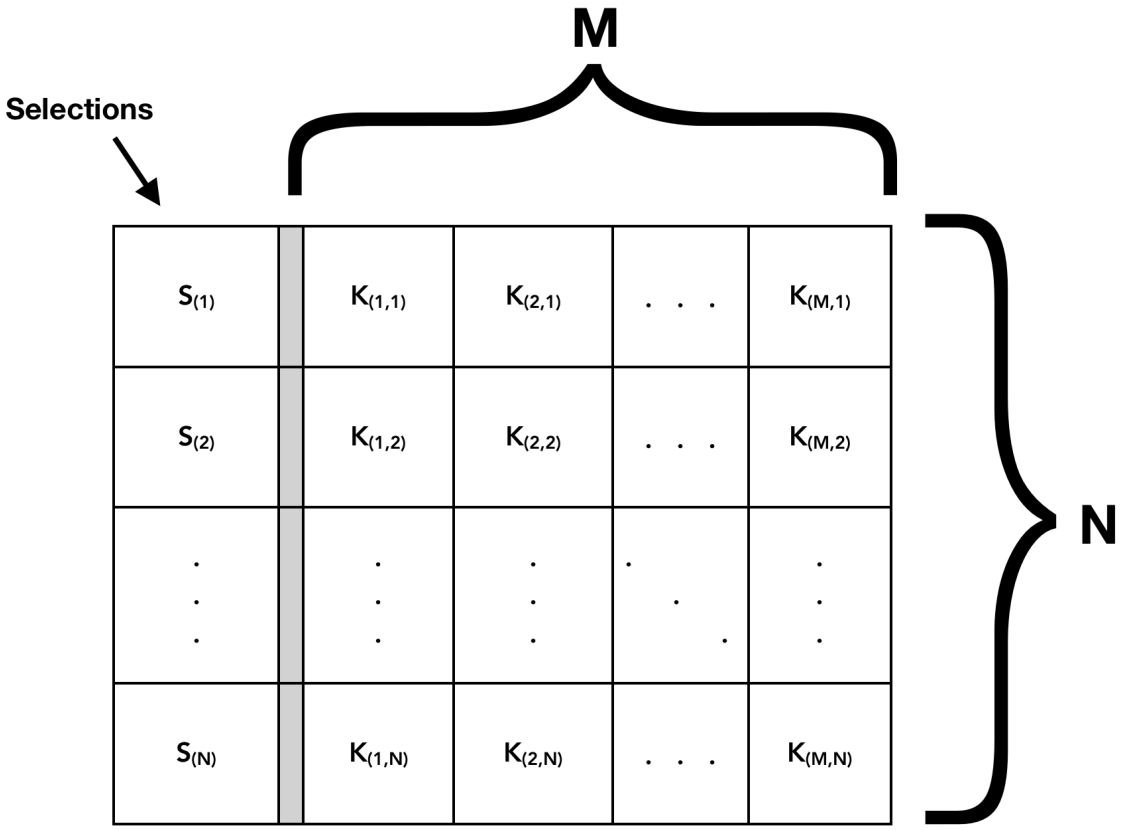

Figure 5 - Example Kalman Matrix

The Kalman matrix is two-dimensional matrix of single dimension Kalman filter types where one dimension is tracked POIs and the other is new POIs to be analyzed. Each row of tracked POIs shall select the most probable selection after each cycle of updates using new POIs. When implemented with sorting, this structure allows for dynamic accuracy tuning meaning that for N number of objects being tracked, the minimum processing time is updating the Kalman filters lying in the lowest NxN square. After this point, all further processing only improves the accuracy and resilience of the current predictions. The following five sections describe how the Kalman matrix works.

3.2.1.1. Adding

Values are added in a matrix, empty or filled, when the number of new POIs incoming are larger than the number of POIs being tracked. These values are added diagonally so that each new row and column hold one unique value. They are added by being run through the Kalman filter at that location. When the number of new POIs is equal or less than then number of tracked POIs, this step can be skipped.

While each filter in the matrix should perform location filtering, additional information such as density should be stored as well. By nature, Kalman filters operate best on information that is physical, thus because the pixel density produced by the image compression discussed above is virtual, it does not benefit from Kalman filtering.

3.2.1.2. Updating

The goal of the update step is to produce at least one highly accurate Kalman filter update per tracking row. This is achieved by updating the previous most probable POI with each new POI. When a tracked object does not overlap another, it will be strengthen and more resilient when overlap does occur. Because of the velocity tracking nature of Kalman filters, this method can distinguish two or more identical objects passing each other. Furthermore, if the objects stop while overlapped and velocity information is lost, their previous tracking information is still stored and can be simply separated into different rows when the objects separate in the image. As mentioned above, this method is resilient against incorrect assumptions. This is the reason tracking more POIs than only the physical number of objects being tracked. Many cases may cause the objects being tracked to not be visualized correctly such as glare, thus more prominent false POIs may take precedence over the true POIs. The assumption is that each POI hold some probability of being the true POI. This probability can be accounted for by tracking less significant POIs. When corrections are made by feedback process described in **Section** 3.3 and noise is eliminate to reveal the true POIs, they will already have been tracked and simply rise to the top of the matrix during sorting.

3.2.1.3. Selecting

Selection is the most consistent and least complex portion of the dynamic matrix operation. Each row produces a single selection, typically the Kalman filter that produced the most

accurate prediction in that row. Because Kalman filters produce the most probable calculations based on the data provided, little to no further processing is required. An example of a further selection factor is density where density consistency is factored into encouragement. Once each tracking row produces a selection, the matrix can be sorted as to bring the best of the selected to the top.

3.2.1.4. Coupling

When more than one similar objects are being tracked, coupling can be implemented to encourage both to be reinforced together and thus their resilience when noise impairs one or both. A typical coupling implementation will reward POIs with similar density to be sorted higher in the matrix together than unpaired POIs. Probability holds priority when coupling POIs where if many have the same density, only the number of objects being tracked will be coupled together. By being sorted higher in the matrix, tracked POIs have a higher likelihood of being selected as the final predictions thus this method encourages higher similarity between these final predictions.

3.2.1.5. Removing

Finally, improbable and expired Kalman filters must be purged from the matrix. Expiration can be implemented as a simple "time since last update" check. When the Kalman matrix holds more tracked POIs than new POIs, many rows may not update. Second, improbable tracking selections low in the Kalman matrix should be removed. A third method for cleaning the matrix is position similarity punishment. As discussed above, information of all objects passing should be preserved specifically velocity information, however, when they stop, many Kalman filters hold identical information and become redundant. By checking for redundancy, similar predictions can be merged into a single Kalman filter by slowly punishing the extraneous filters by a percentage. This ensures information is not irreversibly lost during a noisy frame while still punishing possibly irrelevant filters.

3.3. Bayesian State Control

While the description of Tau+ to this point can properly convert a frame from the camera into probable location predictions, a feedback loop should be added to ensure convergence. This can be achieved by implementing a form of state recognition and appropriate value tuning. Examples of tuning include changes of threshold value for image compression, variant method selection of POI calculation, or altering tolerances used in Kalman matrix selection or coupling.

The following is an example implementation of a Bayesian state machine feedback method. The Bayesian 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).

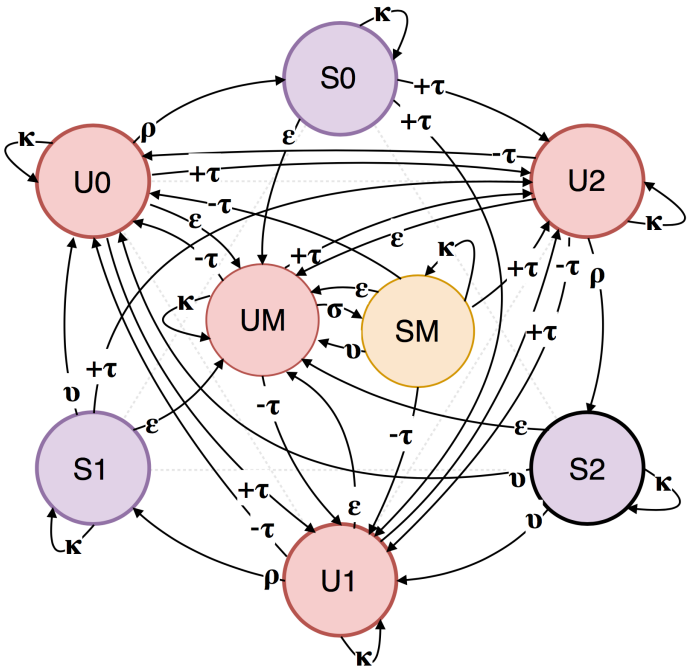

Figure 6 - Example Bayesian State Machine

The primary probabilities and their corresponding actions are stabilization(ρ and σ), de-stabilization(υ), tuning(τ). A stable state can only be reach through stabilization from the corresponding unstable state. Similarly, generic stabilization(σ) 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(τ) where positive results are can be assumed to be caused by alterations in the system. All states may follow either self-reinforcement(κ) and chaos(ε) 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 memoriless, popularly referred to as a Markov chain, where state updates require only the current state as shown in this figure where S_n is the current state and S_{n+1} is the next state.

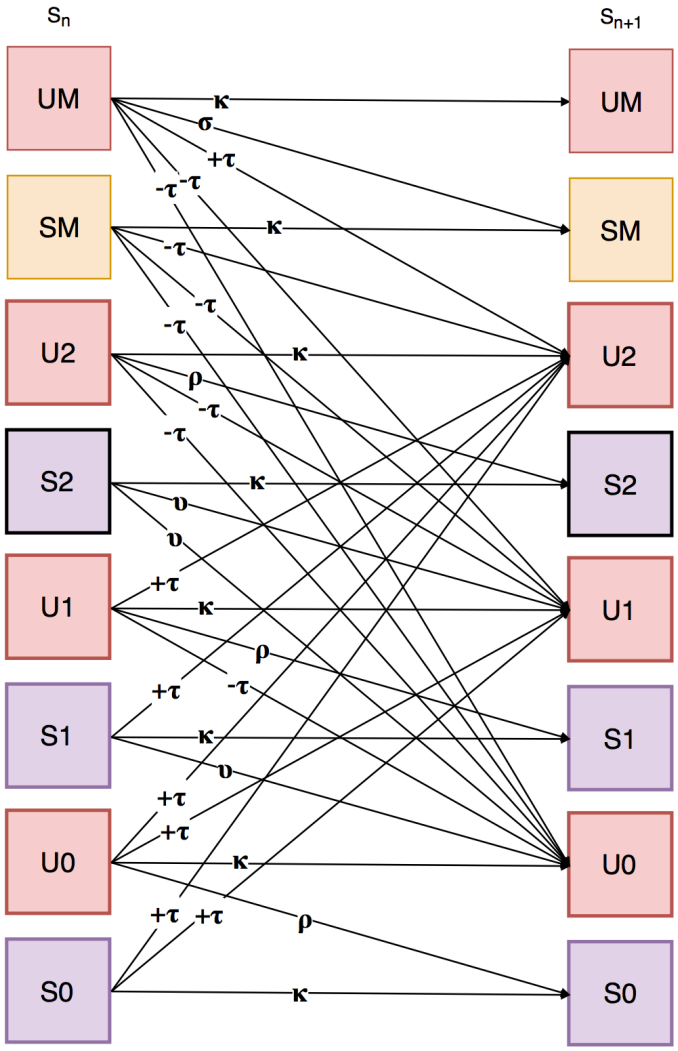

Figure 7 - Markov Nature of Bayesian State Machine

4. Results

Table 1 shows the speed performance comparison of a prototype Tau+ implementation against many popular object tracking and detection algorithms as included in the OpenCV 3.1 library. These benchmark tests were performed against the same set of 26 images measuring time from algorithm function call to return. Tau+ tied with MedianFlow for best time and significantly out-performed the others.

Algorithm	Average Performance (ms)**	Approximate Comparison
Tau+	**4.766**	**N**
MedianFlow*	**5.047**	**N**
Centroids	**8.079**	**2N**
BlobDetection*	**11.24**	**2N**
KCF*	**13.36**	**3N**
MIL*	**37.84**	**8N**
BOOSTING*	**54.93**	**12N**

* OpenCV 3.1 Implementations
** Average of 10,000 iterations

Table 1 - Benchmark Performance Data

These benchmark values demonstrate the benefit of a highly specified object tracking algorithm such as Tau+. The other algorithms tested are designed for complex objects while Tau+ is designed specifically for tracking density-convertible i.e. high-contrast or visually homogeneously featured objects. Applications such as virtual reality and three-dimensional position tracking often operate by tracking a single point. This point is designed to be highly visible to the detector or camera and most importantly has minimal complexity beyond color size, and similar features that are directly convertible into pixel density as described above. Such applications are ideal for Tau+ implementation due to object simplicity whereas applications requiring object feature identification such as facial or text recognition are not.

Tau+'s performance advantage is attributed to its early image compression. As described above, by immediately compressing an image of size NxN into two arrays of length N, the root complexity of Tau+ to $O(n)$ whereas traditional object tracking and detection algorithms not implemented with access to the camera's direct output must analyze the entire at complexity $O(n^2)$. The term "root complexity" is used here as there are two caveats: 1) All image processing algorithms must on the lowest level gather an entire image of pixels implying a

complexity element of $O(n^2)$, and 2) the specific density map smoothing technique implemented add a complexity element of an order higher than $O(n)$. Then major difference being that when Tau+ at the lowest level, the initial image read from the camera, the follow processing is $O(n)$ whereas others that are implemented downstream must not only re-read the entire image but also designed without complex-order reducing compression and are therefore locked into $O(n^2)$ complexity. See Appendix B for more in depth complexity analysis.

Furthermore, Tau+ compression significantly reduces memory requirements. Table 2 illustrates the approximate size of significant elements based on a prototype implementation designed for processing 1 megapixel images.

	Description	Approx. Size (Bytes)
Image Compression	Density Maps	12000
	POI Lists	8000
Object Tracking	Smoothing Filter	4000
	Kalman Matrix	3200
	Prediction	40
State Control	Bayesian Map	350
Total		27590

Table 2 - Memory Usage Approximations

The resulting 27.59KB size requirement including density mapped image and all processing data is less then space required for the most simple conventionally compressed image and more practically is not one, not two, but three orders of magnitude less than the 24MB ($3 \times 8 \times 1000 \times 1000$) required to store a raw RGB image. This drastic size reduction is crucial for advancement of virtual reality and three-dimensional position tracking into the handheld and mobile device market.

Not only is does Tau+ significantly reduce object tracking size requirements, but also computational complexity into a range capable of microcontrollers and similar low-level, micro-footprint devices. In conclusion, Tau+ is a revolutionary algorithm that can provide object tracking capabilities that have been only accessible to larger microprocessor systems now to micro-hardware devices.

5. Future Work

While Tau+ is designed to be flexible from implementation to implementation, the performance given data is based off a prototype simulation. A highly optimized, fully hardware implemented version is yet to be created for more accurate testing and benchmarking. Further work includes testing reliability, power efficiency, accuracy, tolerance, and flexibility both for optimization and for benchmarking against other object tracking and detection algorithms.

Furthermore, as is the core goal for the development of Tau+, broader testing in virtual reality and position tracking systems will be performed to demonstrate and explore the new hardware miniaturization possibilities opened by Tau+. As demonstrated by the example implementation of Appendix A, hardware implementations can be distributed across many processing and memory devices. A future goal is to test the benefits of centralized or single-processor systems versus distributed or multi-processor systems.

Finally, the central image compression technique and resulting $O(n)$ complexity of Tau+ should be explored for non-density based implementations. This would open the door for more complex pattern tracking and multi-point identification allowing for systems with much broader tracking abilities beyond the single-point tracking system proposed above.

Altogether, Tau+ is a promising solution to hardware and size limitations of modern virtual reality technologies.

6. Works Cited

6.1. Appendix A - Example Hardware Implementation

This can be performed directly to the camera's out

Consider the following example application implementing this thresholding:

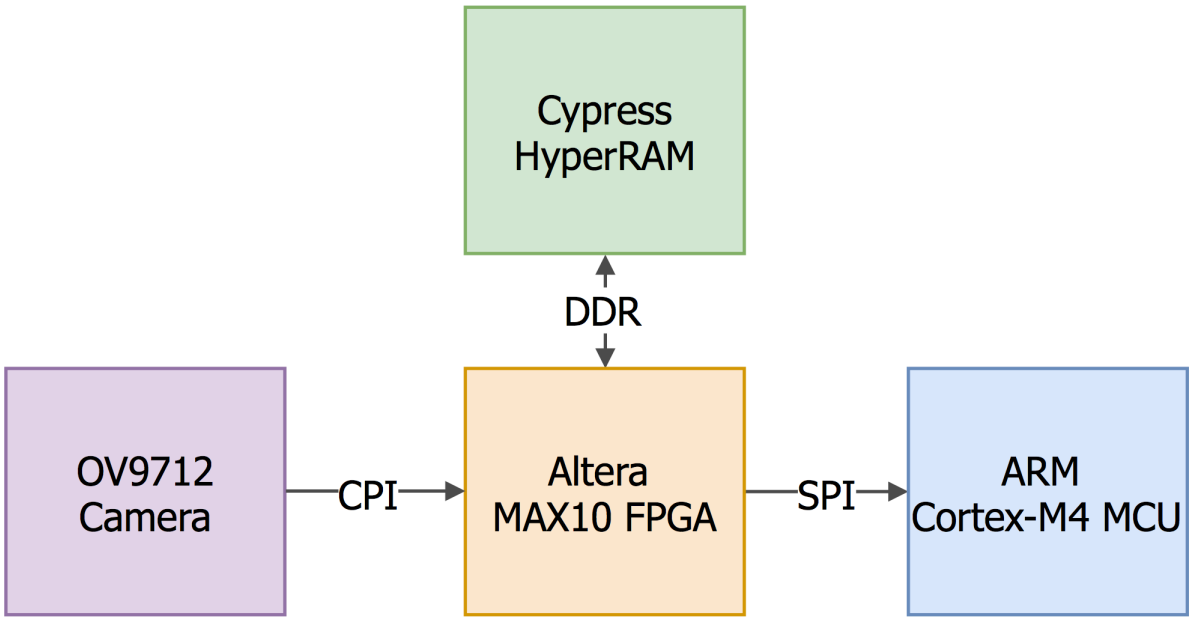

Figure 8 - Example Hardware Implementation

- OV9712 Camera
  - Bayer RG-GB
  - Resolution 1280x800 pixels
  - 30 frames-per-second or 30Hz,
- Cypress DRAM
  - Bandwidth of 300MB/s
  - Frequency of 150MHz

Image Compression

The minimum processing speed is then:

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

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

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

Each threshold operation requires red and blue data and thus for every row pair, the first is saved and the second is added to the first, thresholded, and the result is saved.

$$\frac{1280\textbf{B}}{2} \times \frac{1}{300\frac{\textbf{MB}}{\textbf{s}}} = 2.13\textbf{µs per transfer}$$

$$2.13\textbf{µs}s \times 2 \times 30\,\% \textbf{ blocking delay} = 4.91\textbf{µs or } \approx 3.83\frac{\textbf{ns}}{\textbf{pixel}}\textbf{ per Read/Write}$$

$$65.2\frac{\textbf{ns}}{\textbf{pixel}} = 3.83\frac{\textbf{ns}}{\textbf{pixel}} + \frac{X\textbf{cycles}}{150\textbf{MHz}}$$

$X \approx 9$ **single edge or** 18 **dual edge triggers** per red/blue pixel pair.

Clocks speeds above 48MHz are typically only available to microprocessors and FPGA's. With major advancements in scaling and miniaturization, a possible FPGA implementation using Altera's MAX10 and Cypress' HyperRAM allows for sizes below one square inch.

Data Transfer

Given a POI description of one 2 byte integer for location and one 2 byte integer for density, a maximum transmission of 10 POIs per dimension, and a typical camera with a frame-to-frame delay of 1ms **(See Appendix ….)**, the minimum transmission time is:

POI Packet:

$$2\textbf{ bytes(location)} + 2\textbf{ bytes(density)} + 2\textbf{ bytes(header)} = 6\textbf{bytes} = 48\textbf{bits}$$

Minimum frequency:

$$\frac{48\textbf{ bits}}{1\textbf{ ms}} = 48\textbf{ kbps}$$

Popular communication protocols capable of handling this bandwidth are I2C (1Mbps) and SPI (25Mbps)WORK SITED #. In conclusion, the given implementation shown in Figure 8 is a viable micro-hardware implementation capable of image compression required by Tau+.

6.2. Appendix B - Performance Analysis

Performance:

complexity $= a + b + c + d + e + f$

a : image reading

b : smoothing

c : image analysis

d : kalman matrix filter

e : generate prediction

f : update state machine

Approximate Complexity:

$$= n^2 + k(n + k + 1) + 2n + (p + 8m) + mlog(m) + 2s$$

Typical values are:

$$m = \frac{n}{2^8}, p = \frac{n}{2^6}, s = \frac{n}{2^7}, \text{ and } k = \frac{n}{2^6}$$

Condensed Equation:

$$= n^2 + \frac{n^2}{2^6} + 2n + \frac{nlog(\frac{n}{2^8})}{2^8} = A + B + C + D$$

A : n^2 (Image reading)

B : $\frac{n^2}{2^6}$ (smoothing)

C : $2n$ (root complexity)

D : Negligible for typical frame dimensions $2^9 \leq n \leq 2^{12}$:

$$0.0027n \leq \frac{nlog\left(\frac{n}{2^8}\right)}{2^8} \leq 0.0108n$$

Root complexity is $O(2n) = O(n)$**.**