This ICCV paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the version available on IEEE Xplore.

Visual Odometry for Pixel Processor Arrays

Laurie Bose[1] Jianing Chen[2] Stephen J. Carey[2] Piotr Dudek[2] Walterio Mayol-Cuevas[1]
[1]University of Bristol, Bristol, United Kingdom
[2]University of Manchester, Manchester, United Kingdom

Abstract

We present an approach of estimating constrained ego-motion on a Pixel Processor Array (PPA). These devices embed processing and data storage capability into the pixels of the image sensor, allowing for fast and low power parallel computation directly on the image-plane. Rather than the standard visual pipeline whereby whole images are transferred to an external general processing unit, our approach performs all computation upon the PPA itself, with the camera's estimated motion as the only information output. Our approach estimates 3D rotation and a 1D scale-less estimate of translation. We introduce methods of image scaling, rotation and alignment which are performed solely upon the PPA itself and form the basis for conducting motion estimation. We demonstrate the algorithms on a SCAMP-5 vision chip, achieving frame rates >1000Hz at ∼2W power consumption.

1. Introduction

PPA sensors consist of a parallel array of neighbor-connected processing elements, each of which features light capture, processing and storage capabilities allowing for various image processing tasks to be efficiently performed directly on the focal plane itself [4],[15],[9],[1],[2],[19].

This is in contrast to traditional camera sensors in which each pixel element is only capable of light capture, meaning that whole images must be transferred to a separate external device for processing. PPA devices need only output specific information e.g. feature point locations or even higher level events such as ego-motion coordinates. This results in requiring little bandwidth, energy consumption and allowing for high speed visual processing.

Low power and bandwidth consumption draw comparison between such PPA cameras and Dynamic Vision Sensors (DVS) [14]. These devices produce an asynchronous stream of events encoding brightness changes incurred at specific pixels, allowing very fast response times and dynamic ranges. DVS have been demonstrated in agile UAVs maneuvers [17], orientation tracking [7], [11], visual odometry [13], 6DOF tracking and SLAM [12],[20]. However, DVS based systems still require the actual processing of visual data to be conducted on a separate external device, often via having to reconstruct a whole image. In contrast, PPA cameras are capable of carrying out all processing within the sensor device itself.

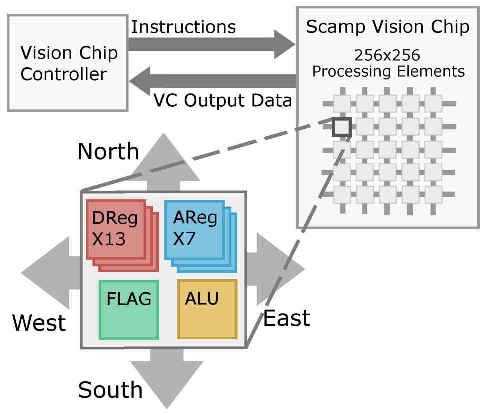

Figure 1: The architecture of SCAMP-5. Each processing element / pixel has own storage and processing capabilities.

In this work, we investigate one potential approach for estimating camera ego-motion on PPA devices, estimating both rotation and motion along the camera's forward axis. Specifically our approach involves conducting image alignment between the current image and a previously acquired key-frame image. This involves determining the set of image transformations (in terms of image rotation, scaling and translation) needed to align these two images. The camera orientation and translation along the camera axis relative to the key-frame is then deduced from these transformations. The estimation process is conducted entirely upon the SCAMP-5 camera [2] at rates that can exceed 1000Hz.

Section 2 introduces the SCAMP-5 vision system. Section 3 and Section 4 present our approaches for image scaling and rotation, which form the basis for our image stabilization based approach for motion estimation presented in Section 5. Results vs accurate groundtruth as well as outdoor explorations are then presented in Section 6.

2. Hardware Architecture

The Scamp-5 system [16],[2] is a novel vision sensor following a Cellular Processor Array (CPA) design, with the focal plane of the sensor consisting of a grid of 256x256 processing elements (PEs), each capable of storage and processing visual data. The architecture of SCAMP-5 is illustrated in Figure 1. Every PE in the array contains 7 analog registers (AReg), and 13 1-bit digital registers (DReg), effectively allowing for the direct in-plane storage and manipulation of 7 different grayscale images and 13 binary images, each of 256x256 resolution. Various parallel operations can be conducted upon these stored images, such as the addition and subtraction of grayscale (analog) images, and bit-wise operations between binary (digital) images. It should be noted however that the content of analog registers is subject to decay over time, making them unsuitable to store information longer than a couple of seconds. We denote images by functions, with grayscale images by $A_i : \Omega \to \mathbb{R}$, and binary images by $D_i : \Omega \to \{0, 1\}$ where $\Omega \subset \mathbb{N}^2$ is the set of all pixel coordinates in the 256x256 array.

Additionally each PE is capable of communication with its four neighbors, enabling each PE to retrieve and copy data from its neighboring elements into its own registers. This allows images stored in the PE registers to be shifted around horizontally and vertically within the image plane along the "North","South","East" and "West" directions as indicated in Figure 1. For brevity we let I^N denote the image resulting from shifting an image I (being either greyscale of digital) in the "North" direction, such that $I^N(x, y) = I(x, y - 1)$. I^S, I^E and I^W are similarly defined for the other directions.

Finally each PE also contains a digital 1-bit activity "FLAG" register which controls whether a PE executes the latest received instruction, thus enabling conditional execution. The value of FLAG can be set by a number of different mechanisms including selection of an arbitrary-shaped rectangle of processing elements, or based upon thresholding the content of a given register (either analog or digital).

Featuring both neighbor communication and conditional execution makes it possible to perform many visual processes directly on the focal plane in an efficient parallel manner. This enables the PPA device to capture and process visual information at very high frame rates, with specific tasks such as simple target identification achievable at frame rates exceeding 100,000 fps given sufficient lighting conditions [2].

3. In-Plane Image Scaling

This section describes the method used to achieve scaling of images stored on SCAMP-5 hardware, examples of which can be seen in Figure 2. Scaling involves compressing or expanding image content in order to fit into a smaller or larger array of pixels, while simultaneously preserving as much information as possible.

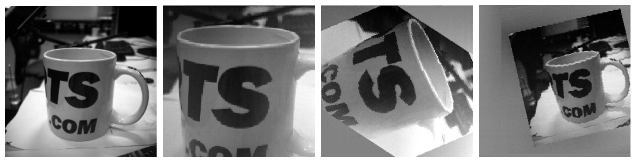
Figure 2: Examples of in-plane image transformations performed on SCAMP-5. L-R: original, zoom-in, zoom-in+CWrotate and zoom-out+CCWrotate+Rshift

The value of each pixel in such a scaled image consists of a weighted average of some subset of pixels taken from the original image. However such an arbitrary mapping between the pixels of one image to those of another cannot directly be performed on SCAMP-5 hardware, on which direct communication can only occur between neighboring processing elements as described in Section 2.

Instead, our approach performs nearest neighbor image scaling incrementally, where at each step two columns and rows of pixels are either duplicated (for up-scaling) or removed (for down-scaling) from the image. Specifically one column from both the right and left halves of the image, and one row from both the top and bottom halves, thus either increasing or decreasing the image's width and height by two pixels.

3.1. Column and Row Manipulation Order

In order to produce a uniformly scaled image different rows and columns must be manipulated at each scaling step. Let us examine the horizontal downscaling of the right hand half of a captured image as shown in Figure 3. Let $n \in \mathbb{N}$ denote the current scaling step and then consider which column of pixels to remove in the first step $n = 1$. Though somewhat arbitrary, one intuitive choice would be the central column of pixels (column 64) indicated in red. When performing subsequent scaling steps however more consideration must be given into what column of pixels to next remove. To achieve uniform scaling, columns of pixels should be removed in an order such that the resulting loss of content is spread equally across the image. Thus the column to remove at each step should be that furthest from the locations at which any columns have been previously removed, and also furthest from the edges of the half-image to avoid scaling artifacts. Thus on the second scaling step ($n = 2$) one potential candidate to remove would be column 32 as shown in blue in Figure 3, after which column 96 ($n = 3$), column 16 ($n = 4$) and column 80 ($n = 5$) would be removed. Thus the ordered indexes of the columns to remove under this scheme follow the pattern of 64, 32, 96, 16, 80 and so on. This sequence of indexes follows the pattern ob-

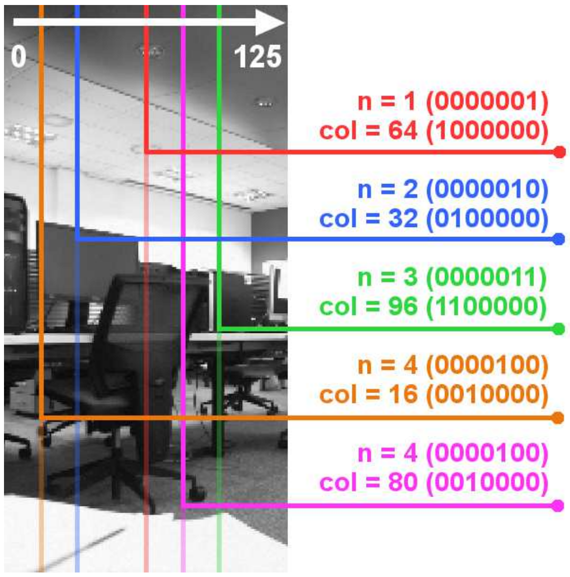

Figure 3: An illustration of which column of pixels is selected for manipulation at each scaling step.

tained by reversing the order of the seven least significant bits of the scaling step number n as shown in Figure 3. Let $r : \mathbb{N} \rightarrow \mathbb{N}$ represent the function which performs this bit reversal process. Then for any image scaling task the bit reversal of the current scaling step $r(n)$ is used to determine which rows and columns to remove or duplicate in each of the half images. It should be noted the sequence resulting from this bit reversal does not account for the current size of the image being scaled, leading to artifacts when significant image scaling has taken place. It is possible to refine this sequence separately for both image up-scaling and down-scaling to account for this issue. However, for our purposes of detecting camera motion between frames captured at high fps, significant image scaling is not required.

3.2. Column and Row Duplication and Removal

The process of removing rows and columns from the image involves shifting the appropriate section of the image to overwrite the desired row or column. For example to eliminate the i^{th} column from the right hand half image, all columns from i to 256 are selected such that only their FLAG registers are active. With these columns selected, a one pixel shift to the left (WEST) is then performed on the desired image. This causes each selected PE to take the pixel value it is storing for the image in question and replace it with the pixel value from its EAST neighbor, resulting in the elimination of the i^{th} column. Insertion of a duplicate column in the right hand half image is similarly achieved but by shifting the image once to the right (EAST). Elimination and duplication of columns from the left hand half image, and rows from the top and bottom half images are

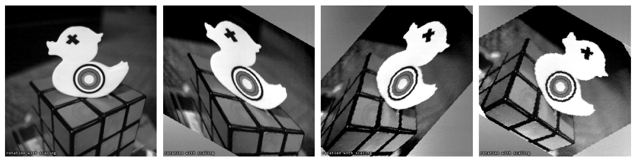

Figure 4: Example of using three consecutive shear operations to perform image rotation.

achieved in a similar fashion.

4. In-Plane Image Rotation

It is also possible to perform image rotation operations directly upon images stored on SCAMP-5, examples of which can be seen in Figure 2.

Similar to image scaling, image rotation involves mapping each pixel's value to a weighted average of a subset of pixels from the original image. We again restrict ourselves to only conducting nearest neighbor image rotation, in which each pixel's value is directly mapped to that of another pixel from the original image.

Sequentially manipulating pixels individually would require a vast number of operations to rotate the entire image, making such an approach infeasible for real-time applications. Instead we first describe a method of conducting efficient parallel image shearing, and then demonstrate how image rotation can be conducted by performing a combination of these shear operations.

4.1. In-Plane Image Shearing

A shear transformation is a linear mapping in which each point undergoes a displacement along a given axis, proportional to the point's signed distance from another axis orthogonal to the first. Consider the 2D shear parallel to the x axis associated with the matrix below.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & \alpha \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

This shear matrix results in each point (x, y) being mapped to $(x', y') = (x + \alpha y, y)$.

Consider applying this shear to the top half of a SCAMP-5 image. Clearly the number of horizontal shifts that need to be applied to each row increases linearly with y according $|\alpha y|$, with the sign of αy dictating the shift direction. Further a minimum number of $|128\alpha|$ horizontal shifts are required to correctly shift the top row. Note that each row of the image needs to be shifted an equal or greater number of times than all rows below it. This fact can be exploited to efficiently perform the shear operation by shifting subsets of contiguous rows in parallel. Specifically subsets of the form of rows $\{i, ..., 256 : 128 < i \leq 256\}$ (i.e. the i^{th} row and all rows above it). This parallel shifting approach allows the entire shear to be performed in the smallest possible number of $|128\alpha|$ shift operations.

Let n denote the current shearing step number, and $\{i_n, ..., 256 : 128 < i_n \leq 256\}$ denote the subset of rows to horizontally shift at the n^{th} step shearing. In order to produce a correctly sheared image the sequence of values $i_1, i_2, i_3...$ must be equally spread across the rows of the half image (as this results in the number of times a given row is shifted being proportional to its location in y). The same bit reversal function r introduced previously for image scaling in Section 3 can be again used to determine a sequence of i_n values ($i_n = r(n)$) that are approximately evenly spread at every scaling step. This allows for shearing to be conducted in a stepwise manner where at each step the intermediate image is sheared as needed.

Horizontal shearing upon the bottom half image and vertical shearing upon the left and right half images are conducted in an a similar fashion, thus allowing arbitrary shearing to be conducted about the center of the image.

4.2. Rotation by Shearing

Consider that the standard 2D rotation matrix (Equation 1) can be pulled apart and expressed in terms of three shear matrices of the form shown in Equation 2.

A rotation by angle θ can thus be performed by a sequence of three shearing operations determined by the matrices of Equation 2. Specifically, two identical horizontal shearing operations and one vertical (however these directions can be flipped). This approach to image rotation was first introduced by [18] for use in computer graphics.

$$\begin{bmatrix} cos(\theta) & sin(\theta) \\ -sin(\theta) & cos(\theta) \end{bmatrix} \tag{1}$$

$$\begin{bmatrix} 1 & -tan(\frac{\theta}{2}) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ sin(\theta) & 1 \end{bmatrix} \begin{bmatrix} 1 & -tan(\frac{\theta}{2}) \\ 0 & 1 \end{bmatrix} \tag{2}$$

By performing shearing operations in the stepwise manner described in Section 4.1 image rotation can likewise be performed incrementally in small rotation steps, as illustrated in Figure 4.

5. Odometry on the focal Plane

This section presents the proposed method of estimating ego-motion on the SCAMP-5 system pictured in Figure 6. The entire process is conducted upon SCAMP-5 itself with only the estimated motion information being communicated to external devices.

Each new camera image is processed into a binary edge image $E : \Omega \rightarrow \{0, 1\}$ as shown in Figure 5, and stored within a single digital register. An iterative image alignment process is then conducted between the latest edge image denoted E and a previously acquired edge image key-frame denoted K. Each iteration tests various image transformations upon both images, in order to find those which result in the best local alignment between the two. Specifically this involves determining the horizontal and vertical 1 pixel shifts to apply to K, and the image rotation and scaling steps to apply to E, which achieve this best alignment. The number and direction of each of these transformations are denoted by the values $\alpha_r, \beta_r, \gamma_r, \lambda_r \in \mathbb{Z}$.

$$\begin{aligned} \alpha_r &: \text{Horizontal shifts applied to } K \\ \beta_r &: \text{Vertical shifts applied to } K \\ \gamma_r &: \text{Rotation steps applied to } E \\ \lambda_r &: \text{Scaling steps applied to } E \end{aligned}$$

The values of $\alpha_r, \beta_r, \gamma_r$ and λ_r determined for the previous frame are used as a prior to initialize this image alignment process. By applying the transformations associated with these previous values fewer iterations are required to achieve image alignment.

The estimated camera pose relative to the pose at which the key-frame K was acquired is then deduced from the transformations determined to result in best image alignment. The values of α_r, β_r and γ_r are used to determine the camera's yaw, pitch and roll, while λ_r is related to translation along the camera's forward axis. This deduction is made under the assumption that there is no camera translation parallel to the image plane, restricting the range of camera motions under which this estimation is valid. Further, let us use $\alpha_k, \beta_k, \gamma_k, \lambda_k$ to denote the cumulative values of α_r, β_r, γ_r, λ_r up until the time of acquisition of the latest key-frame K. The estimated orientation and distance traveled since the initial key-frame acquired at the start of the program can be deduced from the combined values of $\alpha_k + \alpha_r$, $\beta_k + \beta_r$, $\gamma_k + \gamma_r$ and $\lambda_k + \lambda_r$.

A new key-frame is acquired to replace the current K whenever the number of times a certain transformation needs to be applied to achieve image alignment between E and K exceeds a specific threshold. This is to ensure that there will be sufficient overlapping information between the K and E for the following camera image to perform image alignment. These thresholds are denoted by $T_\alpha, T_\beta, T_\gamma, T_\lambda \in \mathbb{N}$ for $\alpha_r, \beta_r, \gamma_r, \lambda_r$ respectively.

These concepts are now described in greater depth in the following subsections.

5.1. Binary Edge Extraction

The edges within an image typically contain a great deal of information regarding the scene in view, and are those parts of the image most sensitive to changes in camera pose. As has been demonstrated in many prior works [6],[5],[10],[8] such edges can provide sufficient information to determine camera motion. Additionally the edges extracted from an image can be stored within a single binary image upon SCAMP-5 providing a good fit both to the

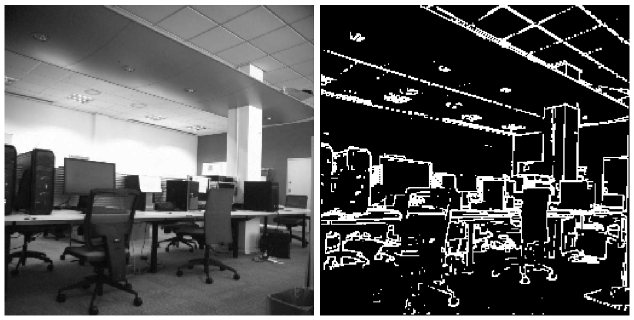

Figure 5: An example of edge extraction on SCAMP-5.

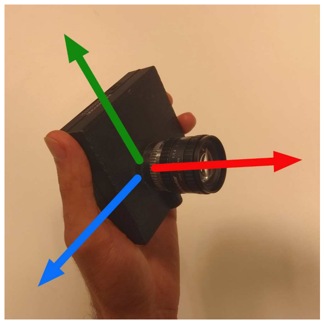

Figure 6: Left : SCAMP-5 camera with yaw (green), pitch (blue) and roll (red) axi.

hardware resources available on SCAMP-5 and for the purposes of determining camera motion.

A simple edge detection method is used to process each camera image $C : \Omega \rightarrow \mathbb{R}$ into a binary edge image $E : \Omega \rightarrow \{0, 1\}$. This involves generating the absolute difference image $D_V(\mathbf{x}) = |C(\mathbf{x}) - C^W(\mathbf{x})|$ between the current camera image and a horizontally shifted copy of itself. Pixels of high value within D_V then indicate the presence of vertically orientated edges. Similarly the absolute difference image $D_H(\mathbf{x}) = |C(\mathbf{x}) - C^S(\mathbf{x})|$ representing horizontally orientated edges is also generated. These two differences images are then combined into a single image $D_E(\mathbf{x}) = D_H(\mathbf{x}) + D_V(\mathbf{x})$, which is then thresholded to a binary image of edge pixels $E : \Omega \rightarrow \{0, 1\}$ such that.

$$E(\mathbf{x}) = \begin{cases} 0 & \text{if } D(\mathbf{x}) < \delta \\ 1 & \text{if } D(\mathbf{x}) > \delta \end{cases}$$

Where $\delta \in \mathbb{Z}$ is a specified edge detection threshold.

5.2. Image Alignment Overview

The implemented image alignment process is performed iteratively, where at each step a number of potential transformations to apply to either E_T or K_T (copies of E and K) are evaluated to determine which would improve the alignment between the two images. Those transformations that improve alignment are then applied to their respective images. This process is then repeated for the updated E_T and K_T images. Given sufficient iterations, this results in E_T and K_T converging to a pair of aligned edge images. The camera pose relative to the pose at which the key-frame K was acquired is determined from the transformations applied to E_T and K_T.

It should be noted that due to the high frame-rate capabilities of SCAMP-5, it is possible to run at a rate at which only a single iteration is required to achieve best image alignment even under violent camera motion. This is due to there being only very small image changes from one frame to the next at high frame-rates.

To evaluate if a given transformation would improve the alignment between E_T and K_T requires a method of measuring the quality of the alignment between two different edge images. This is performed by counting the number of overlapping edge pixels between the two images. For two edge images E_1, K_2 this is performed by first generating the image $E_O(\mathbf{x}) = \text{AND}(E_1, K_2)$ of overlapping edge pixels, using an AND operation to combine E_1 and K_2. A global sum of all pixels of the image E_O is then performed. The higher the resulting sum value the greater the overlap between the edges of two images, indicating a better image alignment.

The various image transformations tested at each iteration of the image alignment process along with their interpretation in terms of change in camera pose are described in the following subsections. This entire alignment process is listed in Algorithm 1.

5.3. Image Transformations : Yaw And Pitch

The transformations tested upon K_T at each iteration consist of four different translations $(K_T^N, K_T^S, K_T^E, K_T^W)$, shifting K_T either up, down, left or right across the image plane by a single pixel. The values α_r and β_r denote the horizontal and vertical shifts applied to K_T determined to achieve best alignment with E_T. Under the constraints upon camera motion previously described, these two values α_r and β_r are proportional to the current yaw and pitch of the camera relative to the orientation at which the current key-frame K was acquired. Combined with the cumulative values α_k and β_k the current estimated yaw $\alpha \in \mathbb{R}$ and pitch $\beta \in \mathbb{R}$ of the camera in radians is then given by

$$\alpha = f(\alpha_r + \alpha_k)/M$$
$$\beta = f(\beta_r + \beta_k)/M$$

where where f is the horizontal field of view of the camera (being the same as the vertical), and with $M = 256$ being the width and height of the PE grid of SCAMP-5.

5.4. Image Transformations : Roll

Additionally two rotation operations are tested upon E_T each iteration. These consist of rotating the image by a sin-

Algorithm 1 Edge Image Alignment

```
$E_T = E$    // create a copy of $E$
$K_T = K$    // create a copy of $K$
Translate $K_T$ according to $\alpha_r$ and $\beta_r$
Rotate and scale $E_T$ according to $\gamma_r$ and $\lambda_r$
for $n = 1$ to $N$ do
    $S_\alpha = \{K_T, K_T^N, K_T^S\}$
    $K_T = \arg\max_{k \in S_\alpha}(gsum(\text{AND}(E_T, k)))$
    $S_\beta = \{K_T, K_T^E, K_T^W\}$
    $K_T = \arg\max_{k \in S_\beta}(gsum(\text{AND}(E_T, k)))$
    $S_\gamma = \{E_T, E_T^{R+}, E_T^{R-}\}$
    $E_T = \arg\max_{e \in S_\gamma}(gsum(\text{AND}(e, K_T)))$
    $S_\lambda = \{E_T, E_T^{S+}, E_T^{S-}\}$
    $E_T = \arg\max_{e \in S_\lambda}(gsum(\text{AND}(e, K_T)))$
    Update $\alpha_r, \beta_r, \gamma_r, \lambda_r$ based on selected transforms
end for
if $|\alpha_r| > T_\alpha \vee |\beta_r| > T_\beta \vee |\gamma_r| > T_\gamma \vee |\lambda_r| > T_\lambda$ then
    // update key-frame and transformation values
    $K = E$
    $\alpha_k = \alpha_k + \alpha_r$    $\beta_k = \beta_k + \beta_r$
    $\gamma_k = \gamma_k + \gamma_r$    $\lambda_k = \lambda_k + \lambda_r$
    $\alpha_r, \beta_r, \gamma_r, \lambda_r = 0$
end if
```

gle step in the clock-wise direction and the second rotating by a single step anti clock-wise, producing the rotated images E_T^{R+} and E_T^{R-}. The number and direction of the rotation steps determined to achieve best alignment between E_T and K_T is denoted by γ_r. Intuitively γ_r is proportional to the camera's current roll angle relative to key-frame K. Combined with the value of cumulative rotation steps γ_k the current estimated roll angle γ of the camera in radians is given by $\gamma = \gamma_r + \gamma_k$.

5.5. Image Transformations : Translation

Finally two scaling operations are tested upon E_T at each iteration of the image alignment process. One upscaling the image by a single step, duplicating two rows and columns and a second down-scaling by a single step by removing two rows and columns, producing the images E_T^{S+} and E_T^{S-} respectively. The number and direction (up vs down scaling) of scaling steps that result in best alignment between E_t and K_t is denoted by λ_r.

Generally forward camera motion results in an increase in the size of objects observed within the camera's image, backwards motion on the other hand results in a decrease in object scale. Thus the scaling applied to the image E_T deemed to produce the best alignment with K_T can be used to infer the presence of forward or backward camera translation that has occurred since the acquisition of K. It must be noted that the rate at which objects within the camera's image change in size from forward and backward translation is inversely proportional to their distance from the camera. Thus the reported rate of forward and backward motion is both scaleless and also dependent upon the camera's distance from the objects in view, rather than being proportional to the actual true rate of forward or backward translation.

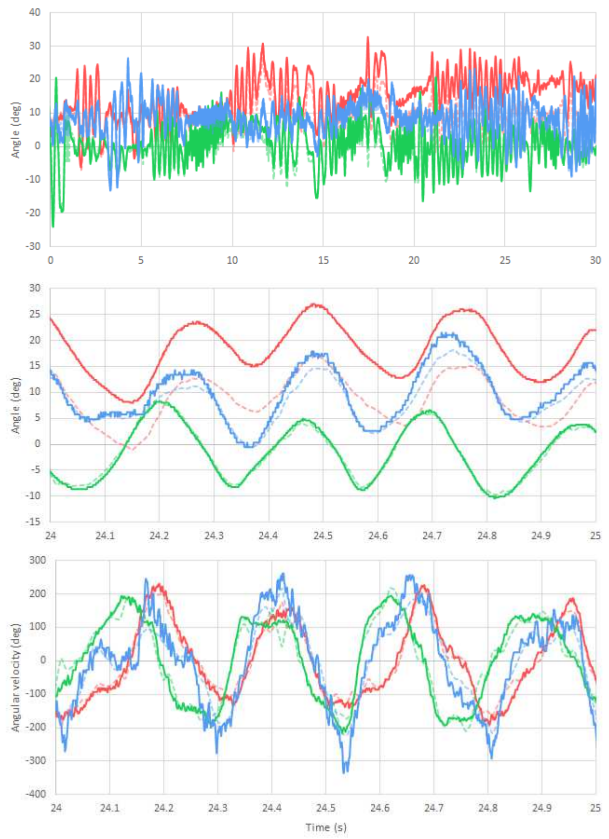

Figure 7: Rotational estimates under rapid shaking motion at 1000fps. Yaw, Pitch and Roll are shown in Red,Green and Blue respectively. Dashed lines are the ground-truth provided by motion capture. See text for details.

Thus from this the forward and backward translation along the camera's axis relative to the pose of the latest keyframe K can be inferred from λ_r. A scaleless measure λ of the total translation along the camera axis is then given by $\lambda = \lambda_r + \lambda_k$.

6. Experiments and Results

The accuracy of the proposed method of estimating orientation and forward and backward translation was evaluated by subjecting the camera to various sequences of hand held motion and comparing the estimated motion to

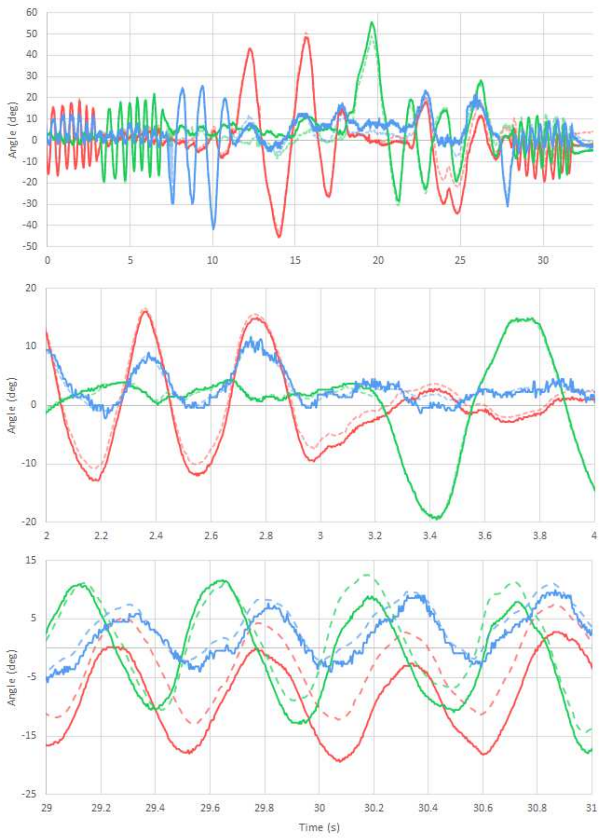

Figure 8: Rotational estimates under steady motion at 500fps. Yaw, Pitch and Roll are shown in Red,Green and Blue respectively. Dashed lines are the ground-truth provided by motion capture. See text for details.

ground-truth from an OptiTrack motion capture system. The SCAMP-5 system was connected to a laptop to record the estimated motion, no other information was transferred. The key-frame acquisition thresholds were taken to be $T_\alpha = 60, T_\beta = 60, T_\gamma = 30, T_\lambda = 15$ for all sequences. The time taken to conduct edge detection as described in Section 5.1 averaged around 50μs, while the time taken to conduct a single iteration of the image alignment search described in Section 5.2 averaged 300μs. While the time taken for these two processes remained roughly constant the time taken to apply the prior transformations to copies of the current image and key-frame varied depending upon the magnitude of the transformations meaning that the frame rate was variable during execution.

The proposed method was able to achieve frame rates in excess of 1000Hz upon the SCAMP-5 hardware (varying between 1000-1500Hz), consuming less than 2W of power in the process. However due to the associated low exposure time sufficient lighting was required to obtain images usable for the estimation process. A small construction site lamp was used to illuminate a small area inside of the motion tracking volume to conduct experiments at this high frame-rate. It should be noted that daytime outdoor scenes have sufficient illumination to operate at such frame-rates even on cloudy *English* overcast days. At such high frame-rates it was sufficient to perform only a single iteration per frame of the image alignment process as described in Section 5.2.

In all sequences, it was observed that the proposed approach produced highly accurate estimates of the camera's angular velocity. Figure 7 shows an example of results of the algorithm working at 1000Hz in which the camera was subject to violent hand held motion (5 shakes per second). A closer examination of this sequence over a one second interval is given by Figure 7 middle. From this it can be seen that in this scenario the estimated yaw has drifted from the ground truth while the pitch and roll estimations remain highly accurate. Figure 7 bottom shows the estimated rates of yaw,pitch and roll for the same one second interval, which are seen to closely follow the ground-truth angular velocity. A total of ten sequences of such rapid shaking camera motion were evaluated, ranging between 30 to 120 seconds each. The average rate of angular drift from the ground-truth orientation was found to be 0.14 deg/s. The standard deviation of the estimated angular velocity from that derived from motion capture was measured at 18.03 deg/s.

Experiments were also conducted at a lower frame-rate of 500Hz allowing a greater range of orientations changes due to the increased exposure time and hence reduced need for sufficient scene illumination. At this reduced frame-rate five iterations were used in the image alignment process of Section 5.2. The estimated orientation closely followed the ground truth in many such 500Hz sequences such as that shown in Figure 8. Magnified views of the estimations near the start and end of this sequence are given in Figure 8 middle and Figure 8 bottom illustrating the increasing deviation from the ground truth over time. Again the accuracy was evaluated over ten such 500Hz sequences with the average rate of drift being 0.19 deg/s. This is greater than that of the rapid shaking motion sequences at 1000Hz largely due to the larger range of motion leading to a greater number of key-frames being necessary, each of which may introduce additional deviation from the ground-truth.

For all sequences the estimated translational motion was scaled such that the magnitude of the minimum and maximum estimated translation matched that of the ground-truth in order to perform a direct comparison between the two. As with the estimation of the camera's rotation, the estimated translation always exhibited increasing drift away from ground-truth over time. Additionally the fact that the estimated rate of translational change is dependent upon the camera's distance from objects within it's view also contributed to this deviation. Despite this however, we see that

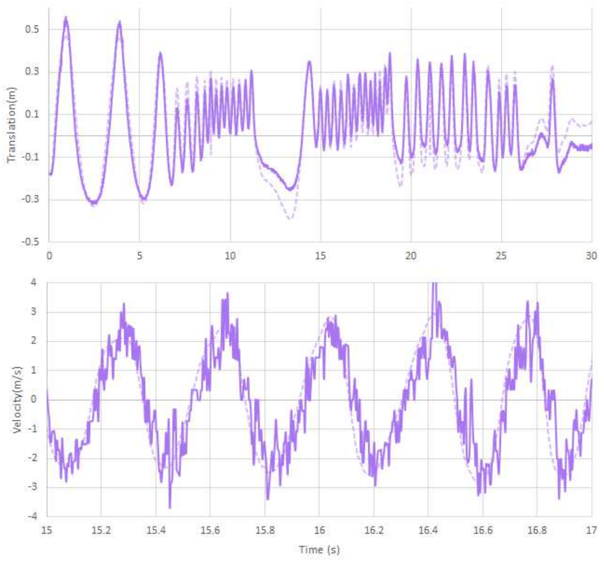

Figure 9: Top: forward and backward translation estimation. Bottom: translation rate estimation. Goundtruth is shown by the dashed lines.

the produced translational estimate, once scaled, is highly correlated with the ground-truth motion as shown in Figure 9 with an average drift of 0.08 meters/s over 10 sequences between 30 to 120 seconds in length of near constant translational hand held motion. Note that due to the nature of the process employed for detecting translational motion (based upon comparing discretely scaled images to a key-frame), a certain degree of motion must occur before a definite change in the estimated translation is made. This fact can also be observed in Figure 9 in which the estimated translation is seen to change in small discrete increments. Due to the discrete nature of this estimation the raw estimated velocity along the camera's axis features a large degree of jitter and discontinuous jumps, however the estimation still very closely follows the trend of the ground-truth velocity.

In addition to these motion tracked sequences we tested our approach outdoors following simple paths starting and ending in the same location. Figure 9 shows one such path taken around the perimeter of a 53m x 30m court yard (the total path length thus being approximately 166m) with the camera facing forward in the direction of motion produces a scaleless estimation of forward motion which is affected greatly by the scene in view. This leads to two sides of the rectangular path as seen in Figure 9 being estimated at different lengths to one another. The accuracy of these outdoor sequences in terms of final distance from start location divided by total estimated path length was found to be 10% on average. This is on the same order of error wrt recent visual odometry systems even when using stereo imagery [3].

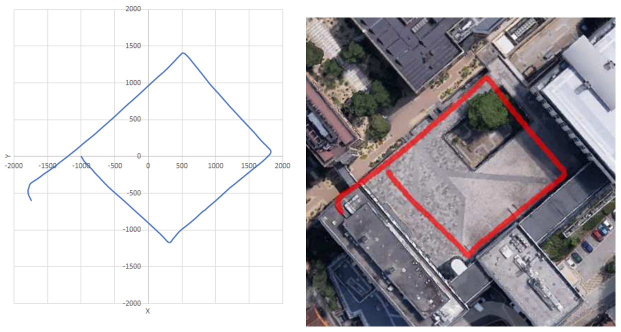

Figure 10: The estimated path of the sensor following a long rectangular path.

Though these outdoor results are qualitative in nature, they still provide a useful indication of the approaches accuracy in practical scenarios.

7. Conclusions and Future Directions

This work investigated an image alignment based approach for conducting constrained ego-motion estimation upon PPA cameras, specifically implemented on the SCAMP-5 vision system. We presented methods of conducting in-plane image transformations and their use in estimating the camera's rotational motion, along with tracking forward and backward translational motion parallel to the camera's axis. Evaluation of the proposed approach for rotational motion estimation was conducted against ground-truth from motion tracking, revealing highly accurate results for derived angular velocity estimation. At 1000Hz under rapid rotational hand held motion the raw estimations of yaw, pitch and roll were seen to drift at an average rate of 0.14 deg/s, while the standard deviation of angular velocity error was measured at 18.03 deg/s. The estimation of translational camera motion, despite being scaleless was also seen to closely follow the ground-truth trend with a drift of 0.08 m/s once scale corrected. The demonstrated accuracy and high frame rate capabilities illustrate the great potential such a device carries in terms of application to areas such as high speed robotics.

Data Access and Acknowledgements

Supported by UK EPSRC EP/M019454/1 and EP/M019284/1. The nature of the PPA means that the data used for evaluation in this work is never recorded.

References

[1] S. J. Carey, D. R. Barr, B. Wang, A. Lopich, and P. Dudek. Mixed signal simd processor array vision chip for real-time image processing. *Analog Integrated Circuits and Signal Processing*, 77(3):385–399, 2013.

[2] S. J. Carey, A. Lopich, D. R. Barr, B. Wang, and P. Dudek. A 100,000 fps vision sensor with embedded 535gops/w 256$\times$ 256 simd processor array. In *VLSI Circuits (VLSIC), 2013 Symposium on*, pages C182–C183. IEEE, 2013.

[3] A. Comport, E. Malis, and P. Rives. Real-time quadrifocal visual odometry. *International Journal of Robotics Research*, 29:245–266, 2010.

[4] P. Dudek and P. J. Hicks. A general-purpose processor-per-pixel analog simd vision chip. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 52(1):13–20, 2005.

[5] J. Engel, V. Koltun, and D. Cremers. Direct sparse odometry. *arXiv preprint arXiv:1607.02565*, 2016.

[6] J. Engel, T. Schöps, and D. Cremers. Lsd-slam: Large-scale direct monocular slam. In *European Conference on Computer Vision*, pages 834–849. Springer, 2014.

[7] G. Gallego and D. Scaramuzza. Accurate angular velocity estimation with an event camera.

[8] A. P. Gee and W. Mayol-Cuevas. Real-time model-based slam using line segments. In *International Symposium on Visual Computing*, pages 354–363. Springer, 2006.

[9] M. Ishikawa, K. Ogawa, T. Komuro, and I. Ishii. A cmos vision chip with simd processing element array for 1 ms image processing. In *Solid-State Circuits Conference, 1999. Digest of Technical Papers. ISSCC. 1999 IEEE International*, pages 206–207. IEEE, 1999.

[10] J. Jose Tarrio and S. Pedre. Realtime edge-based visual odometry for a monocular camera. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 702–710, 2015.

[11] H. Kim, A. Handa, R. Benosman, S.-H. Ieng, and A. J. Davison. Simultaneous mosaicing and tracking with an event camera. In *BMVC*, 2014.

[12] H. Kim, S. Leutenegger, and A. J. Davison. Real-time 3d reconstruction and 6-dof tracking with an event camera. In *European Conference on Computer Vision*, pages 349–364. Springer, 2016.

[13] B. Kueng, E. Mueggler, G. Gallego, and D. Scaramuzza. Low-latency visual odometry using event-based feature tracks. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016.

[14] P. Lichtsteiner, C. Posch, and T. Delbruck. A 128$\times$128 120 db 15μ s latency asynchronous temporal contrast vision sensor. *IEEE journal of solid-state circuits*, 43(2):566–576, 2008.

[15] A. Lopich and P. Dudek. A simd cellular processor array vision chip with asynchronous processing capabilities. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 58(10):2420–2431, 2011.

[16] J. N. Martel and P. Dudek. Vision chips with in-pixel processors for high-performance low-power embedded vision systems. 2016.

[17] E. Mueggler, N. Baumli, F. Fontana, and D. Scaramuzza. Towards evasive maneuvers with quadrotors using dynamic vision sensors. *European Conference on Mobile Robots (ECMR)*, 2015.

[18] A. W. Paeth. A fast algorithm for general raster rotation. In *Graphics Interface*, volume 86, 1986.

[19] J. Poikonen, M. Laiho, and A. Paasio. Mipa4k: A 64$\times$ 64 cell mixed-mode image processor array. In *Circuits and Systems, 2009. ISCAS 2009. IEEE International Symposium on*, pages 1927–1930. IEEE, 2009.

[20] H. Rebecq, T. Horstschafer, G. Gallego, and D. Scaramuzza. Evo: A geometric approach to event-based 6-dof parallel tracking and mapping in real-time. *IEEE Robotics and Automation Letters*, 2016.