PAS: Visual Odometry with Perspective Alignment Search

Andrew Richardson and Edwin Olson

***Abstract*— Visual odometry is typically formulated as a descriptor-based image feature tracking problem, followed by outlier rejection and simultaneous estimation of the scene structure and camera motion. We propose a fundamentally different formulation for the stereo case: a *multi-scale search over pose* to estimate the transformation that best aligns two sparse point clouds in image space. This has three main consequences. First, data association is *descriptorless* and *implicit*, supporting the use of features with indistinct appearance, such as edge features. Second, outlier rejection is subsumed by the use of a robust kernel and a joint feature alignment objective. Third, the method is robust to local minima, in contrast to coarse-to-fine or iterative approaches. This paper details the proposed method, which we call Perspective Alignment Search (PAS), integrated into an edge feature visual odometry system, and an evaluation against a LIDAR-based SLAM solution.**

I. Introduction

This paper focuses on an alternative formulation for visual odometry on mobile robots. Visual odometry is a common method of state estimation where visual features in one or more cameras are tracked in order to reconstruct the camera, and thus robot, motion. Most methods use point feature detectors, such as Harris [1] or FAST [2], and maintain descriptors for these features, such as pixel patches [3] or a sequence of local intensity comparisons [4]. These features are then tracked across time by matching descriptors with a metric such as a distance or ratio threshold [5].

In many environments, these systems are capable of producing highly-accurate results. However, the lack of texture in some environments, such as plain office buildings, can prove problematic for point feature tracking. In these environments, it is beneficial to use more plentiful image features, such as edges, and rely on the unique scene geometry to constrain the matching problem instead of relying on distinct feature descriptors. We will show that a visual odometry system using PAS works well in these environments using descriptorless, joint feature alignment.

Another candidate use of such a descriptorless method is visual localization. Like visual odometry, visual localization is concerned with estimating the pose of the camera. However, in contrast to odometry, localization is concerned with recognizing the surrounding location given a prior map of the environment. A common problem in visual localization is descriptor *lighting invariance*, as visual appearance is subject to changes in lighting. A descriptorless method would be robust to lighting changes, as long as feature locations remain stationary under such conditions. While localization is outside of the scope of this paper, it does motivate this approach.

The authors are with the Computer Science and Engineering Department, University of Michigan, Ann Arbor, MI 48104, USA {chardson,ebolson}@umich.edu http://april.eecs.umich.edu

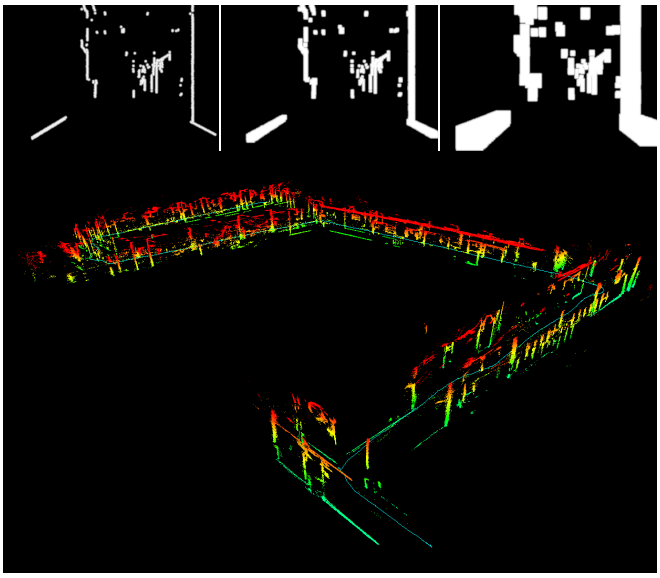


Fig. 1: (Top) PAS translation-image pyramid for a 3-level search with 560 matched edge features. See Figure 3 for explanation. (Bottom) Accumulated edge features from PAS visual odometry on a mobile robot. Individual points colored by height and accumulated only when well-aligned at short range. Camera trajectory in cyan.

In contrast to descriptor-based visual odometry, we propose *Perspective Alignment Search* (PAS), a method for estimating the transformation that best aligns the entire scene, without use of descriptor matching or outlier rejection. PAS is formulated as a multi-scale search over the camera motion. It is a descriptorless method with only implicit data association, a byproduct of computing the transformation that best aligns both observations of the scene. As a result, PAS can leverage simple edge features detected by a 1D gradient filter, whereas a standard descriptor matching approach would fail to find unique matches amongst a set of very similar descriptors.

PAS was inspired by a correlative approach to LIDAR *scan-matching*, which is robust to local minima and posed as a multi-scale search over pose in the planar, 3 DoF case [6], [7]. In contrast to planar LIDAR scan-matching, PAS is enabled by a bound on perspective motion of points in the scene under bounded 3D motion.

The contributions of this work are:

- Adaptation of a multi-scale pose search algorithm from LIDAR scan-matching to the perspective case using the pinhole camera model

- Development of a simple bound on the perspective motion of a point under 3D translation
- Performance evaluation against both synthetic data and a LIDAR SLAM solution on a mobile ground robot

II. Related work

Visual state estimation is often divided between Visual Simultaneous Localization and Mapping (Visual SLAM) methods and Visual Odometry (VO). In Visual SLAM, features are remembered to serve as landmarks for future localization and to develop high-confidence position estimates for them [8], [9]. Typically, such systems use highly-recognizable features for later re-observation. In contrast, VO systems are primarily concerned with short-term feature tracking and motion estimation [10], [11].

While many vision-based state estimation solutions exist, camera-only methods can fail in low-texture environments. Such failures can significantly disrupt the operation of a robot acting autonomously. However, inertial sensors paired with cameras can mitigate these failures, in addition to constraining the original estimation problem. Visual odometry with an inertial sensor is sometimes known as *Visual-Inertial Odometry* (VIO) [12].

Point features are the most common visual feature in VO. These features are typically detected with methods like Harris [1] or FAST [2]. Point feature detectors composed of convolutional filters have also been used for VO [13] and can be learned from ground-truthed data [11]. While point features are, by design, constrained in two dimensions, edge features are only well-constrained in one. Despite this ambiguity, edge features can still be used for visual state estimation. For example, Eade and Drummond showed how to use *edgelets* in a particle-filter SLAM system [14].

Stereo systems are especially well suited to make use of edge features. Vertical edges in stereo rectified images can be easily matched along epipolar lines, similar to dense disparity estimation. As a result, edge feature matching can generate cheap, semi-dense point clouds. This is even true in low-texture, indoor environments where point features are rare. However, unlike point features, edge feature appearance is frequently repeated within an image. This makes it hard to match edge features across time, as in most point feature matching pipelines.

As a set, edge point clouds typically capture unique scene geometry reminiscent of planar LIDAR scans, suggesting that similar methods could be employed for edge features. This was demonstrated by Tomono, who used a variant of Iterative Closest Point (ICP) for point-to-edge alignment in VO with an image space error function [15], [16]. Iterative minimization is subject to local minima, however. This led Tomono to run ICP with multiple random initializations when used for localization.

Local minima are also a problem for LIDAR point cloud alignment, especially when iterative methods are used from poor initializations. For this reason, Pandey et al. described a method for visually bootstrapping a variant of ICP [17]. A different approach was taken by Olson for planar LIDAR scan-matching, who showed that a correlative search is robust to local minima while a multi-resolution formulation exhibits significant runtime improvements [6]. This approach was later extended to more than two levels in the multi-scale search [7]. These works form the basis for PAS.

Other approaches to descriptorless and indistinct feature matching have been explored in the literature. Rodríguez et al. described a method for descriptorless tracking of a set of point features [18]. This method relies on a motion model restricted to the image plane, which assumes that no perspective effects occur apart from an overall scaling term, thus limiting the accuracy of this approximation to high framerates relative to the camera motion. Methods for associating weak features have also been explored. Hsiao et al. described a method to use features with similar descriptors, effectively delaying the match rejection step until the pose hypothesis stage [19]. Unlike the proposed work, Hsiao's method was tailored for the object recognition domain.

III. Method

We wish to compute the camera motion that best-aligns two observations of a scene with a stereo camera pair, and to do so without the use of descriptors. We achieve this by posing the problem as a search over the relative camera motion between two poses, using a multi-scale search representation in a branch-and-bound framework. The key components of this formulation are a bound on the perspective motion of a point, integration into a branch-and-bound search, and an appropriate choice of features.

To make this process efficient and suited for online use, PAS employs an external orientation estimate, such as from an Inertial Measurement Unit (IMU). Given this estimate of the orientation, PAS searches for the unknown translation that best-aligns the observations. While relying on an orientation estimate is a limitation of this method, IMU ICs resulting in orientation drift of a few degrees/minute can cost less than $15 (e.g. InvenSense MPU-6050).

A. Multi-scale search

In a typical bundle adjustment framework, we would compute the Maximum Likelihood (ML) solution to estimate the rigid-body transformations $\hat{\mathbf{T}}_i$ for cameras $i \in \{1, \dots n\}$, which convert a point from the body frame to the world frame, in addition to the positions $\hat{\mathbf{X}}_j = [x_j, y_j, z_j]^T$ for 3D points $j \in \{1, \dots m\}$, defined in the world frame. Here the point $\hat{\mathbf{X}}_j$ may have an associated observation in image i, denoted $\mathbf{z}_{i,j}$. Then, the ML solution corresponds to:

$$\underset{\hat{\mathbf{T}}_i, \hat{\mathbf{X}}_j}{\arg\min} \sum_{i,j} \|\mathbf{z}_{i,j} - project(\mathbf{K}, \hat{\mathbf{T}}_i^{-1} \hat{\mathbf{X}}_j)\|^2 \qquad (1)$$

where $\mathbf{K}$ is a typical skew-free camera matrix with focal lengths f_x, f_y and focal centers c_x, c_y:

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \qquad (2)$$

In the stereo case, we can estimate the positions of image features from a single pair of images. Like Tomono [15],

we refer to each image pair as a *stereo frame*. We can then *redefine* $\hat{\mathbf{X}}_j$ to be the *camera frame* position in the current stereo frame, and can compute the reprojection error against a previous observation via:

$$\|\mathbf{z}_{i-1,j} - project(\mathbf{K}, \hat{\mathbf{T}}_{i-1}^{-1}\hat{\mathbf{T}}_i\hat{\mathbf{X}}_j)\|^2 \quad (3)$$

Because we take the position $\hat{\mathbf{X}}_j$ to be the triangulated point in the current stereo frame, the only unknowns are the correspondences between current and previous observations, as well as the camera poses. If sequentially-estimating the camera motion, as in the visual odometry problem, we need only estimate a single relative transformation, $\hat{\mathbf{T}}$, which can be parameterized by a rotation $\hat{\mathbf{R}}$ and translation $\hat{\mathbf{t}}$ as $\hat{\mathbf{T}} = [\hat{\mathbf{R}}, \hat{\mathbf{t}}; \mathbf{0}, 1]$. Using an external estimate of $\hat{\mathbf{R}}$, PAS searches over a discrete set of candidate translations to compute the ML estimate of $\hat{\mathbf{t}}$.

PAS is formulated as a branch-and-bound search with a multi-scale representation and a fixed depth. The search is parameterized by the number of levels in the multi-scale search, and the difference in translation, s, between adjacent leaf nodes. The nodes are spaced at integer multiples of s in each dimension and nodes at Level l have at most 3 child nodes in each dimension at Level $l-1$ (up to 27 child nodes for 3D translation). Thus, the node spacing on each axis is $s3^l$. A one-dimensional illustration of the tree topology is shown in Figure 2.

The search begins by initializing all nodes at the top level of the search to cover the search range. Nodes are scored using a lookup table appropriate for their search level, where higher scores are better, and added to a max heap. At Level 0, the lookup table is simply an image containing a finite-range, or *robust*, quadratic kernel rendered for each matched feature in the reference image. Higher-level lookup tables specify upper bounds on the kernel value for points under some unknown translation up to $\pm\Delta x_l$. In this way, a single lookup at Level l can rule out entire branches of the search tree, greatly speeding up the search.

Once the initial set of scored nodes has been added to the max heap, the search proceeds by removing the best node from the heap and *expanding* it, creating scored child nodes and placing them in the heap. If the best node is at the maximum search depth, Level 0, the search is complete – no other nodes at Level 0 have higher scores, nor do any nodes exist whose children could yield higher scores.

Given a lookup table at Level l, the score for a node with translation $\hat{\mathbf{t}}$ can be computed by projecting all translated points into the lookup table.

$$score(l, \hat{\mathbf{t}}) = \frac{1}{n}\sum_{j=1}^{n} lookup(l, project(\mathbf{K}, \hat{\mathbf{R}}\hat{\mathbf{X}}_j + \hat{\mathbf{t}})) \quad (4)$$

where the function $lookup(l, \mathbf{p_{xy}})$ simply loads the value from lookup table l at position $\mathbf{p_{xy}}$ determined by the projection of the transformed point.

Lookup tables are generated for a single reference camera, e.g. the left camera. This results in comparable accuracy and a significant runtime reduction. The Level 0 lookup table is generated by rendering a kernel at each feature position using the max operator, where the kernel is scaled such that higher values are better. For a feature which projects to a given pixel location $\mathbf{p_{xy}}$ and a kernel whose value changes monotonically with distance, looking up the value at $\mathbf{p_{xy}}$ corresponds to selecting the nearest observation $\mathbf{z}_{i-1,j}$, computing the distance between $\mathbf{p_{xy}}$ and $\mathbf{z}_{i-1,j}$, and evaluating the kernel function for that distance. Because the kernel values are pre-rendered into the lookup table, the score for a given pose can be computed without computing the data association between observations and projected points. We use a quadratic kernel with a fixed radius, computing the negative log-likelihood for points within range of an observation, and scale the kernel to fill the full range using 8-bit images.

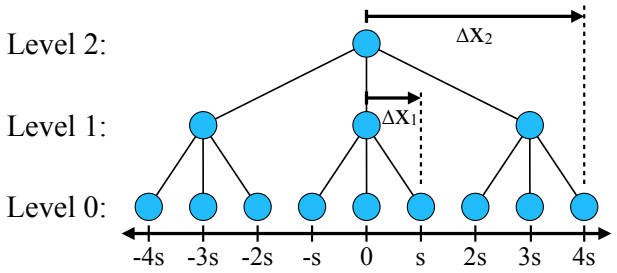


Fig. 2: Illustration of the PAS search tree in one dimension along the X axis. Nodes at Level 0 are positioned at integer multiples of the minimum resolution s. Nodes at Level 1 and above have at most 3 child nodes per dimension. The parameter Δx corresponds to the maximum position offset for any child node.

At higher levels in the search, these lookup tables store upper-bounds on the score for any child nodes. In the planar, LIDAR scan-matching problem [6], [7], these tables can be computed by filtering the Level l table with a max operator whose total width corresponds to the translation between nodes at Level $l+1$. In the perspective case, the search space, the unknown camera translation, is related to pixel space by a perspective projection. Section III-B describes the derivation of a bound for this case.

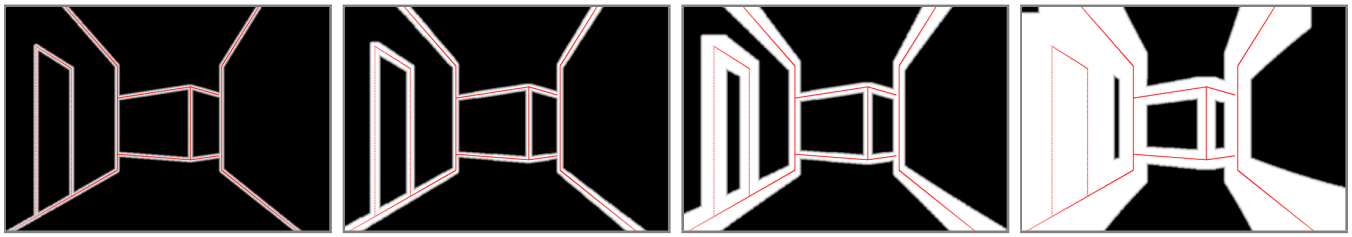
B. Perspective motion bound under 3D translation

PAS's multi-scale search is enabled by a bound on the perspective motion of a translated point. By using this bound to render special lookup tables, PAS can evaluate an upper-bound on the score of a range of translations at once.

Consider a 3D point $\mathbf{X_a} = [X_a, Y_a, Z_a]^T$ in frame A and another measurement $\mathbf{X_b}$ of the same point related by rotation $\mathbf{R}$ and translation $\mathbf{t}$. Let $\mathbf{X_a} = \mathbf{R}\mathbf{X_b} + \mathbf{t}$. The standard pinhole projection equation for a camera not exhibiting distortion or skew relates $\mathbf{X_a}$ to its pixel coordinate in the x-axis by:

$$p_x = f_x\frac{X_a}{Z_a} + c_x \quad (5)$$

and similarly for y. Given the following definition for $\mathbf{R}$ in



(a) Level 0 (b) Level 1 (c) Level 2 (d) Level 3

Fig. 3: Translation-image pyramid for PAS in a synthetic environment. Projected points sampled along wireframe are shown in red. Images correspond to a 4-level search with translation levels of {2 cm, 6 cm, 18 cm, 54 cm}. Level 0 uses the kernel image as described in Section III-A, while Levels 1-3 use bounds images using the translation bound derived in Section III-B. Gray border added for clarity.

terms of its row vectors $\mathbf{R}_1$, $\mathbf{R}_2$, and $\mathbf{R}_3$:

$$\mathbf{R} = \begin{bmatrix} — & \mathbf{R}_1 & — \\ — & \mathbf{R}_2 & — \\ — & \mathbf{R}_3 & — \end{bmatrix} \tag{6}$$

Equation 5 is trivially equal to:

$$p_x = f_x \frac{\mathbf{R}_1 \mathbf{X_b} + t_x}{\mathbf{R}_3 \mathbf{X_b} + t_z} + c_x \tag{7}$$

If the translation is unknown, we can rotate and project the point $\mathbf{X_b}$ without translation and relate the new observation of $\mathbf{X_b}$ to the previous observation of $\mathbf{X_a}$ via:

$$p_{tx} = f_x \frac{\mathbf{R}_1 \mathbf{X_b}}{\mathbf{R}_3 \mathbf{X_b}} + c_x = f_x \frac{X_a - t_x}{Z_a - t_z} + c_x \tag{8}$$

Where the absolute deviation of p_{tx} from p_x is given by $dp_{tx} = |p_{tx} - p_x|$. We wish to compute a single bound on dp_{tx} given a bounded translation. We first pick an upper limit on the translation on all axes, Δx, such that:

$$-\Delta x \leq t_x, t_y, t_z \leq \Delta x \tag{9}$$

We can define $t_x = \alpha \Delta x$, $t_y = \beta \Delta x$, and $t_z = \gamma \Delta x$ for

$$-1 \leq \alpha, \beta, \gamma \leq 1 \tag{10}$$

We then manipulate dp_{tx} into a convenient form in terms of $\mathbf{X_a}$:

$$\begin{aligned} dp_{tx} &= |p_{tx} - p_x| \\ &= f_x \left| \frac{X_a - t_x}{Z_a - t_z} - \frac{X_a}{Z_a} \right| \\ &= f_x \left| \frac{Z_a(X_a - t_x)}{Z_a^2 - Z_a t_z} - \frac{X_a(Z_a - t_z)}{Z_a^2 - Z_a t_z} \right| \\ &= f_x \left| \frac{X_a t_z - Z_a t_x}{Z_a^2 - Z_a t_z} \right| \end{aligned} \tag{11}$$

and finally substitute in the new definitions of t_x, t_y, and t_z:

$$dp_{tx} = f_x \Delta x \left| \frac{X_a \gamma - Z_a \alpha}{Z_a^2 - Z_a \gamma \Delta x} \right| \tag{12}$$

We wish to compute an upper bound for dp_{tx} given the constraints on α and γ. By noting that $Z_a \geq 0$ and requiring that $Z_a > \Delta x$ for all points, it can be shown that $Z_a^2 - Z_a \gamma \Delta x$ is strictly positive. Further, by noting that $|X_a \gamma - Z_a \alpha| \leq |X_a| + Z_a$ given the limits on α, γ, and Z_a, we merely need to minimize the denominator, which can be done by setting γ to 1. This yields the desired bound on dp_{tx}:

$$dp_{tx} \leq f_x \Delta x \left(\frac{|X_a| + Z_a}{Z_a^2 - Z_a \Delta x} \right) \tag{13}$$

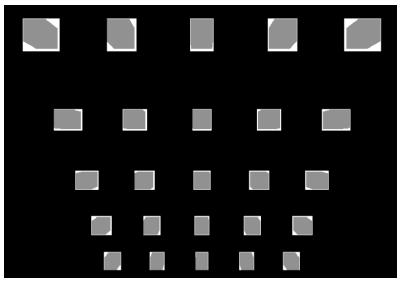


Fig. 4: Visualization of the perspective motion bound. A set of 25 points, arranged in a 5×5 grid with each *row* at an increasing depth, is used to visualize the bound on motion in the image of translated points. The pixel motion bound is rendered first, in white, then the set of points is translated numerous times in a fine sweep over the translation volume and associated pixels colored over in gray.

For any point $\mathbf{X_a}$ and a limit on the translation Δx, we can bound the maximum perspective motion. This bound is visualized in Figure 4 for a set of 25 points. After the bound is rendered in white, the pixels where points could actually project are colored in gray by sweeping over all valid translations and projecting the translated points. This figure verifies that the bound is correct – no pixels colored in gray fall outside of the white bounds. Further, the bound is tight – few white pixels remain visible.

This perspective motion bound allows us to construct our translation-image pyramid. For a given search level l, we set the translation limit to $\Delta x_l = \frac{s}{2}(3^l - 1)$, where s is the node spacing at Level 0. As depicted in Figure 2, this translation limit is equal to the distance of the furthest child node. We then iterate through all points at A and render a rectangle with width and height equal to twice the value of the bound on dp_{tx} from Equation 13, effectively dilating the pixel (p_x, p_y) by the maximum possible motion in image space. The value of the rectangle is set to the optimal value of the kernel. After all rectangles are rendered, we render the kernel with the $\max$ operator centered at every border pixel. Figure 3 shows the result for a 4-level search.

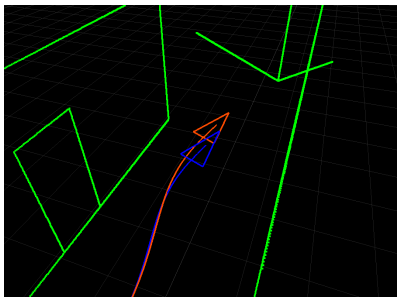


Fig. 5: Simulation environment for PAS for a deliberately-drifting pose estimate. Estimated trajectory in red, true trajectory in blue.

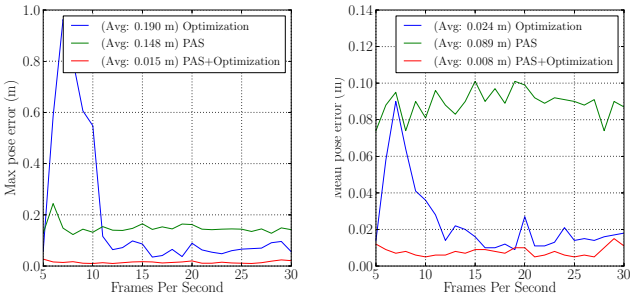


Fig. 6: Mean and max pose error for three configurations vs. true, simulated pose as framerate is varied. Average errors included in legend.

C. 6 DoF Non-linear Optimization

Once the global minima has been found given an initial camera orientation, we can perform non-linear optimization over the full pose state to improve the position estimate. For example, to reduce error due to inertial sensor drift or pose discretization.

We perform Gauss-Newton minimization with Tikhonov regularization. The Jacobian is computed numerically from the Level 0 kernel image and the state update is scaled to ensure that the mean squared error decreases. If no suitable scaling term can be found, optimization halts.

IV. Evaluation using synthetic data

A custom simulation environment was used to evaluate PAS in controlled conditions with imperfect orientation estimates. A wireframe description of the world was used to create simulated feature matches between the left and right cameras by stepping down each line and projecting discrete points into all cameras. A fixed forward velocity with sinusoidal gyro rates about the camera-frame Y and Z axes was used for the true robot motion. Orientation measurements are simulated and optionally degraded with a drift parameter specified in degrees/second. Figure 5 shows this environment with a deliberately-drifting pose estimate.

Figure 6 shows the mean and max error between the simulated and the estimated robot pose for frame rates between 5 and 30 FPS. Gyro drift is fixed to 0.2°/s on all axes, and the pose is estimated in three modes: using PAS, using optimization only, and using both. When PAS is used alone, the angular drift is uncorrected and the pose estimate exhibits a large, constant error. Using non-linear optimization alone produces estimates with a mean error as low as 1 cm. However, it is sensitive to the size of the kernel – for low frame rates, optimization alone fails to converge, as is clear from the max error plot. Optimizing after computing the solution with PAS yields the best results, with an average error of less than 1 cm. This combination is able to reject the gyro drift even at low frame rates and produces more accurate estimates than when using optimization alone, even at higher frame rates.

Figure 7 shows the mean pose error for PAS followed by optimization for three frame rates. The amount of gyro drift, shown on the X-axis, is varied from 0 to 5°/s. As the framerate is increased, the basin of convergence for PAS with optimization increases, where a drift rate of nearly 3°/s is rejected at an update rate of 30 FPS. For a lower drift rate of 0.5°/s, an update rate of 10 FPS is sufficient.

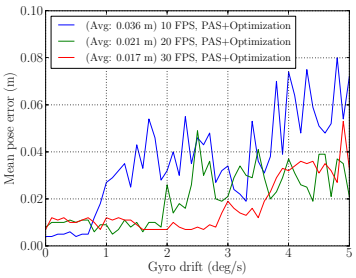


Fig. 7: Mean pose error for three configurations vs. true, simulated pose as the simulated gyro drift rate is varied. As the frame rate increases, more gyro drift is successfully rejected.

V. Evaluation using real data

A. Implementation details

PAS is implemented in C and all results computed on an Apple MacBook Pro 10,1 Intel Core i7-3740QM (4 cores) running Ubuntu 12.10 (native). Video is capture via a custom stereo rig composed of two grayscale Point Grey Firefly MV 1394a cameras with Boowon BW3M30B-2000 M12 lenses separated by a 17.5 cm baseline. Images are captured at 640×480 resolution at 30 frames per second. The cameras automatically synchronize, and images are paired using host times and a passive time synchronization algorithm [20]. In addition, a MEMS-grade IMU is used to estimate orientation rates [7]. The data is collected by the ground robot platform used in the MAGIC 2010 Robotics Competition [7]. LIDAR data is collected for comparison to pose estimates from SLAM.

A horizontal filter designed for gradient detection is used to extract edge features in stereo-rectified imagery. Detection is performed on alternating rows to improve runtime without a noticeable impact on accuracy. A 16-pixel horizontal patch is used as a descriptor for matching between the left and right images, leveraging epipolar constraints, after which the descriptors are discarded. Note that the PAS algorithm, which

is a component in the larger visual odometry system, does not use feature descriptors when estimating the camera motion.

PAS is implemented using a max heap and 8-bit lookup tables. A quadratic kernel with a 7 pixel radius was used, scaled to fill the full 8-bit range. The kernel is inverted for use with the max heap. Keyframes are used to reduce error accumulation, with new keyframes created every 0.5 m or 15°of rotation. Point clouds used in search are decimated uniformly by a factor of two to reduce runtimes.

We collected data to evaluate the PAS-based visual odometry system on a mobile robot as described in Section V-A. The robot was driven through an office environment at an average speed of 0.73 m/s and top speed of 1.2 m/s. The video stream is stereo rectified, and a proportional controller on the feature detection threshold maintains 800 matched features per stereo frame. The average computation time for image preprocessing and feature detection, matching, and triangulation is shown in Table I.

Stage	**Time (ms)**
Rectification	4.2
Detection	6.0
Matching, triangulation	1.7
Total	11.9

TABLE I: Average computation time broken down by stage.

B. Experiments

To evaluate the trade-off between framerate and total CPU load for PAS, we conducted an experiment in which a log was processed at 5, 10, 15, and 30 FPS with search parameters matched to the robot's velocity. Using a 2 m/s upper estimate of the maximum velocity, we computed search parameters to enclose the unknown motion and ensure that all searches have a terminal resolution at Level 0 of 2 cm.

Table II shows the parameters and results for each of the settings. Computation times per update for PAS search are shown in milliseconds, as well as the estimated runtime per second given the update rate (FPS), single-update runtime, and additional runtime required for image preprocessing: rectification plus feature detection, matching, and triangulation. Additionally, rendering the translation image pyramid for each new keyframe took 4.7 ms for a 2-level search and 10.1 ms for a 3-level search, averaged over the entire run.

FPS	**Range** cm	**Levels**	**Time** ms	**CPU** %	**Time Ratio** P95:P50	**Nodes** %
5	±40	3	19.2	15.6	2.73	2.0
10	±20	3	10.0	21.9	2.17	7.5
15	±14	3	7.9	29.7	1.85	15.2
30	±8	2	4.8	50.1	1.69	41.1

TABLE II: Parameter sweep for PAS to find the best update rate and search resolution. Image preprocessing (See Table I) excluded in runtime measurement, but included in estimated CPU load. The runtime ratio of the 95th to the 50th percentile runtime is shown, as well as the percentage of nodes evaluated in the multi-scale search.

The computation time per update in this table varies significantly with the update rate (FPS). A single search with the 5 FPS settings takes approximately 4x as long as a single 30 FPS search. However, when accounting for the number of updates per second as well as the computation time to rectify the images and detect features, the 30 FPS configuration requires 34.5% more total CPU load, denoted by the 'CPU' column.

While the 30 FPS setting requires more runtime, its runtimes are the most consistent. This is demonstrated by the 'Time Ratio' column, which shows the ratio of runtimes for the 95th percentile vs. the 50th percentile. This lack of variation is explained by the percentage of search nodes that are evaluated, on average, in the course of a single update. 41.1% of the nodes in the 30 FPS search volume are evaluated, while at 5 FPS, only 2.0% of the nodes need to be evaluated. Choosing an appropriate update rate requires balancing the trade-off between CPU load and near-real-time performance.

Table III shows the runtime, estimated load, and average trajectory error against SLAM for the 5 and 30 FPS configurations. The trajectory errors are computed by stepping down the trajectory in 1 second intervals and evaluating the error of the estimated trajectory against the SLAM reference trajectory. A ± 5 second sliding window is used for each evaluation and the transformation that best-aligns the two trajectories is applied before computing the error.

Method	**5 FPS**			**30 FPS**		
	Time ms	**CPU** %	**Error** m	**Time** ms	**CPU** %	**Error** m
Optim.	36.4	24.2	0.229	28.3	120.6	0.093
PAS	19.2	15.6	0.117	4.8	50.1	0.080
PAS+Optim.	45.2	28.6	0.092	30.3	126.6	0.088

TABLE III: Comparison of methods for motion estimation for two frame rates on a 3 minute indoor log. Methods include non-linear optimization used alone, PAS, and PAS followed by optimization. Image preprocessing time (See Table I) excluded in runtime measurement, but included in estimated CPU load. Error denotes average sliding-window error.

This table shows that PAS without optimization is comparable in terms of accuracy to the alternatives, and that PAS without optimization can be much faster. At 5 FPS, using only non-linear optimization results in much higher error, as may be predicted following the evaluation in Figure 7. PAS followed by optimization yields a slight improvement in the error. At 30 FPS, PAS requires only 40% as much runtime as the alternatives. For these settings, only PAS without optimization could be run online at 30 FPS, though with additional optimizations or greater decimation that would not be the case. Finally, all variants at 30 FPS yield similar local errors.

An example trajectory from real data is shown in Figure 8. PAS was run at 30 FPS and compared to the output of a LIDAR SLAM solution [7]. The SLAM solution is computed with loop closures and batch optimization, as opposed to PAS, which is run open-loop. The cyan trajectory

corresponds to PAS with optimization off, after applying the transformation that best aligns the entire PAS trajectory to that of SLAM. The SLAM trajectory is shown with colors indicating the average sliding-window trajectory error described previously. Errors are primarily present during large turns, which may be explained by feature density as the camera faces a wall, or by the IMU measurements, which slowly drift over the course of the run. The PAS trajectory exhibits drift over time, mostly due to accumulation of gyro errors. However, this error could be small enough for closed-loop control or as a prior for a Visual SLAM or pure localization system.

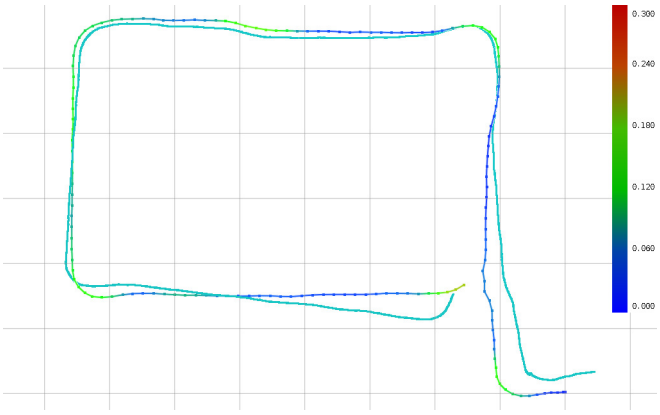


Fig. 8: PAS vs. LIDAR-based SLAM for a 3 minute indoor trial. (Color-mapped) The SLAM trajectory is assumed to be the ideal result, as it is computed from LIDAR scan-matching with loop closures and global optimization. (Cyan) PAS was run open-loop with keyframes, and with optimization disabled. Color map applied to SLAM trajectory corresponds to relative trajectory error against PAS. Grid spacing is 5 meters.

VI. Summary

We have described Perspective Alignment Search, or PAS, an algorithm for efficient joint alignment of sparse point clouds. We have shown it applied to visual odometry in low-texture environments with indistinct visual features. PAS uses implicit and descriptorless data association to compute the translation that best-aligns two observations of the scene. Because it is descriptorless, even edge features which would produce similar descriptors can be used. This method is fast enough to run in real time for odometry estimates and is not susceptible to local minima, as it employs a multi-scale branch-and-bound search.

Acknowledgments

This work was funded by U.S. DoD Grant FA2386-11-1-4024.

References

[1] C. Harris and M. Stephens, "A combined corner and edge detector," in *Alvey vision conference*, vol. 15. Manchester, UK, 1988, p. 50.

[2] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 430–443, 2006.

[3] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, Nara, Japan, November 2007.

[4] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "BRIEF: Binary robust independent elementary features," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2010, pp. 778–792.

[5] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, November 2004.

[6] E. Olson, "Real-time correlative scan matching," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Kobe, Japan, June 2009, pp. 4387–4393.

[7] E. Olson, J. Strom, R. Morton, A. Richardson, P. Ranganathan, R. Goeddel, M. Bulic, J. Crossman, and B. Marinier, "Progress towards multi-robot reconnaissance and the MAGIC 2010 competition," *Journal of Field Robotics*, vol. 29, no. 5, pp. 762–792, September 2012.

[8] A. Davison, "Real-time simultaneous localisation and mapping with a single camera," in *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2, 2003, pp. 1403–1410.

[9] C. Mei, G. Sibley, M. Cummins, P. Newman, and I. Reid, "RSLAM: A system for large-scale mapping in constant-time using stereo," *International Journal of Computer Vision*, pp. 1–17, 2010.

[10] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, 2004, pp. I–652.

[11] A. Richardson and E. Olson, "Learning convolutional filters for interest point detection," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2013.

[12] M. Li and A. Mourikis, "Improving the accuracy of EKF-based visual-inertial odometry," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2012, pp. 828–835.

[13] A. Geiger, J. Ziegler, and C. Stiller, "StereoScan: Dense 3D reconstruction in real-time," in *IEEE Intelligent Vehicles Symposium*, Baden-Baden, Germany, June 2011.

[14] E. Eade and T. Drummond, "Edge landmarks in monocular SLAM," in *British Machine Vision Conference*, 2006, pp. 7–16.

[15] M. Tomono, "Robust 3D SLAM with a stereo camera based on an edge-point ICP algorithm," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2009, pp. 4306–4311.

[16] ——, "3D localization based on visual odometry and landmark recognition using image edge points," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010, pp. 5953–5959.

[17] G. Pandey, J. McBride, S. Savarese, and R. Eustice, "Visually bootstrapped generalized ICP," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Shanghai, China, May 2011, pp. 2660–2667.

[18] A. L. Rodríguez, P. E. López-de Teruel, and A. Ruiz, "Real-time descriptorless feature tracking," in *Proceedings of the International Conference on Image Analysis and Processing (ICIAP)*, 2009, pp. 853–862.

[19] E. Hsiao, A. Collet, and M. Hebert, "Making specific features less discriminative to improve point-based 3D object recognition," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010, pp. 2653–2660.

[20] E. Olson, "A passive solution to the sensor synchronization problem," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2010.