Texture-Aware SLAM Using Stereo Imagery And Inertial Information

Travis Manderson and Florian Shkurti and Gregory Dudek
Center for Intelligent Machines, McGill University, Montreal, Canada
{*travism, florian, dudek*}*@cim.mcgill.ca*

***Abstract*—We present a gaze control method that augments an existing stereo and inertial Simultaneous Localization And Mapping (SLAM) system by directing the stereo camera towards feature-rich regions of the scene. Our integrated active SLAM system is based on careful triangulation of visual features, existing successful nonlinear optimization, and visual loop closing frameworks. It relies on the tight coupling of IMU measurements with constraints imposed by visual correspondences from both stereo and motion. Alongside the SLAM system, the gaze control module also runs in real-time and includes an efficient online classifier that segments the scene into texture classes and assigns a quality score to each class that correlates with the availability of reliable features for tracking. Based on this quality score, the gaze selection module controls a pan-tilt unit that directs the camera to focus on high-reward texture classes. We validate our system in both indoor and outdoor spaces, and we show that active gaze control crucially improves the robustness and long-term operation of the localization system.**

I. INTRODUCTION

In this paper we present an integrated system that combines our prior work on visual and inertial simultaneous localization and mapping (SLAM) with active gaze selection in order to enhance the tracking robustness of the localization system and to avoid feature-poor scenes. Our SLAM system combines inertial measurement unit (IMU) measurements with visual correspondences obtained from camera motion and stereo image pairs in a nonlinear graph optimization framework, yielding true-scale pose and velocity estimates, as well as robustness during pathological motions such as in-place rotations. This paper looks at the enhancement of this existing system through the incorporation of a gaze control module, which directs the stereo camera towards regions of the scene that are classified in real-time to be conducive to good tracking based on its texture classification. We demonstrate that the addition of this module is crucial for the long-term operation of the SLAM system.

While optical flow, motion estimation and stereo are core areas of computer vision that have been under active development for decades, substantially fewer complete integrated systems have been developed that exploit these technologies and take advantage of their complementary attributes for robust, portable, real-time localization. It is in this context that we have developed a system that combines core estimation technologies with a behavioral layer that makes them suitable for use in turn-key robotic systems in the field. Our emphasis is on the development of an end-to-end system that combines motion estimation, inertial data, stereo depth estimates and active view selection in a single hardware/software entity that provides localization and motion tracking outdoors in real-time. This system does these tasks in a sufficiently resource-efficient manner to be deployable on a small (or large) robotic platform.

In our work, an ego-motion system, a stereo camera, and an IMU are placed on a pan-tilt unit used for gaze control. All of these are deployed on a robotic platform to allow for active gaze control during navigation. Our motion/stereo/inertial subsystem dubbed "MORESLAM" provides robust and efficient motion estimates. However, being vision-based, and despite the robustness provided by the IMU, MORESLAM remains at the mercy of a suitable view of the scene. To reduce this dependency, our system exploits the ability to shift the gaze direction in order to focus on texture-rich areas, which leads to increased tracking robustness and maintenance of promising features within the field of view, as well as avoidance of feature-poor scenes that pose a significant challenge to any visual keypoint-based localization algorithm. Our system controls solely the motion of the pan-tilt unit and assumes that the robot's navigation is handled externally and independently.

We are interested in metric SLAM due to the fact that it is one of the types of vision-based feedback loop that can be incorporated in the autonomous navigation of most mobile robotic vehicles. Under this feedback architecture, maintaining constant scale or very low scale drift is crucial for the controllers of these robots. This is especially relevant for vehicles that operate in GPS-denied environments (e.g. underwater) or in settings where motion instability puts the vehicle at risk (e.g. aerial vehicles). We show that texture-seeking gaze control is a major enhancing component to SLAM in both of these scenarios.

Our system uses Local Binary Patterns (LBP) as descriptors for texture and relies on an offline-trained k-means classifier to discriminate between image regions that are conducive to tracking and others that should be avoided. The training phase of this classifier incorporates only appearance information about different types of textures in the environment and no geometric cues. This enables the classifier to perform rudimentary texture segmentation in real-time. During the online operation of the SLAM system,

each class is associated with the quality of visual landmarks that are visible in the field-of-view and project to the region of the image that it occupies. The quality metric that our system uses depends on the lifetime of the feature across past keyframes and on the uncertainty about its triangulated position in space. These quality scores are used to update the goodness of each texture class in an online fashion, and it is this information that the gaze selection uses to focus its attention on a specific region. We show that this simple and fast classification scheme, coupled with and updated by the quality scores obtained from the SLAM system, provide useful feedback to a gaze control module. We demonstrate multiple missions (e.g. UAV flying, and indoor mapping) where its role is critical in the successful operation of the SLAM module.

II. RELATED WORK

Our work is similar to other proposed SLAM methods, such as MCPTAM [1], SVO [2], and works by Shen [3] and others [4], [5], [6], [7], [8]. All these methods use a form of feature based tracking to estimate the system's pose and orientation with respect to an arbitrary world-fixed frame of reference. For example, MCPTAM uses multiple cameras with non-overlapping fields of view, while SVO attempts to reduce the computational requirements by mixing feature-based and intensity-based tracking to reduce its reliance on keypoint matching. Works by Shen propose methods for tightly coupling an IMU with mono and stereo vision in filtering frameworks, whereas we rely on nonlinear optimization.

None of these methods perform single-frame sub-map initialization, however, and none of them have been augmented by active gaze selection. In this regard, our system shares a lot of common ground with works that examine the interplay between texture, shape, and depth [9], [10], as well as with active SLAM approaches, such as [11], [12], [13]. Compared to these works, the main difference with our work is that we assume the navigation of the robot is handled externally and independently of our system, and we only have control over the pan-tilt unit. Most similar in spirit to our work is [14], which estimates the utility of future feature observations according to the placement of future keyframes, and [15], which shows that the yaw direction of the camera with respect to the motion of the robot significantly affects the accuracy of the system.

III. SIMULTANEOUS LOCALIZATION AND MAPPING

A. Background

Our system is based on the well-established monocular vision-based ORBSLAM [16], but we make use of our richer availability of sensory information and manage to overcome most of its limitations. The main drawback that applies to all triangulation-based monocular SLAM systems is that they suffer from scale drift, partly due to erroneously estimating the position of visual landmarks, particularly during in-place rotations where there is insufficient baseline. Our system drastically reduces the danger of in-place rotation of one camera by using visual correspondences of ORB features: (i) from the motion of the other camera, which usually has sufficient baseline to perform correct triangulation and (ii) from the synchronized stereo pair, which has a fixed baseline and can provide a steady supply of new map points.

In addition to these advantages, the inclusion of the IMU in the estimation process provides a few other tangible benefits to our system. First, it enables the prediction of keypoint locations in the next frame, thus reducing matching time and boosting overall efficiency. Second, it increases the accuracy of the system by involving its measurements tightly in the optimization. Most importantly though, the IMU increases the robustness of the system because it enables effective recovery from loss of tracking whenever possible. Our system uses the stereo pair to initialize a new sub-map and then uses the IMU to align the sub-map with the existing map. This sub-map initialization is done from a single stereo pair without requiring any special motions such as a swiping translation.

In the following sections we describe the most important aspects of our SLAM system.

We define the following four frames: W is the fixed world frame of reference in which all the quantities of interest are expressed; L and R are the moving frames of the left and right cameras respectively, and I is the moving IMU frame. The two camera frames and the IMU frame are fixed with respect to each other, and we assume to know in advance the transformations that link them. We have done the calibration using Kalibr [17]. In our particular case, the rotation part of W is the local North-East-Down frame and the translation is the origin of the navigation mission.

The *localization state vector* that we want to estimate is the following:

$$\mathbf{S} = \begin{bmatrix} {}^{L}_{W}\mathbf{q} & {}^{L}\mathbf{p}_{W} & {}^{W}\mathbf{v}_{I} & \mathbf{b}_{g} & \mathbf{b}_{a} \end{bmatrix} \tag{1}$$

where ${}^{L}_{W}\mathbf{q}$ is a unit quaternion that expresses the rotation from the fixed world frame to the left camera frame, ${}^{L}\mathbf{p}_{W}$ is the position of the world frame in the coordinates of the left camera frame, ${}^{W}\mathbf{v}_{I}$ is the velocity vector of the IMU in world frame, $\mathbf{b}_g$ is the gyroscope bias, and $\mathbf{b}_a$ is the accelerometer bias affecting the IMU measurements. Upon every incoming pair of stereo images, we create a frame object that stores this instantaneous localization information as its state vector. In our system, frames are created at the camera frame rate (15Hz) after an image has been received. They also store all the IMU measurements since the last stereo image pair was received, which means that on average each frame has about 7 IMU messages that precede it, since we receive IMU data at 100Hz.

Our system currently assumes that both cameras use the pinhole projection model, are in-phase with the IMU measurements using hardware triggering, and share the same exposure times.

The *mapping state vector* that we want to estimate is a dynamic map object that comprises a list of selected map points and a list of selected keyframes.

Keyframes are sparse snapshots of frame objects along the trajectory that are dispersed in a way that balances sufficient visual overlap to exploit for triangulation and enough visual differences to lead to significant baseline for motion stereo. Ensuring that many of the keypoints associated with keyframes are correctly linked to 3D map points is an essential ingredient to the success of our SLAM system.

Each map point includes a mean 3D position in the world frame and a 3×3 marginal covariance matrix that describes the uncertainty in the position. It is worth noting that in the current version of our SLAM system, we do not model the uncertainty of keyframe poses or cross-correlation terms between keyframes and map points.

Just like in [16], we simultaneously run three interacting threads: tracking, local mapping, and loop closing. Our system builds upon theirs, and uses the g2o framework [18] for graph-based nonlinear optimization. In the following sections, we describe the tracking and local mapping threads and we refer to [16] for the loop closing thread. The tracking thread mainly deals with the correct state estimation of frames and IMU data processing, while the other two threads deal exclusively with estimating the pose of keyframes without processing any IMU data. In other words, we do not model IMU measurements as factors in the connected graph of keyframes in the same way as [5], but rather take a more "filtering-type" approach, which we explain below.

B. Tracking Thread

The tracking thread is the one that receives each stereo image pair and the IMU data. It extracts and undistorts Harris keypoints and computes ORB descriptors on each image. This happens in parallel on two separate sub-threads. The tracking thread is also the one receiving the IMU measurements. It initializes stereo frames and populates them with the IMU messages that were received since the previous frame. It computes high-quality stereo matches from the left descriptors to the right at 15Hz. These matches satisfy strict thresholds over the epipolar errors. Stereo keypoint matches are used to initialize the map for the first time, as well as sub-maps.

The tracking thread also checks if the current frame is a good candidate for being cloned into a keyframe. If so, it dispatches the frame to the local mapping thread, and starts processing the next set of incoming stereo and IMU measurements.

In the following section we describe the main elements involved in the tracking thread.

1) Single-Frame Map Initialization Via Stereo Triangulation: One of the advantages of relying on stereo is that it allows our system to initialize a map from a single stereo image pair. As long as the observed scene is textured enough and close enough to the camera to perform accurate triangulation this step is going to be successful. Our system performs stereo triangulation in two steps:

In the first step, we use Hartley's linear triangulation method [19], which is a linear least squares problem with norm constraints. Our system further improves on this triangulation estimate by doing a second optimization, which minimizes a weighted-norm reprojection error. Keypoints detected at smaller scales are weighed higher than those detected at higher scales in the scale pyramid, under the assumption that small-scale keypoints are better localized in image coordinates. This is encoded by the diagonal matrix $\mathbf{\Sigma}_z$, which we set similarly to [16]. Then the second optimization for refining the 3D map point position, denoted by ${}^W\mathbf{f}$ is:

$$\underset{{}^W\mathbf{f}}{\operatorname{argmin}} \left\| [u_L \; v_L \; u_R \; v_R]^T - \mathbf{h}({}^L_W\mathbf{q}, {}^L\mathbf{p}_W, {}^R_W\mathbf{q}, {}^R\mathbf{p}_W, {}^W\mathbf{f}) \right\|^2_{\mathbf{\Sigma}_z} \tag{2}$$

where $\mathbf{h}$ is a measurement model that projects ${}^W\mathbf{f}$ to the pair of pixel coordinates (u_L, v_L) and (u_R, v_R) of the stereo camera. The solution to this unconstrained nonlinear optimization problem is obtained iteratively and is initialized by the solution found in the first optimization, described above. We compute the covariance of the unconstrained nonlinear least squares estimator, which describes the uncertainty of the triangulation, as follows:

$$\operatorname{cov}({}^W\mathbf{f}) = (\mathbf{J}_h^T \mathbf{\Sigma}_z^{-1} \mathbf{J}_h)^{-1} \tag{3}$$

where $\mathbf{J}_h$ is the Jacobian of the stereo projection function $\mathbf{h}$ evaluated at the map point estimate from the linear triangulation step. We assume a Gaussian distribution of the noise in the estimate of the map point position. We also note that whether this covariance underestimates the real error (making this least squares estimator inconsistent) strongly depends on whether the initial triangulation estimate is close to the optimal solution, which in turn depends on the pixel noise characteristics of the camera and the pixel localization of the keypoint. Given these considerations, we accept the triangulation as successful if a series of outlier detection tests are passed. The tests include the low-parallax test from [16], χ^2 tests on the reprojection error under the Huber loss, negative estimated depth, excessive depth variance from the position covariance computed above, tests on the conditioning of the least-squares problem itself, and also maximum/minimum cutoffs on the estimated depth.

2) Feature Prediction Using IMU: Upon receiving a new stereo image pair the tracking thread computes a rough estimate of the pose of the new frame, given the pose

of the previous frame. In order to do this, our system performs IMU integration on the measurements that have arrived between the previous and the current frame. Thus, it computes the relative motion of the IMU and the cameras since the previous frame. Although this scheme is accurate for the relative rotation, it is not always accurate for the relative translation. Therefore, during the step of keypoint prediction, we ignore the relative translation of the IMU integration, and instead use the relative displacement of the two previous frames. This method seems to work quite well even under heavy rotations.

Given IMU measurements $\boldsymbol{\omega}_m$ and $\boldsymbol{\alpha}_m$ for the angular velocity and linear acceleration of the IMU frame, we model the errors affecting the measurements as follows:

$$\begin{aligned} \boldsymbol{\omega}_m &= {}^{I}\boldsymbol{\omega} + \mathbf{b}_g + \mathbf{n}_g \\ \boldsymbol{\alpha}_m &= \mathbf{R}({}^{I}_{W}\mathbf{q})({}^{W}\boldsymbol{\alpha} - {}^{W}\mathbf{g}) + \mathbf{b}_a + \mathbf{n}_a \end{aligned} \tag{4}$$

where ${}^{W}\boldsymbol{\alpha}$ and ${}^{I}\boldsymbol{\omega}$ are the true values, the noise is white Gaussian, and the biases are modeled as Brownian motion processes with $\dot{\mathbf{b}}_g = \mathbf{0}$, $\dot{\mathbf{b}}_a = \mathbf{0}$, and $\mathbf{R}()$ denotes the rotation matrix resulting from the corresponding quaternion. The linear acceleration and angular velocity estimates used in the IMU state integration are $\hat{\boldsymbol{\alpha}} = \boldsymbol{\alpha}_m - \hat{\mathbf{b}}_a$ and $\hat{\boldsymbol{\omega}} = \boldsymbol{\omega}_m - \hat{\mathbf{b}}_g$. The gravity vector ${}^{W}\mathbf{g}$ is expressed in the world frame. We perform forward Euler integration of the IMU state using the following equations:

$$\begin{aligned} {}^{W}_{I}\hat{\mathbf{q}}_{t_{k+1}} &= {}^{W}_{I}\hat{\mathbf{q}}_{t_k} \otimes \begin{bmatrix} \frac{\hat{\boldsymbol{\omega}}}{\|\hat{\boldsymbol{\omega}}\|}\sin(\|\hat{\boldsymbol{\omega}}\|\frac{(t_{k+1}-t_k)}{2}) \\ \cos(\|\hat{\boldsymbol{\omega}}\|\frac{(t_{k+1}-t_k)}{2}) \end{bmatrix} \\ \dot{\hat{\mathbf{b}}}_g &= \mathbf{0}_{3\times 1}\ , \quad {}^{W}\dot{\hat{\mathbf{v}}}_I = \mathbf{R}({}^{W}_{I}\hat{\mathbf{q}})\hat{\boldsymbol{\alpha}} + {}^{W}\mathbf{g} \\ \dot{\hat{\mathbf{b}}}_a &= \mathbf{0}_{3\times 1}\ , \quad {}^{W}\dot{\hat{\mathbf{p}}}_I = {}^{W}\hat{\mathbf{v}}_I \end{aligned} \tag{5}$$

This is similar to [20], with the exception of the convention for quaternion multiplication. In our work we use the Hamiltonian notation, whereas that work uses the JPL convention.

3) Frame Pose Refinement From Previous Frame & Tight IMU Coupling: After having assigned a rough estimate to the pose of the newly-created frame the tracking thread has established correspondences between the keypoints of said frame and the existing map points of the previous frame. It then refines that estimate, by optimizing over the poses of the two frames in order to minimize two costs: the first is the reprojection error of those map points (denoted by RE), and the second is a relative motion error from the IMU motion (denoted by RME). During this optimization the map point positions are taken to be fixed. Let $\mathbf{S}_0$ denote the localization state vector of the previous frame, and $\mathbf{S}_1$ denote the localization vector of the current frame, as explained in Eq. 1. Then the optimization problem our system solves at this step is the following:

$$\underset{\mathbf{S}_0,\mathbf{S}_1}{\operatorname{argmin}} \quad \mathrm{RE}(\mathbf{S}_1) + \mathrm{RME}(\mathbf{S}_0, \mathbf{S}_1) \tag{6}$$

where the reprojection error is

$$\mathrm{RE}(\mathbf{S}) = \sum_{{}^{W}\mathbf{f}\leftrightarrow\mathbf{z}_f} \left\|\mathbf{z}_f - \mathbf{g}({}^{L}_{W}\mathbf{q}, {}^{L}\mathbf{p}_W, \mathbf{f})\right\|^2_{\boldsymbol{\Sigma}_z} \tag{7}$$

where $\mathbf{g}$ is the function that projects a point from the world frame to a pixel in the left image. We denote the relative camera motion, obtained from the relative IMU motion, by $\bar{\Delta\mathbf{q}}^0_1, \bar{\Delta\mathbf{p}}^0_1$. This is expressed in the previous frame's coordinates. Let the same quantity obtained from the state vectors of the two frames be $\hat{\Delta\mathbf{q}}^0_1, \hat{\Delta\mathbf{p}}^0_1$. We denote the integrated IMU velocity starting from the state vector $\mathbf{S}_0$ until the time of the current frame by ${}^{W}\bar{\mathbf{v}}_{I_1}$. Finally, let the IMU velocity in the state vector $\mathbf{S}_1$ be ${}^{W}\mathbf{v}_{I_1}$. Then the relative motion error is:

$$\mathrm{RME}(\mathbf{S}_0, \mathbf{S}_1) = \left\| \begin{bmatrix} \bar{\Delta\mathbf{q}}^0_1 \otimes \hat{\Delta\mathbf{q}}^1_0 \\ \bar{\Delta\mathbf{p}}^0_1 - \hat{\Delta\mathbf{p}}^0_1 \\ {}^{W}\bar{\mathbf{v}}_{I_1} - {}^{W}\mathbf{v}_{I_1} \\ \mathbf{b}_{g_1} - \mathbf{b}_{g_0} \\ \mathbf{b}_{a_1} - \mathbf{b}_{a_0} \end{bmatrix} \right\|^2_{\Sigma_{\mathrm{IMU}}} \tag{8}$$

where Σ_{IMU} expresses the uncertainty of the IMU propagation, which can be estimated as in [20]. This tight integration of the IMU in the tracking and optimization loop enables improved rotation estimates as well as accurate velocity estimates.

C. Local Mapping Thread

The local mapping thread receives newly-created and initialized keyframes from the tracking thread. It is responsible for using them in order to triangulate new map points for the system. Its role is critical as it needs to supply the other two threads with a sufficient number of well-triangulated map points, so that their optimizations can produce meaningful results.

Once new map points have been created, the local mapping thread tries to merge them with existing map points, to avoid duplicate representations. This is done efficiently in the visual neighborhood of the new keyframe. Once duplicate map points have been merged, the system proceeds to its most computationally demanding step: local bundle adjustment, which is going to be described in more detail below.

1) Creating New Map Points: Our system exploits a few options for triangulating new map points based on the following correspondences: between the current left and right images, between nearby left keyframes, and between nearby left and right keyframes. In most cases this means that if the left camera is undergoing in-place rotation the right camera is not[1]. We exploit this property to increase the availability of a steady supply of newly-triangulated map points.

[1]Rotation of the two cameras about their shared x-axis is still a problem

2) Merging New Map Points With Existing Map: Once new map points are created they need to be integrated into the map and duplicates need to be merged. For each visually neighboring keyframe (significant visual overlap), we project the map points of the new keyframe to the neighbor, and efficiently identify matches using Hamming distances on the space of binary ORB descriptors.

The map points that are merged need to have their mean and covariance updated, so that two sources of information can be combined into one. We regard the act of merging as obtaining a new pixel measurement for an existing map point, and we update its estimate so as to minimize the minimum mean squared error criterion, given a prior estimate. This effectively is akin to maintaining an EKF for each map point, and updating it whenever new observations are established. Note that the propagation model in this case is the identity function.

More concretely, if the prior Gaussian estimate of the map point is given by $({}^{W}\mathbf{f}_{t-1},\ \mathbf{\Sigma}_{t-1})$, and we merge it with an existing map point with pixel observation z on a neighboring keyframe with pose ${}^{L}_{W}\mathbf{q}, {}^{L}\mathbf{p}_{W}$, then the minimum mean squared error update is as follows:

$$\begin{aligned} \mathbf{K} &= \mathbf{\Sigma}_{t-1}\mathbf{J}_g^T(\mathbf{J}_g\mathbf{\Sigma}_{t-1}\mathbf{J}_g^T + \mathbf{\Sigma}_z)^{-1} \\ {}^{W}\Delta\mathbf{f} &= \mathbf{K}(\mathbf{z} - \mathbf{g}({}^{L}_{W}\mathbf{q}, {}^{L}\mathbf{p}_{W}, {}^{W}\mathbf{f}_{t-1})) \\ \mathbf{\Sigma}_t &= \mathbf{\Sigma}_{t-1} - \mathbf{K}\mathbf{J}_g\mathbf{\Sigma}_{t-1} \end{aligned} \tag{9}$$

where $\mathbf{g}$ is the same reprojection function introduced in Eq. 7, and $\mathbf{J}_g$ is its Jacobian evaluated at the previous mean estimate ${}^{W}\mathbf{f}_{t-1}$.

3) Local Bundle Adjustment: Local BA takes the newly-created keyframe and its visual neighbors, as well as all their map points, and it performs a fixed number of reprojection error minimization steps. This is similar to Eq. 7, with the difference that local BA optimizes over the keyframe poses as well as over the map point positions. Hence, there is usually a large number of variables involved in this step, in the order of 50 keyframes and 1500 map points. We are currently working on methods to exploit the map point covariance information to reduce the number of variables involved. On an i7 computer this step usually takes in the order of 0.2-0.5 seconds, and it is critical in refining the estimates of the localization and mapping variables, but also in the process of identifying spurious map points and badly-estimated keyframes that need to be pruned out. In this step we also remove all map points whose reprojection error in *any* of the keyframes that observed them exceeds a preset threshold, similarly to [21].

IV. TEXTURE CLASSIFICATION

A. Background

Our texture classifier is used to segment an image based on textures that are conducive to good tracking. The classifier is divided into two processes: offline k-means clustering and online classification labeling. Since the online component is intended to be run concurrently with SLAM, it must be fast enough to not interfere with the other threads running. For this reason, we chose to use a histogram of LBP [22] feature descriptors which are known to be faster than alternatives such as SIFT or SURF. The offline k-means clustering component operates on a set of images and builds a model that groups image patches based on their histogram of LBP descriptors. The online component takes the image frames from SLAM and uses the k-means model to classify the image patches. The online component also takes the visible keypoints (and their attributes) in the current frame to label the texture classes with a weight that corresponds to the chance that a good keypoint will be found in it.

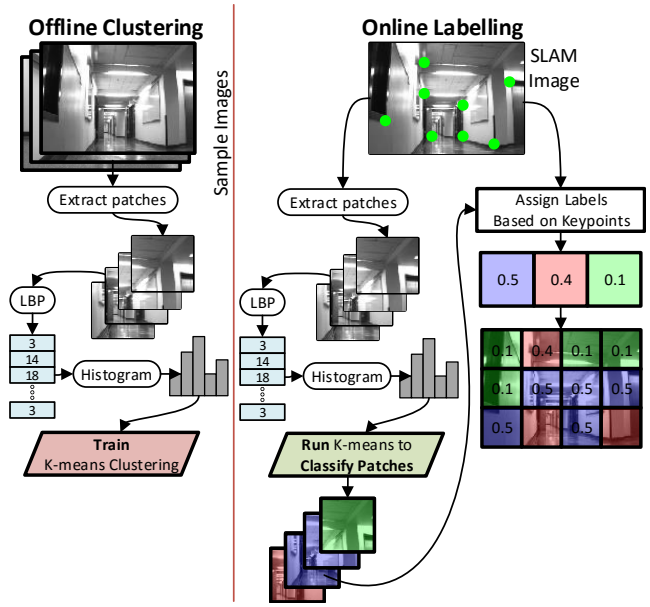


Figure 1: Classification pipeline: offline texture grouping is followed by online classification, see Sec. IV(b)

B. Offline Clustering

Our process begins with an offline clustering task as shown in the left side of Fig. 1. We collect a training set of images taken from a similar scene to where SLAM will be run and extract 40 by 40 pixel patches. Next we compute the LBP descriptor for each pixel in the patch.

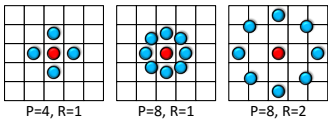


Figure 2: LBP set for (P=4,R=1), (P=8,R=1), (P=8,R=2)

1) Local Binary Patterns: For a given reference pixel c, the LBP descriptor is computed by comparing its grey level g_c intensity to a set of P surrounding pixel intensities g_p found on an evenly spaced circle with radius R, centered on c. Fig. 2 shows three examples of LBP sets. The LBP

descriptor is given by:

$$LBP_{P,R} = \sum_{p=0}^{P-1} \mathbb{1}_{\{g_p - g_c \geq 0\}} 2^p, \tag{10}$$

where $\mathbb{1}_{\{\cdot\}}$ is the indicator function. As proposed by [22], rotationally invariant LBPs, $LBP^{ri}_{P,R}$, are achieved by left-shifting the P-bit LBP number until it is maximized. The shifted LBP is then labeled according to its uniformity, U, which is the number of $0/1$ transitions. LBPs with uniformity less than or equal to two are called *uniform* and LBPs with more than two transitions are called *nonuniform*. Rotationally invariant LBPs are labeled as:

$$LBP^{ri}_{P,R} = \begin{cases} \sum_{p=0}^{P-1} \mathbb{1}_{\{g_p - g_c \geq 0\}} & \text{if } U \leq 2 \\ P+1 & \text{otherwise} \end{cases} \tag{11}$$

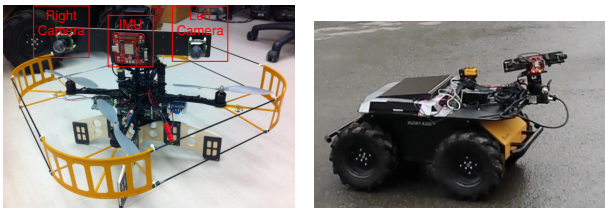
Once we have the LBP descriptor for each pixel in the patch, they are combined into a $P+1$ histogram to form a descriptor. In our implementation, we concatenated three histograms for three LBP patterns to provide a compromise between efficiency and expressive power. These were parameterized as (P=8, R=1), (P=16, R=2) and (P=24, R=3), forming our final patch descriptor. Using bigger radii would result in higher scale-invariance at the expense of detecting higher-frequency texture. We computed the LBP image descriptors for each patch in our training dataset and clustered them using the unsupervised k-means clustering algorithm and saved the model for later use (in Fig. 1). We found three or four clusters to perform well in our experiments.

C. Online Labeling

Online labeling is used to assign the texture classes a value corresponding to their goodness for tracking. The online labeling process is shown on the right side of Fig. 1 and begins similarly to offline clustering. Each image frame used by MORESLAM is divided into patches and the patch descriptors are found. The nearest texture class is predicted for each patch descriptor using the k-means clustering model.

We assign a score to each keypoint extracted by MORESLAM from each left image frame, using a simple quality heuristic s_i, namely the number of times the keypoint's corresponding visible map point has been observed by keyframes, divided by its depth variance. We then associate each of these keypoints with the texture class k that corresponds to the region of the image they are observed in, and bin the quality scores of all visible keypoints under each texture class to obtain the bin b_k. For each texture class we track the mean and variance of these bins, μ_{b_k} and $\sigma^2_{b_k}$ respectively. Finally, each class is given a score, S_k equal to:

$$S_k = \frac{\mu_{b_k}}{\sigma^2_{b_k}} \tag{12}$$



(a) (b)

Figure 3: (a) Pelican UAV with mounted sensor module (b) Husky with the stereo + IMU sensor on a pan-tilt unit.

which are then normalized across all texture classes to sum to one, as is shown in Fig. 1.

V. ACTIVE GAZE CONTROL

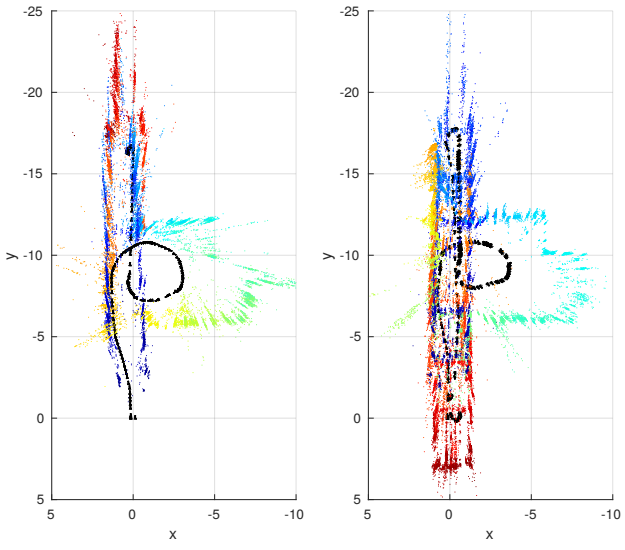
Active gaze control is used to update the gaze of the camera at each new image frame based on the scores of the classified image patches (in Fig. 1, each patch is colored according to its texture class). Each patch inherits the score of the texture class it belongs to, indicated by the numbers on each patch in Fig. 1. For each patch, a vector from its center to the image center is computed and multiplied by the patch score. Summing these weighted vectors results in a centroid which is a position on the image that represents the direction of interest. An example of this is shown in Fig. 6 as the red dot. A pan-tilt unit is then used to point the camera towards the direction of the centroid. We limit the movement of the pan-tilt motors by one degree for each camera frame to limit abrupt motions and the effect of motion blur. Our cameras record at approximately 15Hz, so in our experiments the PTU moves slowly, and so does the robot.

VI. EXPERIMENT SETUP AND RESULTS

A. Stereo + IMU + PTU Setup

Our custom sensor module (as shown in Fig. 3) consists of two IDS Imaging uEye LE USB2 cameras along with a VectorNav VN-100 IMU mounted to a rigid platform. The cameras are mounted in a stereo fashion with a baseline of 30cm and capture 752 x 480 resolution images. The IMU runs at 100Hz and provides a synchronous trigger to the cameras at 15Hz. The left camera runs with auto-exposure, and upon each change it sets the right camera's exposure setting through software. The rigid transformation between the two cameras and the IMU was estimated using the Kalibr library, using both a checkerboard pattern and an AprilTag pattern, and choosing the calibration estimate that achieved average reprojection errors of less than 0.2 pixels. A dual core 3.4 GHz Intel NUC5i7RY baseboard, with 16GB of RAM and a 512GB SSD is used to acquire the images and perform state estimation. The sensor module and computer together weigh $\sim$450 grams.

The stereo and IMU sensor module is attached to a pan-tilt base from Directed Perception, which is mounted on a



(a) (b)

Figure 4: Indoor stereo + IMU SLAM estimates projected on the xy-plane (a) Under fixed gaze, the system got lost during the in-place turn at (x,y)=(0,-17) (b) Under actively-controlled gaze, the system reconstructed the entire trajectory, without getting lost, by looking towards the ceiling during the in-place turn. Keyframes are shown in black. Blue map points were triangulated first. Best viewed in color.

Clearpath Husky mobile robot, as shown in Fig. 3. It is worth mentioning that due to being small and lightweight, we have been able to use the stereo and IMU module in multiple experimental configurations in numerous missions, taking place in the air, on the ground, and underwater.

B. Indoor Texture-Aware SLAM

We compared the performance of our SLAM system in indoor hallway spaces using fixed gaze versus the gaze control scheme described in V. The trajectory of the robot was remote-controlled by a human operator, who executed almost the same trajectory at the same speed in both experimental scenarios. The trajectory includes straight lines, a gradual turn around a column, and a very abrupt in-place turn at the end of the hallway. The latter is the most challenging part of the trajectory because its surroundings include white walls and narrow corridors.

As is shown in Fig. 4 the fixed gaze trajectory lost visual track during the in-place turn, while under active gaze control, the camera switched to tracking visual features on the ceiling, avoiding the effects of the feature-poor walls that were surrounding it. As a result, under active gaze control the SLAM system managed to reconstruct the entire path of the robot, as well as the map of the corridors, while under fixed gaze the reconstruction failed halfway. In the experiments conducted during this scenario we set the texture classifier to distinguish between three texture classes, which we will refer to as Red, Green, and Blue. Of these classes, Red and Blue correspond to feature-rich surfaces, while Green seems to be associated with white walls and visual uniformity. Switching from fixed gaze to active gaze increased the tendency of the system to avoid Green regions, which is shown more clearly in Fig. 5. It also increased the preference of gazing at map points in the Blue texture class by 10%. At the level of feature statistics, the difference between the average number of observations of each visible map point is insignificant, with both the fixed gaze and the active gaze tracking map points across 3 different keyframes on average. We made similar observations regarding the variance of the triangulated features: enabling the gaze control module did not seem to have a significant effect on the uncertainty, although that is environment-dependent. In addition, a small reduction in average reprojection error was achieved during the active gaze session, where the mean reprojection error was 0.59 pixels, compared to the fixed gaze session where it was 0.65 pixels. Overall, the largest contribution of the gaze control

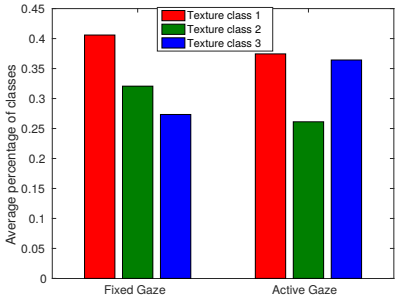


Figure 5: Average percentage of keypoint classification into three classes during fixed gaze vs. during active gaze.

module is the marked increase in tracking robustness, while the increase in accuracy is less significant.

C. Texture Classification During Flight

In our prior work on MORESLAM we validated our system in underwater and aerial medium-altitude settings. Underwater we performed localization and 3D mapping of a shipwreck. In flight deployments we demonstrated that the scene needed to be sufficiently close to the camera to allow triangulation, otherwise scale drift occurred. In both of these missions, the SLAM system spent significant computation effort in trying to triangulate points that were too far away. In the case of the UAV data this included clouds in the sky and mountains on the horizon. Although we did not have the opportunity to perform active gaze control experiments during these missions, we present classification results on the data collected during the UAV missions. These results, shown in Fig. 6, indicate that our texture classification scheme can be used to focus the feature extraction efforts

of the SLAM system to the most promising areas for triangulation, even though the exact relationship between the depth of the scene and the texture segmentation remains open for future work.

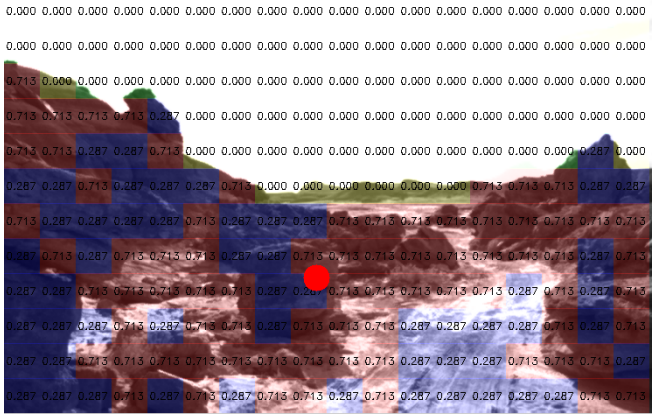


Figure 6: Texture classification on data collected from a multi-rotor flying at 10 meters altitude in the deserts of Utah. The sky and the horizon are classified under the Green class, signaling visual uniformity. The quality scores of the Green class have been updated through SLAM feedback to be zero. The red circle indicates where the camera should focus next. (Best viewed in color)

VII. CONCLUSIONS

We presented a fully integrated active SLAM system that incorporates visual and inertial information, as well as a gaze selection module that directs the camera towards feature-rich regions of the scene. We have demonstrated that this module is a crucial component that not only facilitates the long-term operation of the SLAM system, but is indispensable in avoiding many pathological motion scenarios where triangulating new map points is impossible. Vice-versa the gaze selection module is also informed by the SLAM system in updating its evaluation of texture classes, based on user-defined quality metrics. The two systems work in unison in real-time and result in increased robustness.

REFERENCES

[1] A. Harmat, I. Sharf, and M. Trentini, "Parallel tracking and mapping with multiple cameras on an unmanned aerial vehicle," *Intelligent Robotics and Applications*, pp. 421–432, 2012.

[2] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast Semi-Direct Monocular Visual Odometry ," in *IEEE ICRA*, 2014.

[3] S. Shen, N. Michael, and V. Kumar, "Tightly-coupled monocular visual-inertial fusion for autonomous flight of rotorcraft MAVs," in *IEEE ICRA*, May 2015, pp. 5303–5310.

[4] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visualinertial odometry using nonlinear optimization," *Intl. Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, 2015.

[5] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "Imu preintegration on manifold for efficient visual-inertial map estimation," in *Robotics Science and Systems*, June 2015.

[6] J. Hernandez, K. Tsotsos, and S. Soatto, "Observability, identifiability and sensitivity of vision-aided inertial navigation," in *IEEE Intl. Conference on Intelligent Robots and Systems*, May 2015, pp. 2319–2325.

[7] J. A. Hesch, D. G. Kottas, S. L. Bowman, and S. I. Roumeliotis, "Towards consistent vision-aided inertial navigation," in *Algorithmic Foundations of Robotics X*. Springer, 2013, pp. 559–574.

[8] R. Sim and G. Dudek, "Effective exploration strategies for the construction of visual maps," in *Proc. Intelligent Robots and Systems (IROS)*, vol. 4. IEEE, 2003, pp. 3224–3231.

[9] J. Malik and R. Rosenholtz, "Computing local surface orientation and shape from texture for curved surfaces," *International Journal of Computer Vision*, vol. 23, no. 2, pp. 149–168.

[10] A. P. Witkin, "Recovering surface shape and orientation from texture," *Artificial Intelligence*, vol. 17, no. 1, pp. 17 – 45, 1981.

[11] T. Kollar and N. Roy, "Efficient optimization of information-theoretic exploration in slam," in *23rd National Conference on Artificial Intelligence - V3*. AAAI Press, 2008, pp. 1369–1375.

[12] R. He, S. Prentice, and N. Roy, "Planning in information space for a quadrotor helicopter in a gps-denied environment," in *Robotics and Automation*, May 2008, pp. 1814–1820.

[13] S. Isler, R. Sabzevari, J. Delmerico, and D. Scaramuzza, "An information gain formulation for active volumetric 3d reconstruction," in *IEEE Intl. Conference on Robotics and Automation*, 2016.

[14] A. Das and S. Waslander, "Entropy based keyframe selection for multi-camera visual slam," in *IEEE Intelligent Robots and Systems*, Sept 2015, pp. 3676–3681.

[15] V. Peretroukhin, J. Kelly, and T. D. Barfoot, "Optimizing camera perspective for stereo visual odometry," in *Computer and Robot Vision (CRV)*, May 2014.

[16] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: A versatile and accurate monocular slam system," *Trans. on Robotics*, vol. 31, no. 5, pp. 1147–1163, Oct 2015.

[17] P. Furgale, J. Rehder, and R. Siegwart, "Unified temporal and spatial calibration for multi-sensor systems," in *IEEE Intelligent Robots and Systems*, Nov 2013, pp. 1280–1286.

[18] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g2o: A general framework for graph optimization," in *IEEE Intl. Conference on Robotics and Automation*, May 2011, pp. 3607–3613.

[19] R. Hartley and P. Sturm, "Triangulation," in *CVIU*, November 1997, pp. 146–157.

[20] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint Kalman filter for vision-aided inertial navigation," in *IEEE Intl. Conference on Robotics and Automation*, Rome, Italy, 2007, pp. 3565–3572.

[21] D. Zou and P. Tan, "Coslam: Collaborative visual slam in dynamic environments," *PAMI*, 2013.

[22] T. Ojala, M. Pietikainen, and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *PAMI*, vol. 24, no. 7, pp. 971–987.