2017 IEEE International Conference on Robotics and Automation (ICRA)
Singapore, May 29 - June 3, 2017

Attention and Anticipation in Fast Visual-Inertial Navigation

Luca Carlone and Sertac Karaman

Abstract— ***Visual attention*** **is the cognitive process that allows humans to parse a large amount of sensory data by selecting relevant information and filtering out irrelevant stimuli. This papers develops a computational approach for visual attention in robots. We consider a Visual-Inertial Navigation (VIN) problem in which a robot needs to estimate its state using an on-board camera and an inertial sensor. The robot can allocate limited resources to VIN, due to time and energy constraints. Therefore, we answer the following question: under limited resources, what are the most relevant visual cues to maximize the performance of visual-inertial navigation? Our approach has four key features. First, it is** ***task-driven*****, in that the selection of the visual cues is guided by a metric quantifying the task performance. Second, it exploits the notion of** ***anticipation*****, since it uses a simplified model for forward-simulation of robot dynamics, predicting the utility of a set of visual cues over a time horizon. Third, it is** ***efficient and easy to implement*****, since it leads to a greedy algorithm for the selection of the most relevant visual cues. Fourth, it provides** ***formal performance guarantees*****: we leverage submodularity to prove that the greedy selection cannot be far from the optimal (combinatorial) selection. Simulations and real experiments on agile micro aerial vehicles show that our approach leads to dramatic improvements in the VIN performance. In the easy scenarios, our approach outperforms the state of the art in terms of localization errors. In the most challenging scenarios, it enables accurate visual-inertial navigation while the state of the art fails to track robot's motion during aggressive maneuvers.**

I. Introduction

The human brain can extract conceptual information from an image in a time lapse as short as 13 ms [26]. One has proof of the human's capability to seamlessly process large amount of sensory data in everyday tasks, including driving a car on a highway, or walking on a crowded street. In the cognitive science literature, there is agreement on the fact that efficiency in processing the large amount of data we are confronted with is due to our ability to prioritize some aspects of the visual scene, while ignoring others [6]. One can imagine that sensory inputs compete to have access to the limited computational resources of our brain. These resource constraints are dictated by the fixed amount of energy available to the brain as well as time constraints imposed by time-critical tasks. *Attention* is the selective process that chooses the most relevant stimuli so to maximize performance, under limited resources.

The astonishing progress in robotics and computer vision over the last three decades might induce us to ask: how far is robot perception from human performance? Let us approach this question by looking at the state of the art in visual processing for different tasks. Without any claim to be exhaustive, we consider few representative papers (sampled over the last 2 years) and we only look at timing performance. A state-of-the-art approach for object detection [27] detects objects in 22ms on a GPU. A high-performance approach for stereo reconstruction [25] builds a triangular mesh of a scene in 10-100ms on a single CPU (at resolution 800×600). A state-of-the-art vision-based SLAM approach [21] requires around 400ms for local mapping and motion tracking and more than 1s for global refinement (CPU, multiple cores). For each task, in isolation, modern algorithms require more time than what a human needs to parse an entire scene. Arguably, while a merit of the robotic and computer vision communities has been to push performance in each task, we are quite far from a computational model in which all these tasks (pose estimation, 3D reconstruction, scene understanding) are concurrently executed in the blink of an eye.

L. Carlone and S. Karaman are with the Laboratory for Information & Decision Systems, Massachusetts Institute of Technology, Cambridge, MA, USA, {lcarlone,sertac}@mit.edu

One might argue that catching up with human efficiency is only a matter of time: according to Moore's law, the available computation grows at exponential rate, hence we only need to wait for more powerful computers. An analogous argument would suggest that using GPU rather than CPU would boost performance in some of the tasks mentioned above. By comparison with human performance, we realize that this argument is not completely accurate. While it is true that we can keep increasing computation to meet given time constraints (i.e., enable faster processing of sensory data), the increase in computation implies an increase in energy consumption. On the other hand, human processing constantly deals with limited time and energy constraints, and is parsimonious in allocating only the resources that are necessary to accomplish its goals.

In this paper we hint to a different solution: *we argue that fast processing results from prioritizing sensory data in a task-dependent fashion*. We provide a concrete example in the context of visual-inertial navigation (VIN): our robot has to use visual and inertial data to estimate its location in the environment and it has to do so under given resource constraints. Therefore, our goal is to design an approach that can optimally allocate the limited on-board resources of a robot towards maximizing localization performance.

Contribution. We propose an approach for visual-inertial navigation under limited computational resources. Due to constraints on the on-board processing the robot can only use a small number of visual features in the environment to support motion estimation. We design a visual attention mechanism that selects a suitable set of visual features to maximize localization accuracy; the general framework is given in Section III. Our approach is *task-driven*: it selects features that maximize a task-dependent performance metric, that we present in Section III-A. Our attention mechanism is *predictive* in nature: when deciding which feature is more useful, our approach performs fast forward-simulations of the state of the robot leading to a feature selection that is aware of the dynamics and the "intentions" of the robot. The forward simulation is based on a simplified model that we present in Section III-B. Since the optimal allocation of the resources in a hard combinatorial problem, in Section IV we present a greedy algorithm for attention allocation. In the same section, we leverage recent results on submodularity to provide formal performance guarantees on the greedy algorithm. Section V provides an experimental evaluation of the proposed approach. The results show that our approach boosts VIN performance, and enables accurate localization under agile motion and strict resource constraints.

A longer version of this paper has been released as a technical report [5]. We refer the reader to [5] for technical proofs, a broader coverage of related work, more algorithms (including convex relaxations), and extra experiments.

Notation. We use lower and upper case bold letters to denote vectors ($\boldsymbol{v}$) and matrices ($\boldsymbol{M}$), respectively. Sets are denoted by sans script fonts (A). Non-bold face letters are used for scalars, indices, and functions. The symbol $|\mathsf{A}|$ denotes the cardinality of A, while $\|\cdot\|$ is the Euclidean norm for vectors and the spectral norm for matrices.

II. Related Work

Attention and Saliency in Neuroscience and Psychology. Attention is a central topic in human and animal vision research with more than 2500 papers published since the 1980s [6]. Here we review few basic concepts, using the surveys of Carrasco [6], and Borji and Itti [2] as main references. Carrasco [6] identifies three types of attention: *spatial*, *feature-based*, and *object-based*. Spatial attention prioritizes parts of the scene by moving the eyes towards a specific location (*overt* attention) or by focusing on relevant locations within the field of view (*covert* attention). Feature-based attention prioritizes specific features (e.g., color, motion direction) independently on their location. Object-based attention prioritizes specific objects. In this work, we are interested in covert spatial attention: which locations in the field of view are the most informative for navigation?

Feature selection in Robotics and Computer Vision. The idea of enhancing visual odometry and SLAM via active feature selection is not novel. Sim and Dudek [32] and Peretroukhin *et al.* [24] use training data to learn a model of the quality of the visual features. Ouerhani *et al.* [23] construct a topological map of attentional landmarks. Newman and Ho [22] consider a robot equipped with camera and laser and perform appearance-based feature selection. Sala *et al.* [28] use co-visibility to select good landmarks. Siagian and Itti [30] investigate a bio-inpired attention model within Monte Carlo localization. Frintrop and Jensfelt [10] use an attention model for landmark selection and gaze control. Strasdat *et al.* [33] use reinforcement learning for landmark selection. Chli and Davison [7], and Kopitkov and Indelman [15] propose active approaches to improve feature matching and support decision-making.

The contributions that are most related to our proposal are the one of Lerner *et al.* [17], Mu *et al.* [20], and Zhang and Vela [34]. Lerner *et al.* [17] study landmark selection in a localization problem with known landmarks. Mu *et al.* [20] propose a two-stage approach to select a subset of landmarks to minimize the probability of collision, and a subset of measurements to accurately localize those landmarks. Zhang and Vela [34] perform feature selection using an observability score and provide sub-optimality guarantees using submodularity. Our proposal differs from these works in a number of aspects. First, we study feature selection within a more challenging problem (visual-inertial odometry), assuming no prior knowledge of the environment. Second, our approach relies on *anticipation*: we use a simplified linear model for forward dynamics simulation, which informs feature selection about the future intention of the robot. Third, we consider different metrics that quantify the motion estimation error and we purposely disregard the map reconstruction quality.

Sensor scheduling and submodularity. Feature selection is deeply related to sensor selection and scheduling in control theory. A common setup for sensor selection is the case in which N sensors monitor a phenomenon of interest and one has to choose κ out of the N sensors to maximize some information-collection metric. Related work includes approaches based on convex relaxation, Bayesian optimal design, and submodular optimization [16], [29].

Visual-Inertial Navigation. The literature on VIN is vast with many recent contributions, including approaches based on filtering [11], fixed-lag smoothing [31], [18], and full smoothing [19], [9]. We refer the reader to [9] for a review.

III. Attention in Visual-Inertial Navigation

We design an attention mechanism that selects κ visual features -out of the N features seen in the current frame- to maximize the performance in VIN. The approach can deal with both monocular and stereo cameras and we only distinguish the two setups when necessary.

We call F the set of all available visual features (with $|\mathsf{F}|= N$). Denoting with $f(\cdot)$ our task-dependent performance metric, we state our feature selection problem as follows:

$$\max_{\mathsf{S}\subset\mathsf{F}} \quad f(\mathsf{S}) \quad \text{subject to} \quad |\mathsf{S}|\leq \kappa \tag{1}$$

The problem looks for a subset of features S, containing no more than κ features, which optimizes the task performance $f(\cdot)$. The budget κ is dictated by the available on-board resources. In the rest of this paper we are interested in designing a suitable performance metric $f(\mathsf{S})$ for VIN, and in providing fast approximation algorithms to solve (1).

Our goal is to design a task-dependent performance metric $f(\cdot)$ that captures aspects already deemed relevant in related work. First, the metric has to reward the selection of the distinctive features (in terms of appearance) since these are more likely to be re-observed. Second, the metric has to reward features that remain within the field of view. Third, the metric has to reward features that are informative for motion estimation. In the following we propose two performance metrics that seamlessly capture all these aspects.

A. Task-dependent Performance Metrics

Here we propose two metrics that quantify the accumulation of estimation errors over an horizon H, under the selection of a set of visual features S. Assume that k is the time instant at which the features need to be selected. Let us call $\boldsymbol{x}_k$ the state of the robot at time k: we will be more precise about the variables included in $\boldsymbol{x}_k$ in Section III-B.1. We denote with $\boldsymbol{x}_{k:k+H} \doteq [\boldsymbol{x}_k \; \boldsymbol{x}_{k+1} \; \ldots \; \boldsymbol{x}_{k+H}]$ the future states within the horizon H. Moreover, we call $\boldsymbol{P}_{k:k+H}$ the covariance matrix of our estimate of $\boldsymbol{x}_{k:k+H}$, and $\boldsymbol{\Omega}_{k:k+H} \doteq \boldsymbol{P}^{-1}_{k:k+H}$ the corresponding information matrix. Two natural metrics that quantify the estimation error are described in the following.

Worst-case Estimation Error. The worst-case estimation error variance is quantified by the maximum eigenvalue $\lambda_{\max}(\boldsymbol{P}_{k:k+H})$ of the covariance $\boldsymbol{P}_{k:k+H}$, see e.g., [14]. Calling $\lambda_{\min}(\boldsymbol{\Omega}_{k:k+H})$ the smallest eigenvalue of the information matrix $\boldsymbol{\Omega}_{k:k+H}$, if follows that $\lambda_{\max}(\boldsymbol{P}_{k:k+H}) = 1/\lambda_{\min}(\boldsymbol{\Omega}_{k:k+H})$, hence minimizing the worst-case error is the same as maximizing $\lambda_{\min}(\boldsymbol{\Omega}_{k:k+H})$. Note that the information matrix $\boldsymbol{\Omega}_{k:k+H}$ is function of the measurements, hence we write $\lambda_{\min}(\boldsymbol{\Omega}_{k:k+H}(\mathsf{S}))$. Hence our first metric is:

$$f_\lambda(\mathsf{S}) = \lambda_{\min}(\boldsymbol{\Omega}_{k:k+H}(\mathsf{S})) = \lambda_{\min}\left(\bar{\boldsymbol{\Omega}}_{k:k+H} + \textstyle\sum_{l\in\mathsf{S}} \Delta_l\right) \tag{2}$$

where on the right-hand-side, we exploited the additive structure of the information matrix, where $\bar{\boldsymbol{\Omega}}_{k:k+H}$ is the information matrix of the estimate when no features are selected (intuitively, this is the inverse of the covariance resulting from the IMU integration), while Δ_l is the information matrix associated with the selection of the l-th feature. We will give an explicit expression to $\bar{\boldsymbol{\Omega}}_{k:k+H}$ and Δ_l in Section III-B.

Volume and Mean Radius of the Confidence Ellipsoid. The ϵ-confidence ellipsoid contains the estimation error with probability ϵ. Its volume and mean radius are proportional to the determinant of the covariance. Therefore to minimize the volume and the mean radius of the confidence ellipsoid we can equivalently minimize the determinant of the covariance. Moreover, since $\log\det(\boldsymbol{P}_{k:k+H}) = -\log\det(\boldsymbol{\Omega}_{k:k+H})$, then minimizing the size of the confidence ellipsoid is the same as maximizing the log-determinant of the information matrix, leading to our second performance metric:

$$f_{\det}(\mathsf{S}) = \log\det(\boldsymbol{\Omega}_{k:k+H}(\mathsf{S})) = \log\det\left(\bar{\boldsymbol{\Omega}}_{k:k+H} + \textstyle\sum_{l\in\mathsf{S}} \Delta_l\right) \tag{3}$$

Probabilistic Feature Tracks. The performance metrics described so far are task-dependent in that they both quantify the motion estimation accuracy; moreover, they are predictive, in the sense that they look at the result of selecting a set of features over a short (future) horizon. The only aspect that is not yet modeled is that, even when a feature is in the field of view, there is some chance that it will not be re-detected and the corresponding feature track will be lost.

To model the probability that a feature track is lost, we introduce N Bernoulli random variables $b_1, \ldots, b_N$. Each variable b_l represents the outcome of the tracking of feature l: if $b_l = 1$, then the feature is successfully tracked, otherwise, the feature track is lost. For each feature we assume $p_l = \text{Prob}(b_l = 1)$ to be given; in practice one can correlate p_l to the appearance of feature l, such that more distinctive features have higher probability of being tracked if they are in the field of view. Using the binary variables $\boldsymbol{b} \doteq \{b_1, \ldots, b_N\}$, we write the information matrix at the end of the horizon as:

$$\boldsymbol{\Omega}_{k:k+H}(\mathsf{S}, \boldsymbol{b}) = \bar{\boldsymbol{\Omega}}_{k:k+H} + \textstyle\sum_{l\in\mathsf{S}} b_l \Delta_l \tag{4}$$

which has a clear interpretation: if the l-th feature is correctly tracked, then $b_l = 1$ and the corresponding information matrix Δ_l is added to $\bar{\boldsymbol{\Omega}}_{k:k+H}$; on the other hand, if the feature tracks is lost, then $b_l = 0$ and Δ_l disappears from (4).

Since $\boldsymbol{b}$ is a random vector, our information matrix is now a stochastic quantity $\boldsymbol{\Omega}_{k:k+H}(\mathsf{S}, \boldsymbol{b})$, hence we redefine our performance metrics to include the expectation over $\boldsymbol{b}$:

$$f(\mathsf{S}) = \mathbb{E}\left[f(\boldsymbol{\Omega}_{k:k+H}(\mathsf{S}, \boldsymbol{b}))\right] \tag{5}$$

where we use the symbol f to denote either f_λ or $f_{\det}$. Computing the expectation (5) leads to a sum with a combinatorial number of terms, which is hard to even evaluate. To avoid the combinatorial explosion, we use Jensen's inequality:

$$\mathbb{E}\left[f(\boldsymbol{\Omega}_{k:k+H}(\mathsf{S}, \boldsymbol{b}))\right] \geq f(\mathbb{E}\left[\boldsymbol{\Omega}_{k:k+H}(\mathsf{S}, \boldsymbol{b})\right]) \tag{6}$$

which produces a lower bound for our expected cost. In the follow we maximize this lower bound, rather that the original cost. The advantage of doing so is that the right-hand-side of (6) can be efficiently computed as:

$$f(\mathbb{E}\left[\boldsymbol{\Omega}_{k:k+H}(\mathsf{S}, \boldsymbol{b})\right] = f(\bar{\boldsymbol{\Omega}}_{k:k+H} + \textstyle\sum_{l\in\mathsf{S}} p_l \Delta_l) \tag{7}$$

where we used the definition (4), the fact that the expectation is a linear operator, and that $\mathbb{E}[b_l] = p_l$. Therefore, our performance metrics can be written explicitly as:

$$\begin{aligned} f_\lambda(\mathsf{S}) &= \lambda_{\min}\left(\bar{\boldsymbol{\Omega}}_{k:k+H} + \textstyle\sum_{l\in\mathsf{S}} p_l \Delta_l\right) \\ f_{\det}(\mathsf{S}) &= \log\det\left(\bar{\boldsymbol{\Omega}}_{k:k+H} + \textstyle\sum_{l\in\mathsf{S}} p_l \Delta_l\right) \end{aligned} \tag{8}$$

which coincide with the deterministic counterparts (2), (3) when $p_l = 1, \forall l$. Interestingly, in (8) the probability that a feature is not tracked simply discounts the corresponding information content. Therefore, features that are more likely to get lost are considered less informative, which is a desired behavior. In the following we focus on VIN and we give explicit expressions for $\bar{\boldsymbol{\Omega}}_{k:k+H}$ and Δ_l.

B. Forward-simulation Model

The feature selection model proposed in Section III and the metrics in Section III-A require to predict the evolution of the information matrix over the horizon H. The evolution of the information matrix depends on the future IMU and camera measurements. In the following we show how to forward-simulate the evolution of the information matrix; we remark that we do not require to simulate actual IMU measurements, but only the corresponding information matrix.

Our forward-simulation model depends on the future motion of the robot. This enables a more clever selection of features during sharp turns and aggressive maneuvers. In practice, the future poses correspond to the desired trajectory that the controller is trying to follow.

1) IMU Model: Our simplified IMU model is based on a key assumption: the accumulation of the rotation error from gyroscope integration over the time horizon H is negligible. In other words, the relative rotation estimates predicted by the gyroscope are accurate. This assumption is realistic, even for inexpensive IMUs, since the rotation drift is small over the time horizon considered in our attention system ($\leq$ 3s).

Assuming that the rotations are accurately known allows restricting the state to the robot position, velocity, and accelerometer bias. Therefore, in the rest of this paper, the state of the robot at time k is $\boldsymbol{x}_k \doteq [\boldsymbol{t}_k \ \boldsymbol{v}_k \ \boldsymbol{b}_k]$, where $\boldsymbol{t}_k \in \mathbb{R}^3$ is the position of the robot, $\boldsymbol{v}_k \in \mathbb{R}^3$ is its velocity, and $\boldsymbol{b}_k$ is the (time-varying) accelerometer bias. We use the symbol $\boldsymbol{R}_k$ to denote the attitude of the robot at time k.

We estimate the state of the robot at each frame (or, equivalently, keyframe). Therefore, the goal of this subsection, similarly to [9], is to reformulate a set of IMU measurements between two consecutive frames k and j as a single measurement that constrains $\boldsymbol{x}_k$ and $\boldsymbol{x}_j$. Differently from [9], we show how to get a *linear* measurement model.

The on-board accelerometer measures the acceleration $\boldsymbol{a}_k$ of the sensor with respect to an inertial frame, and is affected by additive white noise $\boldsymbol{\eta}_k$ and a slowly-varying sensor bias $\boldsymbol{b}_k$. Therefore, the measurement $\tilde{\boldsymbol{a}}_k \in \mathbb{R}^3$ acquired by the accelerometer at time k is modeled as [9]:

$$\tilde{\boldsymbol{a}}_k = \boldsymbol{R}_k^{\mathsf{T}} (\boldsymbol{a}_k - \mathrm{g}) + \boldsymbol{b}_k + \boldsymbol{\eta}_k, \tag{9}$$

where g is the gravity vector, expressed in the inertial frame.

Given position $\boldsymbol{t}_k$ and velocity $\boldsymbol{v}_k$ at time k, we can forward-integrate and obtain $\boldsymbol{t}_j$ and $\boldsymbol{v}_j$ at time $j > k$:

$$\begin{aligned}
\boldsymbol{v}_j &= \boldsymbol{v}_k + \textstyle\sum_{i=k}^{j-1} \boldsymbol{a}_i \delta \\
&\quad \text{(from (9) we know } \boldsymbol{a}_i = \mathrm{g} + \boldsymbol{R}_i(\tilde{\boldsymbol{a}}_i - \boldsymbol{b}_i - \boldsymbol{\eta}_i), \\
&\quad \text{and assuming constant bias between frames, } \boldsymbol{b}_i = \boldsymbol{b}_k) \\
&= \boldsymbol{v}_k + \mathrm{g}\delta_{kj} + \textstyle\sum_{i=k}^{j-1} \boldsymbol{R}_k (\tilde{\boldsymbol{a}}_i - \boldsymbol{b}_k - \boldsymbol{\eta}_i)\, \delta \qquad (10) \\
\boldsymbol{t}_j &= \boldsymbol{t}_k + \textstyle\sum_{i=k}^{j-1} \left(\boldsymbol{v}_i \delta + \frac{1}{2}\boldsymbol{a}_i \delta^2\right) \\
&\quad \text{(substituting } \boldsymbol{a}_i = \mathrm{g} + \boldsymbol{R}_i(\tilde{\boldsymbol{a}}_i - \boldsymbol{b}_k - \boldsymbol{\eta}_i)) \\
&= \boldsymbol{t}_k + \textstyle\sum_{i=k}^{j-1} (\boldsymbol{v}_i \delta + \frac{1}{2}\mathrm{g}\delta^2 + \frac{1}{2}\boldsymbol{R}_k (\tilde{\boldsymbol{a}}_i - \boldsymbol{b}_k - \boldsymbol{\eta}_i)\, \delta^2) \\
&\quad \text{(substituting } \boldsymbol{v}_j \text{ from (10) with } j = i) \\
&= \boldsymbol{t}_k + \tfrac{1}{2}\mathrm{g}\hat{\delta}_{kj}^2 + \textstyle\sum_{i=k}^{j-1} \frac{1}{2}\boldsymbol{R}_i (\tilde{\boldsymbol{a}}_i - \boldsymbol{b}_k - \boldsymbol{\eta}_i)\, \delta^2 \\
&+ \textstyle\sum_{i=k}^{j-1} (\boldsymbol{v}_k + \mathrm{g}\delta_{ki} + \sum_{h=k}^{i-1} \boldsymbol{R}_h (\tilde{\boldsymbol{a}}_h - \boldsymbol{b}_k - \boldsymbol{\eta}_h)\, \delta) \qquad (11)
\end{aligned}$$

where δ is the sampling time of the IMU, $\delta_{kj} \doteq \sum_{i=k}^{j-1} \delta$, and $\hat{\delta}_{kj}^2 \doteq \sum_{i=k}^{j-1} \delta^2$; as in [9], we assumed that the IMU bias remains constant between two frames. The evolution of the bias across frames can be modeled as a random walk:

$$\boldsymbol{b}_j = \boldsymbol{b}_k - \boldsymbol{\eta}_{kj}^{\mathbf{b}} \tag{12}$$

where $\boldsymbol{\eta}_{kj}^{\mathbf{b}}$ is a zero-mean random vector.

Noting that the state appears linearly in (10)-(12), we rewrite the three expressions together in matrix form:

$$\boldsymbol{z}_{kj}^{\text{IMU}} = \boldsymbol{A}_{kj} \boldsymbol{x}_{k:k+H} + \boldsymbol{\eta}_{kj}^{\text{IMU}} \tag{13}$$

where $\boldsymbol{z}_{kj}^{\text{IMU}} \in \mathbb{R}^9$ is a suitable vector, and $\boldsymbol{\eta}_{kj}^{\text{IMU}} \in \mathbb{R}^9$ is zero-mean random noise. While $\boldsymbol{z}_{kj}^{\text{IMU}}$ is function of the future IMU measurements, this vector is not actually used in our approach (what matters is $\boldsymbol{A}_{kj}$ and the information matrix of $\boldsymbol{\eta}_{kj}^{\text{IMU}}$), hence we do not need to simulate future measurements. An explicit expression for the matrix $\boldsymbol{A}_{kj} \in \mathbb{R}^{9 \times 9H}$, the vector $\boldsymbol{z}_{kj}^{\text{IMU}}$, and the covariance of $\boldsymbol{\eta}_{kj}^{\text{IMU}}$ are given in [5].

Linear estimation theory tells us that the information matrix of the optimal estimate of $\boldsymbol{x}_{k:k+H}$, given the measurements (13) for all frames k,j in the horizon H, is:

$$\bar{\boldsymbol{\Omega}}_{k:k+H} = \textstyle\sum_{kj \in \mathsf{H}} (\boldsymbol{A}_{kj}^{\mathsf{T}} \boldsymbol{\Omega}_{kj}^{\text{IMU}} \boldsymbol{A}_{kj}) \tag{14}$$

where H is the set of consecutive frames within the time horizon H, and $\boldsymbol{\Omega}_{kj}^{\text{IMU}} \in \mathbb{R}^{9\times 9}$ is the information matrix of the noise vector $\boldsymbol{\eta}_{kj}^{\text{IMU}}$. The matrix $\bar{\boldsymbol{\Omega}}_{k:k+H}$ is precisely the information matrix of the state estimate before any vision measurement is selected, that we already introduced in (4).

2) Vision Model: Also for the vision measurements, we are interested in designing a linear measurement model, which simplifies the actual (nonlinear) perspective projection. A (calibrated) pixel measurement of an external 3D point (or landmark) l identifies the 3D bearing of the landmark in the camera frame. Mathematically, if we call $\boldsymbol{u}_{kl}$ the unit vector, corresponding to the (calibrated) pixel observation of l from the robot pose at time k, $\boldsymbol{u}_{kl}$ satisfies the following relation:

$$\boldsymbol{u}_{kl} \times \left((\boldsymbol{R}_{\text{cam},k}^{\text{w}})^{\mathsf{T}} (\boldsymbol{p}_l - \boldsymbol{t}_{\text{cam},k}^{\text{w}})\right) = \mathbf{0}_3 \tag{15}$$

where $\times$ is the cross product between two vectors, $\boldsymbol{p}_l$ is the 3D position of landmark l (in the world frame), $\boldsymbol{R}_{\text{cam},k}^{\text{w}}$ and $\boldsymbol{t}_{\text{cam},k}^{\text{w}}$ are the rotation and translation describing the camera pose at time k (w.r.t. the world frame). In words, eq. (15) requires the observed point (transformed to the camera frame) to be collinear to the measured direction $\boldsymbol{u}_{kl}$, since the cross product measures deviation from collinearity.

For two vectors $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$, the cross product $\boldsymbol{v}_1 \times \boldsymbol{v}_2 = [\boldsymbol{v}_1]_\times \boldsymbol{v}_2$, where $[\boldsymbol{v}_1]_\times$ is a skew symmetric matrix built from $\boldsymbol{v}_1$. Moreover, the camera pose w.r.t. the world frame, $(\boldsymbol{R}_{\text{cam},k}^{\text{w}}, \boldsymbol{t}_{\text{cam},k}^{\text{w}})$, can be written as the composition of the IMU pose w.r.t. the world frame, $(\boldsymbol{R}_k, \boldsymbol{t}_k)$, and the relative pose of the camera w.r.t. the IMU, $(\boldsymbol{R}_{\text{cam}}^{\text{IMU}}, \boldsymbol{t}_{\text{cam}}^{\text{IMU}})$ (known from calibration). Using these considerations, we rewrite (15) as:

$$[\boldsymbol{u}_{kl}]_\times \left((\boldsymbol{R}_k \boldsymbol{R}_{\text{cam}}^{\text{IMU}})^{\mathsf{T}} (\boldsymbol{p}_l - (\boldsymbol{t}_k + \boldsymbol{R}_k \boldsymbol{t}_{\text{cam}}^{\text{IMU}}))\right) = \mathbf{0}_3 \tag{16}$$

In presence of measurement noise, (16) becomes:

$$[\boldsymbol{u}_{kl}]_\times \left((\boldsymbol{R}_k \boldsymbol{R}_{\text{cam}}^{\text{IMU}})^{\mathsf{T}} (\boldsymbol{p}_l - (\boldsymbol{t}_k + \boldsymbol{R}_k \boldsymbol{t}_{\text{cam}}^{\text{IMU}}))\right) = \boldsymbol{\eta}_{kl}^{\text{cam}} \tag{17}$$

where $\boldsymbol{\eta}_{kl}^{\text{cam}}$ is a zero-mean random noise with known covariance. Under the assumptions that rotations are known from gyroscope integration, the unknowns in model (17) are the robot position $\boldsymbol{t}_k$ (which is part of our state vector $\boldsymbol{x}_{k:k+H}$) and the position of the observed 3D landmark $\boldsymbol{p}_l$. The model is linear and can be written in matrix form as:

$$\boldsymbol{z}_{kl}^{\text{cam}} = \boldsymbol{F}_{kl} \boldsymbol{x}_{k:k+H} + \boldsymbol{E}_{kl} \boldsymbol{p}_l + \boldsymbol{\eta}_{kl}^{\text{cam}} \tag{18}$$

for a suitable vector $\boldsymbol{z}_{kl}^{\text{cam}}$, and matrices $\boldsymbol{F}_{kl}$ and $\boldsymbol{E}_{kl}$. Without loss of generality we assume $\boldsymbol{\eta}_{kl}^{\text{cam}}$ to have identical covariance: if this is not the case we can always multiply both members of (18) by the square-root information matrix of $\boldsymbol{\eta}_{kl}^{\text{cam}}$ and get a model with additive standard Gaussian noise.

Eq. (18) describes a single pixel measurement. In order to be triangulated, a point $\boldsymbol{p}_l$ has to be observed across multiple frames in the horizon H. Stacking the linear system (18) for each frame k from which l is visible, we get:

$$\boldsymbol{z}_l^{\text{cam}} = \boldsymbol{F}_l \boldsymbol{x}_{k:k+H} + \boldsymbol{E}_l \boldsymbol{p}_l + \boldsymbol{\eta}_l^{\text{cam}} \tag{19}$$

where $\boldsymbol{z}_l^{\text{cam}}$, $\boldsymbol{F}_l$, and $\boldsymbol{E}_l$ are obtained by stacking (row-wise) $\boldsymbol{z}_{kl}^{\text{cam}}$, $\boldsymbol{F}_{kl}$, and $\boldsymbol{E}_{kl}$, respectively, for all frames $k, \ldots, k+H$. As for the IMU model, the expression of $\boldsymbol{z}_l^{\text{cam}}$ is inconsequential for our derivation, as it does not influence the future state covariance. On the other hand $\boldsymbol{F}_l$ and $\boldsymbol{E}_l$ depend on the future measurements $\boldsymbol{u}_{kl}$: for this reason, computing these matrices require simulating pixel projections of $\boldsymbol{p}_l$ for each frame in the horizon. When using a stereo camera, we have an estimate of $\boldsymbol{p}_l$ hence we can easily project it to the future frames. In a monocular setup, we can guess the depth of new features from the existing features in the VIN back-end.

We cannot directly use the linear model (19) to estimate our state vector $\boldsymbol{x}_{k:k+H}$, since it contains the unknown position of landmark l. One may include the landmarks as part of the state but that would largely increase the dimension of the state space (and hence of the matrices in (8)). Moreover, it may create undesirable behaviors, e.g., the metrics might induce to select features that minimize the uncertainty of a far 3D point rather than focusing on the variables we are actually interested in (i.e., the robot state).

To avoid this undesirable effects, we analytically eliminate the 3D point from the estimation using the Schur complement trick [4]. We first write the information matrix of the joint state $[\boldsymbol{x}_{k:k+H}\ \boldsymbol{p}_l]$ from the linear measurements (19):

$$\boldsymbol{\Omega}_{k:k+H}^{(l)} = \left[\begin{array}{cc} \boldsymbol{F}_l^{\mathsf{T}}\boldsymbol{F}_l & \boldsymbol{F}_l^{\mathsf{T}}\boldsymbol{E}_l \\ \boldsymbol{E}_l^{\mathsf{T}}\boldsymbol{F}_l & \boldsymbol{E}_l^{\mathsf{T}}\boldsymbol{E}_l \end{array}\right] \tag{20}$$

Using the Schur complement trick we marginalize out the landmark l and obtain the information matrix of our state $\boldsymbol{x}_{k:k+H}$ given the measurements (19):

$$\Delta_l = \boldsymbol{F}_l^{\mathsf{T}}\boldsymbol{F}_l - \boldsymbol{F}_l^{\mathsf{T}}\boldsymbol{E}_l(\boldsymbol{E}_l^{\mathsf{T}}\boldsymbol{E}_l)^{-1}\boldsymbol{E}_l^{\mathsf{T}}\boldsymbol{F}_l \tag{21}$$

Eq. (21) is the (additive) contribution to the information matrix of our state estimate due to the measurements of a single landmark l. This is the matrix Δ_l that we already used in (4). It is worth noticing that $(\boldsymbol{E}_l^{\mathsf{T}}\boldsymbol{E}_l)^{-1}$ is the covariance of the estimate of the landmark position [4], and it is invertible as long as the landmark l can be triangulated.

IV. Attention Allocation: Algorithms and Performance Guarantees

We now discuss computational approaches to solve problem (1). It is known that finding the optimal subset $\mathsf{S}^\star$ which solves (1) exactly is NP-hard [1]. The approach we adopt in this paper is to design *approximation algorithms*, which are computationally efficient and provide performance guarantees (roughly speaking, produce a set which is not far from the optimal subset $\mathsf{S}^\star$). In particular, we discuss the use of greedy algorithms to approximately solve (1).

A. Greedy Algorithms and Lazy Evaluation

This section presents a greedy algorithm that selects κ features that (approximately) maximize the objective $f(\cdot)$ (being either f_λ or $f_{\det}$). The algorithm starts with an empty set $\mathsf{S}^\#$ and performs κ iterations. At each iteration, it adds the feature that induces the largest increase in the objective function. The pseudocode of the algorithm is given in Algorithm 1. In line 3 the algorithm starts with an empty set. The "for" loop in line 4 iterates κ times: at each time the best feature is added to the subset $\mathsf{S}^\#$ (line 17). The role of the "for" loop in line 9 is to compute the feature that induces the maximum increase in the cost (lines 13-15). The remaining lines provide a lazy evaluation mechanism. For each feature l we compute an upper bound on the cost $f(\mathsf{S}^\# \cup l)$ (line 6). The features are sorted (in descending order) according to this upper bound (line 7). The advantage of this is that by comparing the current best feature with this upper bound (line 10) we can avoid checking features that are guaranteed to attain a smaller cost.

Algorithm 1: Greedy algorithm with lazy evaluation

```
1  Input: $\bar{\boldsymbol{\Omega}}_{k:k+H}$, $\Delta_l$, for $l = 1, \ldots, N$, and $\kappa < N$ ;
2  Output: feature subset $\mathsf{S}^\#$, with $|\mathsf{S}^\#| = \kappa$ ;
3  $\mathsf{S}^\# = \emptyset$ ;
4  for $i = 1, \ldots, \kappa$ do
5      % Compute upper bound for $f(\mathsf{S}^\# \cup l)$, $l = 1, \ldots, N$
6      $[U_1, \ldots, U_N]$ = upperBounds($\bar{\boldsymbol{\Omega}}_{k:k+H}, \Delta_1, \ldots, \Delta_N$) ;
7      $\mathsf{F}^\downarrow$ = sort($U_1, \ldots, U_N$)  % Sort using upper bound
8      $f_{\max} = -1$ ;  $l_{\max} = -1$  % Initialize best feature
9      for $l \in \mathsf{F}^\downarrow$ do
10         if $U_l < f_{\max}$ then
11             break ;
12         end
13         if $f(\mathsf{S}^\# \cup l) > f_{\max}$ then
14             $f_{\max} = f(\mathsf{S}^\# \cup l)$ ;  $l_{\max} = l$ ;
15         end
16     end
17     $\mathsf{S}^\# = \mathsf{S}^\# \cup l_{\max}$ ;
18 end
```

In the following we provide two (computationally cheap) upper bounds for our objective functions.

Proposition 1 (Upper bounds for $\log\det$*, Thm 7.8.1 [12]):* For a positive definite matrix $\boldsymbol{M} \in \mathbb{R}^{n\times n}$ with diagonal elements $\boldsymbol{M}_{ii}$, it holds $\det(\boldsymbol{M}) \leq \prod_{i=1}^n \boldsymbol{M}_{ii}$ (Hadamard's inequality), or, equivalently, $\log\det(\boldsymbol{M}) \leq \sum_{i=1}^n \log \boldsymbol{M}_{ii}$.

Proposition 2 (Upper bounds for $\lambda_{\min}$*):* Given two symmetric and positive semidefinite matrices $\boldsymbol{M}, \Delta \in \mathbb{R}^{n\times n}$ the following inequality holds: $\lambda_{\min}(\boldsymbol{M} + \Delta) \leq \lambda_{\min}(\boldsymbol{M}) + \|\Delta\, \boldsymbol{\mu}_{\min}\|$, where $\boldsymbol{\mu}_{\min}$ is the eigenvector of $\boldsymbol{M}$ associated to the smallest eigenvalue $\lambda_{\min}(\boldsymbol{M})$.

B. Performance Guarantees for the Greedy Algorithm

This section shows that the greedy algorithm (Algorithm 1) admits provable sub-optimality bounds. The section tackles separately the two metrics presented in Section III-A, since the corresponding performance guarantees are fundamentally different. Our results are based on the recent literature on submodular maximization. We refer the non-expert reader to [5] for a more introductory discussion on submodularity.

1) Sub-optimality guarantees for $\log\det$*:* It is possible to show that $\log\det$ is submodular with respect to the set of measurements used for estimation. This result and the resulting performance guarantees are formalized as follows.

Proposition 3 (Submodularity of $\log\det$ *[29]):* The set function $f_{\det}(\mathsf{S})$ defined in (3) is monotone and submodular. Moreover, the greedy algorithm applied to (1) using $f_{\det}(\mathsf{S})$ as objective enjoys the following performance guarantees:

$$f_{\det}(\mathsf{S}) \geq (1 - 1/\mathrm{e}) f_{\det}(\mathsf{S}^{\star}) + \frac{f_{\det}(\emptyset)}{\mathrm{e}} \tag{22}$$

The result is proven in [29] and has been later rectified to account for the need of normalized functions in [13].

2) Sub-optimality guarantees for $\lambda_{\min}$*:* No result has been proven to bound the suboptimality of the greedy algorithm applied to the maximization of the smallest eigenvalue of the information matrix. Related work provides counterexamples, showing that this metric is not submodular in general, while the greedy algorithm works well in practice [13].

Currently, we have an initial technical result that uses a milder notion, known as *approximate submodularity*. The result states that under technical conditions on the eigenvectors of the information matrix, the solution $\mathsf{S}^{\#}$ returned by the greedy algorithm that maximizes $f_{\lambda}(\cdot)$ is such that:

$$f_{\lambda}(\mathsf{S}^{\#}) \geq (1 - \mathrm{e}^{-\gamma_{\mathsf{S}^{\#}}}) f_{\lambda}(\mathsf{S}^{\star}) \tag{23}$$

for some $\gamma_{\mathsf{S}^{\#}} > 0$. We omit this result for space reasons and we refer the interested reader to [5].

V. Experiments

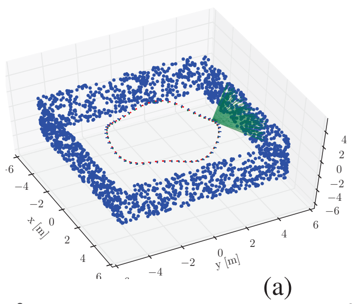
Section V-A evaluates our attention mechanism in realistic Monte Carlo simulations, showing that our feature selection techniques boost VIN performance. Section V-B evaluates our approach on real data collected by a micro aerial vehicle.

A. Monte Carlo Runs: Importance of Feature Selection

Testing setup. We adopt the benchmarking problem of [9], pictured in Fig. 1(a), as testing setup. We simulate a robot that follows a circular trajectory with a sinusoidal vertical motion. The total length of the trajectory is $120\mathrm{m}$. The on-board camera has focal length of 315 pixels and runs at a rate of 2.5Hz (simulating keyframes).

Implementation and evaluation metrics. We implemented our greedy algorithms and the construction of the matrices required in the functions (8) in c++, using eigen for the computation of the log-determinant and the smallest eigenvalue. Our feature selection approach is used as an add-on to a visual-inertial pipeline similar to the one described in [9]. Our VIN pipeline estimates the navigation state (robot pose, velocity, and IMU biases) using the structureless visual model and the pre-integrated IMU model described in [9]. The entire implementation is based on the GTSAM optimization library [8]. Our implementation differs from [9] in two important ways. First, we do not use SVO, but resort to a simpler front-end, described in the following section (no front-end is needed for the simulations in this section). Second, rather than feeding to the VIN estimator all available measurements, we use the feature selection algorithms described in this paper to select a small set of informative visual observations.

We use two main benchmarking metrics: the *absolute translation error*, which is the Euclidean distance between the estimated and the actual position, and the *relative translation error*, which computes the Euclidean norm of the difference between the estimated translation between time k and time $k+1$ and the actual translation. The relative translation error quantifies how quickly the estimate drifts from the ground truth. Since absolute positions are not observable in VIN, the relative error is a more suitable metric. When useful, we also report rotation errors.



	Time [s]
minEig naive	0.238
minEig lazy	0.195
logDet naive	0.070
logDet lazy	0.069

(b)

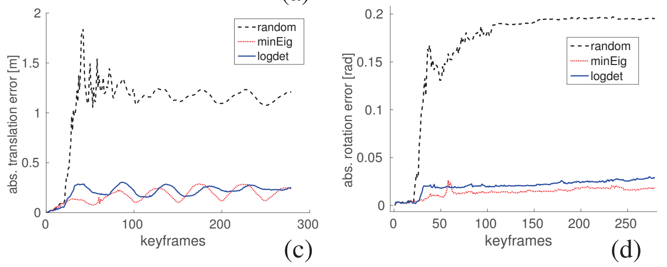


Fig. 1. Simulation results: (a) simulated environment, (b) CPU times for different implementations of the greedy algorithms, (c) absolute translation errors, (d) absolute rotation errors.

Results. We simulate 50 Monte Carlo runs; in each run we add random noise to the accelerometer, gyroscope, and camera measurements. The noise statistics are identical to the ones used in the real tests of Section V-B. In each run, the robot performs VIN and, at each frame, it selects $\kappa = 20$ visual features out of all the features in the field of view. We compare three feature selection strategies. The greedy selection resulting from Algorithm 1 with the eigenvalue objective f_{λ} (label: minEig), the greedy selection with the log-determinant objective $f_{\det}$ (label: logDet), and a random selection which randomly draws κ features (label: random).

Fig. 1(c)-(d) show the absolute translation and absolute rotation errors, averaged over 50 Monte Carlo runs. From the figure is clear that a clever selection of the features deeply impacts performance in VIN. Our techniques largely improve estimation errors, when compared against the random selection; both minEig and logDet result in similar performance. From the figure we note that the absolute errors have some oscillations: this is a consequence of the fact that the trajectory is circular; in general, this stresses the fact that absolute metrics may be poor indicators of performance in visual-inertial odometry. In this case, the relative error metrics confirm the results of Fig. 1(c)-(d): the relative errors are given in Table I; in parenthesis we report the error reduction percentage with respect to the random baseline.

Technique	Rel. Translation Error [m]	Rel. Rotation Error [rad]
random	0.0103	0.0049
minEig	0.0064 (-37%)	0.0025 (-48%)
logDet	0.0053 (-48%)	0.0018 (-63%)

TABLE I
Monte Carlo Runs : Relative errors.

		Rel. Tran. Error [m]	Abs. Tran. Error [m]	Time [s]
MH_01_easy	quality	6.25e−3	0.168	-
	minEig	5.67e−3 (−9%)	0.25	0.63
	logDet	5.46e−3 (−13%)	0.139	0.21
MH_02_easy	quality	8.88e−3	0.502	-
	minEig	6.11e−3 (−31%)	0.418	0.73
	logDet	6.69e−3 (−25%)	0.36	0.24
MH_03_medium	quality	1.59e−2	0.637	-
	minEig	1.42e−2 (−11%)	0.851	0.64
	logDet	1.58e−2 (−1%)	0.38	0.22
MH_04_difficult	quality	6.34e−2	1.3	-
	minEig	5.07e−2 (−20%)	1.43	1.2
	logDet	4.7e−2 (−26%)	0.992	0.36
MH_05_difficult	quality	6.72e−2	1.29	-
	minEig	6.01e−2 (−11%)	2.56	1.2
	logDet	2.83e−2 (−58%)	0.846	0.36
V1_01_easy	quality	1.09e−2	0.326	-
	minEig	1.09e−2 (−1%)	0.401	0.49
	logDet	1.07e−2 (−2%)	0.269	0.2
V1_02_medium	quality	1.19e−1	136	-
	minEig	2.67e−2 (−78%)	0.538	0.89
	logDet	2.39e−2 (−80%)	0.665	0.28

TABLE II
EuRoC datasets results on a 50-feature budget.

Fig. 1(b) reports the CPU time required for feature selection. The figure considers both objective functions (logDet and minEig) and compares the timing when using our lazy evaluation, against a naive implementation of the greedy algorithm which always tests the marginal gain of every feature. The naive greedy (without lazy evaluation) always results in κN objective evaluation. When using lazy evaluation, the number of objective evaluations depends on the tightness of the upper bounds used in Algorithm 1. Fig. 1(b) shows that the advantage of using the lazy evaluation is marginal for the log-determinant objective (the Hadamard's inequality of Proposition 1 usually gives a fairly loose bound). On the other hard, the advantage of using the lazy evaluation is significant for the minEig, resulting in a 20%-reduction of the computational time.

B. Real Tests: Agile Navigation on Micro Aerial Vehicles

Here we show that our feature selection approach enhances VIN performance in real-world navigation problems with micro-aerial vehicles (MAVs) using the *EuRoC* dataset [3].

Testing setup. The EuRoC datasets are collected with an AscTech Firefly hex-rotor helicopter equipped with a VI (stereo) visual-inertial sensor [3]. Our VIN front-end uses OpenCV's *goodFeaturesToTrack* for feature detection; as input to the detector we specify a minimum quality level for the features and a desired number of features to extract (N). From these N features our selector has to retain $\kappa = 50$ that will be used by the back-end. The feature selector uses a predictive horizon of 3s; in our tests, we compute the future poses by attaching ground truth motion increments to the current pose estimate. Our only assumption in doing so is that the control loop and the estimation quality are good enough to track a desired set of future poses; this is the case in VIN in which the short-term estimation drift is small.

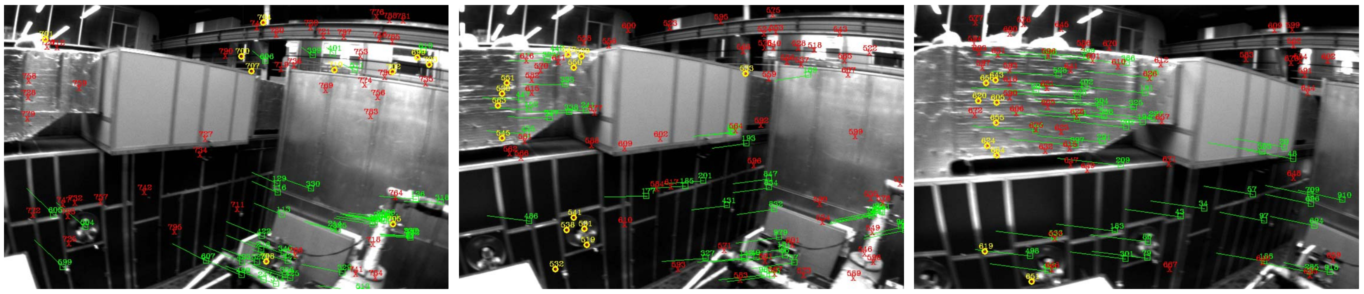
Techniques. In this section we compare the minEig and the logDet selectors proposed in this paper, against a selector that retains the κ feature with highest quality (i.e., higher score in *goodFeaturesToTrack*). The latter is commonly used in VIN and only accounts for the appearance of the visual feature. We denote this last approach with the label "quality", following OpenCV's terminology. In order to compute the tracking probabilities p_l in (8), we modified OpenCV's *goodFeaturesToTrack* in order to have access to the features' scores. Then, we normalized the scores in $[0, 1]$ and we set the probabilities to be equal to the normalized scores; as a consequence, more distinguishable features have higher tracking probabilities p_l. In most datasets we used 0.2 as minimum quality level for *goodFeaturesToTrack*. For the datasets labeled as "difficult", the presence of dark images and motion blur makes feature detection challenging; for these datasets we set the minimum quality to 0.01 in order to detect enough features.

Results. Table II shows the results on 7 datasets from the EuRoC benchmark. The tests are performed on datasets of different levels of complexity, with the difficult datasets being challenging due to the fast motion of the MAV. Here we show that we can obtain accurate motion estimation with as few as $\kappa = 50$ features; this budget is enforced for each frame: if we are tracking r features from the previous frame, then in the current frame we only retain $\kappa - r$ features.

Table II reports the relative translation errors for each technique (the absolute translation error is also given for completeness). For the minEig and logDet, we also report the error reduction with respect to the baseline quality. The table shows that the proposed feature selectors result in smaller drift across all datasets. In the easy datasets (e.g., V1_01_easy) the advantage is negligible, while in all the others the gain in accuracy is substantial. In the first 5 datasets the error reduction ranges from 10% to 30% (the gain is smaller for logDet in MH_03_medium). In the last dataset, the estimate resulting from the quality-based selector diverged after a sharp turn, while our techniques enabled accurate motion estimation.

To get a better intuition behind the large performance boost induced by the proposed techniques, we report few snapshots produced by our pipeline in Fig. 2. Each subfigure shows, for the current frame, the tracked features (green squares), the available features (red crosses), and the features selected (yellow circles) by (a) quality, (b) minEig, and (c) logDet. The frames are captured during a sharp left turn from the MH_03_medium dataset. The quality selector simply picks the most distinguishable features, resulting in many features selected on the right-hand side of the image; these features are of scarce utility as they will soon disappear from the field of view due to the motion of the MAV. On the other hand, logDet and minEig are aware of the motion of the platform and select features on the left-hand side of the image.

The rightmost column in Table II reports the average CPU time required by the feature selection in each dataset. For the datasets MH_04_difficult and MH_05_difficult the CPU times are larger: this is due to the fact that for those we use a lower quality level for feature detection (0.01), resulting in larger number of features fed to the selector. While we report the CPU time for completeness, Section VI discusses extensions that can make the selection time negligible.



(a) quality (b) minEig (c) logDet

Fig. 2. Snapshots of the feature selection performed by the different techniques during a sharp left turn. Features tracked from previous frames are shown as green squares (with their optical flow vectors), newly detected features are shown as red crosses, selected features are shown as yellow circles.

VI. Conclusion and Future Work

We proposed an attention mechanism applied to a visual-inertial navigation problem. This mechanism takes the form of a feature selector, which retains only the most informative visual features detected by the front-end and feeds them to the estimation back-end. We proposed two algorithms for feature selection. Both algorithms have four desirable properties: they are predictive in nature, they are task-driven, they can be implemented via simple greedy techniques, and they enjoy performance guarantees. We demonstrated that our feature selectors deeply impact VIN performance in both realistic simulations and real-world data collected by a MAV.

This work opens several avenues for future work. The first avenue consists in reducing the computational time of feature selection. Two main ideas can make the selection time negligible. The first idea is parallelization: the greedy algorithm is trivially parallelizable in that the marginal gain of each feature can be computed independently. The second is the use sparse matrix manipulation to compute the determinant and the smallest eigenvalue; our current implementation uses dense matrices in eigen. A second direction for future work consists in extending our approach to other tasks: for instance, *how many visual features the robot needs to sense in order to avoid crashing into nearby obstacles?* These are necessary steps towards a task-driven perception theory, that enables autonomy on resource-constrained robots with strict budget on sensing and computation.

Acknowledgments. The authors are grateful to Christian Forster for the useful discussions and for sharing the simulator used for the Monte Carlo analysis. This work was partially funded by the ARL MAST CTA Project 1436607.

References

[1] F. Bian, D. Kempe, and R. Govindan. Utility based sensor selection. In *5th Int. Conf. Information Processing Sensor Networks*, pages 11–18, 2006.

[2] A. Borji and L. Itti. State-of-the-art in visual attention modeling. *IEEE Trans. Pattern Anal. Machine Intell.*, 35(1), 2013.

[3] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M.W. Achtelik, and R. Siegwart. The EuRoC micro aerial vehicle datasets. *Intl. J. of Robotics Research*, 2016.

[4] L. Carlone, P. F. Alcantarilla, H.P. Chiu, K. Zsolt, and F. Dellaert. Mining structure fragments for smart bundle adjustment. In *British Machine Vision Conf. (BMVC)*, 2014.

[5] L. Carlone and S. Karaman. Attention and anticipation in fast visual inertial navigation, arxiv preprint: 1610.03344, 2016.

[6] M. Carrasco. Visual attention: The past 25 years. *Vision Research*, 51:1484–1525, 2011.

[7] M. Chli and A.J. Davison. Active matching for visual tracking. *Robotics and Autonomous Systems*, 57(12):1173–1187, December 2009.

[8] Frank Dellaert. Factor graphs and GTSAM: A hands-on introduction. Technical Report GT-RIM-CP&R-2012-002, Georgia Institute of Technology, September 2012.

[9] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza. IMU preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation. In *Robotics: Science and Systems (RSS)*, 2015.

[10] S. Frintrop and P. Jensfelt. Attentional landmarks and active gaze control for visual SLAM. *IEEE Trans. Robotics*, 24(5):1054–1065, 2008.

[11] J.A. Hesch, D.G. Kottas, S.L. Bowman, and S.I. Roumeliotis. Camera-imu-based localization: Observability analysis and consistency improvement. *Intl. J. of Robotics Research*, 33(1):182–201, 2014.

[12] R.A. Horn and C.R. Johnson. *Matrix Analysis*. Cambridge University Press, 1985.

[13] S.T. Jawaid and S.L. Smith. Submodularity and greedy algorithms in sensor scheduling for linear dynamical systems. *Automatica*, 61:282–288, 2015.

[14] S. Joshi and S. Boyd. Sensor selection via convex optimization. *IEEE Trans. Signal Processing*, 57:451–462, 2009.

[15] D. Kopitkov and V. Indelman. Computationally efficient active inference in high-dimensional state spaces. In *workshop on AI for Long-term Autonomy, in conjunction with the Intl. Conf. on Robotics and Automation (ICRA)*, 2016.

[16] A. Krause, A. Singh, and C. Guestrin. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, 9:235–284, 2008.

[17] R. Lerner, E. Rivlin, and I. Shimshoni. Landmark selection for task-oriented navigation. *IEEE Trans. Robotics*, 23(3):494–505, 2007.

[18] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale. Keyframe-based visual-inertial slam using nonlinear optimization. *Intl. J. of Robotics Research*, 2015.

[19] T. Lupton and S. Sukkarieh. Visual-inertial-aided navigation for high-dynamic motion in built environments without initial conditions. *IEEE Trans. Robotics*, 28(1):61–76, Feb 2012.

[20] B. Mu, A. Agha-mohammadi, L. Paull, M. Graham, J. How, and J. Leonard. Two-stage focused inference for resource-constrained collision-free navigation. In *Robotics: Science and Systems (RSS)*, 2015.

[21] R. Mur-Artal, J.M.M. Montiel, and J.D. Tardós. ORB-SLAM: A versatile and accurate monocular SLAM system. *IEEE Trans. Robotics*, 31(5):1147–1163, 2015.

[22] P. Newman and K. Ho. Slam-loop closing with visually salient features. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 635–642, 2005.

[23] N. Ouerhani, A. Bur, and H. Hügli. Visual attention-based robot self-localization. In *ECMR*, pages 8–13, 2005.

[24] V. Peretroukhin, L. Clement, M. Giamou, and J. Kelly. PROBE: Predictive robust estimation for visual-inertial navigation. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2015.

[25] S. Pillai, S. Ramalingam, and J. Leonard. High-performance and tunable stereo reconstruction. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2016.

[26] M.C. Potter, B. Wyble, C.E Hagmann, and E.S. McCourt. Detecting meaning in RSVP at 13 ms per picture. *Attention, Perception, & Psychophysics*, 76(2):270–279, 2014.

[27] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[28] P. Sala, R. Sim, A. Shokoufandeh, and S. Dickinson. Landmark selection for vision-based navigation. *IEEE Trans. Robotics*, 22(2):334–349, 2006.

[29] M. Shamaiah, S. Banerjee, and H. Vikalo. Greedy sensor selection: leveraging submodularity. In *IEEE Conf. on Decision and Control*, 2010.

[30] C. Siagian and L. Itti. Biologically-inspired robotics vision monte-carlo localization in the outdoor environment. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 1723–1730, 2007.

[31] G. Sibley, L. Matthies, and G. Sukhatme. Sliding window filter with application to planetary landing. *J. of Field Robotics*, 27(5):587–608, 2010.

[32] R. Sim and G. Dudek. Learning and evaluating visual features for pose estimation. In *Intl. Conf. on Computer Vision (ICCV)*, pages 1217–1222, 1999.

[33] H. Strasdat, C. Stachniss, and W. Burgard. Which landmark is useful? learning selection policies for navigation in unknown environments. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 1410–1415, 2009.

[34] G. Zhang and P.A. Vela. Good features to track for visual slam. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015.