ViCoMoR 2012

2nd Workshop on Visual Control of Mobile Robots (ViCoMoR)

Half Day Workshop

October 11th, 2012, Vilamoura, Algarve, Portugal, in conjunction with the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2012)

http://vicomor.unizar.es

Organizers

Youcef Mezouar
Institut Pascal- IFMA, France

Gonzalo López-Nicolás
I3A - Universidad de Zaragoza, Spain

Contents

Web: http://vicomor.unizar.es
2nd Workshop on Visual Control of Mobile Robots (ViCoMoR)
October 11th, 2012, Vilamoura, Algarve, Portugal, in conjunction with the
IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2012)

The organization of this workshop was supported by Ministerio de Ciencia e Innovación / European Union (projects DPI2009-08126 and DPI2009-14664-C02-01), DGA-FSE (T04), ANR ARMEN project and grant I09200 from Gyeonggi Technology Development Program funded by Gyeonggi Province.

Aims and Scope

The purpose of this workshop is to discuss topics related to the challenging problems of visual control of mobile robots. Visual control refers to the capability of a robot to visually perceive the environment and use this information for autonomous navigation. This task involves solving multidisciplinary problems related with vision and robotics, for example: motion constraints, vision systems, visual perception, safety, real-time constraints, robustness, stability issues, obstacle avoidance… The problem of the vision-based autonomous navigation is also compounded of the different constraints imposed by the particular features of the platform involved (ground platforms, aerial vehicles, underwater robots, humanoids…)

Over the last years, increasing efforts have been made to integrate robotic control and vision. Although there is an important number of works in the area of visual control for manipulation, which is a mature field of research, the use of mobile robots add new challenges in a still open research area. The interest in this subject lies in the many potential robotic applications in industrial as well as in domestic settings that involve visual control of mobile robots (automation industry, material transportation, assistance to disabled people, surveillance, rescue, etc).

This workshop is aimed to promote exchange and sharing of experiences among researchers in the field of visual control of mobile robots. Previously, the first edition of ViCoMoR was held in San Francisco during IROS'11. This new edition of the workshop will consist of invited talks and selected papers for oral presentation.

Topics

Topics of interest include:

- Autonomous navigation and visual servoing techniques for mobile robots.
- Visual perception for visual control, visual sensors and integration of image information in the control loop.
- Visual control with constraints: nonholonomic constraints, motion in formation, distributed visual control, obstacle avoidance, etc.
- New trends in visual control, innovative solutions or proposals in the framework of computer vision and control theory.

Program committee

Helder Araujo	(ISR, University of Coimbra, Portugal)
Antonis Argyros	(FORTH, Heraklion, Greece)
Hector M. Becerra	(CIMAT, Guanajuato, Mexico)
Enric Cervera	(Universitat Jaume-I, Spain)
François Chaumette	(INRIA Rennes - IRISA, France)
Peter Corke	(Queensland Univ. of Technology, Australia)
Francisco Escolano	(Universidad de Alicante, Spain)
Nicholas R. Gans	(University of Texas at Dallas, USA)
Andrea Gasparri	(Università degli Studi Roma Tre, Roma, Italy)
Jose J. Guerrero	(Universidad de Zaragoza, Spain)
Koichi Hashimoto	(Tohoku University, Sendai, Japan)
Seth Hutchinson	(University of Illinois at Urbana-Champaign, USA)
Patric Jensfelt	(CAS, KTH, Sweden)
Nicolas Mansard	(LAAS/CNRS, France)
Roberto Naldi	(Universita' di Bologna, Italy)
Patrick Rives	(INRIA Sophia Antipolis, France)
Carlos Sagues	(Universidad de Zaragoza, Spain)
Omar Tahri	(ISR, University of Coimbra, Portugal)
Dimitris P. Tsakiris	(FORTH, Heraklion, Greece)
Andrew Vardy	(Memorial Univ. of Newfoundland, Canada)
Xenophon Zabulis	(FORTH, Heraklion, Greece)

Organizers

Youcef Mezouar
Clermont Université, IFMA, Institut Pascal,
BP 10448, F-63000 Clermont-Ferrand, France
CNRS, UMR 6602, IP, F-63171 Aubière, France
Email: youcef.mezouar@ifma.fr
Web: http://wwwlasmea.univ-bpclermont.fr/Personnel/Youcef.Mezouar

Gonzalo López-Nicolás
Instituto de Investigación en Ingeniería de Aragón - Universidad de Zaragoza
María de Luna 1, E-50018 Zaragoza. Spain
Email: gonlopez@unizar.es
Web: http://webdiis.unizar.es/~glopez

Instituto Universitario de Investigación
en Ingeniería de Aragón
Universidad Zaragoza

Invited speakers

Patrick Rives

INRIA Sophia Antipolis Mediterranee
2004 Route des Lucioles BP 93, Sophia Antipolis, France.

Web: http://www-sop.inria.fr/icare/WEB/Personnel/modele-rives.html

Title: Dense RGB-D mapping of large scale environments for real-time localisation and autonomous navigation

Abstract: We present a method and apparatus for building 3D dense visual maps of large scale environments for real-time localisation and autonomous navigation. The method relies on a spherical ego-centric representation of the environment which is able to reproduce photo-realistic omnidirectional views of captured environments. It is shown that this representation can be used to accurately localise a vehicle navigating within a graph of locally accurate spherical views, using only a monocular camera. Autonomous navigation results are shown in challenging urban environments, containing pedestrians and other vehicles.

Cédric Pradalier

Autonomous Systems Lab ETH Zürich
Inst. f. Robotik u. Intell. Syst. CLA E 14.3, Tannenstrasse 3, 8092 Zuerich, Switzerland

Web: http://www.asl.ethz.ch/people/cedricp

Title: Visual homing with omnidirectional vision

Abstract: Visual homing is the process by which a mobile robot moves to a home position using only information extracted from visual data. This idea is often inspired by the mechanisms that certain animal species, such as insects, utilize to return to their known home location. This talk will present an overview of the results obtained recently in visual homing in the Autonomous Systems Lab at ETH Zurich.

Program ViCoMoR 2012 (October 11th, 2012, Vilamoura, Algarve, Portugal)

14:00 – 14:10	Presentation of the workshop
14:10 – 14:40	**Invited speaker: Cédric Pradalier**
14:40 – 15:00	**From the general navigation problem to its image based solutions** Durand Petiteville Adrien, Cadenat Viviane
15:00 – 15:20	**Vistas and wall-floor intersection features: enabling autonomous flight in man-made environments** Kyel Ok, Duy-Nguyen Ta, Frank Dellaert
15:20 – 15:40	**Distributed policies for neighbor selection in multi-robot visual consensus** Eduardo Montijano, Johan Thunberg, Xiaoming Hu, Carlos Sagues
15:40 – 16:00	**Target tracking and obstacle avoidance for a VTOL UAV using optical flow** Aurélie Treil, Philippe Mouyon, Tarek Hamel, Alain Piquereau, Yoko Watanabe
16:00 – 16:30	Coffee break
16:30 – 17:00	**Invited speaker: Patrick Rives**
17:00 – 17:20	**Homography based visual odometry with known vertical direction and weak Manhattan world assumption** Olivier Saurer, Friedrich Fraundorfer, Marc Pollefeys
17:20 – 17:40	**Anisotropic vision-based coverage control for mobile robots** Carlos Franco, Gonzalo Lopez-Nicolas, Dusan Stipanovic, Carlos Sagues
17:40 – 18:00	**FSM-based visual navigation for autonomous vehicles** Daniel Oliva Sales, and Fernando Santos Osório
18:00 – 18:20	**Accurate figure flying with a quadrocopter using onboard visual and inertial sensing** Jakob Engel, Jurgen Sturm, Daniel Cremers

IROS Workshop on Visual Control of Mobile Robots (ViCoMoR 2012)
October 11th, 2012, Vilamoura, Algarve, Portugal

From the general navigation problem to its image based solutions

Adrien Durand Petiteville[1] and Viviane Cadenat[1]

Abstract— **This article presents a brief study of mobile robots navigation. In a first part, we provide an overview of this problem, analyzing the different involved processes and showing several architectures allowing to organize them. In a second step, we consider the vision based navigation problem. From the previous analysis, we highlight the interest of using topological maps in this context and propose an overview of existing works in this area. Finally, we present our own solution to the problem, showing its relevance and its efficiency.**

I. INTRODUCTION

In this paper we consider the well known autonomous navigation problem. It consists for the robot in reaching a goal through a given environment while dealing with unexpected events [1]. Thus, the navigation generally involves six processes: perception, modelling, planning, localization, action and decision. A wide range of techniques are available in the literature for each of them. As these processes cooperate within an architecture to perform the navigation, they cannot be designed independently and it is necessary to have an overview of the problem to select suitably the different methods. This article aims at (i) providing such an overview, (ii) presenting the visual solutions and (iii) positioning our own work in this general context.

II. THE NAVIGATION FRAMEWORK

In this section, we present the different processes and the associated methods before highlighting several architectures.

A. The navigation processes

1) Perception: In order to acquire the data required by the navigation, a robot can be equipped with both proprioceptive and exteroceptive sensors. The first ones (odometers, gyroscopes, ...) provide data relative to the robot internal state whereas the second ones (camera, laser telemeters, bumpers, ...) give information about the environment. The sensory data may be used by four processes: environment modelling, localization, decision and robot control.

2) Modelling: A navigation process generally requires an environment model. This model is initially built using *a priori* data. It is not always complete, and can evolve with time. In this case, the model is updated thanks to the data acquired during the navigation. There exists two kinds of models, namely the metric and/or topologic maps [2].

The metric map is a continuous or discrete representation of the free and occupied spaces. A global frame is defined and the robot and obstacles poses are known with more or less precision in this frame. The data must then be expressed in this frame to update the model [1].

The topologic map is a discrete representation of the environment based on graphs [1]. Each node represents a continuous area of the scene defined by a characteristic property. The areas are naturally connected and can be limited to a unique scene point. The characteristic property, chosen by the user, may be the feature visibility or belonging to a same room. Moreover, if a couple of nodes verifies an adjacency condition, then they are connected. The adjacency condition is chosen by the user and may correspond for example to the existence of a path allowing to connect two sets, each of them represented by a node. A topologic map is less sensitive to the scene evolutions: it has to be updated only if there are modifications concerning the area represented by the nodes or the adjacency between two nodes.

Metric and topologic maps can be enhanced by adding to nodes sensory data, actions or control inputs. These informations may be required to localize or control the robot. There also exists hybrid representations of the environment based on both metric and topologic maps [3], [4], [5].

3) Localization: For navigation, two kinds of localizations are identified: the metric one and the topologic one [2]. The metric localization consists in calculating the robot pose with respect to a global or a local frame. To do so, a first solution is to use only proprioceptive data. However this solution can lead to significant errors [6] [7] for three reasons. The first one comes from the pose computation process which consists in successively integrating the acquired data, which induces a drift. The second one is due to the model which is used to determine the pose: an error which occurs during the modelling step is automatically transferred to the pose computation. Finally, the last one is related to phenomenons such as sliding which are not taken into account. It is then necessary to consider additional exteroceptive information to be able to localize the robot properly. Visual odometry [8] [9] [10] is an example of such a fusion.

The topological localization [3] [11] [12] [13] consists in relating the data provided by the sensors with the ones associated to the graph nodes which model the environment. The goal is to determine the situation in a graph and not with respect to a frame [2]. Topological localization is not very sensitive to measurement errors unlike its metric alter-ego. The precision depends on the accuracy with which the environment has been described.

4) Planning: The planning step consists in computing, using the environment model, an itinerary allowing the robot to reach its final pose. The itinerary may be a path, a trajectory, a set of poses to reach successively, ... There

[1]CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France, Univ de Toulouse, UPS, LAAS ; F-31400 Toulouse, France [adurandp,cadenat]at laas.fr

exists a large variety of planning methods depending on the environment modeling. An overview is presented hereafter.

A first approach consists in computing the path or the trajectory using a metric map. To do so, the geometric space is transposed into the configuration space. The configuration corresponds to a parametrization of the static robot state. Thus a robot with a complex geometry in the workspace is represented by a point in the configuration space [2]. Then planning consists in finding a path or a trajectory in the configuration space allowing to reach the final configuration from the initial one [1]. With a continuous representation of the environment, a path or a trajectory can be obtained using visibility graphs or Voronoi diagrams [14]. With a discrete scene model, planning is performed thanks to methods from graph theory such as A^* or Dijkstra algorithms[15] [16]. For the two kinds of maps, planning may be time consuming. A solution consists in using probabilistic planning: *probabilistic roadmap* [17] [18] or *rapidly exploring random tree* [19].

When the environment model is incomplete at the beginning of the navigation, unexpected obstacles may lie on the robot itinerary. A first solution to overcome this problem consists in adding the obstacles to the model and then to plan a new path or a new trajectory [1]. In [20], [21], authors propose to consider the trajectory as an elastic band which can be deformed if necessary. A global re-planning step can then be required for a major environment modification.

When using a topological map without any metric data, the planned itinerary is generally composed of a set of poses to reach successively. These poses can be expressed in a frame associated to the scene or to a sensor. The itinerary is computed using general methods from the graph theory. Depending on the precision degree used to describe the environment, it is possible that the planned itinerary does not take into account all the obstacles. This issue has to be considered during the motion realization.

5) Action: To perform the tasks required by the navigation, two kinds of controllers can be designed: state feedback or output feedback [22]. In the first case, the task is generally defined by an error between the current robot state[1] and the desired one. To make this error vanish, a state feedback is designed. The control law implementation requires to know the current state value and therefore a metric localization is needed. In the second case, it consists in making the error between the current measure and the desired one vanish. This measure depends on the relative pose with respect to an element of the environment, called feature or landmark. When using vision, the measures correspond to image data (points, lines, moments; etc. [25]). For proximity sensors, they are defined by distances provided by ultrasound [26] and laser [5] [27] telemeters. The measures are directly used in the control law computation, which means that no metric localization is required. However the landmark must be perceptible during the entire navigation to compute the control inputs.

[1]The state may correspond to the robot pose with respect to the global frame or to a given landmark [23] [24].

6) Decision: To perform a navigation, it may be necessary to take decisions at different process states. These decisions may concern high level, e.g. a re-planning step [21], or low level, e.g. the applied control law [26]. They are usually taken by supervision algorithms based on exteroceptive data.

B. The navigation architectures

The different processes required by a navigation have now been identified. Here, we present examples of navigation architectures based on the previously presented processes. We propose to organize our presentation around the controllers.

1) "State feedback" based architecture: First, we consider a robot controlled using a state feedback controller in a free space. The initial and final configurations are defined in a world frame. To compute the control inputs, the state value has to be known at any time. The robot capacity to geometrically localize itself is a necessary condition to successfully perform the navigation. Let us note that, the distance that the robot can cover is only limited by the localization precision. Indeed, a too large error on the state value will result in inconsistent control inputs.

We now consider a cluttered environment. In this case there are two solutions, either reactive or planning based. The reactive one consists in controlling the robot using two controllers : a first one making the error between the current and the desired poses vanish, allowing to reach the goal, and a second one performing the obstacle avoidance using exteroceptive data. It is then necessary to develop a supervision module selecting the adequate controller. This solution, which guarantees the non-collision with obstacles, does not allow to ensure the navigation success. Indeed, the obstacle avoidance is locally performed and does not take into account the goal. The second solution consists in following a previously planned collision free path. To this aim, the environment has to be modeled using a map. If the model represents the whole scene, then the navigation simply consists in following the planned itinerary using a state feedback controller. A supervision module is no more required. If the environment is not completely modeled, it may be necessary to update it when an obstacle appears on the robot path. After the update, a re-planning step is performed. In this case, a supervision module which decides to update and re-plan is mandatory. Finally, it should be noticed that the metric localization is required and limits the navigation range for each solution.

As a conclusion, when the robot is controlled using state feedback controllers, the metric localization is a decisive element, as the navigation success depends on the localization quality. Moreover, in a cluttered environment, a model is quickly mandatory to converge towards the desired pose or to avoid obstacles. The metric localization and modeling processes are very sensitive to measurement errors. It is then necessary to pay attention to the methods performances when the navigation is based on state feedback controllers.

2) "Output feedback" based architecture: We now consider a robot controlled using an output feedback controller in a free space. The initial pose is unknown whereas the desired

one with respect to a landmark is defined by measures. The robot can converge toward the desired pose if the landmark can be perceived at any instant. It is now the sensor range which limits the navigation range. In this case no metric localization is required.

When the environment is cluttered, a first solution consists in using a sole output feedback controller to reach the desired pose while avoiding obstacles. A second idea is to control the robot thanks to two output feedback controllers : the first one allows to reach the desired pose and the second one guarantees non collision. A supervisor selecting the adequate controller is then required. For both solutions, local minima problems may occur. Moreover, the navigation range is still limited by the sensors range. Global informations must then be used to perform a long range navigation. These global information can be added using a metric map or a topological map. In the first case, it is possible to plan a path taking into account the features availability at each pose. The planned itinerary is then composed by several landmarks successively used to compute the control inputs. Moreover, for a static environment, joint limits, visibility and obstacles can also be considered in the planning step. Nevertheless, this approach requires environment, robot and sensors reliable models. In the second case, a topological map is used to provide the necessary global information. Here, the additional data associated to the graph nodes correspond usually to the desired features or landmarks. As previously, the planned itinerary is made of measures or landmarks set to reach. This approach is based on a partial environment representation. The model is then less sensitive to the environment modifications, but does not allow to take into account several constraints such as obstacles or joints limits during the planning step. A topologic localization is needed.

III. THE VISUAL NAVIGATION

Now we focus on the vision based navigation problem. The camera is then used as the main sensor, which still allows to select any of the previous presented approaches as shown in [28] and [29]. In these works, the authors propose an overview of visual navigation splitting the methods into two main categories: the metric map based ones and the topological map based ones. Following our previous analysis, we have selected the topological approach. Indeed, in this case, the metric localization is no more required, limiting the inaccuracy due to the use of noisy data in the state computation process. Furthermore, a topological map provides sufficient data to perform a navigation task, without significantly increasing the problem complexity. Finally, this representation is less sensitive to scene modifications. We present hereafter methods based on such an approach.

A. Related works

In [30], the scene is modelled by a graph whose nodes correspond to the corridors. The robot navigates into the corridors using an image based visual servoing relying on the vanishing point as visual feature. This method is then limited to an environment composed of corridors. Other approaches propose to model the environment during a pre-navigation step. During this phase, images obtained for several close robot configurations are memorized. A topological map, also called visual memory, is built by organizing the images [4]. The planned itinerary is called visual road [31]. This approach is performed using omnidirectional [32] [33] [34] [35] [36] or pinhole camera [37] [38] [39] [40] [36]. However, none of these approaches take into account the two major problems of visual navigation : occlusions, *i.e.* the landmarks loss, and collisions with obstacles. A set of works [41] [42] [43] has produced a visual navigation allowing to avoid unexpected obstacles while tolerating partial occlusions. The topological map is also built during a pre-navigation step. Time variant visual features are used by the visual servoing while the obstacle avoidance is performed thanks to a potential fields based control law. Thus, the learnt path can be replayed using a topological map while avoiding collisions.

B. Our approach

We propose a similar approach to perform the navigation while dealing with collisions and total occlusions [44]. Following the above analysis, we have chosen to use a camera, a topological map and several output feedback controllers organized in a supervision algorithm. We present hereafter our approach detailing our choices for each process. More details can be found in [44].

1) Perception: Our robot is equipped with a camera and a laser able to detect respectively the landmarks of interest and the obstacles. Our approach will rely on these two data.

2) Modelling: We now focus on the topological map, which consists of a directed graph. Each node corresponds to a landmark present in the scene. If there are n_l landmarks, then the graph is composed of n_l nodes. A point, corresponding to the desired robot pose S_i with respect to the landmark T_i, is associated to each node N_i, with $i \in [1,...,n_l]$. An arc $A(N_j,N_k)$ is created if the landmark T_k, associated to the node N_k, can be seen from the pose S_j, associated to the node N_j, with $j \in [1,...,n_l]$, $k \in [1,...,n_l]$ and $j \neq k$. Moreover, sensory data D_i extracted from an image of the landmark T_i taken at the pose S_i is associated to the node N_i.

3) Localization: During the navigation, and especially at the beginning, the robot has to localize itself into the graph. The localization process identifies the landmarks that are in the field of view of the camera. Localization is performed using the sensory data associated to each node. It consists in making a test of similarity between two images. To this aim, the descriptors of the current image and those from the data base are matched. The image from the data base which has the best similarity with the current image is selected. Then we consider that the robot situation in the graph corresponds to the node containing the selected image.

4) Planning: The initial and final poses are obtained from the localization process and from the user. They are now considered as known. The path T_P made of a sequence of n_P landmarks $[T_{P1},...,T_{Pn_P}]$ to reach, is planned using the Dijkstra algorithm [16] which provides the shortest path.

5) Action: To perform a long range navigation, we use three output feedback controllers [44]. The first one allows to perform a short range navigation with respect to a landmark, which will be referred to as "sub-navigation". This controller is defined by a classical image based visual servoing [45], which makes the error between the current and desired images vanish. The second one performs the obstacle avoidance by stabilizing the robot on a path defined thanks to telemetric data [46]. The last one is intended to avoid unsuitable motions when switching from one landmark to the other [44]. The transition between each controller is performed by a dynamic sequencing allowing to guarantee the control law continuity [47]. Thus, using these two controllers, the robot can successively reach the landmarks composing the path while avoiding the obstacles.

To manage the occlusions problem, we have used the algorithm [48]. It allows to predict the visual features next position from the previous visual data and the visual features depth. The latter is estimated thanks to a predictor/corrector using a number n_{pc} of images allowing to provide an accurate estimation even in the presence of noisy data [49]. Using these tools we can deal with total occlusions.

6) Decision: The decision process has to activate or deactivate the available tools in order to guarantee the long range navigation success. We propose to use a supervision algorithm to perform the decision process. The algorithm, summarized in figure 1, is built using the following strategy.

First of all, the robot localizes itself to determine the initial node in the graph. Then, knowing the desired pose, a path T_P composed by a set of landmarks to reach is computed. Then, the initialization phase is executed. It consists in making small rotations to estimate the visual features depth of landmark T_{P1}. Thus occlusions can be managed during the sub-navigation with respect to T_{P1}. When the initialization phase is over, the sub-navigation to T_{P1} is launched. If the robot is too close to an obstacle, the obstacle avoidance controller is used. During the sub-navigation, the robot regularly looks for the next landmark T_{P2}. If this latter is not found, the robot continues the current sub-navigation and restarts the depth estimation process. When it converges, T_{P2} is one more time looked for. This loop is repeated until the next landmark is found or the current sub-navigation is over. In the latter case, the robot turns on itself to identify the next target. If it is not found, then the graph is updated and a new path is planned. If there is no path to reach the desired landmark, the navigation fails. We consider now that the landmark T_{P2} has been found. The sub-navigation, obstacles avoidance and looking for the next landmark processes are repeated using the same conditions as previously until the robot reaches the desired pose or the navigation fails.

IV. SIMULATIONS

We have simulated a long range navigation using Matlab^{TM}software. The considered cart-like robot is equipped with a camera mounted on a pan-platform and a laser telemeter. In the scene shown in figure 2(a), there are $n_l = 9$ artificial landmarks made of a different number

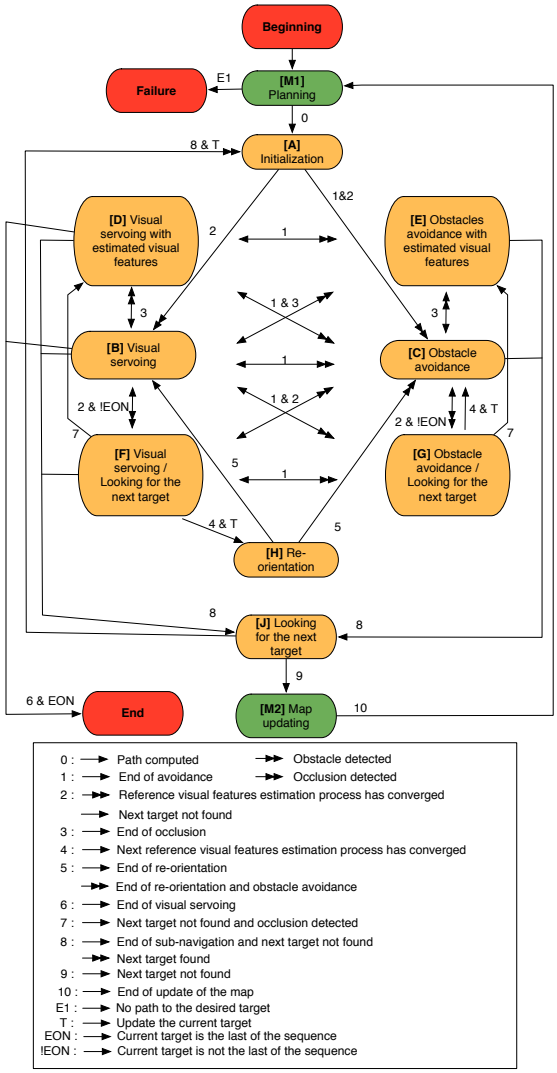
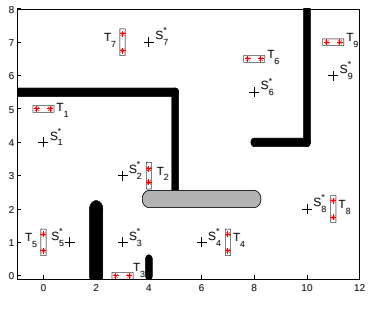


Fig. 1. Supervision algorithm for a long range navigation

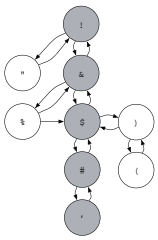
of points. The occluding obstacles are represented in black whereas the non-occluding one is in gray. To model the environment, the robot is successively placed at the desired poses S_i^*, with $i \in [1,...,n_l]$. Thus, we obtain the topological map presented in figure 2(b). It should be noticed that the topological map is not complete, as the chosen poses S_i^* do not allow to connect all the nodes. For example, the selected S_8^* does not allow to relate N_8 and N_6, although other choices would have permitted it. However, it is not a problem as there exists a path allowing to reach the desired pose.

From its initial pose near S_5^*, the robot must reach S_7^* with respect to the landmark T_7. After a localization and using the topological map, the shortest path $T_P = [T_1, T_2, T_4, T_6, T_7]$ is computed (see Fig. 2(b)). Then, the mission starts and the supervision algorithm selects the current task to perform until T_7 is reached. Figure 3 shows the corresponding task sequencing and robot trajectory. As we can see, the mission is successfully realized despite the unexpected obstacles.

We propose a second simulation to illustrate the re-

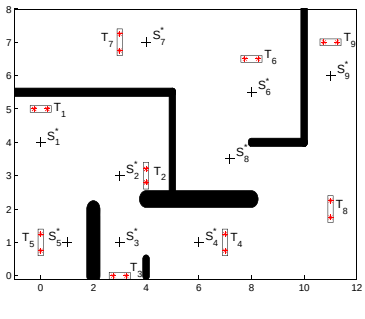


(a) Environment of navigation #1

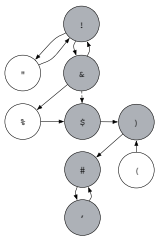


(b) Environment #1 topological map

Fig. 2. Mapping #1



(a) Environment of navigation #2



(b) Environment #2 topological map

Fig. 4. Mapping #2

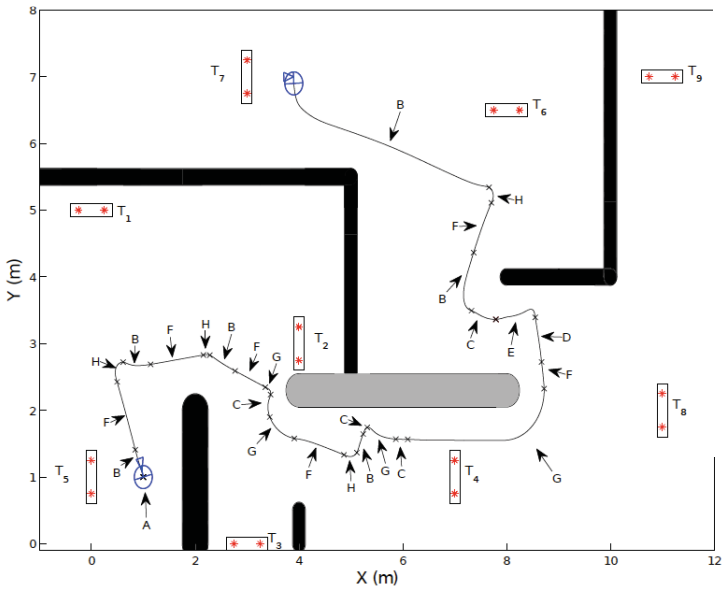


Fig. 3. Robot long range navigation #1 (letters correspond to the current executed task (see figure 1))

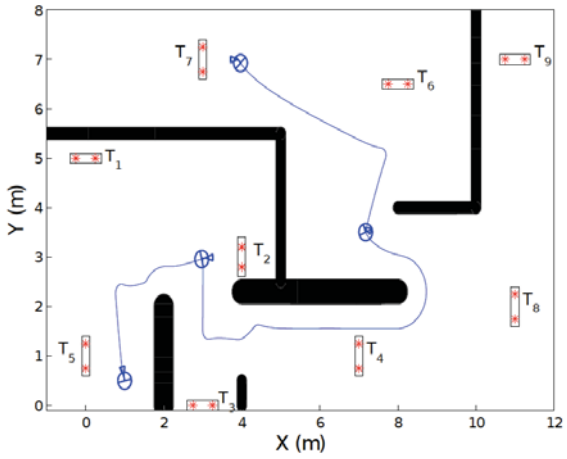


Fig. 5. Robot long range navigation #2

planning phase and the necessity to provide the most complete topological map. We consider the same environment as previously, except that all obstacles are now occluding. If we use the same poses S_i^*, reaching S_7^* from S_5^* is impossible because nodes N_4 and N_6 cannot be connected anymore. To overcome this problem, a new pose S_8^* allowing to relate N_8 and N_6 is defined (see Fig 4(a)). The proposed map is then more complete than the previous one, showing the importance of the choice of each S_i^*. The corresponding environment map is shown in figure 4(b). Note that we have willingly introduced an error in the map by connecting nodes N_2 and N_4 whereas this relation does not exist anymore.

To reach S_7^* the robot now plans the following path $T_P = [T_1, T_2, T_4, T_8, T_6, T_7]$. Then the navigation starts and the robot reaches S_2^* but cannot find T_4. At this time, a localization process is performed, showing that only T_2 and T_3 can be perceived from the robot current position. The map is then updated by suppressing the link between N_2 and N_4. A new path $T_P = [T_3, T_4, T_8, T_6, T_7]$ is computed, and the navigation is launched again. Using the adequate controllers the robot then performs the task and reaches S_7^*.

V. CONCLUSION

This paper was focused on the navigation problem. We have first highlighted the different processes involved in the navigation and shown their organization within several possible architectures. Then, we have considered the vision based solutions, showing the interest of using a topological map. We have finally positioned our own works in this general framework, demonstrating its efficiency to perform a visual navigation despite collisions and occlusions. One of the next challenges will be to take into account the presence of mobile obstacles (vehicles, human beings, ...) to improve the robot autonomy in a real environment.

REFERENCES

[1] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, and S. Thrun, *Principles of Robot Motion*. MIT Press, Boston, 2005.

[2] R. Siegwart and I. Nourbakhsh, *Introduction to autonomous mobile robots*, ser. A bradford book, Intelligent robotics and autonomous agents series. The MIT Press, 2004.

[3] S. Segvic, A. Remazeilles, A. Diosi, and F. Chaumette, "A mapping and localization framework for scalable appearance-based navigation," *Computer Vision and Image Understanding*, vol. 113, no. 2, pp. 172–187, February 2009.

[4] E. Royer, M. Lhuillier, M. Dhome, and J.-M. Lavest, "Monocular vision for mobile robot localization and autonomous navigation," *International Journal of Computer Vision*, vol. 74, no. 3, pp. 237–260, 2007.

[5] A. Victorino and P. Rives, "An hybrid representation well-adapted to the exploration of large scale indoors environments," in *IEEE International Conference on Robotics and Automation*, New Orleans, USA, 2004, pp. 2930–2935.

[6] D. Cobzas and H. Zhang, "Mobile robot localization using planar patches and a stereo panoramic model," in *Vision Interface*, Ottawa, Canada, June 2001, pp. 04–99.

[7] J. Wolf, W. Burgard, and H. Burkhardt, "Robust vision-based localization for mobile robots using an image retrieval system based on invariant features," in *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, vol. 1, 2002, pp. 359 – 365 vol.1.

[8] D. Nister, O. Naroditsky, and J. Bergen, "Visual odometry," in *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, vol. 1, june-2 july 2004, pp. I–652 – I–659 Vol.1.

[9] A. Comport, E. Malis, and P. Rives, "Real-time quadrifocal visual odometry," *International Journal of Robotics Research, Special issue on Robot Vision*, vol. 29, 2010.

[10] Y. Cheng, M. Maimone, and L. Matthies, "Visual odometry on the mars exploration rovers - a tool to ensure accurate driving and science imaging," *Robotics Automation Magazine, IEEE*, vol. 13, no. 2, pp. 54 –62, june 2006.

[11] R. Sim and G. Dudek, "Learning visual landmarks for pose estimation," in *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, vol. 3, 1999, pp. 1972 –1978 vol.3.

[12] L. Paletta, S. Frintrop, and J. Hertzberg, "Robust localization using context in omnidirectional imaging," in *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, vol. 2, 2001, pp. 2072 – 2077 vol.2.

[13] B. Krösea, N. Vlassisa, R. Bunschotena, and Y. Motomura, "A probabilistic model for appearance-based robot localization," *Image and Vision Computing*, vol. 19, pp. 381–391, 2001.

[14] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu, *Spatial Tessellations - Concepts and Applications of Voronoi Diagrams*. John Wiley, 2000.

[15] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination od minimum cost paths," *Systems Science and Cybernetics, IEEE Transactions on*, vol. 4, no. 2, pp. 100–107, 1968.

[16] E. W. Dijkstra, "A short introduction to the art of programming," Aug 1971.

[17] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *Robotics and Automation, IEEE Transactions on*, vol. 12, no. 4, pp. 566 –580, aug 1996.

[18] R. Geraerts and M. H. Overmars, "A comparative study of probabilistic roadmap planners," in *Proc. Workshop on the Algorithmic Foundations of Robotics (WAFR'02)*, Nice, France, December 2002, pp. 43–57.

[19] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," *In*, vol. TR 98-11, no. 98-11, pp. 98–11, 1998.

[20] Khatib, Jaouni, Chatila, and Laumond, "Dynamic path modification for car-like nonholonomoic mobile robots," in *IEEE Int. Conf. on Robotics and Automation*, April 1997, pp. 490–496.

[21] F. Lamiraux, D. Bonnafous, and O. Lefebvre, "Reactive path deformation for nonholonomic mobile robots," *Robotics, IEEE Transactions on*, vol. 20, no. 6, pp. 967 – 977, dec. 2004.

[22] W. S. Levine, *The Control Handbook*. CRC Press Handbook, 1996.

[23] S. Hutchinson, G. Hager, and P. Corke, "A tutorial on visual servo control," *IEEE Trans. on Rob. and Automation*, vol. 12, no. 5, pp. 651–670, 1996.

[24] D. Bellot and P. Danes, "Towards an lmi approach to multiobjective visual servoing," in *European Control Conference 2001*, Porto (Portugal), September 2001.

[25] F. Chaumette, "Image moments: a general and useful set of features for visual servoing," *Robotics, IEEE Transactions on*, vol. 20, no. 4, pp. 713 – 723, aug. 2004.

[26] D. Folio and V. Cadenat, "A controller to avoid both occlusions and obstacles during a vision-based navigation task in a cluttered environment," in *European Control Conference (ECC05)*, Seville, Espagne, December 2005, pp. 3898–3903.

[27] S. Thrun, W. Burgard, and D. Fox, "A real time algorithm for mobile robot mapping with applications to multi-robot and 3d mapping," in *IEEE International Conference on Robotics and Automation*, San Francisco, CA, USA, April 2000.

[28] G. Desouza and A. Kak, "Vision for mobile robot navigation: a survey," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 2, pp. 237 –267, feb 2002.

[29] F. Bonin-Font, F. Ortiz, and G. Oliver, "Visual navigation for mobile robots : a survey," *Journal of intelligent and robotic systems*, vol. 53, no. 3, p. 263, 2008.

[30] R. Vassalo, H. Schneebeli, and J. Santos-Victor, "Visual servoing and appearance for navigation," *Robotics and autonomous systems*, 2000.

[31] Y. Matsumoto, M. Inaba, and H. Inoue, "Visual navigation using viewsequenced route representation," in *IEEE Int. Conf. on Robotics and Automation*, Minneapolis, USA, 1996, pp. 83–88 –2692.

[32] J. Gaspar, N. Winters, and J. Santos-Victor, "Vision-based navigation and environmental representations with an omni-directional camera," *IEEE transactions on robotics and automation*, vol. 6, no. 6, pp. 890–898, 2000.

[33] Y. Yagi, K. Imai, K. Tsuji, and M. Yachida, "Iconic memory-based omnidirectional route panorama navigation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, pp. 78–87, 2005.

[34] T. Goedemé, M. Nuttin, T. Tuytelaars, and L. V. Gool, "Omnidirectional vision based topological navigation," *International Journal of Computer Vision*, vol. 74, no. 3, pp. 219–236, 2007.

[35] O. Booij, B. Terwijn, Z. Zivkovic, and B. Krose, "Navigation using an appearance based topological map," in *IEEE Int. Conf. on Robotics and Automation*, Rome, Italy, 2007, pp. 3927– 3932.

[36] J. Courbon, Y. Mezouar, and P. Martinet, "Autonomous navigation of vehicles from a visual memory using a generic camera model," *Intelligent Transport System (ITS)*, vol. 10, pp. 392–402, 2009.

[37] S. Jones, C. Andresen, and J. Crowley, "Appearance based process for visual navigation," in *Intelligent Robots and Systems, 1997. IROS '97., Proceedings of the 1997 IEEE/RSJ International Conference on*, vol. 2, sep 1997, pp. 551 –557 vol.2.

[38] G. Blanc, Y. Mezouar, and P. Martinet, "Indoor navigation of a wheeled mobile robot along visual routes," in *International Conference on Robotics and Automation*, Barcelona, Spain, 2005.

[39] Z. Chen and S. Birchfield, "Qualitative vision-based mobile robot navigation," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, may 2006, pp. 2686 –2692.

[40] T. Krajnìk and L. Přeučil, *A simple visual navigation system with convergence property*. H. Bruyninckx et al. (Eds.), 2008.

[41] A. Cherubini and F. Chaumette, "Visual navigation with a time-independent varying reference," in *IEEE Int. Conf. on Intelligent Robots and Systems, IROS'09*, St Louis, USA, October 2009, pp. 5968–5973.

[42] ——, "A redundancy-based approach to obstacle avoidance applied to mobile robot navigation," in *Proc. of IEEE Int. Conf. on Intelligent Robots and Systems*, Taipei, Taiwan, 2010.

[43] A. Cherubini, F. Spindler, and F. Chaumette, "A redundancy-based approach for visual navigation with collision avoidance," in *ICVTS proceedings*, 2011.

[44] A. Durand Petiteville, S. Hutchinson, V. Cadeant, and M. Courdesses, "2d visual servoing for a long range navigation in a cluttered environment," in 50^{th} *IEEE Conference on Decision and Control and European Control Conference*, Orlando, USA, December 2011.

[45] F. Chaumette and S. Hutchinson, "Visual servo control, part 1 : Basic approaches," *IEEE Robotics and Automation Magazine*, vol. 13, no. 4, 2006.

[46] P. Souères, T. Hamel, and V. Cadenat, "A path following controller for wheeled robots wich allows to avoid obstacles during the transition phase," in *IEEE, Int. Conf. on Robotics and Automation*, Leuven, Belgium, May 1998.

[47] P. Souères and V. Cadenat, "Dynamical sequence of multi-sensor based tasks for mobile robots navigation," in *SYROCO*, Wroclaw, Poland, September 2003.

[48] D. Folio and V. Cadenat, *Computer Vision - Treating Image Loss by using the Vision/Motion Link: A Generic Framework*. IN-TECH, 2008, ch. 4.

[49] A. Durand Petiteville, M. Courdesses, V. Cadenat, and P. Baillion, "On-line estimation of the reference visual features. application to a vision based long range navigation task," in *IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems*, Taipei, Taiwan, October 2010.

IROS Workshop on Visual Control of Mobile Robots (ViCoMoR 2012)
October 11th, 2012, Vilamoura, Algarve, Portugal

Vistas and Wall-Floor Intersection Features: Enabling Autonomous Flight in Man-made Environments

Kyel Ok, Duy-Nguyen Ta and Frank Dellaert

***Abstract*— We propose a solution toward the problem of autonomous flight and exploration in man-made indoor environments with a micro aerial vehicle (MAV), using a frontal camera, a downward-facing sonar, and an IMU. We present a general method to detect and steer an MAV toward distant features that we call *vistas* while building a map of the environment to detect unexplored regions. Our method enables autonomous exploration capabilities while working reliably in textureless indoor environments that are challenging for traditional monocular SLAM approaches. We overcome the difficulties faced by traditional approaches with *Wall-Floor Intersection Features*, a novel type of low-dimensional landmarks that are specifically designed for man-made environments to capture the geometric structure of the scene. We demonstrate our results on a small, commercially available quadrotor platform.**

I. Introduction

We address the problem of vision-based autonomous navigation and exploration in man-made environments for Micro Aerial Vehicles (MAVs). With its wide range of applications in military and civilian services, research in autonomous navigation and exploration for MAVs has been growing significantly in recent years. Despite many similar characteristics to ground robots, problems such as autonomous navigation, obstacle avoidance, and map building on an aerial robot have been much more challenging due to payload limitations, power availability, and extra degrees-of-freedom.

Recent work in autonomous MAV navigation and exploration has been insufficient due to aforementioned challenges. Related work, described in Section II, either neglects to address the power and payload limitations by using heavy and power-hungry sensors or uses vision-only but comes short of achieving autonomous exploration capabilities.

We present an autonomous navigation and exploration method, using a lightweight frontal camera, an IMU and a downward-facing sonar for height measurements. Our key contribution is combining map-building and detection of distant features, which we call *vistas,* to enable exploration strategies that could not be achieved before. For example, we utilize our map of inferred structure to detect unexplored regions of interest, such as new hallway openings. This type of capability could not be achieved in previous vision-based MAVs, without dedicating additional sensors for this purpose (i.e. frontal and side-facing sonars [1]).

Our first contribution is using vistas to determine the robot steering direction, enabling robust navigation. Our vistas are derived from first principles of what it means to be *distant*; hence, they are not hallway-specific like the previous work that depends on vanishing points detected from spurious edges [1] or hallway-specific cues [2]. Moreover, vistas are also derived from scale-space features and inherit the properties such that they are easily and reliably detected and tracked in many types of environments.

The authors are with the Center for Robotics and Intelligent Machines, Georgia Institute of Technology, Atlanta, Georgia, USA,
{kyelok, duynguyen, dellaert}@gatech.edu

Fig. 1: Our method uses *vistas* (bottom left) to maintain long-term orientation consistency and relies on a map of *Wall-Floor Intersection Features* (bottom right) to infer the scene structure. We present our results in an indoor setting using a Parrot AR.Drone (top).

Our second contribution is an indoor mapping paradigm that allows full exploration. In addition to vistas, for intelligent exploration schemes, the MAV needs some knowledge about the scene structure. We infer the structure from a map of compact and low-dimensional landmarks that are informative enough to capture the most important geometric information about the scene. Our novel landmarks that we call *Wall-Floor Intersection Features* lie on the perpendicular intersection of vertical lines on the wall and the horizontal floor plane. They encode the direction of the wall and can capture any type of corners whether straight, convex or concave. We build a map of our landmarks online using the state-of-the-art inferencing engine, iSAM2 [3].

We combine our contributions to demonstrate an autonomous exploration system on an inexpensive quadrotor.

II. Related Work

Recent work [4], [5], [6] successfully demonstrates MAV navigation and exploration in indoor environments using a map built with laser scanners. [7] present a full SLAM solution for an MAV equipped with a laser scanner to autonomously navigate in indoor environments. [8] presents a helicopter navigating with a laser scanner to avoid different types of objects such as buildings, trees, and 6mm wires in the city. However, these methods are severely limited to short-term operations due to their heavy payload and high power usage. Moreover, active sensors such as laser scanners are undesirable in many applications (e.g., military), due to the risk of cross-talk and ineligibility for covert operations. Therefore, we preclude the use of laser scanner and other heavy and power-hungry sensors.

Recent work in vision-based autonomous navigation neglects to provide exploration capabilities enabled by building a map of the environment. For example, [1] detects the vanishing point at the end of the hallway by finding intersection of long lines along the corridors. Similarly, on a ground robot, [2] fuses many specific properties present at the end of hallways such as high entropy, symmetry, self-similarity, etc. to infer the hallway directions. However, neither methods have a vision-based exploration capability to steer the robot toward undiscovered regions. [1] attempts to solve the problem but relies on supplementary sonar sensors to detect openings to the sides. However, this method does not infer the scene structure and cannot support any planning algorithms to efficiently explore the area, whereas our combined method can support any planning algorithm to navigate toward unexplored regions.

On the other hand, state-of-the-art map-building methods are insufficient for usage in indoor navigation. Some work relies on a downward camera for building a map [9], [10], [11] but lacks the ability to avoid obstacles. Many other vision-based methods build 3D point-cloud based maps [9], [10], [11] but in textureless indoor environments, the point-clouds are too sparse to reveal the 3D structure needed for path/motion planning. Although some [12], [13] build a map from edges in the environment, they neglect to infer the environment structure crucial for robot navigation. Furthermore, state-of-the-art vision-based methods that reconstruct the indoor scene [14], [15], [16] either rely on the indoor Manhattan world assumption or require expensive multi-hypothesis inference methods [15], [16]. Our method based on Wall-Floor Intersection Features improves on previous work with the ability to work in textureless environments, using sparse yet informative scene representation, and not relying on the indoor Manhattan world assumption.

III. Autonomous Navigation Toward Vistas

One of the first tasks in autonomous navigation and exploration is to determine the direction toward open space. In this section, we derive from first principles a general approach that can potentially be applied to any type of environment to steer the robot.

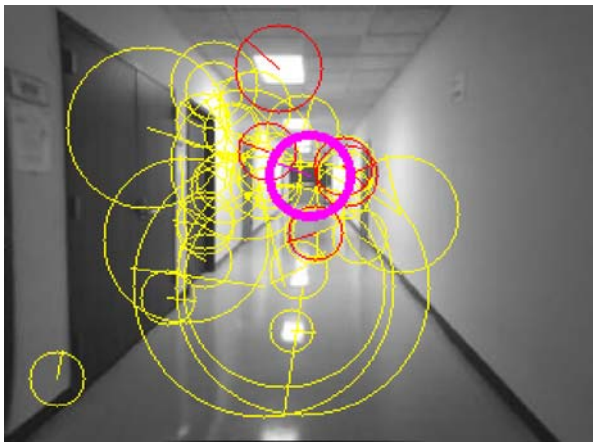


Fig. 2: Detected *vistas* (in red) and features that do not satisfy the *vista* criteria (in yellow) are shown. The closest *vista* to the mean of all the detected *vistas* (pink feature) is selected as the steering direction for the robot.

A. Vista Size Change Criterion

We use *vistas* to refer to those landmarks that are far away from the robot and can be used as a steering direction toward empty space when exploring in an unknown environment.

One important property of vistas is that, due to their far distance to the robot, the size of their projection in the camera frame does not change significantly when flying toward them. This property is already well-known in perceptual psychology under the τ-theory [17] by David Lee, saying that the time-to-collision (TTC) to an object is the ratio τ of the object's image size to the rate of its size change. Some work has utilized this property to compute TTC using optical flow [18], [19] or direct methods [20], [2].

Using this property, we derive vistas from relative size change of scale-space features such as SIFT [21] or SURF [22]. The optimal size of these features are computed by fitting a 3D quadratic function to the feature responses in scale-space around the max response [21].

Let s_1, s_2 be feature sizes and Z_1, Z_2 be their distance from the camera at frames 1 and 2. Since $s_i = f\frac{S}{Z_i}$, where f is the camera focal length and S is the true size of landmark, we have $s_1/s_2 = Z_2/Z_1$. It can be easily shown that $\frac{\Delta s}{s_2} = -\frac{\Delta Z}{Z_1} = \frac{t_z}{Z_1}$ where $\Delta s = s_2 - s_1$ is the absolute size change of the feature and $t_z = -\Delta Z = Z_1 - Z_2$ is the amount of forward movement of the robot between two frames, easily obtained from integrating an IMU, using a motion model, or fusing optical flow and corner tracking on a bottom-looking camera, as already implemented on the AR.Drone [23].

Let Z_{1min} be the minimum safety distance to the landmark in camera frame 1 so that any landmarks with $Z_1 \geq Z_{1min}$ can be considered vistas. The relative size change of vistas must satisfy

$$\frac{\Delta s}{s_2} \leq \frac{t_z}{Z_{1min}} \tag{1}$$

As shown in Figure 2, this criterion leads to a simple yet efficient way to detect distant landmarks in the environment.

B. Vista Rotation-predictability Criterion

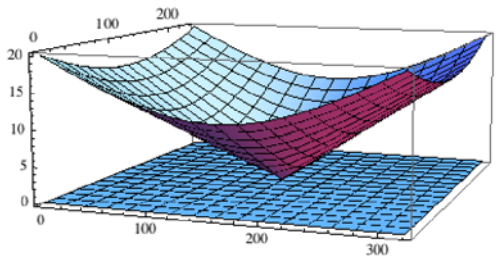


Fig. 3: Minimum Z^r_{1min} distances for rotation-predictable features for $t_x = t_y = 0,\ t_z = 0.1$. The horizontal xy-plane is the image pixel coordinate, and the vertical z-axis is the minimum Z_1 required at each pixel. Plot with camera calibration: $o_x = 160,\ o_y = 120,\ f_x = f_y = 210$.

The minimum safety distance Z_{1min} of vistas in the previous section could be chosen arbitrarily as long as it is safe for the robot to avoid collision with the wall at the moving speed. However, to ease the prediction and tracking of the vistas, we enforce another geometric property of distant landmarks that their projection in the image should be predictable using pure camera rotation, unaffected by the translation. We call this "rotation-predictability" criterion.

We derive this additional requirement for our vistas basing on a well-known fact that if a point is at infinity, its projection in the camera image can be purely determined by the camera rotation. In our case, the camera translation between two consecutive frames is insignificant compared to the distance from the camera to the landmarks, hence has no effect on the landmark position in the image.

More specifically, let p_1 and p_2 be the 2D homogeneous forms of the landmark projections in camera frames 1 and 2. Also, let $K = \begin{bmatrix} f_x & 0 & o_x \\ 0 & f_y & o_y \\ 0 & 0 & 1 \end{bmatrix}$ be the camera calibration matrix, and $X^1_2 = \{R, t\} \in \mathbb{SE}(3)$ be the odometry of the camera from frame 1 to frame 2. If the landmark P is at infinity or if the camera motion is under a pure rotation ($t = 0$), its projections p_1 and p_2 are related by the infinite homography $H = KR^2_1K^1$ between the two images [24]:

$$p_2 = p^r_2 \sim KR^2_1K^{-1}p_1,$$

where $R^2_1 = R^\top$, and $\sim$ denotes the equivalent up to a constant factor.

However, if the camera motion also involves a translation, i.e. $t \neq 0$, and the landmark is not at infinity, the relationship between p_1 and p_2 is:

$$\begin{aligned} p_2 = p^t_2 &\sim K(R^2_1 Z_1 K^{-1}p_1 + t^2_1) \\ &\sim p^r_2 + \frac{1}{Z_1}Kt^2_1, \end{aligned}$$

where $t^2_1 = -R^\top t$.

Consequently, the rotation-predictability criterion infers that p^t_2 must be well approximated by p^r_2. In this case, the effect of the camera translation t on p_2 is negligible and insensible by the camera, i.e., in homogeneous form, $\frac{1}{Z_1}Kt^2_1 \approx kp^r_2$, for some scalar $k \in \mathbb{R}$. To satisfy this constraint, we impose the condition that the non-homogeneous distance between p^t_2 and p^r_2 has to be less than 1 pixel, i.e.,

$$||\frac{1}{z_{p^r_2}}p^r_2 - \frac{1}{z_{p^t_2}}p^t_2||^2 \leq 1,$$

where $z_{p^r_2}$ and $z_{p^t_2}$ are the third components of p^r_2 and p^t_2, respectively. Solving for this constraint leads to the minimum depth Z^r_{1min} of the landmark such that its image projection can be purely determined by the camera rotation as follows:

$$\begin{aligned} &Z^r_{1min}(x, y, t) = \\ &\quad t_z + \sqrt{[f_x t_x + t_z(o_x - x)]^2 + [f_y t_y + t_z(o_y - y)]^2} \end{aligned} \quad (2)$$

where $t = \begin{bmatrix} t_x & t_y & t_z \end{bmatrix}^\top$, and (x, y) is the non-homogeneous coordinate of p_1.

This formula shows that the minimum Z^r_{1min} distance of the landmark in the first camera view depends on its position (x, y) in the first image, and also the camera translation t. Figure 3 shows the Z^r_{1min} required for each pixel landmark location in the image where the camera moves in z direction.

Note that at the Focus of Expansion (FoE), where the camera translation vector intersects with the camera image plane, the minimum Z^r_{1min} is very close to the camera. As a trivial example, when the camera moves forward without rotation, $R = I_{3\times3}$, the minimum distance for rotation-predictability criterion is $Z^r_{1min} = t_z$; i.e., any point along the camera optical axis will not be affected by the camera translation as long as it is in front of the second camera view.

Although such limitations exist in the FoE region, the rotation-predictability criterion is still useful to reject false vistas outside the region. Thus, we use $\max(Z^r_{1min}, Z_{1min})$ for the minimum distance in equation (1) to create the final criteria to track vistas on a frame to frame basis.

IV. Wall-Floor Intersection Features for Smoothing and Mapping

Despite vistas' ability to steer a robot toward open space, vistas alone can not grant fully autonomous exploration capabilities. In order to detect directions toward unexplored regions and adopt an intelligent planning scheme, it is critical to obtain a map of the environment. In other words, while flying toward vistas, we need to build a map of landmarks that contains sufficient information about the environment to simultaneously localize the robot and plan exploration strategies. Although, the well-studied problem of Simultaneous Localization and Mapping can be solved using state-of-the-art incremental smoothing and mapping algorithms such as iSAM2 [3], the problem of lack of texture in indoor environments still imposes difficulties in the landmark representation. We address this problem with our novel landmarks, *Wall-Floor Intersection Features*.

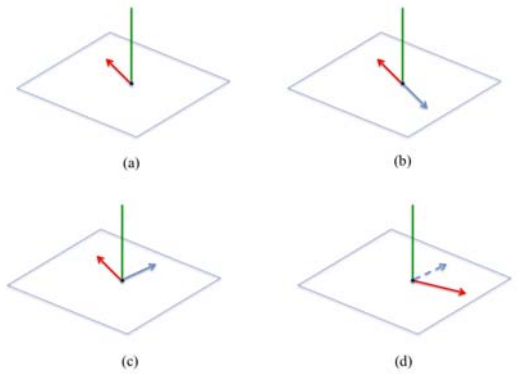


Fig. 4: (a) Our landmark encodes a vertical line position and a wall direction. (b) Two landmarks with opposite wall directions can share the same vertical line. (c) Two landmarks encoding an edge. (d) Two landmarks, one invisible.

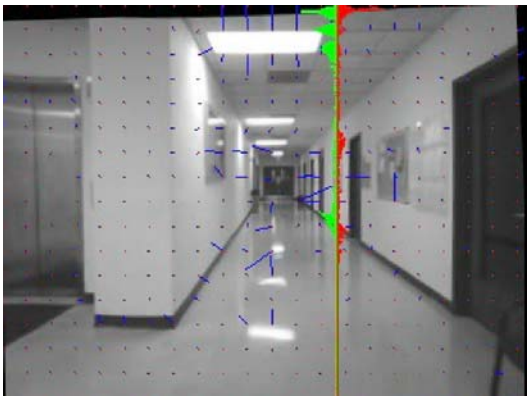
A. Wall-Floor Intersection Features

Choosing the right type of landmarks is challenging for indoor vision-based SLAM, due to the textureless scene. [25] proposes to recognize the floor-wall boundary in each column of the input image. [15], on the other hand, categorizes all possible types of corners in indoor environments to generates hypotheses of the environment structure. Recently, [16] generates and evaluates multiple hypotheses of wall-floor intersection lines from detected edges in the images, whereas [14] utilizes the floor-ceiling planar homology.

Inspired by these previous work, we propose a landmark representation that can encode the intersection of a vertical line on the wall and the intersecting floor plane. These new landmarks, named *Wall-Floor Intersection Feature,* are derived from our observation that a vertical line in the scene is most likely associated with a wall and an intersecting floor plane, whose location is estimated by the downward sonar sensor, allowing easy localization of the landmark in space.

Our landmark representation, shown in Figure 4, can employ different wall configurations by encoding only **a single wall direction** in each landmark and allowing two landmarks with different wall directions to co-exist at the same vertical line. This alleviates the need to explicitly model all types of concave/convex corners as previously done in [15], and can deal with non-right wall angles by allowing arbitrary angles between wall directions at the same vertical edge. For example, if a vertical line is on a single wall (ie. vertical edge of a door), the angle between the landmarks would be 180° and if the line is an intersection of two different walls (ie. corners), then the two landmarks will form an angle other than 180°. As such, our representation can efficiently capture the structure of the scene.

Requiring only a 2D position and a single direction, our landmarks can be represented as $\mathbb{SE}(2)$, an element of the Lie-group, where the representation is compact and standard Gauss-Newton optimization is straightforward. Moreover, our landmarks are also easy to detect for both vertical lines and wall directions, as discussed in the next section.



(a) Steerable filter responses along a vertical line. The maximum sum responses on each side are shown in red and green. Blue segments display dominant gradient direction at each point.

(b) Wall-floor corner detection. Detected features are shown in yellow and images of landmarks are shown in red (positive horizontal gradient) and blue (negative horizontal gradient).

Fig. 5: Detection results

B. Detection and Measurement Model

First, to detect the vertical line in the landmark, we rectify the image using an IMU, so that vertical lines in the 3D space are also vertical lines in the image, as shown in Figure 5. Then, the vertical line candidates are local maxima in the sum of horizontal image gradients $\mathcal{I}_x$ along each column of the image. Using height estimate from the sonar sensor, each point on the vertical line in the image is associated with one point on the floor plane by back-projection. Then, we only select points with high vertical image gradients $\mathcal{I}_y$ on the bottom half of the image, near the floor.

Then, we detect wall directions for the remaining candidates by (1) quantizing all possible directions on the left and right side of the detected vertical line, (2) summing up steerable filter responses [26] at every pixel along each bin direction and (3) choosing the directions with maximum sum responses on each side (see Figure 5).

Finally, we traverse the image in the detected wall directions as far as the steerable filter response is similar to the original detection. When the response differs by more than a threshold, we stop and store the length of the wall-floor intersection traversed. We finally choose features with lengths larger than a threshold as our landmarks.

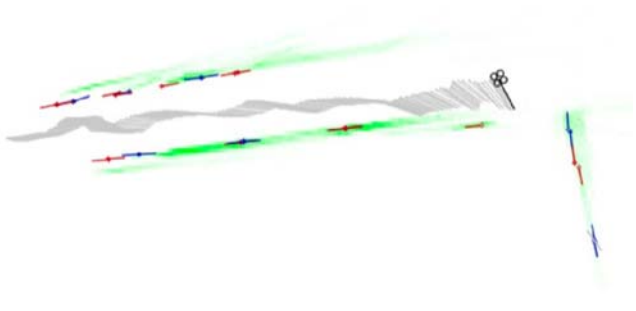


Fig. 6: Wall inference results (green) and an estimated map of Wall-Floor Intersection Features (red and blue).

C. Wall Inference

Our landmarks only capture local information about the wall structure. At places where there are no vertical lines on the wall, no landmarks exist. However, our Wall-Floor Intersection Features are capable of revealing the skeleton structure of the hallway. We perform an additional step to "fill in" the space between the features to yield a complete knowledge of the environment by accumulating evidence of walls in an occupancy grid, with each cell's evidence calculated by extending our features in the wall directions and summing up the image gradient strength along the extended direction. Figure 6 shows our inferred wall structure in the occupancy grid when the drone is turning a corner.

V. Experiments

A. Complete System

In order to obtain an autonomous system, we combine vistas and Wall-Floor Intersection Features with two additional supplementary navigation strategies.

Vistas: We use vistas to choose the steering direction toward open space. This governs the yaw direction of the robot and prevents head-on collision with obstacles.

Wall-Floor Intersection Features: Using iSAM2 [3] Smoothing and Mapping algorithm and our novel landmarks, we create a sparse map of our features and a grid map of inferred wall structure to be used in the later strategies.

Avoiding Side Collisions: Given the local occupancy grid centered at the current robot position, we infer the distance to the walls on the sides of the robot by attempting to equate the distance to the left and the right by changing the MAV's roll rates. This strategy prevents side collisions and navigates the robot in the middle of the environment.

Detecting directions to unexplored regions: We create two masks with openings on the left and right sides and apply them on the local grid map to find salient matches. Matches above a threshold is considered a new opening, and is used to re-direct the MAV.

Combining the four strategies, we obtain a system that can avoid collisions in both forward and side directions, fly toward unexplored open areas, while also simultaneously localizing and building a map of the environment.

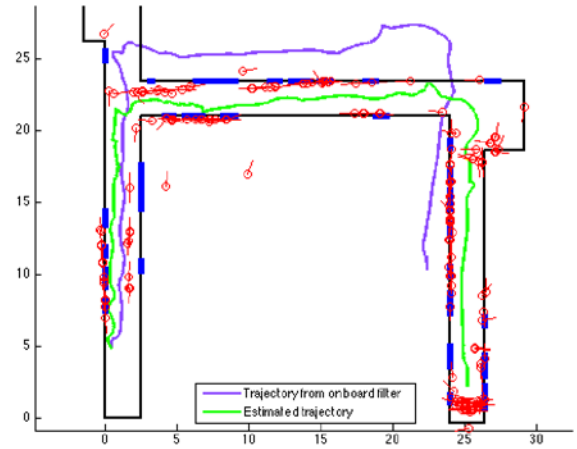


Fig. 7: Comparison between the ground-truth map of the environment and our manually-aligned estimated map. Our Wall-Floor Intersection Features are shown in red and the ground-truth floor layout in black (walls) and blue (doors). Our estimated trajectory of the quadrotor is in green, and the AR.Drone onboard estimate in purple. Accuracy of our estimate, completely constrained in the ground-truth map, shows advantages in using our features in textureless environments.

B. Experimental Setup

For the evaluation of our complete system using vistas and Wall-Floor Intersection Features, we fly a commercially-available AR.Drone quadrotor through a hallway, as shown in Figure 1. Using the 468 MHz processor on the AR.Drone, we stream 320×240 gray-scale images from the front facing camera at 10 Hz along with the IMU and sonar measurements. The main computing is done off-board, and the control outputs are streamed back to the quadrotor.

C. Map-building Results

We evaluate the quality of our map by first running our system on a set of video frames and sensor data recorded from a manual flight and compare the results with the hand-measured ground-truth of the test environment. Due to drift and unreliability in sensor readings during AR.Drone's take-off sequence [23], we only start our system once the drone stabilizes in the air. Since the entire map depends on the first robot pose at the system start, which is arbitrary due to the drift, we manually rotate our map to match the ground-truth map orientation. Figure 7 demonstrates our map of landmarks approximating the ground-truth structure sufficiently. Although there are some spurious features inside the walls that escaped rejection based on uncertainty, and large features are congregated at the bottom-right corner during the unstable landing sequence, our exploration strategies are unaffected by these small shortcomings in the map.

Furthermore, we compare the quadrotor trajectory estimated by our system with the one provided by the AR.Drone software. As shown in Figure 7, our estimated trajectory is much more accurate, being well-bounded inside the hallway interior, while the AR.Drone's estimate drifts significantly.

D. Autonomous Exploration Results

We also test our system in a hallway environment for (1) autonomously steering toward vistas, (2) building a map of the environment online, and (3) finding new corners and hallway openings to turn to. As shown in Figure 2, the vista detection was robust enough to detect and focus on distant features at the end of hallways, and effectively steer the robot toward that direction. As shown in our attached video[1], the skeleton map was accurate enough for inferring a grid map of the wall-structure, keeping the robot stay in the middle of the hallway and detecting new corners effectively. In addition, our full system could run in real-time at around 8 to 9 fps.

VI. Discussion and Future Work

We have presented a vision-based system that enables autonomous exploration strategies on an MAV in textureless indoor environments, which could not be achieved in previous work in the absence of heavy and power-hungry sensors. With our map of Wall-Floor Intersection Features, we are able to infer the entire scene structure and with vistas, steer toward open areas. We have demonstrated our complete system with two additional strategies (1) to keep the robot in the middle of the hallway, and (2) to detect opening directions to undiscovered regions. Our experiments show promising results toward a fully robust autonomous system for MAV navigation and exploration.

Although our method of combining vistas and Wall-Floor Intersection Features advances autonomous navigation and exploration capabilities of MAVs, there remain some limitations as future work. Our criteria for vistas (1) and (2) may have some practical limitations without perfect projective camera imaging. For example, the rolling shutter effect of the low-quality camera on the AR.Drone may affect the size of the features and violate (1). In addition, the Wall-Floor Intersection Features require future work for mapping with special chessboard-type floors where strong gradient lines exist in the floor's texture. Lastly, our wall-inference and hallway opening detection schemes are sensitive to thresholding values, and require fine-tuning for the specific lighting condition in the environment. A more sophisticated top-down algorithm remains as future work to make the complete system less sensitive to thresholds and be more robust to other lighting conditions.

Acknowledgements

This work is supported by an ARO MURI grant, award number W911NF-11-1-0046.

References

[1] C. Bills, J. Chen, and A. Saxena, "Autonomous MAV flight in indoor environments using single image perspective cues," in *Robotics and Automation, 2011. ICRA 2011. Proceedings of the 2011 IEEE International Conference on*, 2011.

[2] V. Murali and S. Birchfield, "Autonomous exploration using rapid perception of low-resolution image information," *Autonomous Robots*, pp. 1–14, 2012.

[3] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert, "iSAM2: Incremental smoothing and mapping using the Bayes tree," *Intl. J. of Robotics Research*, vol. 31, pp. 217–236, Feb 2012.

[4] S. Grzonka, G. Grisetti, and W. Burgard, "Towards a navigation system for autonomous indoor flying," in *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, 2009, pp. 2878–2883.

[5] A. Bachrach, R. He, and N. Roy, "Autonomous flight in unknown indoor environments," *International Journal of Micro Air Vehicles*, vol. 1, no. 4, p. 217–228, 2009.

[6] M. Achtelik, A. Bachrach, R. He, S. Prentice, and N. Roy, "Stereo vision and laser odometry for autonomous helicopters in GPS-denied indoor environments," *Unmanned Systems Technology XI. Ed. Grant R. Gerhart, Douglas W. Gage, & Charles M. Shoemaker. Orlando, FL, USA: SPIE*, 2009.

[7] S. Grzonka, G. Grisetti, and W. Burgard, "A fully autonomous indoor quadrotor," *Robotics, IEEE Transactions on*, no. 99, pp. 1–11, 2012.

[8] S. Scherer, S. Singh, L. Chamberlain, and S. Saripalli, "Flying fast and low among obstacles," in *Robotics and Automation, 2007 IEEE International Conference on*, 2007, p. 2023–2029.

[9] M. Blösch, S. Weiss, D. Scaramuzza, and R. Siegwart, "Vision based mav navigation in unknown and unstructured environments," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, 2010, p. 21–28.

[10] M. Achtelik, M. Achtelik, S. Weiss, and R. Siegwart, "Onboard imu and monocular vision based control for mavs in unknown in- and outdoor environments," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[11] S. Weiss, M. Achtelik, L. Kneip, D. Scaramuzza, and R. Siegwart, "Intuitive 3d maps for mav terrain exploration and obstacle avoidance," *Journal of Intelligent and Robotic Systems*, vol. 61, pp. 473–493, 2011.

[12] G. Klein and D. Murray, "Improving the agility of keyframe-based SLAM," in *Eur. Conf. on Computer Vision (ECCV)*, Marseille, France, 2008.

[13] E. Eade and T. Drummond, "Edge landmarks in monocular slam," in *Proc. British Machine Vision Conf*, 2006.

[14] M. D. Flint A. and R. I., "Manhattan scene understanding using monocular, stereo, and 3d features," in *International Conference on Computer Vision*. IEEE, 2011.

[15] D. Lee, M. Hebert, and T. Kanade, "Geometric reasoning for single image structure recovery," in *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2009, pp. 2136–2143.

[16] G. Tsai, C. Xu, J. Liu, and B. Kuipers, "Real-time indoor scene understanding using bayesian filtering with motion cues," in *International Conference on Computer Vision*, 2011.

[17] D. Lee *et al.*, "A theory of visual control of braking based on information about time-to-collision," *Perception*, vol. 5, no. 4, pp. 437–459, 1976.

[18] N. Ancona and T. Poggio, "Optical flow from 1-d correlation: Application to a simple time-to-crash detector," *International Journal of Computer Vision*, vol. 14, no. 2, pp. 131–146, 1995.

[19] D. Coombs, M. Herman, T. Hong, and M. Nashman, "Real-time obstacle avoidance using central flow divergence and peripheral flow," in *Computer Vision, 1995. Proceedings., Fifth International Conference on*. IEEE, 1995, pp. 276–283.

[20] B. Horn, Y. Fang, and I. Masaki, "Time to contact relative to a planar surface," in *Intelligent Vehicles Symposium, 2007 IEEE*. IEEE, 2007, pp. 68–74.

[21] D. Lowe, "Distinctive image features from scale-invariant keypoints," *Intl. J. of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[22] H. Bay, T. Tuytelaars, and L. V. Gool, "Surf: speeded up robust features," in *Eur. Conf. on Computer Vision (ECCV)*, 2006.

[23] P. Bristeau, F. Callou, D. Vissière, and N. Petit, "The navigation and control technology inside the ar. drone micro uav," in *World Congress*, vol. 18, no. 1, 2011, pp. 1477–1484.

[24] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[25] E. Delage, H. Lee, and A. Ng, "A dynamic bayesian network model for autonomous 3d reconstruction from a single indoor image," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 2. IEEE, 2006, pp. 2418–2428.

[26] W. Freeman and E. Adelson, "The design and use of steerable filters," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1991.

[1] http://youtu.be/x8oyld2m9Cw

IROS Workshop on Visual Control of Mobile Robots (ViCoMoR 2012)
October 11th, 2012, Vilamoura, Algarve, Portugal

Distributed Policies for Neighbor Selection in Multi-Robot Visual Consensus

Eduardo Montijano Johan Thunberg Xiaoming Hu Carlos Sagues

***Abstract*— In this paper we propose a distributed algorithm for choosing the appropriate neighbors to compute the control inputs of a team of robots. We consider a scheme where the motion of each robot is decided using nearest neighbor rules. In this scheme each robot is equipped with a camera and can only exchange visual information with a subset of the robots. Using the information provided by their neighbors, the robots compute their control inputs, eventually reaching a consensus in their motion. However, if a robot has too many neighbors (e.g., a star topology), then it will require a long time to process all the received information, leading to long loop times or synchronization problems. In the paper we provide two distributed policies for the robots to select at each iteration the information of a fixed number of neighbors. In both cases we demonstrate convergence to the consensus with a considerable reduction on the amount of required computations. Simulations in a virtual environment show the effectiveness of the proposed policies.**

I. Introduction

The idea of multiple robots working in cooperation to achieve a common goal is of high interest in many tasks, such as exploration, surveillance or transportation. Multi-robot systems can perform these tasks with more robustness or in less time than one robot working alone. On the other hand, in order to carry on with these tasks, the robots need to be able to move in coordination.

A generalized problem in this context is the problem of reaching a consensus by all the robots. From the control perspective, the consensus problem [14] consists of making a team of robots to move all together in a common direction, with the peculiarity that this goal is achieved by each robot using only partial information given by the nearest-neighbors in the team. In this way all the robots play the same role in the formation, conferring the system a natural robustness against changes in the topology and individual failures. A key aspect left aside in most of the existing work in this topic, e.g., [2], [6], [12], [15], is how the robots estimate their neighbors positions to control their motion.

Vision sensors can play a fundamental role in this part of the process due to the big amount of information that images contain. Additionally, all the research done in the field of computer vision during the past decades can be exploited in a multi-robot framework in order to achieve the desired goal. Some works consider a single camera and a central unit to control all the robots [7]. Distributed solutions using omnidirectional cameras can be found in [13], [16] where the robots can see all their neighbors. If the robots are equipped with monocular cameras with limited field of view, then the observation of all the neighbors may not always be possible. A leader-follower solution is adopted in [5], where each robot only needs to observe another robot, leading to tree configurations. Geometry constraints are used in [11] to allow the network to be configured in any arbitrary topology.

In the latter approach each robot computes its control input using the epipoles between its current image and the images of its neighbors in the communication graph. The use of the epipolar constraint presents some advantages over other approaches. First of all, it has been successfully used to control the motion of a single robot on several occasions [1], [8], which gives this constraint reliability to be used in a multi-robot context. Secondly, the robots can reach the consensus even if they are not directly observing each other as long as they have common observations of the environment. Lastly, the controller does not impose any constraint on the network topology as each robot can compute as many pairs of epipoles as neighbors in the communication graph it has.

On the other hand, in the approach presented in [11], the number of neighbors determines the amount of time each robot will require to compute its control input. In a distributed scenario, if one robot has many neighbors, e.g., a star topology, with one robot connected to all the others, then it will receive many images. Processing all the images may take a long time, depending on the computation capabilities of the robots. This can lead to long times in the control loop or even to synchronization problems between the robots with different number of neighbors. Therefore, additional mechanisms are required to keep the amount of computations under control for all the robots.

In this paper we contribute to the state of the art presenting two distributed policies that allow the robots to select only a subset of their neighbors to compute the control input. In this way the team is still able to reach the consensus but the computational demands of each robot are bounded and equal. Additionally, we discuss the convergence of the considered controller for directed graphs.

The rest of the paper is organized as follows: In section II we review the distributed control law using epipoles to reach the consensus of the team of robots. All the formal details about the distributed policies for neighbor selection are explained in section III. Section IV shows simulations in a virtual environment where the two proposed policies

E. Montijano is with Centro Universitario de la Defensa (CUD) and Instituto de Investigación en Ingeniería de Aragón (I3A), Zaragoza, Spain.

J. Thunberg and X. Hu are with Department of Math, Division of Optimization and Systems Theory, Royal Institute of Technology (KTH), Sweden.

C. Sagues is with Departamento de Informática e Ingeniería de Sistemas - Instituto de Investigación en Ingeniería de Aragón (I3A), Universidad de Zaragoza, Spain.

are tested and compared with the standard distributed controller. Finally, in section V the conclusions of the work are presented.

II. Distributed Consensus using Epipoles

In this section we review the distributed controller based on the epipolar geometry to achieve the consensus. For additional details we refer the reader to [11].

We consider a set $\mathcal{V}$ of N homogeneous autonomous robots. Communications between the robots are defined with a connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with $\mathcal{E}$ the set of communication links. In this way, if robots i and j are able to exchange messages with each other, then $(i, j) \in \mathcal{E}$. The neighbors of robot i are defined as $\mathcal{N}_i = \{j \in \mathcal{V} \mid (i, j) \in \mathcal{E}\}$.

The robots move on the plane with non-holonomic motion constraints. Given two robots, i and j, their relative positions can be defined by a distance, r_{ij}, a bearing angle, ψ_{ij}, and relative orientation, θ_{ij}. The goal of the consensus problem is to make all the robots achieve the same orientation, i.e., $\theta_{ij} \to 0$, $\forall i, j \in \mathcal{V}$, as $t \to \infty$. To achieve this goal, each robot has two control inputs, v_i and w_i, which are the linear and angular velocity respectively. Since the linear velocity is not required to make the robots achieve the consensus, along the paper we consider it constant for all the robots, $v_i = v \geq 0,\ \forall i$.

In our setup all the robots are equipped with pinhole monocular cameras with limited field of view. We assume that all the robots have identical cameras onboard, with unknown calibration matrix equal to $\mathbf{K} = \text{diag}(\alpha, \alpha, 1)$, with $\alpha > 0$, the focal length of the camera. For any pair of robots, i and j, the lack of calibration implies that r_{ij}, ψ_{ij} and θ_{ij} are not directly available. The output of the system is instead defined by the epipoles of the images acquired by them (see Fig. 1). The robots exchange their images and use the epipolar constraint [10] to compute e_{ij} and e_{ji}, the epipoles in the two images. Specifically, due to the planar motion, we are only interested in the x-coordinate of the epipoles, which satisfies

$$e_{ijx} = \alpha \tan(\psi_{ij}),\ e_{jix} = \alpha \tan(\psi_{ij} - \theta_{ij}). \tag{1}$$

For simplicity purposes, in the following we use e_{ij} and e_{ji} to refer to the expressions in equation (1).

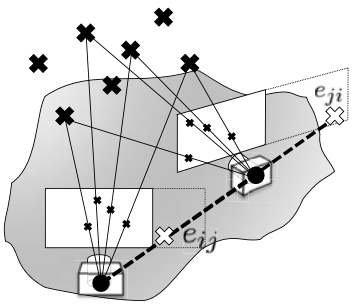


Fig. 1: The robots use the epipoles between their images to compute the control input.

Given a pair of neighbor robots, by eq. (1), a necessary condition for the consensus is that their epipoles must be equal, $\theta_{ij} = 0 \Rightarrow e_{ij} = e_{ji}$. To reach this objective the control input w_i of each robot is defined as:

$$w_i = K \sum_{j \in \mathcal{N}_i} w_{ij}, \tag{2}$$

where $K > 0$ is the controller gain and w_{ij} is the misalignment in the epipoles, defined as

$$w_{ij} = \begin{cases} d_{ij} & \text{if } |d_{ij}| \leq \frac{\pi}{2} \\ -\text{sign}(d_{ij})(\pi - |d_{ij}|) & \text{otherwise} \end{cases}, \tag{3}$$

with

$$d_{ij} = \arctan(\frac{e_{ij}}{\beta}) - \arctan(\frac{e_{ji}}{\beta}) \in (-\pi, \pi], \tag{4}$$

and $0 < \beta < \infty$ some fixed positive constant to choose. Note that, if $\beta = \alpha$, then the setup is calibrated, $d_{ij} = \theta_{ij}$, and the relative orientation between the robots can be computed from the epipoles. However, we assume that this is not the case and $\beta \neq \alpha$.

The following result determines the conditions required for the controller to converge to the consensus:

***Theorem** 2.1 (Theorem 3.2 [11]):* Let the robots be initially oriented in such a way that $|\theta_{ij}| \leq \theta_M < \pi/2,\ \forall i, j \in \mathcal{V}$. If the robots use the control law (2) with β satisfying

$$\alpha \tan(\frac{\theta_M}{2}) < \beta < \frac{\alpha}{\tan(\frac{\theta_M}{2})}, \tag{5}$$

then $\lim_{t \to \infty} \theta_{ij} = 0, \forall i, j \in \mathcal{V}$, i.e., the system will reach consensus. ■

Additionally, the system is robust to changes in the communication topology, as long as the following assumptions are satisfied.

***Assumption** 2.1:* There exists a lower bound, $\delta > 0$, on the time between two consecutive changes in the topology. Denoting $t_k,\ k \in \mathbb{N}$, the discrete time instants when the topology changes, then $t_{k+1} - t_k \geq \delta,\ \forall k$.

***Assumption** 2.2:* There exists a positive time period T such that, for any instant of time, t, the collection of communication topologies in the time interval $(t, t + T)$ is jointly connected.

The problem with the aforementioned controller is the amount of computations that the robots require in order to compute the epipoles between them and all their neighbors. If one robot has too many neighbors then it will have to compute many epipoles. For that reason additional mechanisms are required to keep the computational demands of all the robots bounded and similar.

Along the rest of the paper we will assume that the communication graph is fixed and the conditions in Theorem 2.1 regarding β and the initial orientations are satisfied. Since the neighbor policies will select different neighbors at each iteration, the assumptions regarding the changes in the communication topology will be required to prove the convergence.

III. Distributed Policies for Neighbor Selection

In this section we propose two distributed policies to select, from the subset of neighbors, which one each robot should choose to compute the epipoles. The first policy chooses at each iteration the robot that was not selected for the longest time. The second policy chooses at each iteration the neighbor that supposedly has the orientation farthest away. For both policies we prove convergence to the consensus.

Before explaining in detail our policies for neighbor selection, let us note that by selecting a subset of the neighbors what we are doing in practice is just changing the communication topology at each iteration. However, the proposed policies do not ensure a bi-directional selection, that is robot the fact that robot i chooses j to compute the epipoles does not imply that robot j chooses robot i as well. As a consequence, the communication topologies need to be modeled with time-varying directed graphs. Nevertheless, the proposed controller will also reach the consensus if Assumptions 2.1 and 2.2, see, e.g., [4]. Therefore, it will be enough to prove that the proposed policies ensure that Assumptions 2.1 and 2.2 are satisfied to reach the consensus.

In the following we present the two policies and formally prove the convergence to the consensus.

A. Policy 1: Choose the neighbor that was not selected for the longest time

The first policy we propose consists of selecting a different neighbor at each iteration. Specifically the one that was not selected for the longest time. Let each robot handle a vector $\mathbf{N}_i(t) = [N_{i1}(t), \ldots, N_{iN}(t)]$, with $N_{ij}(t)$ representing the number of communication rounds that has passed since the last time that robot i chose robot j as the selected neighbor to compute the epipoles.

Initially, $N_{ij}(0) = 0$ for all j. Then, at time t, the neighbor selected by robot i, denoted by $j(t)$, will be

$$j(t) = \arg_{j \in \mathcal{N}_i} \max N_{ij}(t), \tag{6}$$

and the control input of the robot

$$w_i(t) = K w_{ij(t)}. \tag{7}$$

Once the robot has computed the epipoles and the control input, it updates the vector $\mathbf{N}_i(t)$ with the following rule

$$N_{ij}(t+1) = \begin{cases} 0 & \text{if } j = \{j(t), i\} \\ N_{ij}(t) + 1 & \text{otherwise} \end{cases}, \tag{8}$$

so that it ensures that $j(t)$ will not be chosen again until all the other possible neighbors have been chosen once.

***Proposition** 3.1:* Let all the robots select, at each iteration, one neighbor to compute the epipoles using equations (6), (8) and move using the controller in eq. (7) considering only this neighbor. Then, the system will reach the consensus.

Proof: Since we are assuming that the communication topology is fixed, $\mathcal{G}$ necessarily is connected and the neighbors of each robot remain constant the whole time. At any iteration, t, each robot selects only one of its neighbors, which may lead to a disconnected digraph, $\mathcal{G}(t) \subseteq \mathcal{G}$. However, in the following iterations the selected neighbor of all those robots with more than one will be different. Denoting $\mathcal{N}_{\max} = \max |\mathcal{N}_i|$, we can see that for any t

$$\mathcal{G}(t) \cup \mathcal{G}(t+1) \cup \ldots \cup \mathcal{G}(t + \mathcal{N}_{\max}) = \mathcal{G},$$

which means that Assumption 2.2 is satisfied when using this policy. Considering that the time required to compute the epipoles is not zero, Assumption 2.1 is also satisfied and then we can conclude that the time varying evolution of the graph satisfies the conditions to reach the consensus. ■

B. Policy 2: Choose the neighbor with more misalignment

The second policy we propose is designed to reduce the orientation error with the neighbor that supposedly is further away at each iteration.

Let each robot have a vector $\hat{\mathbf{d}}_i(t) = [\hat{d}_{i1}(t), \ldots, \hat{d}_{iN}(t)]$, with $\hat{d}_{ij}(t)$ being the last value of d_{ij} computed by robot i using the information provided by robot j, equation (4). Initially, $\hat{d}_{ij}(0) = \infty$ for all j. The neighbor selected at each iteration is chosen by

$$j(t) = \arg_{j \in \mathcal{N}_i(t)} \max \hat{d}_{ij}(t), \tag{9}$$

that is, the one with the estimated most misalignment at current time. The control input of each robot is then assigned as in eq. (7).

With the epipoles computed, the update of $\hat{\mathbf{d}}_i(t)$ executed in this case is

$$\hat{d}_{ij}(t+1) = \begin{cases} d_{ij} & \text{if } j = j(t) \\ \hat{d}_{ij}(t) & \text{otherwise} \end{cases}, \tag{10}$$

with d_{ij} the value computed in equation (4).

***Proposition** 3.2:* Let all the robots select, at each iteration, one neighbor to compute the epipoles using equations (9), (10) and move using the controller in eq. (7) considering only this neighbor. Then, the system will reach the consensus.

Proof: With this policy we cannot ensure that the robots are changing the selected neighbors at each iteration. Let us assume that none of the robots change the selected neighbor, this means that the topology of the network remains fixed for some time. In such case we already now that the team approaches to the consensus, meaning that $d_{ij} \to 0$ for all i and j that are neighbors in the subgraph defined by the policy, and so $\hat{d}_{ij} \to 0$ for the same set. This means that at some point there will be some robot i and some $k \in \mathcal{N}_i$ such that $\hat{d}_{ij} > \hat{d}_{ik}$ and the robot will change the selected neighbor. Noting that any change of the selected neighbor does not change the rest of values, $\hat{d}_{ij}$, in the network, we can use this argument iteratively to see that all the neighbors of each robot will be selected at some point. After that, using the same arguments as in Proposition 3.1 we conclude that the system will reach the consensus. ■

C. Discussion

There are several advantages of using any of the two proposed policies instead of computing the epipoles with the images sent by all the robots. First of all, the consensus is achieved requiring less computations at each communication

round. Each pair of computed epipoles requires an initial step to match the features of the two images plus a robust method to estimate the epipolar constraint, e.g., DLT+RANSAC [10]. By selecting only one neighbor, we are executing this step only once at each iteration instead of $|\mathcal{N}_i|$ times. This computational reduction can be of high interest in situations where the energy of the robots is limited.

Synchronization issues are also solved. Note that the controller requires the images of all the robots to be acquired (approximately) at the same instant. If each robot does not acquire a new image until it has processed all the received information, and assuming that the number of neighbors is not going to be the same for all the robots, then without additional mechanisms to synchronize the network there will appear time discrepancies among the matched images. On the other hand, making all the robots to select only one neighbor to compute the epipoles, will imply similar computation times for all of them, leading to a natural synchronization.

In case the robots want to select more than one neighbor, let us say k neighbors at each iteration, both policies are still be applicable. In the first policy, each robot selects the k neighbors that were not selected for the longest time whereas the in the second policy, each robot selects the k neighbors with the largest value of $\hat{d}_{ij}(t)$. The larger k, the most similar would be the results to the standard case, at the price of more and more computational demands.

IV. Simulations

The properties of the proposed controller are shown in simulations. The experiments have been carried out using Matlab. We have considered a fixed robotic network composed by ten robots with initial positions and orientations depicted in Fig. 2 and communications defined by the black lines. As we can see, there are some robots that have up to five neighbors in the communication graph whereas others only have one or two neighbors. The vision system has been

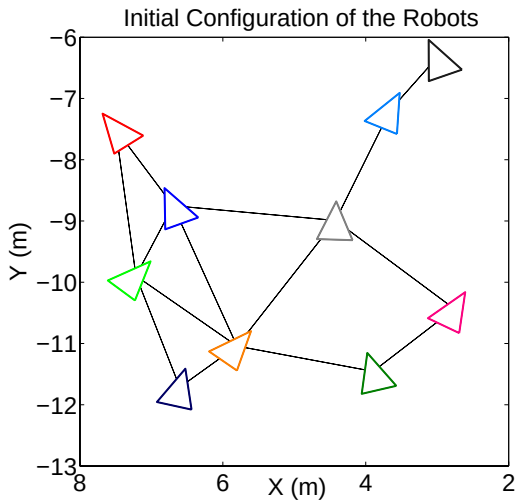


Fig. 2: Initial configuration of the team of robots in the experiments. Black lines represent direct communication between robots.

simulated using the virtual reality toolbox of MatLab. In this way, the robots acquire virtual images of resolution 640×480 pixels depending on their position and orientation. We have extracted SIFT [9] features from the virtual images and the 8 point algorithm with RANSAC [3] to match them in a robust way and to compute the epipoles between pairs of robots. An example of the images acquired by the robots and the features extracted and matched can be found in Figure 3.

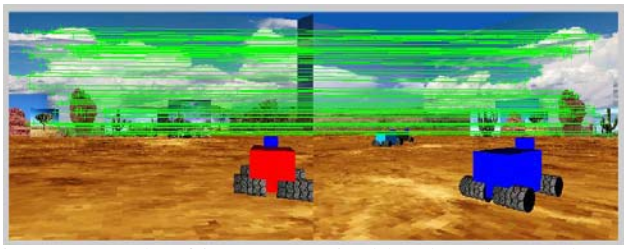


Fig. 3: Example of images acquired by the robots and SIFT matches using the epipolar constraint.

We have executed the controller in eq. (2) without using a neighbor selection policy and the controller (7) using the two policies proposed in section III. The evolution of the orientation of the robots in the three cases can be seen in Figure 4. As expected, all the robots reach the consensus without problems in the three situations. The evolution of the orientation when the information of all the neighbors is used is smoother and the consensus is reached in less time. However, the difference with respect to the other two graphics is negligible and in the simulation we are not considering the real time spent by each robot to compute the epipoles with all the neighbors. The second policy (neighbor with more misalignment) reaches the consensus relatively faster than the first policy, which makes sense because it tries to reduce the error with the robot with most relative orientation.

The control inputs of the robots in each scenario are depicted in Fig. 5. Again, the control inputs when no policies are used are smoother than when using any policy because they are computed using the same set of images the whole time but this simulation is not considering the real time spent to compute the inputs. The second policy also seems to be better than the first one in this aspect.

TABLE I: Computational time (seconds per robot and iteration)

Quantity	No policy	Policy 1	Policy 2
Mean time	12.65	3.80	3.85
Std. dev	5.83	0.77	0.75
Max time	27.99	5.92	5.84
Min time	2.55	1.49	1.28

We have measured the time spent to compute the control inputs when the robots do not use the proposed policies and when they do to point out the real advantages of using a policy to select a subset of neighbors. The computational time spent by each robot at each iteration is depicted in Fig. 6. We can see that using any of the two policies the computational time per iteration and robot remains bounded and similar whereas in the standard case there are big variations in the loop time, depending on the number of

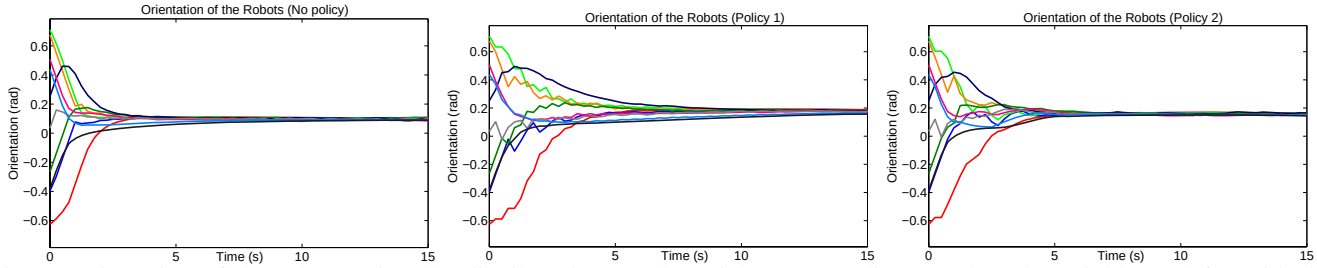


Fig. 4: Orientation of the robots using the distributed controller without any policy to select the neighbors (left), with the policy to choose the neighbor that was not selected for the longest time (middle) and with the policy to choose the neighbor with the most misalignment (right). In the three cases all the robots reach the consensus in a similar time.

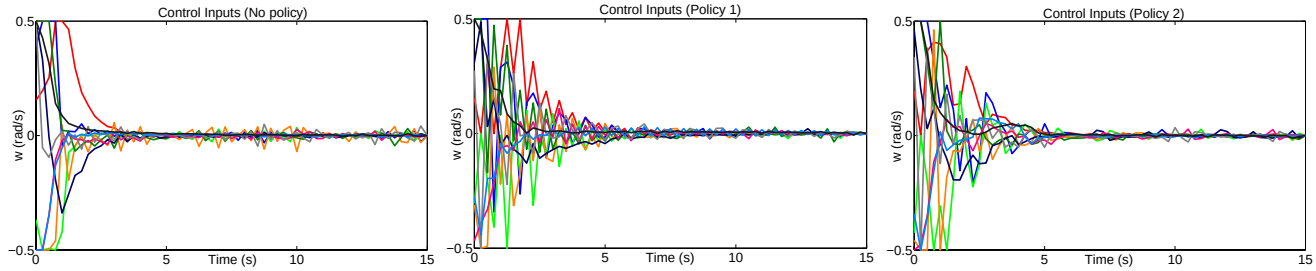


Fig. 5: Control inputs of the ten robots in the three scenarios: without any policy to select the neighbors (left), with the policy to choose the neighbor that was not selected for the longest time (middle) and with the policy to choose the neighbor with the most misalignment (right).

Fig. 6: Computational time spent by each robot at each iteration: without any policy to select the neighbors (left), with the policy to choose the neighbor that was not selected for the longest time (middle) and with the policy to choose the neighbor with the most misalignment (right). As we can see, when the robots use a policy, all of them spend approximately the same computational time, whereas without a policy each robot requires a different time depending on its neighbors.

neighbors of each robot. The statistics of these times are shown in in Table I.

V. Conclusions

In this paper we have proposed two distributed policies for a team of robots to select a subset of their neighbors to compute their control inputs. This selection bounds the time required by the robots to compute the input, making it equal for all of them, avoiding possible synchronization problems that could appear because of the different computation requirements of each robot. The computational reduction is also of high interest when the information shared by the robots is provided by vision sensors, because image processing methods are in general time-demanding. We have proved that both policies ensure convergence to the consensus and we have shown in simulations the benefits of using any of the two approaches compared to the situation in which all the neighbors are considered.

Acknowledgments

This work was supported by the project DPI2009-08126 and partly supported by Swedish Research Council and Swedish Foundation for Strategic Research.

References

[1] R. Basri, E. Rivlin, and I. Shimshoni, *Visual homing: Surfing on the epipoles*, International Journal of Computer Vision **33** (1999), no. 2, 117–137.

[2] J. Cortes, *Global and robust formation-shape stabilization of relative sensing networks*, Automatica **45** (2009), no. 12, 2754 – 2762.

[3] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*, Cambridge University Press, Cambridge, 2000.

[4] Q. Hui and W. M. Haddad, *Distributed nonlinear control algorithms for network consensus*, Automatica **44** (2008), no. 1, 2375–2381.

[5] T. Ibuki, T. Hatanaka, M. Fujita, and M. W. Spong, *Visual feedback attitude synchronization in leader-follower type visibility structures*, 49th IEEE Conference on Decision and Control, December 2010, pp. 2486–2491.

[6] A. Jadbabaie, J. Lin, and A. S. Morse, *Coordination of groups of mobile autonomous agents using nearest neighbor rules*, IEEE Transactions on Automatic Control **48** (2003), no. 6, 988–1001.

[7] G. Lopez-Nicolas, M. Aranda, Y. Mezouar, and C. Sagues, *Visual Control for Multi-Robot Organized Rendezvous*, IEEE Transactions on Systems Man and Cybernetics: Part B (2012), to appear.

[8] G. López-Nicolás, C. Sagues, J.J. Guerrero, D. Kragic, and P. Jensfelt, *Switching visual control based on epipoles for mobile robots*, Robotics and Autonomous Systems **56** (2008), no. 7, 592–603.
[9] D. Lowe, *Distinctive image features from scale-invariant keypoints*, International Journal of Computer Vision **60** (2004), no. 2, 91–110.
[10] Y. Ma, S. Soatto, J. Kosecka, and S. S. Sastry, *An invitation to 3d vision*, SpringerVerlag, 2004.
[11] E. Montijano, J. Thunberg, X. Hu, and C. Sagues, *Multi-Robot Distributed Visual Consensus using Epipoles*, IEEE Conference on Decision and Control, 2011, pp. 2750–2655.
[12] N. Mostagh and A. Jadbabaie, *Distributed geodesic control laws for flocking of nonholonomic agents*, IEEE Transactions on Automatic Control **52** (2007), no. 4, 681–686.
[13] N. Mostagh, N. Michael, A. Jadbabaie, and K. Daniilidis, *Vision-based, distributed control laws for motion coordination of nonholonomic robots*, IEEE Transactions on Robotics **25** (2009), no. 4, 851–860.
[14] Wei Ren and Randal W. Beard, *Distributed consensus in multi-vehicle cooperative control*, Communications and Control Engineering, Springer-Verlag, London, 2008.
[15] J. Thunberg, E. Montijano, and X. Hu, *Distributed attitude synchronization control*, 50th IEEE Conference on Decision and Control and European Control Conference, December 2011, pp. 1962–1967.
[16] R. Vidal, O. Shakernia, and S. Sastry, *Following the flock [formation control]*, IEEE Robotics and Automation Magazine **11** (2004), no. 4, 14–20.

IROS Workshop on Visual Control of Mobile Robots (ViCoMoR 2012)
October 11th, 2012, Vilamoura, Algarve, Portugal

Target Tracking and Obstacle Avoidance for a VTOL UAV using Optical Flow

Aurélie Treil[1], Philippe Mouyon[1], Tarek Hamel[2], Alain Piquereau[1] and Yoko Watanabe [1]

***Abstract*— This paper presents the practical implementation of a nonlinear visuel servo controller for obstacle avoidance and target tracking of an eye-in-hand Vertical Take-Off and Landing Uninhabited Air Vehicle (VTOL UAV). The VTOL vehicle is assumed to be equipped with a minimum sensor suite; a camera and Inertial Measurement Unit (IMU). The control law uses optical flow calculated from the camera images and angular measurement from IMU to ensure obstacle avoidance and target tracking of the UAV while maneuvering over a textured terrain made of planar surfaces. The proposed controller has been tested in simulation as a preliminary step to outdoor flight experiments. Both simulation and experimental results are presented in this paper.**

I. INTRODUCTION

Most of progresses in aeronautics field since the appearance of aircraft are due to military applications. UAVs are not an exception. They are born to limit losses of human pilot lifes during recognition missions. Operation fields, that require the use of UAV, are often urban environment. There are also some civil applications like monitoring traffic congestion, regular inspection of infrastructure such as bridges, etc. The urban environment brings new constraints to UAV operation. Indeed, masking of GPS signals by building walls in the city disables its use. Moreover to follow streets, which can be narrow, or even inside buildings UAVs need to be light and small. Finally, UAVs can be lost during mission, which implies cost constraint.

So, new solutions have been studied to compensate non-availability of GPS in urban environment and to go beyond cost and size constraints. In particular, visual servoing has been for twenty years an important research subject in ground robotics field and more recently in UAV field.

Visual servoing for ground applications has been extensively studied. Question of docking [1] and obstacle detection and avoidance [2], [3] have been treated. For example to detect obstacle, methods using perspective have been set up [4] as well as method based on optical flow [5]. In UAV category optical flow is often used in biomimetic approaches [6], [7], [8]. Optical flow is also used combined with IMU measurements for stabilisation and vertical landing problematics, [9], [10] and for terrain following applications, [11].

Two objectives are identified in this paper. The first is the target tracking for a VTOL UAV, the second is the obstacle avoidance during tracking, by using vision and IMU. Image features considered are the optical flow obtained from image processing algorithms, and using additional information provided by an embedded IMU for derotation of the flow. A non-linear proportional type controller is designed for target tracking. This controller is augmented with a repulsive action via repulsive potential fields around obstacles.

To expose this work we start, in Section II, by presenting the fundamental equations of motion for a VTOL UAV and describing the translational optical flow that is used as an input for the control law. Section III presents the control strategy for target tracking and obstacle avoidance manoeuvres. Section IV describes gain tuning and test in simulation. Section V is about image processing algorithms evaluation. Then, experimental results are presented in Section VI. Finally we conclude and discuss about futur work in section VII.

II. MODELLING

A. UAV dynamics model for control law design

The VTOL UAV is represented by a rigid body of mass m and of tensor of inertia J. To describe the motion of the UAV, two reference frames are introduced: an inertial frame $\mathcal{F}_I$ fixed on the ground and associated with the vector basis $[e_x, e_y, e_z]$ and a body-fixed frame $\mathcal{F}_B$ attached to the UAV at the center of mass and associated with the vector basis $[e_x^b, e_y^b, e_z^b]$.
A translational force F and a torque Γ are applied to the UAV. The translational force F combines thrust, lift and drag. The gravitational force can be separated from F. For a miniature VTOL UAV in quasi-stationary flight, one can reasonably assume that the aerodynamic forces are always in direction e_z^b, since the lift predominates on the other components. Let ξ, v, be the UAV position and velocity in $\mathcal{F}_I$, R the rotation matrix from $\mathcal{F}_B$ to $\mathcal{F}_I$ and Ω the angular velocity in $\mathcal{F}_B$. Then, the UAV dynamics can be written as:

$$\begin{cases} \dot{\xi} = v \\ m\dot{v} = F + mge_z, \quad F = RTe_z^b \\ \dot{R} = R\,\mathrm{sk}(\Omega) \\ J\dot{\Omega} = -\Omega \times J\Omega + \Gamma \end{cases} \tag{1}$$

with T the rotor thrust and g the gravity. sk(.) denotes the skew-symetric matrix representing cross-product.

The VTOL UAV is equipped with a minimum sensor suite, IMU which provides Ω and R, and a camera. For the rotational dynamics of the UAV, a high gain controller is used to ensure that the orientation R of the UAV converges

[1]ONERA, Dept. of Systems Control and Flight Dynamics, Toulouse, France, firstname.name at onera.fr
[2]I3S, CNRS, Sophia Antipolis, France, thamel at i3s.unice.fr

to the desired values. The resulting control problem is then simplified to:

$$\dot{\xi} = v, \quad m\dot{v} = F + mge_z \tag{2}$$

Thus in the part III, only the control of the translational dynamics (2) is considered and the direct control input, $u = F$. This common approach is used in practice and can be justified theoretically using singular perturbation theory [12].

B. Translational optical flow for visual servoing

The translational optical flow on a spherical image, under the following assumptions is presented in the sequel.

Assumption 1:

- The camera is positioned at the UAV center of mass, with its orientation aligned with $\mathcal{F}_B$ and the optical axis is e_z^b,
- Points of the environment are stationary in the inertial frame. Thus, motion of the environment points appeared on images depends only on motion of the camera,
- Surface observed is textured and plane.

The choice of using spherical image is based on the passivity-like property discussed in [13] and the fact that it is possible to convert optical flow and points position in a plane image to a spherical image [14].

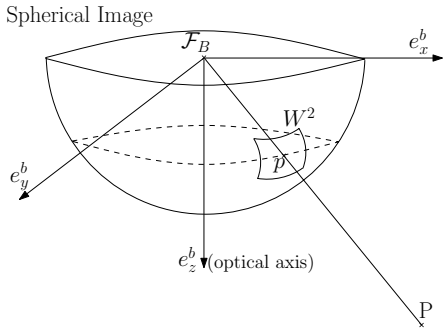


Fig. 1. Spherical Image

Let $P = (X, Y, Z)^T$ be the position of a point on the surface observed expressed in $\mathcal{F}_B$. Figure 1 shows its projection on a unit radius spherical image $f = 1$. The projection equation is given by:

$$p = \frac{P}{|P|} \tag{3}$$

where $|.|$ denotes the Euclidean norm.

Introducing the normal direction η, expressed in $\mathcal{F}_B$, to the surface, the distance to its surface can be written:

$$d =< P, \eta >= |P| < p, \eta > \tag{4}$$

where <.,.> is the inner product. As defined in [15], the translational optical flow is:

$$w = -\frac{v}{d} = RQ^{-1}(\dot{q} + \Omega \times q) \tag{5}$$

where,

$$q = \sum_{i=1}^{n} p_i \tag{6}$$

with p_i the set of points on W^2 (Fig.1), a section of the image. The square matrix Q is defined by :

$$\begin{aligned} \dot{q} &= \sum_{i=1}^{n} \dot{p}_i \\ &= -\Omega \times q - \sum_{i=1}^{n} \left(I - p_i p_i^T\right) < p_i, \eta > R^T \frac{v}{d} \\ &= -\Omega \times q - QR^T \frac{v}{d} \end{aligned} \tag{7}$$

III. TARGET TRACKING AND OBSTACLE AVOIDANCE

A. Target tracking

In this section a control design for target tracking is proposed. Let call target a point of the environment that the UAV has to join. This control law is inspired by [9]. The translational optical flow and the target position on the image are used as an input. The target is a point on the horizontal and plane ground ($R\eta = e_z$). Its coordinates expressed in $\mathcal{F}_I$ are ${}^I P$. The target error expressed in $\mathcal{F}_B$ is thus

$$P = R^T({}^I P - \xi) \tag{8}$$

Its projection on the spherical image is denoted p ans is merged with the target location within the image.

Proposition 3.1: Consider the dynamics (2) and assume that the thrust vector $u = F$ is the control input chosen as follows:

$$u = -mge_z + mk_P(w + k_I Rp), \ k_P, \ k_I > 0 \tag{9}$$

The proof that $\forall t$, $d(t) > 0$ and hence the linear velocity and the position of the UAV converges asymptotically towards zero, is done by using a Lyapunov function. The proof is not presented here but will be the object of a future article.

B. Obstacle avoidance

In this section an obstacle avoidance control design is proposed augmenting the previous target tracking control law. The control law is inspired by [3] and the potential field theory.

1) Avoiding a unique obstacle: First, consider only one obstacle. Let P_o be a point on the obstacle surface, its position on the spherical image is named p_o and B_o be the spherical influence area around the obstacle, in which it is repellant, see Figure 2.

Assumption 2: The target position and the UAV initial position are not inside B_o.

Assumption 3: Obstacle is spherical and its volume is small, so the approximation $p_o = \eta_o$ can be made, with η_o the normal direction to the obstacle. To compute the translational optical flow of the obstacle, the surface around P_o is considered locally plane.

The distance to the obstacle can be writen as follows :

$$d_o =< P_o, \eta_o >= |P_o| \tag{10}$$

As defined in (5) the translational optical flow of the obstacle, w_o, is :

$$w_o = -\frac{v}{d_o} \tag{11}$$

The relative speed to the obstacle in the normal direction can be written as:

$$\begin{aligned}\frac{\dot{d}_o}{d_o} &= \frac{1}{d_o}\left(\eta_o^T \dot{P}_o + \dot{\eta}_o^T P_o\right) \\ &= \eta_o^T R^T w_o\end{aligned} \tag{12}$$

Since $\eta_o = p_o$, the control law hereafter uses an integration of $p_o^T R^T w_o$ to compute $\ln(d_o/d_o(0))$ up to a constant.

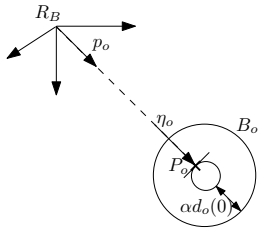


Fig. 2. Obstacle configuration

Proposition 3.2: Consider the dynamics (2) and assume that the thrust vector $u = F$ is the control input chosen as follows:

$$\begin{cases}\dot{\gamma} = p_o^T R^T w_o, \quad \gamma(0) = -\ln(\alpha) \\ u = -mge_z + mk_P(w + k_I Rp) + mk_o f(\gamma) Rp_o\end{cases} \tag{13}$$

with, $0 < \alpha < 1$, $k_P,\ k_I,\ k_o > 0$, and $f(\gamma) = \min(\gamma, 0)$.

Then, the UAV is ensured to avoid the obstacle, to get out of B_o, the repulsive sphere of radius $\alpha d_o(0)$ in a finite time if it enters in, to do never go back in B_o, and to converge asymptotically to the target if

$$k_o > \frac{k_I k_P |P(0)| + v(0)^T v(0)}{\alpha d_o(0)} \tag{14}$$

This constraint depending on the initial distance to the obstacle, UAV initial condition, and control gain of target tracking, determines how important has to be the repulsion. The proof is not presented here but will be the object of a future article.

2) Avoiding more than one obstacle: Considering more than one obstacle and calling B_{o_i} the influence sphere around the ith obstacle.

Assumption 4: B_{o_i} doesn't intersect an other influence sphere B_{o_j}, $\forall i \neq j$, otherwise they are merged in a unique repulsive sphere. B_{o_i} doesn't contain the target, $\forall i$.
As in previous section, the distance to the i^{th} obstacle can be written as follow :

$$d_{o_i} = |P_{o_i}| < p_{o_i}, \eta_{o_i} > \tag{15}$$

Using (5), the translational optical flow of the i^{th} obstacle, w_{o_i} is

$$w_{o_i} = -\frac{v}{d_{o_i}} \tag{16}$$

and the relative speed normal to the i^{th} obstacle,

$$\frac{\dot{d}_{o_i}}{d_{o_i}} = \eta_{o_i}^T R^T w_{o_i} \tag{17}$$

Proposition 3.3: Consider the dynamics (2) and assume that the thrust vector u is the control input chosen as follows

$$\begin{cases}\dot{\gamma}_i = p_{o_i}^T R^T w_{o_i}, \quad \gamma_i(0) = -\ln(\alpha_i) \\ u = -mge_z + mk_P(w + k_I Rp) + mk_{o_i} \sum_{i=1}^{n} f(\gamma_i) Rp_{o_i}\end{cases} \tag{18}$$

with, $0 < \alpha_i < 1$ and $k_P,\ k_I,\ k_{o_i} > 0$.

Then, the UAV is ensured to avoid each obstacle, to get out of each B_{o_i} in a finite time if it enters in, to do never go back in, and to converge asymptotically to the target if

$$k_{o_i} > \frac{k_I k_P |P(0)| + v(0)^T v(0)}{\alpha d_{o_i}(0)} \tag{19}$$

The proof is not presented here but will be the object of a future article.

IV. TEST IN SIMULATION

Control laws presented in part III are tested in simulation. First, the control gain are determined by considering the translational controlled dynamics for target tracking :

$$d\ddot{\xi} + k_P \dot{\xi} + k_P k_I' \xi = 0 \tag{20}$$

where $k_I' = k_I < p, \eta >$. Choosing a double pole at -0.5 and that for d^*=5m the damping ratio be 0.7. Solving the following equations,

$$\begin{cases} k_P^2 - 4dk_P k_I' = 0 \\ \dfrac{-k_P}{2d} = -0.5 \\ \dfrac{\sqrt{-k_P^2 + 4d^* k_P k_I'}}{-k_P} = -1 \end{cases} \tag{21}$$

it comes k_P=2.5 and k_I'=0.25.

Figure 3 shows the architecture of the simulator that includes in the complete UAV dynamics (1).

Control input are T, ϕ_c, θ_c and can be expressed as following:

$$\begin{aligned} T &= |u| \\ \phi_c &= \text{atan}\left(\frac{u_y}{|u_z|}\right) \\ \theta_c &= -\text{atan}\left(\frac{u_x}{|u_z|}\right) \\ \psi_c &= 0 \end{aligned} \tag{22}$$

The attitude controller on the ONERA UAV is :

$$\Gamma = k_1 \left(\begin{pmatrix} \phi_c - \phi \\ \theta_c - \theta \\ \psi_c - \psi \end{pmatrix} - k_2 \Omega \right) \tag{23}$$

with $k_1 = 3$ and $k_2 = 960$.

Figure 4 presents simulation results of UAV position and velocity for the target tracking with the initial conditions : $\xi(0) = [20; 20; -20]$, $v(0) = [0; 0; 0]$. The skid height is taken into account so simulation stops when the center of mass reach the height of 20 cm. Note that for the target tracking orientation dynamics introduce a difference between trajectory and velocity in (x, y) and in z. The maximum

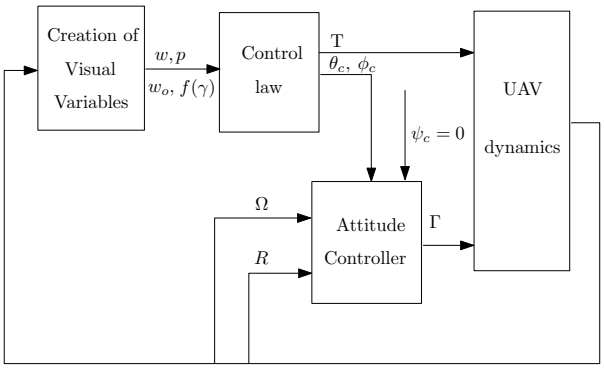


Fig. 3. UAV complete dynamics simulator

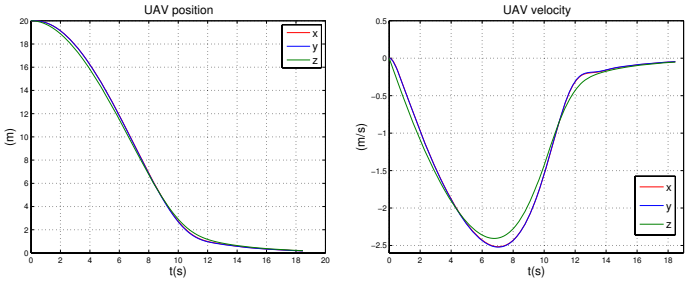


Fig. 4. UAV position and velocity

impact velocity of the UAV on the ground has to be 0.5m/s, this constraint is respected, see Figure 4 and 5.

Figure 5 is that for the obstacle avoidance with $k_o = 21$ and $\alpha = 0.5/d_o(0)$. Initial conditions are the same that for target traking, and the obstacle position in $\mathcal{F}_I$ is $P_o = [3; 3; -2.5]$. UAV 3D trajectory is shown in figure 6. The UAV reaction during the obstacle avoidance is slow due to the delay introduced by the orientation dynamics.

Figure 7 is the simulation results of target tracking with measurement noise on optical flow. The white noise introduced is based on noise estimation presented in the next part. Even with the noise the target tracking and obstacle avoidance is correctly realised.

Moreover, in the simulator the simulation stops when the target get out of the camera field of view. Because the control imposes a 3D deplacement with a constant slope and considering initial condition $v(0) = 0$, it is possible to determine the domain of acceptable UAV initial positions

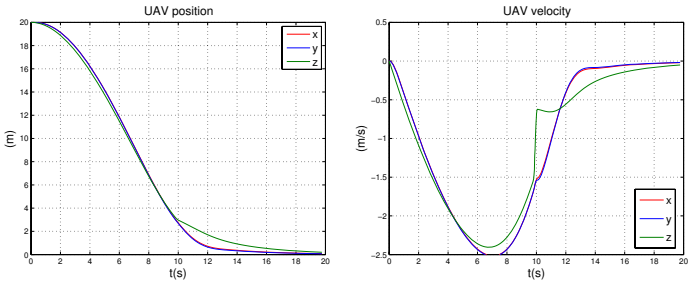


Fig. 5. UAV position and velocity in obstacle avoidance scenario

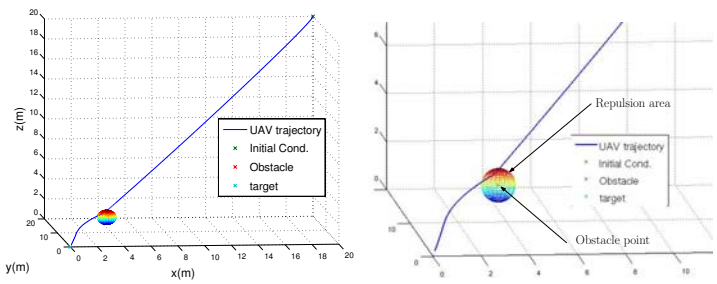


Fig. 6. UAV 3D trajectory with aobstacle avoidance and zoom on the repulsive area

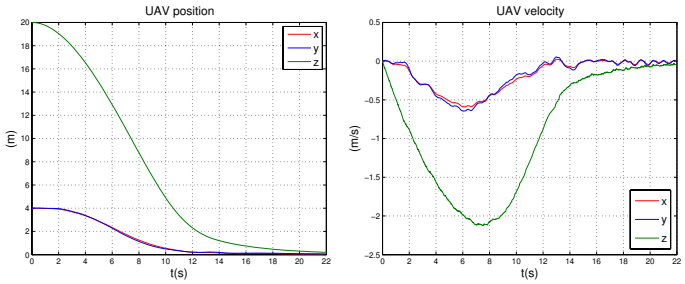


Fig. 7. Target tracking with noisy w,$v(0) = 0$, $\xi(0) = [4; 4; -20]$

for which the target stays in the camera field of view, see Figure 8.

V. IMAGE PROCESSING

This part presents the study of the image processing algorithms and their ouput. The precision of the measurements and the impact of the hypothesis previously made are studied. Image processing algorithms are runned on images of the onboard camera, recorded during flight test. The camera embedded on the ONERA VTOL UAV is a plane camera.

In the sequel are considered only image processing functions that yield to algorithms estimating affine deplacement between images. Calling p the coordinates of a point in the previous image, p' in the currant image, $A|b$ the affine deplacement matrix and H the homographic matrix. The affine deplacement can be written

$$p' = \begin{pmatrix} A & b \\ 0 & 1 \end{pmatrix} p = Hp \tag{24}$$

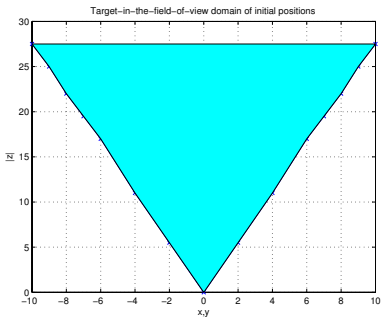


Fig. 8. Initial position from where target stays in the field of view

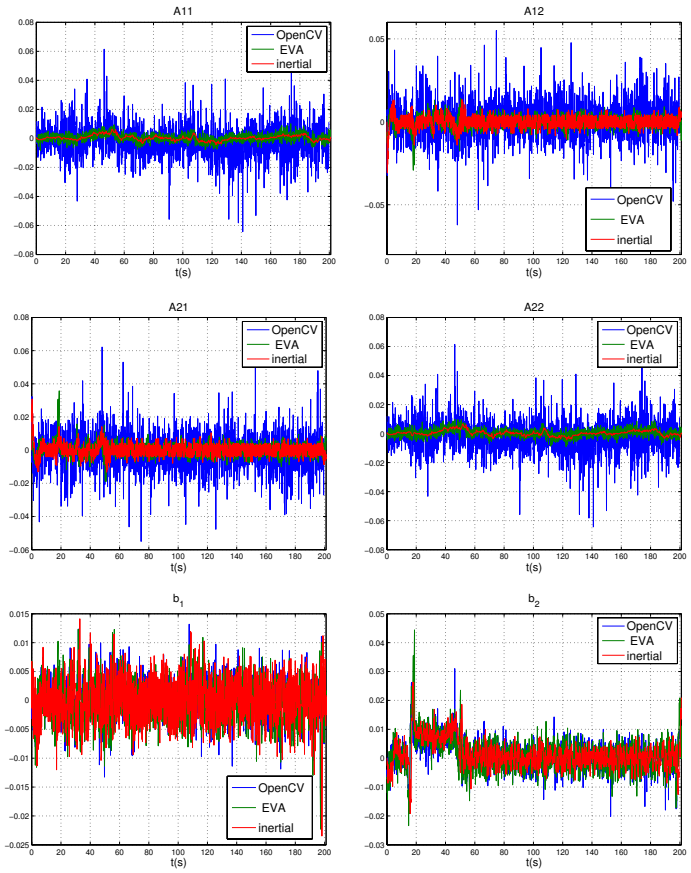


Fig. 9. $A|b$ and $A^*|b^*$

There exists a difference between real affine deplacement $A|b$ and its estimation named $A^*|b^*$. The relation between $A|b$ and $A^*|b^*$ is modeled as

$$\begin{aligned} A^* &= A + \bar{e}_A + \sigma_A \nu \\ b^* &= b + \bar{e}_b + \sigma_b \nu \end{aligned} \tag{25}$$

where, ν is a white noise, $\bar{e}_{A,b}$ and $\sigma_{A,b}$ are respectively the mean of error and the standard deviation of error on A or b.

Two image processing algorithms are compared to determine the most adapted to obtain $A^*|b^*$. The first uses OpenCV, an open source library of image processing functions. The second uses EVA a library equivalent to OpenCV developped by the ONERA/DTIM department. Both algorithms, the one based on OpenCv functions and the one based on EVA functions, provide affine deplacement between consecutive images. Steps that conduce to affine deplacement estimation are: feature point extraction from the previous and current image, matching of the two sets of points based on descriptor, estimation of the affine deplacement.

Figure 9 presents the value of $A|b$ obtained from GPS measurement, considered as true, and their estimations $A^*|b^*$ obtained with the image processing algorithm based on OpenCV functions and with the image processing algorithm based on EVA functions. These datas have been obtained from flight experimental measurements recorded during a representative trajectory of target tracking.

Algorithm	Term of Alb	Mean of error	Standard deviation
EVA	A_{xx}	-1.1543e-04	0.0022
	A_{xy}	1.4080e-04	0.0032
	A_{yx}	-2.0342e-04	0.0034
	A_{yy}	3.6142e-04	0.0038
	b_x	-0.0391	2.0141
	b_y	0.9360	2.9976
OpenCV : Harris + EstimateRigidTransform	A_{xx}	0.0025	0.0110
	A_{xy}	-7.8512e-04	0.0114
	A_{yx}	9.7957e-04	0.0120
	A_{yy}	-5.3694e-04	0.0116
	b_x	-0.0413	2.0106
	b_y	0.9741	3.0523

TABLE I
MEAN OF ERROR AND STANDARD DEVIATION

Table I shows mean and standard deviation of error of each componant of A and b. $\bar{e}_{A,b}$ and $\sigma_{A,b}$ have been calculated from datas presented on the Figure 9, in the case of OpenCV based image processing algorithm and on EVA based image processing algorithm. During this test it has been noticed that the calculation time for the entire operation chain on a 200x200 section of image is about 60ms for EVA and 20ms for OpenCV.

For flight experiment in section IV image processing algorithm based on EVA will be used because it is the more accurate of the two algorithms.

Moreover, it is possible to link $A|b$ to w writing theoretically the affine deplacement. Let Z be the distance to the ground along the optical axis and $V = (V_x, V_y, V_z)^T$ and ${}^b\Omega$ be the velocity and the angular velocity in $\mathcal{F}_B$ and considering that the center of the image section, on which the image processing algorithm is used, coincides with the center of the entire image, it comes (24) with

$$A = I + \begin{pmatrix} \frac{V_z}{Z} & -\Omega_z \\ \Omega_z & \frac{V_z}{Z} \end{pmatrix} dt \tag{26}$$

and

$$b = f \begin{pmatrix} \Omega_y - \frac{V_x}{Z} \\ -\Omega_x - \frac{V_y}{Z} \end{pmatrix} dt \tag{27}$$

Remark that because $\Omega = (\Omega_x, \Omega_y, \Omega_z)^T$ is measured, it is possible to isolate from $A|b$ the term $\frac{V}{Z}$. Using previous assumption of plane and horizontal ground it is possible to link $\frac{V}{Z}$ to w. In reality $A^*|b^*$ are estimated so it is possible to estimate w^*, that will be used in the control laws.

$$w^* = R \begin{pmatrix} \frac{1}{f.dt} b^* - \begin{pmatrix} \Omega_y \\ -\Omega_x \end{pmatrix} \\ \frac{1 - A^*_{11}}{dt} \end{pmatrix} \tag{28}$$

where A^*_{11} is the first row and first column coefficient of the A^* matrix. Because $A^*|b^*$ is noisy w^* is too. A discrete low pass filter is used to filter w^* .

VI. EXPERIMENTAL RESULTS

In order to realise the final scenario of target tracking and obstacle avoidance a plan of experiments is built. First, the work consists in testing control laws in the horizontal plane, the vertical control is managed by a control law using GPS

Fig. 10. ONERA RMAX UAV

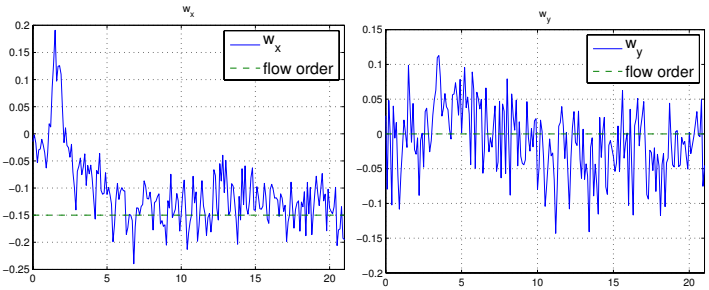


Fig. 11. Experimentation result of constant optical flow in x

datas. Then, the scenario is to move on the x axis with a constant optical flow. Second step is to run the horizontal target tracking. After that the vertical landing is tested to finally run the entire scenario of target tracking in 3D. Then, the same steps are realised with obstacle avoidance. The first step of this plan of experiments is presented in this part. For a move on the x axis with a constant optical flow the control law (9) is reduced to u_x and u_y the two first componants of the control u :

$$\begin{aligned} u_x &= mk_P(w_x + \alpha) \\ u_y &= mk_P(w_y + k_I \int w_y) \end{aligned} \tag{29}$$

where w_x, w_y are the two first componants of w and $\alpha = 0.15$ is the constant optical flow order in x. Initial conditions of the experiment are $v(0) = 0$, $\xi(0) = (0, 0, -20)$. The experiments are realized with an YAMAHA RMAX, Figure 10, on which are embedded the IMU, the camera and a PIP22 allowing the image processing and the control to be embedded. Figure 11 shows that the optical flow in x reaches its reference value. Because the UAV is at the same height during the flight and the ground is horizontal the UAV velocity reaches a constant value, see Figure 12. This experimentation shows that it is possible to control VTOL UAV in real time using optical flow, with an embedded camera and image processing system. Other parts of the plan of experiments are in progress so they are not presented here.

VII. Conclusion

This paper presented a nonlinear controller for obstacle avoidance and target tracking of a VTOL UAV using the measurement of optical flow along with the IMU data. The proposed control algorithm has been tested in simulation in different scenarios to demonstrate the performance of the closed loop system. And a first experimental result shown

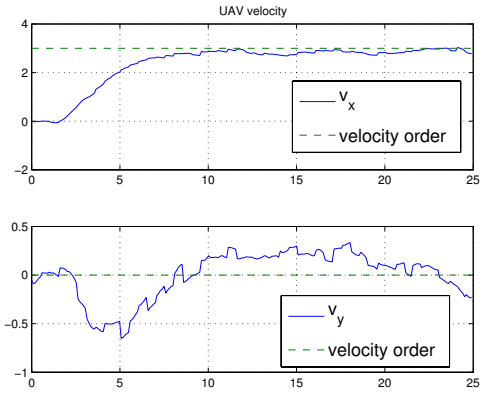


Fig. 12. UAV velocity during first experiment

that it was possible to achieve a closed-loop flight. As a futur work, we would like to complete experimental results with the entire scenario of target tracking and obstacle avoidance.

References

[1] J. Santos-Victor and G. Sandini, "Visual behaviors for docking," *Computer Vision and image understanding*, vol. vol 67, pp. pp 223–238, September 1997.

[2] J. Vanualailai, B. Sharma, and S. ichi Nakagiri, "An asymptotically stable collision-avoidance system," *International Journal of Non-Linear Mechanics*, vol. 43, pp. 925–932, 2008.

[3] E. P. Jiangmin Chunyu, Zhihua Qu and M. Falash, "A new reactive target-tracking control with obstacle avoidance in a dynamic environment," in *American Control Conference*, 2009.

[4] N. Simond, "Obstacle Detection from IPM and Super-Homography," in *IEEE IROS 2007 / International Conference on Intelligent Robots and Systems*, (San Diego, California / USA United States), IEEE Intelligent Robots and Systems, 10 2007.

[5] J. Santos-Victor and G. Sandini, "Visual-based obstacle detection. a purposive approach using the normal flow," *Intelligent Autonomous Systems*, 1995.

[6] J. S. Humbert and A. M. Hyslop, "Bioinspired visuomotor convergence," *Trans. Rob.*, vol. 26, no. 1, pp. 121–130, 2010.

[7] J.-C. Zufferey, *BIO-INSPIRED VISION-BASED FLYING ROBOTS*. PhD thesis, EPFL, 2005.

[8] F. Ruffier and N. Franceschini, "Octave, a bioinspired visuo-motor control system for the guidance of micro-air vehicles," in *Conference on Bioengineered and Bioinspired Systems*, 2003.

[9] B. Herisse, F.-X. Russotto, T. Hamel, and R. Mahony, "Hovering flight and vertical landing control of a vtol unmanned aerial vehicule using optical flow," in *IEEE/RSJ Internationnal Conference on Intelligent Robots and Systems*, 2008.

[10] B. Herisse, T. Hamel, R. Mahony, and F.-X. Russotto, "Landing a VTOL Unmanned Aerial Vehicle on a moving platform using optical flow," *Automatica*, 2010.

[11] B. Herisse, T. Hamel, R. Mahony, and F.-X. Russotto, "A nonlinear terrain following controller for a vtol unmanned aerial vehicule using translationnal optical flow," in *ICRA, Kobe, Japan*, 2009.

[12] H. K. Khalil, *Nonlinear Systems*. Prentice Hall, 2002.

[13] T. Hamel and R. Mahony, "Visual servoing of an under actuated dynamic rigid-body system: An image-based approach," *IEEE Transactions on robotics and automation*, vol. vol 18, pp. pp 187–198, April 2002.

[14] R. F. Vassallo, J. Santos-victor, and H. J. Schneebeli, "A general approach for egomotion estimation with omnidirectional images," *3rd Workshop on Omnidirectional Vision*, pp. 97–103, 2002.

[15] R. Mahony, P. Corke, and T. Hamel, "Dynamic image-based visual servo control using centroid and optic flow features," *Journal of Dynamic Systems, Measurement and Control*, vol. vol 130, pp. pp 011005–1–011005–12, January 2008.

IROS Workshop on Visual Control of Mobile Robots (ViCoMoR 2012)
October 11th, 2012, Vilamoura, Algarve, Portugal

Homography based visual odometry with known vertical direction and weak Manhattan world assumption

Olivier Saurer, Friedrich Fraundorfer, Marc Pollefeys
Computer Vision and Geometry Lab, ETH Zürich, Switzerland

***Abstract*— In this paper we present a novel visual odometry pipeline, that exploits the weak Manhattan world assumption and known vertical direction. A novel 2pt and 2.5pt method for computing the essential matrix under the Manhattan world assumption and known vertical direction is presented that improves the efficiency of relative motion estimation in the visual odometry pipeline. Similarly an efficient 2pt algorithm for absolute camera pose estimation from 3D-2D correspondences is presented that speeds up the visual odometry pipeline as well. We show that the weak Manhattan world assumption and known vertical allow for direct relative scale estimation, without recovering the 3D structure. We evaluate our algorithms on synthetic data and show their application on real data sets from camera phones and robotic micro aerial vehicles. Our experiments show that the weak Manhattan world assumption holds for many real-world scenarios.**

I. Introduction

The Manhattan world assumption is a very strong restriction to a general 3D scene. And yet this assumption is fulfilled for many scenes that contain man-made architectural structures, at least partially. The assumption especially holds true for indoor environments, and also for urban canyons of modern cities. This was successfully demonstrated and exploited in a variety of recent papers [1], [4], [5]. In this work we will refer to the weak Manhattan world, describing a world consisting of vertical planes which are arbitrary oriented around the vertical direction. They are not required to be orthogonal to each other. The only restriction is, that vertical planes are parallel to the gravity vector and the ground planes are orthogonal to the vertical direction.

Especially visual odometry [16] can benefit at a high degree from the Manhattan world assumption. Visual odometry is the means of ego motion estimation of e.g. mobile robots fitted with cameras. One computational bottleneck of visual odometry is the robust motion estimation using RANSAC [2]. The computational complexity of RANSAC depends exponentially on the number of data points needed for hypothesis generation, for unconstrained motion in 3D this would mean 5 data points (feature correspondences) using the 5pt essential matrix algorithm [14]. Constraints on the robot motion (e.g. planar motion), on the environment or using additional sensor data however can reduce the number of necessary data points and make RANSAC more efficient, which is important to achieve real time performance. For the case of a planar motion assumption, which is true for many mobile robot applications, two point correspondences are sufficient to compute an egomotion hypothesis for RANSAC [15].

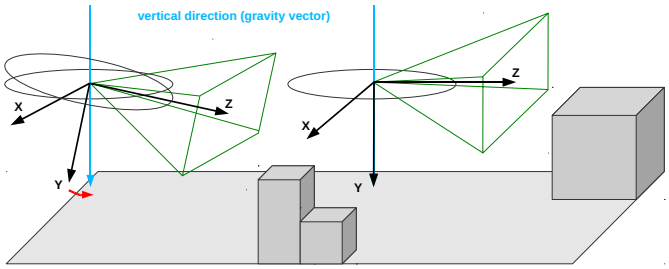


Fig. 1. Knowing the vertical direction, e.g. by measuring the gravity vector with an IMU or from vanishing points, the image can be rotated such that the y-axis of the camera matches the vertical direction. Under the weak Manhattan world assumption this aligns the x-z-plane of the camera with the ground plane and the y-axis with the vertical direction of walls.

In this work we will present two novel relative pose algorithms and a novel absolute pose algorithm which exploits the weak Manhattan world assumption and additionally takes advantage of the knowledge of the vertical direction of the scene structure in the images. The known vertical direction and the assumptions about the environment lead to a simpler formulation for relative 5DOF camera motion, in particular to a 2pt algorithm and a 2.5pt algorithm, in contrast to the standard 5pt method. For successive 6DOF camera pose estimation from 3D-2D matches we propose a new 2pt method, exploiting the known vertical direction.

The vertical direction can be computed from image features, but also from an inertial measurement unit (IMU) (which e.g. measures the earth's gravity vector) attached to a camera. This is the case for almost every state-of-the-art smart phone (e.g. IPhone, Google Nexus S, Nokia N900..) which are all equipped with a camera and an IMU. We conduct synthetic experiments to evaluate the proposed algorithms under different image noise and IMU measurement noise and compare the results to the standard 5-pt relative pose and the 3-pt relative pose with known vertical direction algorithm. We further demonstrate the proposed algorithms on real data from camera phones (which come with IMU sensors) and show visual odometry results for a robotic micro aerial vehicle. The experiments show that the weak Manhattan world assumption holds and can be exploited in real-world scenarios.

II. Related work

Early results on coupling inertial sensors (IMU) and cameras and using the measured gravity normal for ego-motion

estimation have been discussed in [17] or [11]. Most closely related to our paper are the works of [3], [7]. In [3] the authors present an algorithm to compute the essential matrix from 3pt correspondences and a known gravity vector, e.g. from an IMU measurement. This 3pt formulation can speed up RANSAC significantly compared to the standard 5pt algorithm. In [7] vision and IMU information is combined in a similar way to find new algorithms for absolute pose estimation. More recently [8] proposed to combine IMU and vision data from monocular camera to incrementally accumulate the motion and reconstruct the camera trajectory. The incremental approach requires integration of the IMU data over time and is brittle towards IMU inaccuracy.

The Manhattan world assumption [1] has recently been picked up again and successfully been used for multi-view stereo [5], the reconstruction of building interiors [4] and also for scene reconstruction from a single image only [9]. In this paper we combine both ideas, the IMU-vision fusion and a weak Manhattan world assumption. Similar to [3], [7] but now for the case of homographies we derive a formulation for relative motion under weak Manhattan world constraints and a formulation for absolute pose. A similar idea for egomotion estimation has also been described in [12] where the authors derive a homography for vertical walls under a planar motion constraint. However, in our formulation the relative motion is not restricted to motion on a plane.

III. Relative and absolute pose estimation

Knowing the vertical direction in images will simplify the estimation of camera pose and camera motion, which are fundamental methods in any odometry pipeline. It is then possible to align every camera coordinate system with the measured vertical direction such that the y-axis of the camera is parallel to the vertical direction and the x-z-plane of the camera is orthogonal to the vertical direction (illustrated in Fig. 1). Under the Manhattan world assumption this means that the x-z-plane of the camera is now parallel to the world's ground plane and the y-axis is parallel to vertical walls. This alignment can just be done as a coordinate transform for motion estimation algorithms, but also be implemented as image warping such that feature extraction method benefit from it. Relative motion between two such aligned cameras reduces to a 3-DOF motion, which consists of 1 remaining rotation and a 2-DOF translation vector. The absolute camera pose for aligned camera has 4-DOF, again 1 remaining rotation and a 3-DOF translation vector.

The algorithms for estimating the relative pose are derived from computing a homography between a plane in two images and decomposing it. After incorporating the Manhattan world constraint, which restricts the possible planes to vertical and horizontal ones, and after incorporating the vertical direction, which decreases the DOF of the camera orientation, the parameterization of a homography is greatly simplified. This simplification leads to a 2pt and a 2.5pt algorithm for computing homographies and closed form solutions for the decomposition. The homography is represented by

$$\mathbf{H} = \mathbf{R} + \frac{1}{d}\mathbf{t}^T\mathbf{n}, \tag{1}$$

where $\mathbf{R} = \mathbf{R_y}\mathbf{R_x}\mathbf{R_z}$ is a rotation matrix representing the relative camera rotations around the x, y, and z-axis, $\mathbf{t} = [t_x, t_y, t_z]^T$ represents the relative motion, $\mathbf{n} = [n_x, n_y, n_z]^T$ is the plane normal and d is the distance from the first camera center to the plane. In all our derivations, the camera-plane distance is set to 1 and absorbed by $\mathbf{t}$. With the knowledge of the vertical direction the rotation matrix R can be simplified such that $\mathbf{R} = \mathbf{R_y}$ by pre-rotating the feature points with $\mathbf{R_x}\mathbf{R_z}$, which can be measured from the IMU or vanishing points. Under the weak Manhattan world assumption additionally the parameterization of the plane normal $\mathbf{n}$ can be simplified, to be only vertical or horizontal planes.

A. 2pt relative pose for known plane normal

The following algorithm is able to compute the relative pose given 2pt correspondences and the normal of the plane on which the points reside. The derivation will be carried out for a vertical plane but works similar for planes parameterized around other axis.

The homography for a vertical plane can be written as

$$\mathbf{H} = \mathbf{R_y} + [t_x, t_y, t_z]^T[n_x, 0, n_z] \tag{2}$$

where the normal vector is parametrized by $n_x = sin(\phi_n)$ and $n_z = cos(\phi_n)$. The homography then writes as

$$\mathbf{H} = \begin{bmatrix} \cos(\phi_y) + n_x tx & 0 & \sin(\phi_y) + n_z t_x \\ n_x t_y & 1 & n_z t_y \\ n_x t_z - \sin(\phi_y) & 0 & \cos(\phi_y) + n_z t_z \end{bmatrix} \tag{3}$$

$$= \begin{bmatrix} h_{11} & 0 & h_{13} \\ h_{21} & 1 & h_{23} \\ h_{31} & 0 & h_{33} \end{bmatrix} \tag{4}$$

To solve for the 6 entries of $\mathbf{H}$ we solve $\mathbf{x}' \times \mathbf{H}\mathbf{x} = \mathbf{0}$, where $\mathbf{x} = [x\ y\ 1]^T$ and $\mathbf{x}' = [x'\ y'\ 1]^T$ are the point correspondences. By using this relation we get rid of the unknown scaling factor of the homography. Knowing n_x and n_y leads to one additional linear constraint in the entries of the homography, $h_{23} = n_x/n_z h_{21}$.

This leaves 5 entries in $\mathbf{H}$ to be estimated. Each point correspondences gives 2 inhomogeneous linearly independent equations of the form $\mathbf{A}\mathbf{h} = \mathbf{b}$,

$$\begin{bmatrix} 0 & 0 & -x - \frac{n_x}{n_z} & xy' & y' \\ -xy' & -y' & xx' + \frac{n_x}{n_z}x' & 0 & 0 \end{bmatrix} \mathbf{h} = \begin{bmatrix} y \\ -x'y \end{bmatrix} \tag{5}$$

where $\mathbf{h} = [h_{11}\ h_{13}\ h_{21}\ h_{31}\ h_{33}]^T$.

Using 2 point correspondences this gives 4 equations which is a deficient-rank system. The solution is $\mathbf{h} = \mathbf{V}\mathbf{y} + w\mathbf{v}$ (see [6]) where $svd(\mathbf{A}) = \mathbf{U}\mathbf{D}\mathbf{V}^T$ and $\mathbf{v}$ is the last column vector of $\mathbf{V}$. The vector $\mathbf{y}$ is computed by $y_i = b_i/d_i$ where d_i is the $i-th$ diagonal entry of $\mathbf{D}$ and $\mathbf{b}' = \mathbf{U^T}\mathbf{b}$.

This leaves the unknown scalar w which can be computed from the additional constraint, that one of the singular values of the homograph has to be one (i.e., $\det(\mathbf{H}^T\mathbf{H} - \mathbf{I}) = 0$, see [13]). By substituting $\mathbf{h} = \mathbf{V}\mathbf{y} + w\mathbf{v}$ for the entries of $\mathbf{H}$. The determinant is a 4th order polynomial in w which results in 4 solutions for $\mathbf{H}$.

If the plane normal is not known one can sample the ratio n_x/n_z. Each sample represents a hypothesis in the RANSAC loop that are then tested against the other points. Having multiple hypothesis is better than computing the orientation with one additional point sample since this has only a linear instead of an exponential impact on the RANSAC iterations. Knowledge about the approximate orientation of the wall relative to the camera will reduce the number of hypothesis, for example the case when the camera is moving along a corridor it is not always necessary to sample the full $360\,\mathrm{deg}$ for the side walls.

The such parameterized homography can easily be decomposed in the rotation and translation parameters of the relative motion. First step is a proper normalization of the up to scale homography, by dividing it by h_{22}. h_{22} needs to be 1 according to Eq. 3. Using the relation $n_x^2 + n_z^2 = 1$, t_y can be obtained from h_{21} and h_{23} by $t_y = \pm(h_{21}^2 + h_{23}^2)^{\frac{1}{2}}$ which gives two solutions for t_y which differ in the sign. The normal can then be computed as follows that $n_x = h_{21}/t_y$ and $n_z = h_{23}/t_y$. Two pairs of the normal have to be computed for the two t_y. Next we can compute $\sin(\phi_y)$ from a quadratic in the entries $h_{11}, h_{13}, h_{21}, h_{23}$ and the $\cos(\phi_y)$ is obtained from the relation $\sin(\phi_y)^2 + \cos(\phi_y)^2 = 1$. Finally t_x and t_z are obtained as $t_x = (h_{11} - \cos(\phi_y))/n_x$ and $t_z = (h_{33} - \cos(\phi_y))/n_z$.

B. 2pt relative pose for ground plane

This derivation is a special case of the previous one and will work for points on the ground plane. The normal of the ground plane is $\mathbf{n} = [0\ 1\ 0]^T$. This leads to an again simpler formulation for the homography and only 2 point correspondences are enough to compute the full homography. The homography for the ground plane can be written as: $\mathbf{H} = \mathbf{R}_\mathbf{y} + [t_x, t_y, t_z]^T[0, 1, 0]$

The entries of $\mathbf{H}$ are then

$$\mathbf{H} = \begin{bmatrix} \cos(\phi_y) & t_x & \sin(\phi_y) \\ 0 & t_y + 1 & 0 \\ -\sin(\phi_y) & t_z & \cos(\phi_y) \end{bmatrix} \quad (6)$$

$$= \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ 0 & h_{22} & 0 \\ -h_{13} & h_{32} & h_{11} \end{bmatrix} \quad (7)$$

Because of 2 linear constraints in the entries of H, H has only 5 unknowns, for which can linearly be solved using 2 point correspondences. Each point gives 2 constraints of the form $\mathbf{A}\mathbf{h} = \mathbf{0}$,

$$\begin{bmatrix} y' & 0 & -xy' & -y & yy' \\ -xy' & -yy' & -y' & x'y & 0 \end{bmatrix} \mathbf{h} = \mathbf{0} \quad (8)$$

where $\mathbf{h} = [h_{11}\ h_{12}\ h_{13}\ h_{22}\ h_{32}]^T$.

The rotation and translation parameters of the relative motion can be read off the homography matrix directly, after proper normalization. Inspection of $\mathbf{H}$ shows that the following relation $h_{11}^2 + h_{13}^2 = 1$ has to be fulfilled. The proper normalization is by dividing $\mathbf{H}$ by $(h_{11}^2 + h_{13}^2)^{\frac{1}{2}}$. The rotation matrix R and the translation vector $\mathbf{t}$ are then:

$$\mathbf{t} = [h_{12},\ h_{22} - 1,\ h_{32}]^T, \quad \mathbf{R} = \begin{bmatrix} h_{11} & 0 & h_{13} \\ 0 & 1 & 0 \\ -h_{13} & 0 & h_{11} \end{bmatrix} \quad (9)$$

C. 2.5pt relative pose with unknown plane normal

The 2.5pt algorithm is an extension of the 2pt described in section III-A. The homography is designed as in Eq 2. However, when the plane normal $\mathbf{n}$ is not known we can't make use of the same linear constraint, thus all the 6 parameters of $\mathbf{H}$ have to be estimated. To do this, one more equation is needed which can be taken from a third point. Thus we stack the constraint equations of 2 points and 1 of the equations from a third point into an equation system of the form $\mathbf{A}\mathbf{h} = \mathbf{b}$. The two equations from one point are as follows:

$$\begin{bmatrix} 0 & 0 & -x & -1 & xy' & y' \\ -xy' & -y' & xx' & x' & 0 & 0 \end{bmatrix} \mathbf{h} = \begin{bmatrix} y \\ -x'y \end{bmatrix} \quad (10)$$

where $\mathbf{h} = [h_{11}\ h_{13}\ h_{21}\ h_{23}\ h_{31}\ h_{33}]^T$.

As in section III-A the solution to this system is of the form $\mathbf{h} = \mathbf{V}\mathbf{y} + w\mathbf{v}$. The unknown scalar w can again be computed from the additional homography constraint $det(\mathbf{H}^T\mathbf{H} - \mathbf{I}) = 0$ (see [13]). The determinant is a 4th order polynomial in w which results in 4 solutions for $\mathbf{H}$.

The interesting fact in this case is that we used only 1 equation of the 2 available ones for computing the homography. While in a RANSAC loop it is however necessary to sample 3 points for this method, it is now possible to do a consistency check on the 3 point correspondences. To be an outlier free sample the one remaining equation has also to be fulfilled by the estimated homography. This can easily be tested and if it is not fulfilled the hypothesis is discarded. This gives a computational advantage over the standard 3pt essential matrix method [3], because inconsistent samples can be detected without testing on all the other point correspondences.

D. 2pt absolute pose

With known vertical the absolute pose problem gets simplified as well and it is possible to compute the remaining 4DOF absolute pose from 2 3D-2D point correspondences. Here we again assume that the camera is pre-rotated by the vertical direction so that the x-z plane is parallel to the ground plane. The camera matrix is defined as $\mathbf{P} = [\mathbf{R}\ \mathbf{t}]$ which results in

$$\mathbf{P} = \begin{bmatrix} cos(\phi_y) & 0 & sin(\phi_y) & t_x \\ 0 & 1 & 0 & t_y \\ -sin(\phi_y) & 0 & cos(\phi_y) & t_z \end{bmatrix}. \quad (11)$$

There are 4 unknowns which are the rotation angle around the y-axis and one 3D translation vector. Using the relation $\mathbf{x} \times \mathbf{PX} = \mathbf{0}$ we can solve for these unknowns, where $\mathbf{X}$ is a homogeneous 3D point of the form $\mathbf{X} = [X\ Y\ Z\ 1]$ and $\mathbf{x}$ is an image point $\mathbf{x} = [x\ y\ 1]$. One 3D-2D correspondence gives 2 linearly independent equations of the form

$$\begin{bmatrix} yZ & -yX & 0 & -1 & y \\ -yX & -yZ & -y & x & 0 \end{bmatrix} \mathbf{p} = \begin{bmatrix} Y \\ -xY \end{bmatrix} \quad (12)$$

where $\mathbf{p} = [cos(\phi_y)\ sin(\phi_y)\ t_x\ t_y\ t_z]^T$. 2 point correspondences give an equation system with 4 equations. Variable elimination can be used to find expressions for t_x, t_y, t_z and eliminate it from the remaining 4th equation. The remaining equation is in $cos(\phi_y)$ and $sin(\phi_y)$ only and the additional constraint $cos(\phi_y)^2 + sin(\phi_y)^2 = 1$ can be used to get an expression in $sin(\phi_y)$. It is quadratic in $sin(\phi_y)$ and can be solved in closed form. Then the other parameters can be found by back-substitution, which leads to 2 solutions for the camera pose. A similar approach has been described in [7], however for our derivation we assume pre-aligned feature correspondences, while in [7] the measured angles of the IMU are included in the equation system. Furthermore our angles are parameterized in sin and cos while in [7] a representation with tan is used.

IV. Relative scale estimation without 3D structure recovery

The formulation of the homography induced by the ground plane Eq. 1 encodes the inverse distance d of the first camera to the plane. Assuming the ground plane is visible over all views, the relative scale can be propagated between views over the plane. The computation of the homography assumes the distance to the ground plane to be 1, as formulated in Eq. 1. The actual distance between the first camera and the ground plane ise encoded in the translation vector of the camera (i.e., y-component of the camera). This distance can then be used to rescale the relative translation vector of the second camera. The implicit encoding of the scale in the homography allows for direct scale estimation without the need of a computationally expensive recovery of the 3D structure.

V. Experiments

We evaluate the accuracy of the presented algorithms on synthetic data under different image noise and IMU noise. We compare the results to the general 5pt algorithm presented in [14] and the general 3pt algorithm proposed by [3] with two know orientation angles. Finally we demonstrate our algorithms on our own real world datasets.

A. Synthetic evaluation

To evaluate the algorithm on synthetic data we chose the following setup. The average distance of the scene to the first camera center is set to 1. The scene consists of two planes, one ground plane and one vertical plane which is parallel to the image plane of the first camera. Both planes consist of

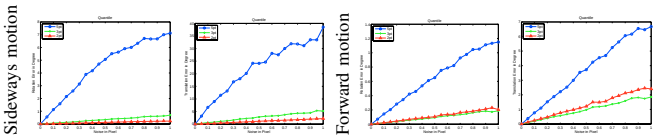


Fig. 2. Evaluation of the 2 point algorithm under different image noise.

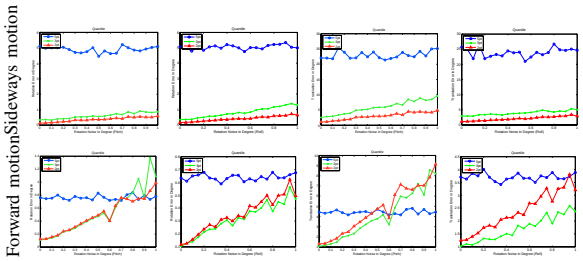


Fig. 3. Evaluation of the 2pt algorithm under different IMU noise and constant image noise with 0.5 pixel standard deviation. (First row sideways motion, second row forward motion.

200 randomly sampled points. The base-line between two cameras is set to be 0.2, i.e., 20% of the average scene distance, and the focal length is set to 1000 pixel, with a field of view of 45 degrees.
Each algorithm is evaluated under varying image noise and increasing IMU noise. Each of the two setups is evaluated under a forward and sideways translation of the second camera. For each configuration we randomly sample 100 cameras.

a) Relative pose:: Fig. 2 and Fig. 3 compare the 2-point algorithm to the general 5-point [14] and 3-point algorithms [3]. Notice, in this experiments the camera poses were computed from points randomly drawn from the ground plane. Since camera poses estimated from coplanar points do not provide a unique solution for the 5pt and 3pt algorithm we evaluate each hypothesis with all points coming from both planes. The solution providing the smallest reprojected error is chosen to be the correct one. This evaluation is used in all our synthetic experiments. Similarly Fig. 4 and Fig. 5 show a comparison of the 2.5pt algorithm with the general 5pt and 3pt algorithm. Here the camera poses are computed from points randomly sampled from the vertical plane only. The evaluation shows that knowing the vertical direction and exploiting the planarity of the scene improves motion estimation. The 2pt and 2.5pt algorithms outperform the 5pt algorithm, in terms of accuracy. Under perfect IMU measurements the algorithms are robust to image noise and perform significantly better than the 5pt algorithm. With increasing IMU noise their performance are still comparable to the 5pt algorithm.

b) Absolute pose:: We compare the presented 2pt absolute pose algorithm to the closed form solution proposed in [10]. We evaluate the algorithm on different noise in the image plane and noise in the roll and pitch measurements of the IMU. The results are shown in Fig. 6 and Fig. 7. With increasing image noise the absolute 2pt algorithm

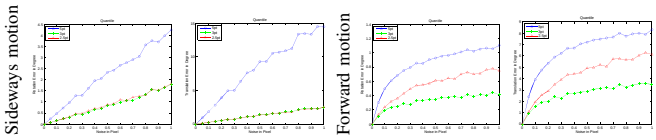


Fig. 4. Evaluation of the 2.5pt algorithm under forward and sideways motion under varying image noise.

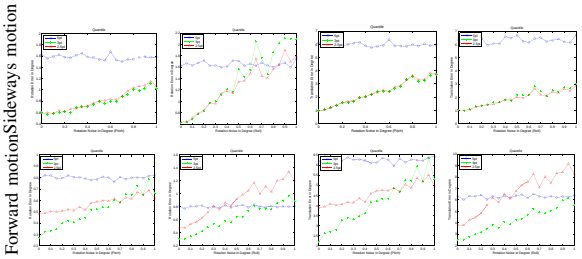


Fig. 5. Evaluation of the 2.5pt algorithm under IMU noise and constant image noise with 0.5 pixel standard deviation. (First row sideways translation, second row forward translation.

outperforms the 4pt algorithm. While with increasing IMU noise their approach has a higher accuracy.

B. Algorithm evaluation on real data

c) Plane detection:: We evaluate our relative motion algorithms on an image pair that contains multiple planes and further demonstrate that the weak Manhattan world assumption holds on real images, where vertical structures might not be perfectly vertical due to construction or due to IMU inaccuracies. Relative motion is computed with the 2pt algorithm as well as with the 2.5pt algorithm. The 2pt algorithm computes the relative motion from matches found on the ground, while the 2.5pt algorithm computes relative motion from matches found on the wall. Fig. 8 shows the inlier sets of the two methods in one image. The motion estimate from the homographies can be refined by constructing the essential matrix of it and finding inliers to the essential matrix (these need not be on planes) and computing a least squares estimate of the essential matrix. Furthermore, the detected inlier sets can be used for plane detection.

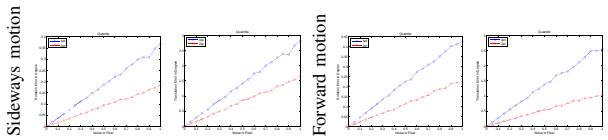


Fig. 6. Evaluation of the 2pt absolute pose algorithm under forward and sideways motion with varying image noise.

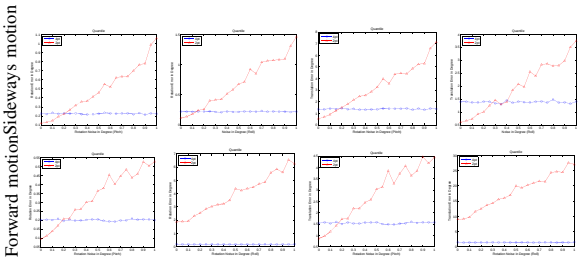


Fig. 7. Evaluation of the 2pt absolute pose algorithm under different IMU noise and image noise of 0.5 pixel standard deviation. (First row, sideways translation, second row forward translation.

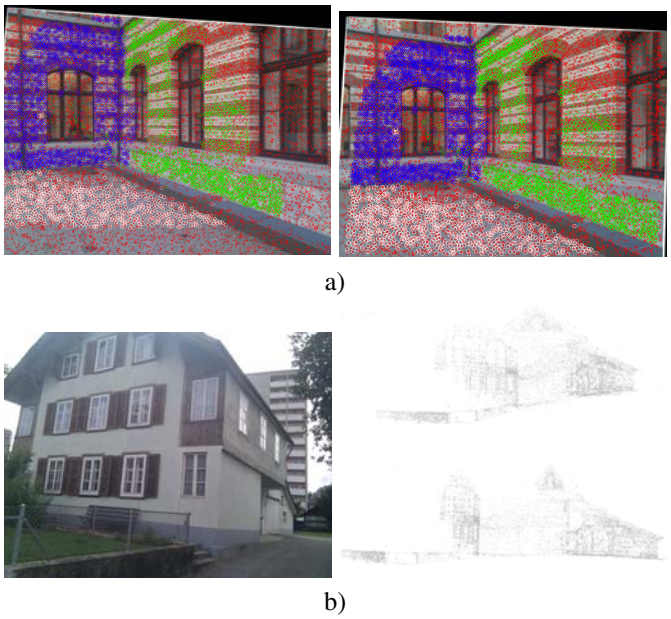


Fig. 8. a) Detected planes using the 2pt and 2.5 algorithm. b) Sample input image and two synthetic views of the reconstructed scene using the absolute 2pt algorithm.

d) Visual Odometry:: We evaluate the 2pt relative pose on our own dataset, recorded with an IMU-camera rig of a micro aerial vehicle. Sample images are shown in Fig. 8a). The experiment consists of 114 images showing a forward motion towards the front wall followed by a backwards motion to the starting point. First, we extract SIFT features and match them to neighboring views. Then, we compute the relative pose between two consecutive views using the 2pt algorithm in a RANSAC scheme. Finally, the solution with most inliers is used to form the camera trajectory by concatenating neighboring poses. Fig. 9 compares the raw odometry obtained from the 2pt algorithm without scaling and without refinement to the odometry obtained after non-linear refinement and proper scaling with the method described in section IV.

e) 2pt absolute pose:: We integrate the 2pt absolute pose algorithm into a standard SfM pipeline and show results of a reconstructed scene in Fig. 8b). The 2pt absolute pose algorithm is used to find a first inlier set from the 3d - 2d correspondences. The full camera pose and 3d points are then refined using bundle adjustment. The dataset was recorded using a Nexus One smartphone, equipped with IMU and camera.

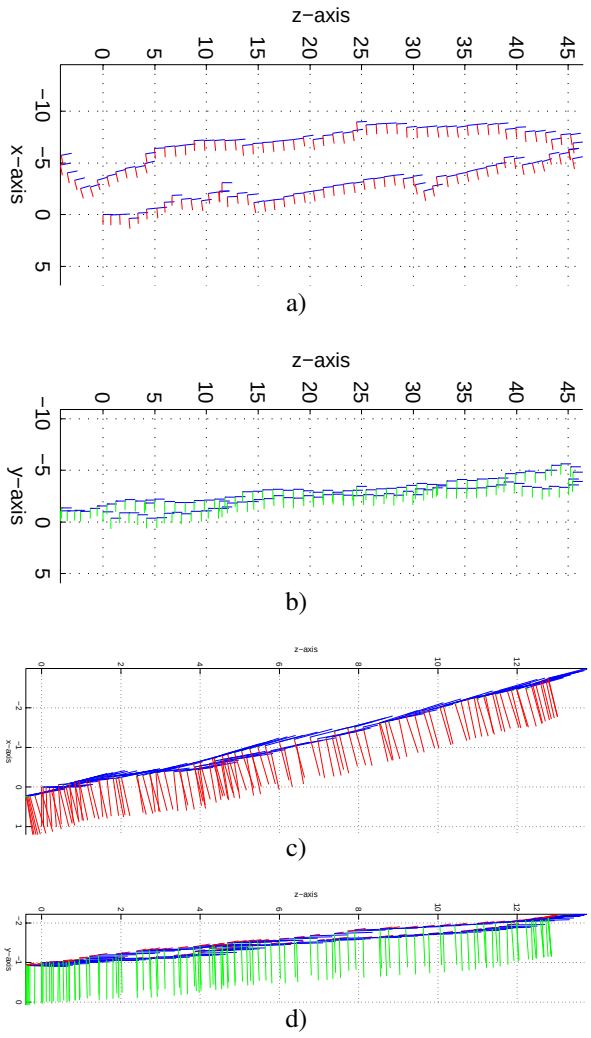


Fig. 9. (a,c) Top view of the camera trajectory. (b,d) Side view of the trajectory. (a,b) Non-refined trajectory obtained from the 2pt algorithm. (c,d) Optimized and properly scaled camera trajectory. (z-axis, motion direction, y-axis gravity direction, blue line optical axis).

VI. Conclusion

In this paper we presented an odometry pipeline that exploits the weak Manhattan world assumption and takes advantage of knowing the vertical direction in images. Our results show that the weak Manhattan assumption holds for real-world scenarios and can be used to derive efficient algorithms for relative motion (2pt, 2.5pt). Furthermore our results confirm that the vertical direction measured from an off-the-shelf IMUs are accurate enough to be used for relative motion estimation and absolute pose estimation (2pt).

References

[1] J.M. Coughlan and A.L. Yuille. Manhattan world: Compass direction from a single image by bayesian inference. In *Proc. 7th International Conference on Computer Vision, Kerkyra, Greece*, pages 941–947, 1999.

[2] M. A. Fischler and R. C. Bolles. RANSAC random sampling concensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of ACM*, 26:381–395, 1981.

[3] F. Fraundorfer, P. Tanskanen, and M. Pollefeys. A minimal case solution to the calibrated relative pose problem for the case of two known orientation angles. In *Proc. 11th European Conference on Computer Vision*, pages IV: 269–282, 2010.

[4] Y. Furukawa, B. Curless, S.M. Seitz, and R. Szeliski. Reconstructing building interiors from images. In *Proc. 12th International Conference on Computer Vision*, pages 80–87, 2009.

[5] Y. Furukawa, B. Curless, S.M. Seitz, and R.S. Szeliski. Manhattan-world stereo. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition, Florida, Miami*, pages 1422–1429, 2009.

[6] R.I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge, 2000.

[7] Z. Kukelova, M. Bujnak, and T. Pajdla. Closed-form solutions to the minimal absolute pose problems with known vertical direction. In *ACCV*, 2010.

[8] Margarita Chli Laurent Kneip and Roland Siegwart. Robust Real-Time Visual Odometry with a Single Camera and an IMU. *Proceedings of the British Machine Vision Conference*, pages 16.1–16.11, 2011. http://dx.doi.org/10.5244/C.25.16.

[9] D.C. Lee, M. Hebert, and T. Kanade. Geometric reasoning for single image structure recovery. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition, Florida, Miami*, pages 2136–2143, 2009.

[10] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Epnp: An accurate o(n) solution to the pnp problem. *Int. J. Comput. Vision*, 81(2):155–166, feb 2009.

[11] J. Lobo and J. Dias. Vision and inertial sensor cooperation using gravity as a vertical reference. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(12):1597–1608, December 2003.

[12] G. López-Nicolás, J.J. Guerrero, and C. Sagüés. Multiple homographies with omnidirectional vision for robot homing. *Robotics and Autonomous Systems*, 58(6):773 – 783, 2010.

[13] Ezio Malis and Manuel Vargas. Deeper understanding of the homography decomposition for vision-based control. Research Report RR-6303, INRIA, 2007.

[14] D. Nistér. An efficient solution to the five-point relative pose problem. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition, Madison, Wisconsin*, pages II: 195–202, 2003.

[15] D. Ortín and J. M. M. Montiel. Indoor robot motion based on monocular images. *Robotica*, 19(3):331–342, 2001.

[16] Davide Scaramuzza and Friedrich Fraundorfer. Visual odometry [tutorial] part1: The first 30 years and fundamentals. *Robotics Automation Magazine, IEEE*, 18(4):80 –92, dec. 2011.

[17] T. Vieville, E. Clergue, and P.E.D. Facao. Computation of ego-motion and structure from visual an inertial sensor using the vertical cue. In *Proc. 4th International Conference on Computer Vision, Berlin, Germany*, pages 591–598, 1993.

IROS Workshop on Visual Control of Mobile Robots (ViCoMoR 2012)
October 11th, 2012, Vilamoura, Algarve, Portugal

Anisotropic Vision-Based Coverage Control for Mobile Robots

C. Franco, G. López-Nicolás, D. M. Stipanović, C. Sagüés

***Abstract*— We consider the problem of vision-based coverage control with a team of robots in the sense of dynamic coverage. Therefore, the aim is to actively cover certain domain by means of robots' visual sensors while they navigate in the workspace. Each robot is equipped with a conventional camera, an anisotropic sensor modeled as a wedge-shaped region in front of the robot. The contribution is a new algorithm for coordinated coverage control which, weights local and global information while avoiding local minima and obeying the particularities of the anisotropic vision-based sensors. The performance of the proposed technique is illustrated with simulation results.**

I. Introduction

Visual control is currently a mature field of research, nevertheless it is still a very active area as new computer vision algorithms or control techniques are being developed and more ambitious applications are envisioned. Fundamental concepts and basic approaches in visual servo control are described in the tutorial [1]. More specific is the survey on vision for mobile robot navigation presented in [2], where visual control refers to the pose control of a vehicle in a closed loop using the input of a visual sensor.

Although visual control and autonomous navigation for a single robot is still an open research area, the use of multiple robots to fulfill particular tasks has been increasingly demanding during the last decade. This is due not only to the advances in hardware devices and commercial software but also to the fact that multiple robots may carry out tasks that are difficult or unfeasible for one single robot such as exploration, surveillance, security or rescue applications. However, not many works in the literature consider the use of vision in the algorithm design for multi-robot systems. Some examples of works using vision to fulfill tasks performed by multiple mobile robots are the localization method presented in [3], the vision-based formation control in [4] or the robot coordination proposed in [5]. Other related works are [6], that aims to enable groups of mobile robots to visually maintain formations in the absence of communication, and [7], that encapsulates the multi-robot system information in a single homography so as to drive the team to a desired formation.

In this work, we focused on the problem of coverage control by using a team of robots equipped with conventional cameras. The coverage task may be found in a wide variety of applications such as demining, cleaning, lawn mowing, painting or surveillance. However, the challenges involved in the multi-robot system control are far from trivial and need to be solved in order to exploit the benefits of multi-robot systems for the efficient coordination of the resources.

The problem of coverage can be classified as static or dynamic depending on how it is addressed. If the resources or robots are static, the problem is known as allocation of resources [8]. The other approach considers mobile resources, and may also consider variable or unknown environment. This problem is often referred to as area coverage and, although multiple applications are possible, literature is mainly focused on sensing tasks. Several approaches tackle the problem by means of an optimization function to be minimized in a decentralized manner with Voronoi partitions [9], [10], by using potential fields [11], [12], or gradient-based approaches [13], [14].

The goal of this work is to explore the feasibility of an anisotropic sensor, i.e. the conventional camera, in the context of dynamic coverage with a team of robots. The problem consists in a coordinated covering of the workspace with the camera field of view. Regarding the type of camera selected, for the last years, the use of omnidirectional cameras is growing because of their effectiveness due to the panoramic view from a single image. This type of camera can be modeled as an isotropic sensor with a circle shape. However, some applications require better resolution rather than a large field of view. In particular, we consider a camera mounted onboard the robots pointing forward. The problem of motion control of a single robot with camera field-of-view constraints has been considered, for instance, in [15], [16]. There, the goal is to keep the camera field of view focused in a particular zone of the environment during the navigation rather than perform visual coverage.

To the best of our knowledge, this is the first dynamic coverage control algorithm proposed for a team of robots considering anisotropic visual sensors. Closely related works are [10] that considers anisotropic sensors modeled with elliptic shape and [17], [18] with the same wedge-shape sensor considered in our work. However, both works focus on the problem of coverage control in the sense of deployment, whereas we are interested in the problem of dynamic coverage. In this paper, we propose a new motion strategy that weights continuously local and global components avoiding local minima and providing an efficient coverage of the domain. The local strategy is based on the gradient, while a blob analysis based approach is defined for the global strategy. The main novelty of this work resides in that the

This work was supported by Ministerio de Ciencia e Innovación/Unión Europea, DPI2009-08126, by project IPT-2011-1158-920000 of subprogram INNPACTO from Ministerio de Economía y Competitividad, by DGA-FSE (grupo T04), and by grant B139/2010 by DGA.

C. Franco, G. López-Nicolás and C. Sagüés are with the Instituto de Investigación en Ingeniería de Aragón (I3A), Universidad de Zaragoza, Mariano Esquillor s/n, 50018, Zaragoza, España. cfranco@unizar.es

D. M. Stipanović is with the Department of Industrial and Enterprise Systems Engineering, and the Coordinated Science Laboratory, University of Illinois, Urbana, IL 61801, USA.

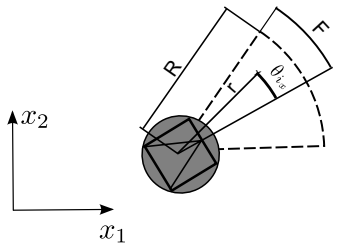


Fig. 1. Scheme of the variables of the sensing function. The dashed line represents the area covered by the camera onboard

proposed approach overcomes the challenges raised by the use of the camera sensor to achieve the coverage objective efficiently.

The paper is organized as follows. Section II introduces the problem formulation. The coverage control laws are presented in Section III. The strategy to select global objectives avoiding local minima is presented in Section IV. Simulations are given in Section V to illustrate the performance of the proposed approach. Conclusion and avenues for future research are given in Section VI.

II. Problem formulation

In this section we describe the framework for a team of nonholonomic agents performing dynamic coverage tasks with anisotropic sensors. The main objective is to reach a coverage level $\Lambda^*(x) > 0$ inside a domain $D_x \subset \mathbb{R}^2$. We assume the agents moves according to the following unicycle model:

$$\begin{aligned} \dot{p}_{i_1} &= v_i \cos(\theta_i), \\ \dot{p}_{i_2} &= v_i \sin(\theta_i), \\ \dot{\theta}_i &= \omega_i. \end{aligned} \tag{1}$$

Here, $p_i = [p_{i_1}, p_{i_2}]^T$ is the position of agent i in a convex domain $D_p \subset \mathbb{R}^2$, and $\theta_i \in [-\pi, \pi]$ is the orientation angle. The positions and the orientation angles of agents are assumed to be known, for instance, by visual localization or by a GPS system. v_i, ω_i are the linear and angular velocity inputs respectively. In this paper, we focus on sensing with vision cameras so let us define the sensing ability $\alpha_i(r, \theta_{i_x})$ as:

$$\alpha_i(r, \theta_{i_x}) = \begin{cases} \alpha_M \frac{(R-r)(F-\theta_{i_x})}{RF}, & r \leq R, \theta_{i_x} \leq F \\ 0, & \text{elsewhere} \end{cases} \tag{2}$$

where α_M is the maximum ability of sensing, R is the sensing range, $r = \|p_i - x\|$ is the distance from the agent to a point x, F is the half of the angle of view of the camera and $\theta_{i_x} = |\theta_i - \theta_x|$ is the angle between the camera and the point x. A graphical depiction of the variables is shown in Fig. 1. This wedge shaped function is maximum in the position of the agent and decreases with the linear and angular distance to the agent. The points that are nearer are better sensed and then it will take less time to cover those points. The coverage action of the team of agents is defined as $\alpha = \sum_{i \in \{1,\dots,N\}} \alpha_i(r, \theta_{i_x})$, with N being the number of robots. Furthermore we define $\Lambda(t, x)$ as the coverage developed by a team of agents over a point x at time t. The coverage information is updated continuously as follows:

$$\frac{\partial \Lambda(t,x)}{\partial t} = \alpha(r, \theta_{i_x}) \tag{3}$$

We assume that the points are initially uncovered $\Lambda(0, x) = 0, \quad \forall x \in D_x$. We introduce the lack of coverage $\Upsilon(t, x)$ over a point x at time t as:

$$\Upsilon(t,x) = \max\left(0, 1 - \frac{\Lambda(t,x)}{\Lambda^*(x)}\right). \tag{4}$$

At this point let us define the error function of the whole domain as:

$$e_{D_x}(t) = \frac{\int_{D_x} \Upsilon(t,x)dx}{S_{D_x}}, \tag{5}$$

where S_{D_x} is the area of the surface of the domain, and the error function of the actuator domain of each agent as:

$$e_{\Omega_i}(t) = \frac{\int_{\Omega_i} \Upsilon(t,x)dx}{S_{\Omega_i}}, \tag{6}$$

where Ω_i is the sensing domain, and S_{Ω_i} is the area of the surface of the sensing domain i.e. the area which the camera sees.

III. Dynamic coverage control laws

We divide our control strategy in two control laws that are weighted during the coverage process. One part of the control law depends on the local error, that is the error of the points that are in the coverage domain of the agent. The other part of the control law depends on the coverage error of the whole domain.

A. Local control law

The speed of the local control law is controlled with the amount of local error:

$$u_i^{loc} = 1 - e_{\Omega_i}. \tag{7}$$

In this way, when the local error is high, the agents slow down to cover the domain, and when the local error is low, they speed up to escape from covered areas. To obtain the angular velocity we start by computing:

$$\tilde{e} = \int_{D_x} \frac{\partial \Upsilon}{\partial t} dx = -\frac{1}{\Lambda^*(x)} \int_{\Omega - \Upsilon_0} \alpha(r, \theta_{i_x}) dx \tag{8}$$

We take out from the integral the points of D_x that are not in the coverage domain of the agents, i.e. the points that are not in $\Omega = \bigcup_{i=1}^{N} \Omega_i$, and the points whose lack of coverage is 0, $\Upsilon_0 = \{x : \Upsilon(t, x) = 0\}$, because both do not contribute to the integral. To optimize the orientation of each robot with respect to the variation of the error we take the partial derivative as:

$$\frac{\partial \tilde{e}}{\partial \theta_i} = \frac{1}{\Lambda^*(x)} \int_{\Omega - \Upsilon_0} \frac{(R-r)(\theta_i - \theta_x)}{RF\theta_{i_x}} dx \tag{9}$$

With this expression we get the direction $\theta_i^{loc^*}$ to obtain the maximum benefit for developing local coverage . Therefore, the contribution of the local control law to the control strategy is given by $(u_i^{loc}, \theta_i^{loc^*})$.

B. Global control law

When an agent falls into an area where the error is constant or symmetric from the point of view of the camera, expression in equation (9) equals zero, and the agent needs another input to reach uncovered areas until the domain is fully covered. In this section we propose a control law by defining $(u_i^{glo}, \theta_i^{glo^*})$ to reach an uncovered point p_i^{obj}, and in section IV we will explain how to choose these points between all uncovered points. The speed of the global control law is controlled with the distance from the agent to the objective, $\|p_i - p_i^{obj}\|$, minus a distance from where the agent can sense the objective, for example $R/2$ as:

$$u_i^{glo} = \frac{2}{\pi} \arctan(\|p_i - p_i^{obj}\| - R/2). \tag{10}$$

u_i^{glo} is close to one until the agent approach the surrounding area of the objective from where can cover it. Then, it decreases to 0 at a distance equal to $R/2$, and is negative inside a circle whose center is p_i and radius $R/2$. Then, when an agent is too near to the objective, it moves away avoiding to drive in circles around the point to see. The orientation to reach global objectives is obtained as:

$$\theta_i^{glo^*} = \arctan 2(p_i - p_i^{obj}). \tag{11}$$

C. Coverage control law

In order to combine both global and local control laws let us introduce a local weight W_i^{loc} and a global weight W_i^{glo} as follows:

$$W_i^{loc}(t) = e_{\Omega_i}^{\beta}(t) \tag{12}$$

$$W_i^{glo}(t) = 1 - e_{\Omega_i}^{\beta}(t) \tag{13}$$

where $\beta \in \mathbb{R}^+$ is a parameter which allows tuning the weights depending on the amount of error. We define the angular error $e_{\theta_i} \in (-\pi, \pi]$ as:

$$e_{\theta_i} = (\theta_i^{loc*} - \theta_i)W_i^{loc} + (\theta_i^{glo*} - \theta_i)W_i^{glo}, \tag{14}$$

and the linear and angular velocities are finally obtained with:

$$v_i = k_v(u_i^{loc}W_i^{loc} + u_i^{glo}W_i^{glo})(1 - \frac{2}{\pi}e_{\theta_i}), \tag{15}$$

$$\omega_i = k_\omega e_{\theta_i}(1 - u_i^{glo}). \tag{16}$$

k_v and k_ω are the control gains of the linear and angular velocity inputs, respectively. When the local error is high, i.e. e_{Ω_i} is close to 1, $W_i^{loc}(t)$ is also close to 1, and the agents obey the local control law which is based on the gradient of the coverage error. Thus, the agents move to get the most coverage benefit. However, when the local error is low, the agents do not get benefit covering its neighborhood. $W_i^{loc}(t)$ is close to 0, and the agents obey the global control law, which direct them to new areas with higher error. These control laws are both bounded by definition, with $v_i \in [-k_v, k_v]$ and $\omega_i \in [-k_\omega\pi, k_\omega\pi]$, allowing a straightforward implementation of the algorithm in real robots by adjusting the gains to the maximum speed of the robots.

Algorithm 1 Blob-based algorithm for the selection of global objectives

Require: D_x, $\Upsilon(t,x)$, Ψ, π_j;
Ensure: Ψ, π_j;

for $j = 1,..,M$ **do**
  if $\Upsilon(t, \psi_j) \leq 0$ **then**
    $\Pi = \Pi - \pi_j$; $\Psi = \Psi - \psi_j$;
  end if
end for
for $j = 1,..,M$ **do**
  $d_j^{min} = min(\|\psi_j - \psi_r\|), r = 1..M/j$
end for
for $j = 1,..,M$ **do**
  if $d_j^{min} < R$ **then**
    $\Pi = \Pi - \pi_j$; $\Psi = \Psi - \psi_j$;
  end if
end for
$D_{blob} = D_x - \Pi - \pi^{\emptyset}$;
while $D_{blob} \neq \emptyset$ **do**
  $(\psi^1, ..., \psi^K, \pi^1, .., \pi^K) = blob(D_{blob})$;
  for k=1,..,K **do**
    if $\psi^k \cup \pi^k \neq \emptyset$ **then**
      $\Psi_{M+1} = \psi^k$; $\pi_{M+1} = \pi^k$;
      $D_{blob} = D_{blob} - \pi^k$;
    end if
  end for
  $D_{blob} = erode(D_{blob})$;
end while
Assign eroded points to the nearest blob;

IV. Selection of global objectives

In this section, we propose a strategy to find areas with large error and to provide the agents with inputs to reach them (i.e. we now describe the procedure to define the values of p_i^{obj} used in section III.B). It is based on blob detection of the uncovered information. We use this image processing technique to find islands of uncovered information in the map $\Lambda(t,x)$, and then we compute their sizes and their centroids. With this information we propose a criterium to select a centroid as a global objective based on the uncertainty and the proximity of the blobs. Hence, a global objective is the centroid of an uncovered area which is close to the agent.

Let us define $\Psi = \{\psi_1, \psi_2, ..., \psi_j, ..., \psi_M\}$ as the collection of M global objectives, $\psi_j \in D_x$. We refer to ψ's as objectives and they represent points that belongs to a set of uncovered points. Let us also define π_j as the collection of points of the domain composing each blob and whose global objective is ψ_j, $\Pi = \bigcup_{j=1}^{M} \pi_j$ as the collection of points of the domain assigned to objectives ψ_j, and $\pi^{\emptyset} = \{x \in D_x | \Upsilon(t,x) \leq 0\}$ as the collection of points that are covered. The method to select the global objectives is described in Algorithm 1.

The algorithm starts by checking if some of the M global objectives ψ_j have been covered. Those covered are erased from the list of global objectives Ψ and the points π_j

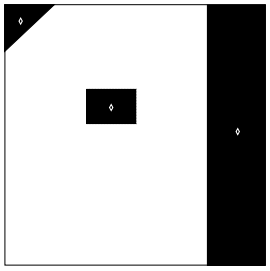


Fig. 2. Example of an uncertainty map that causes the centroid to fall inside of the blob. Black area represents the uncovered zone and white the covered zone whose points belongs to $\pi^{\emptyset}$. The centroids are represented by rhombi. The uncertainty map has 3 blobs and then 3 global objectives are generated. The points π_j in black areas are assigned to their respective centroids ψ_j.

assigned to the objective are released. It also checks if there are objectives that are closer than a distance R, which is the coverage radius of the agents. Those objectives are also erased and their points released to try to merge them to get a bigger blob in the blob searching procedure. Afterwards, the domain to obtain the new blobs of the scene is computed by subtracting the covered points $\pi^{\emptyset}$ and the assigned points Π from the domain to cover D_x. With $blob(D_{blob})$ the centroids ψ^k and the points π^k of the K regions of the space to be covered are obtained (as shown in Fig. 2). Then, the centroids ψ^k are checked to see if they belong to the points π^k of the blob. If the centroids ψ^k are inside the blob, they and the points of the blobs π^k are saved, whereas the blob domain is reduced by the points of the new found blob.

Once the checking is complete, it is possible that some centroids fall outside of their respective blobs. This is not a desired situation because due to the coverage range of the agent, it is possible that once the agent has arrived to the global objective, it cannot reach uncovered points causing a blockage. In this case, the image is eroded (as depicted in Fig. 3). This results in the elimination of the points in the domain that are in contact with covered or assigned points in such a way that the irregularities of the blob that cause the centroid to fall outside the blob are eliminated. Afterwards, blob analysis is repeated while the blob domain is not null. Finally, the eroded points are assigned to the nearest blob. It is possible but unlikely that, due to symmetries, no global objectives are found. In that case a nearest uncovered point is the only global objective until the symmetry breaks.

The choice of the objective p_i^{obj} for each robot i is done according to proximity. First, the distance between the i-th agent and the j-th centroid is calculated to create a matrix D as follows:

$$D(i,j) = \left(\frac{\|p_i - \psi_j\|}{\max(\|p_i - \psi_j\|)} \right). \qquad (17)$$

It is divided by the maximum distance in such a way that the elements of D are in interval $(0, 1]$. Then, with this matrix, the global objectives are assigned using Algorithm 2. It is repeated until all the agents have a global objective. First, the algorithm finds the minimum distance between an agent and a centroid. Then, the agent takes that centroid as the global

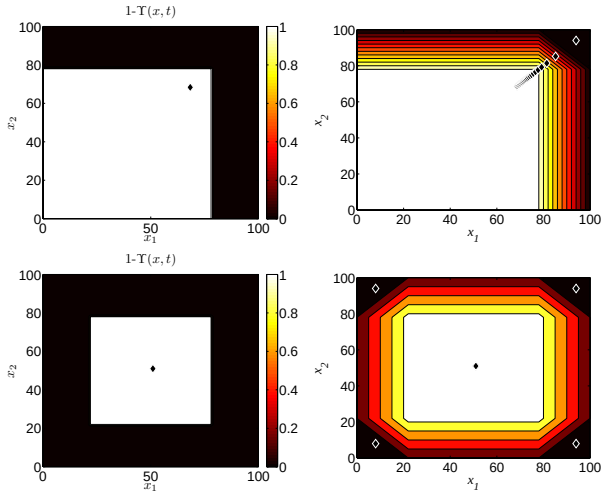


Fig. 3. Two examples of uncertainty maps that cause the centroid to fall outside of the blob. In the left column black areas represent the uncovered zones and white the covered zones. The centroids are represented by rhombi. A blob analysis of these scenes produces in each case only one connected blob with one centroid. Due to the shapes of the blobs, the centroids fall outside their blobs and according to Algorithm 1 an erosion process is needed. In the right column the iterative erosion process is shown until the centroid falls into the blob. After this process is completed, eroded points are assigned to the last centroids obtained according to proximity.

objective. Next, the algorithm adds 1 to the column where the distances of the centroid to other has been calculated to avoid that the centroid is assigned to other agent, distributing the team between all the available centroids. Also, the row of the agent is increased by N units to avoid the assignation of other centroid to the agent. If there are more agents than centroids, when all the centroids have been assigned once, all the distances between centroids and non assigned agents are increased by 1 and then, the centroids can be assigned again with the distance criterium. This process is repeated $\lfloor N/M \rfloor$ times distributing at most $\lceil N/M \rceil$ to each centroid and at least $\lfloor N/M \rfloor$.

Algorithm 2 Assignation of objectives

Require: D, Ψ
Ensure: p_i^{obj}

1: **repeat**
2: $[i,j] = \{i,j : D(i,j) = \min(D)\}$;
3: $p_i^{obj} = \psi_j$;
4: $D(i,:) = D(i,:) + N$;
5: $D(:,j) = D(:,j) + 1$;
6: **until** <All agents have an objective>

V. Simulation results

In this section we present some simulation results that illustrate the behavior of the proposed algorithms. The domain to cover D_x is a square of 100x100 units, whereas $D_p = \mathbb{R}^2$. The coverage objective is $\Lambda^* = 100$. There are four agents with: $R = 20$, $F = 54^{\text{o}}$, $\alpha_M = 50$, $\beta = 1$, $k_v = 1$, and $k_\omega = 1$. Fig. 4 shows the evolution of the normalized error throughout the coverage process. In 592 units of time, the

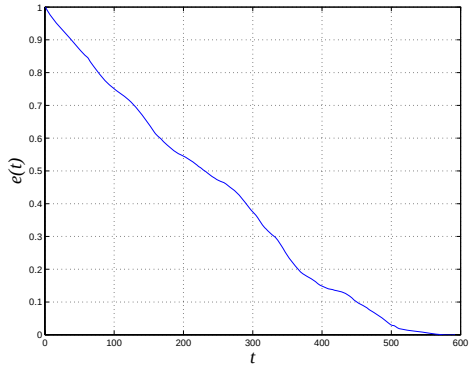


Fig. 4. Evolution of the normalized coverage error

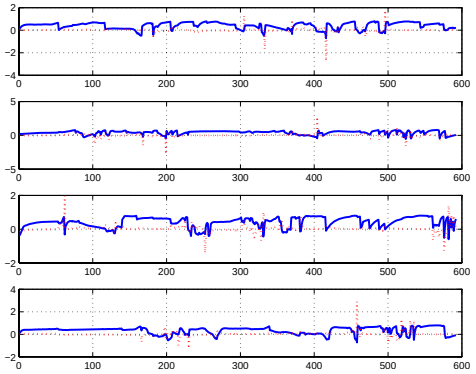


Fig. 5. Motion action of the agents. Solid line represents the linear action and dotted line the angular velocity.

objective has been fully accomplished at all the points of the domain. The motion actions to develop the coverage are shown in Fig. 5, and the map of the lack of coverage in each point at different times is shown in Fig. 6.

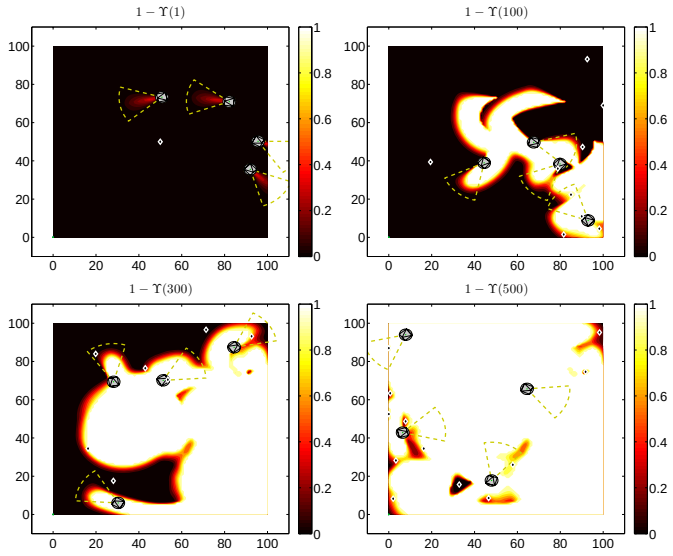


Fig. 6. Evolution of the global coverage map throughout the coverage process. Small circles represent the position of the robots and the coverage domain is represented by a dashed line. The domain is rather covered at t=500 and it is totally covered at t=592.

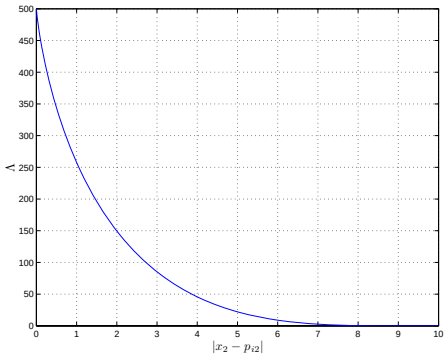


Fig. 7. Amount of coverage developed perpendicular to v_i when the agent moves along x_1.

We also develop simulations to check the efficiency in the usage of the robots. First, we start by computing the total ability of sensing $\mathcal{A}$:

$$\mathcal{A} = 2\int_0^F \int_0^R \alpha(r, \theta_{i_x}) r dr d\theta_{i_x} \tag{18}$$

With this value, we can compute the minimum time to completion for N agents t_N^*:

$$t_N^* = \frac{\int\int_{D_x} \Lambda^*(x_1, x_2) dx_1 dx_2}{\mathcal{A} \cdot N}, \tag{19}$$

Lastly, let us compare our algorithm with a typical path planning coverage trajectory carried out in zig-zag with maximum speed. The amount of coverage $\Lambda(x, t)$ developed by one agent in the perpendicular direction to v_i with $v_i = 1$ and $\omega_i = 0$ is shown in Fig. 7. The agent reaches the coverage objective over the points placed at $d_{100} = 2.72$ units away from the agent and half of the coverage objective over the points placed at $d_{50} = 3.865$ units away from the agent. The trajectory is developed taking into account these parameters and with a length L=140 units as shown in Fig. 8. The time to completion for one agent is $t_1^{zz} = 2080.6$ units of time with a path length of $PL_1^{zz} = 2054.6$ units. Assuming perfect coordination between teams of agents, we have $t_N^{zz} = 2080.6/N$ and we also assume $PL_N^{zz} = 2054.6$. Thus, we compute $E_N^{zz} = t_N^*/t_N^{zz}$.

In Fig. 9 we show the relation between the optimum and our algorithm, and the relation between the optimum and the zig-zag algorithm from 1 to 30 agents. The time to completion t_N of teams of robots varying from 1 unit to 30 units have been computed running 100 simulations in each case and taking the average value. For small teams our proposal takes 4 times the time to completion of the optimum, and as the team grows it tends to 5, compared with the zig-zag path that is 3 times slower. Furthermore, we also compare the sum of the path length of the robots to develop the coverage. We compute the path length of 100 simulations and show the average in Fig.10. Since the speed is regulated with the local error, our algorithm make the most of the traveled path. It takes some more time to cover the domain, but the path length needed in our algorithm is around a 55% of the path length needed with zig-zag strategy.

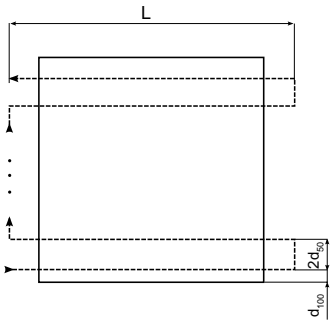


Fig. 8. Zig-zag path to cover the domain with one agent.

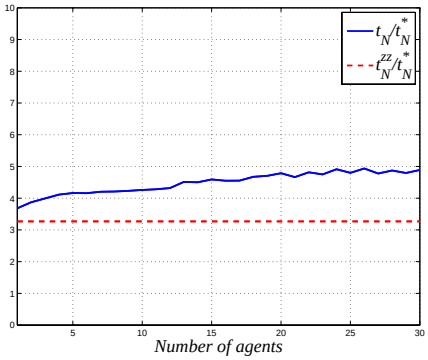


Fig. 9. Comparison of the time to completion of several teams of agents developing optimum coverage with our algorithm and with a zig-zag algorithm.

VI. Conclusion

In this paper we have proposed a new control algorithm for the dynamic coverage of a domain developed by a team of agents with anisotropic vision based sensors. The control law weights local and global actions, depending on local coverage error and global error map, respectively, to give more importance to local objectives when the local error is high and to global objectives when the benefit of developing the coverage in the neighborhood of an agent is small. We also propose, a new strategy to select global objectives based on blob analysis of the whole map. Additionally, we impose bounds on both linear and angular velocities. Finally, simulations are provided to illustrate the approach. Current work focuses on the collision avoidance problem and the coverage with decentralized information. In this work, we assume that the robots do not produce occlusions between them. Even if the convergence is not compromised, future work would study this issue regarding the performance of the proposed algorithm.

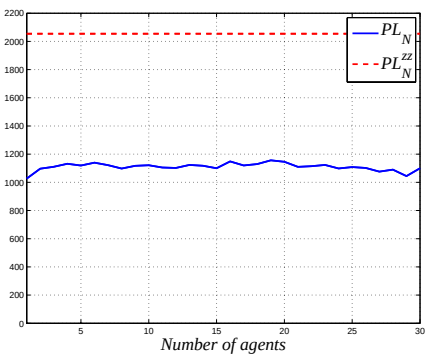


Fig. 10. Comparison of the path length of several teams of agents developing coverage with our algorithm and with a zig-zag algorithm.

References

[1] F. Chaumette and S. Hutchinson, "Visual servo control, part II: Advanced approaches," *IEEE Robotics and Automation Magazine*, vol. 14, pp. 109–118, Mar. 2007.

[2] G. N. DeSouza and A. C. Kak, "Vision for mobile robot navigation: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 2, pp. 237–267, 2002.

[3] H. Chen, D. Sun, and J. Yang, "Global localization of multirobot formations using ceiling vision SLAM strategy," *Mechatronics*, vol. 19, no. 5, pp. 617 – 628, 2009.

[4] A. K. Das, R. Fierro, V. Kumar, J. P. Ostrowski, J. Spletzer, and C. J. Taylor, "A vision-based formation control framework," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, pp. 813–825, 2002.

[5] N. Moshtagh, N. Michael, A. Jadbabaie, and K. Daniilidis, "Vision-based, distributed control laws for motion coordination of nonholonomic robots," *IEEE Transactions on Robotics*, vol. 25, pp. 851–860, Aug. 2009.

[6] R. Vidal, O. Shakernia, and S. Sastry, "Following the flock: Distributed formation control with omnidirectional vision-based motion segmentation and visual servoing," *Robotics and Autonomous Magazine*, vol. 11, no. 4, pp. 14–20, 2004.

[7] G. López-Nicolás, M. Aranda, Y. Mezouar, and C. Sagüés, "Visual control for multi-robot organized rendezvous," *IEEE Transactions on Systems, Man, and Cybernetics: Part B*, 2012.

[8] Z. Drezner, *Facility location: A survey of Applications and Methods*. New york: Springer-Verlag, 1995.

[9] J. Cortés, S. Martínez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 2, pp. 243–255, 2004.

[10] A. Gusrialdi, S. Hirche, T. Hatanaka, and M. Fujita, "Voronoi based coverage control with anisotropic sensors," in *Proceedings of the American Control Conference*, pp. 736–741, 2008.

[11] D. O. Popa, C. Helm, H. E. Stephanou, and A. C. Sanderson, "Robotic deployment of sensor networks using potential fields," in *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, pp. 642–647, 2004.

[12] M. J. Mataric, A. Howard, and G. S. Sukhatme, "Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem," in *Proceedings of the 6th International Symposium on Distributed Autonomous Robotics Systems*, 2002.

[13] I. I. Hussein and D. M. Stipanović, "Effective coverage control for mobile sensor networks with guaranteed collision avoidance," *IEEE Transactions on Control Systems Technology*, vol. 15, no. 4, pp. 642–657, 2007.

[14] S. K. Gan and S. Sukkarieh, "Multi-UAV target search using explicit decentralized gradient-based negotiation," in *2011 IEEE International Conference on Robotics and Automation*, pp. 751 – 756, may 2011.

[15] G. López-Nicolás, N. R. Gans, S. Bhattacharya, J. J. Guerrero, C. Sagüés, and S. Hutchinson, "Homography-based control scheme for mobile robots with nonholonomic and field-of-view constraints," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 40, no. 4, pp. 1115–1127, 2010.

[16] P. Salaris, L. Pallottino, and A. Bicchi, "Shortest paths for finned, winged, legged, and wheeled vehicles with side-looking sensors," *International Journal of Robotics Research*, vol. 31, no. 8, pp. 997–1017, 2012.

[17] K. Laventall and J. Cortes, "Coverage control by robotic networks with limited-range anisotropic sensory," in *American Control Conference, 2008*, pp. 2666 –2671, june 2008.

[18] A. Gusrialdi, T. Hatanaka, and M. Fujita, "Coverage control for mobile networks with limited-range anisotropic sensors," in *47th IEEE Conference on Decision and Control*, pp. 4263 –4268, dec. 2008.

IROS Workshop on Visual Control of Mobile Robots (ViCoMoR 2012)
October 11th, 2012, Vilamoura, Algarve, Portugal

FSM-based Visual Navigation for Autonomous Vehicles

Daniel Oliva Sales[1], Leandro Carlos Fernandes[1], Fernando Santos Osório[1] and Denis Fernando Wolf[1]

Abstract — **In this paper, we present a vision-based navigation system for autonomous vehicles in structured urban environments. This system uses a standard RGB camera as the main sensor. The proposed approach is composed by an association of Finite State Machines (FSM) with Artificial Neural Networks (ANN). First, we identify the navigable areas after processing the input frames. Then, an ANN is trained to recognize patterns on the generated navigability maps. Each pattern is associated to a specific state, related to a unique environment structural feature. A topological-like map is used to represent the environment, so any path can be described as a sequence of states. For example, straight path, right and left turns and intersections. The experiments were performed with an autonomous vehicle in a real urban environment in order to validate and evaluate this approach. The proposed system demonstrated to be a promising approach to autonomous vehicles navigation.**

I. INTRODUCTION

The development of robust autonomous intelligent systems for robotic applications is a very important research topic. Several applications are related to robotics, from industry to military tasks.

The autonomous driving capability is one of the most desirable features for a mobile robot. Researches related to this feature are being developed since the 80´s, and groups such as NavLab have been presenting relevant results on autonomous vehicles navigation.

Nowadays there are many relevant and known researches on autonomous robotics being developed worldwide. Some of them are powered by government initiatives as for example the Darpa Grand Challenge [7][8][9]. The first two editions (2004 and 2005) were held in desert, and 2007 edition in an urban environment. In Brazil, the development of autonomous vehicles is an important research challenge among the main working groups (WG2) of the Brazilian National Institute of Science and Technology on Embedded Critical Systems (INCT-SEC).

Autonomous mobile robots usually perform three main tasks: localization, mapping and navigation [17]. Mapping is the creation of an environment model using the sensorial data, representing its environment structure. Localization task must occur simultaneously to navigation control. It consists in estimating the robot´s position in a previously known environment, using its sensorial data. The Navigation task is therefore the ability to obtain enough information about the environment, process it, and act, moving safely through the navigable area.

*Research supported by FAPESP.

The authors are with the Mobile Robotics Lab at ICMC - University of São Paulo, São Carlos, CEP 13566-590, Brasil (e-mail: dsales, lnd, fosorio, denis @ icmc.usp.br).

In order to develop an intelligent system able to navigate through environments composed by streets and highways, it is desirable to know the robot´s approximate position, the environment map and the path to be followed. So, navigation in this environment lies in following a well-defined path, considering the navigable areas.

This work focuses on this navigation task, describing the development of a Vision-Based Topological Navigation System, able to recognize the navigable area of an urban environment (streets) processing image frames and classifying them into states which represent the current robot context, allowing the robot to autonomously drive through this environment and reach a desired destination.

The adopted navigation approach does not require a very detailed environment map (metric map), only a graph that represents the main elements, in a simpler path representation. Furthermore, accurate pose estimation is not necessary. The approximate robot´s position is enough to navigate. So, the main objective is to detect the current node in a Topological map, being useful to autonomously decide when and how go straight, turn left or right, even when more than one possibility is detected simultaneously (at an intersection, for example).

The developed system uses Artificial Neural Networks (ANN) [19] in two steps: to classify the frame obtained from camera (resulting in a navigability map) and second to detect patterns on these navigability maps (representing the current context). In this second step, a Finite State Machine (FSM) is used to represent the state sequence for any path at the environment. The ANN is trained to recognize all possible states, so a FSM generator can convert any chosen path into a sequence of these known states.

The motion control is based on hierarchical/hybrid control approach. This way, the navigation system combines the high-level deliberative control (path planning) with different reactive behaviors, allowing a safe motion. This FSM-based approach was already successfully applied in previous authors' works for indoor applications [4][25][16] with different sets of sensors and states, showing the feasibility of this implementation and motivating studies concerning the application of this technique in urban environments.

The main objective of this work was obtaining and processing enough information for state (context) detection, allowing a high-level path planning and also safe motion using reactive control for autonomous vehicles.

The next topics of this paper are organized as follows: Section 2 presents some previous related work; Section 3 presents the Topological Navigation System overview, Section 4 presents experiments and results and Section 5 presents the conclusion and future work.

II. Related Work

Several navigation approaches have been used for navigation, using many different sensors (for example laser, sonar, GPS, IMU, compass) solely or combined [9][17][18]. One of the most used approaches is the vision-based navigation [20]. This method uses monocular video cameras as the main sensor. Cameras are very suitable for navigation and obstacle avoiding tasks due to its low weight and energy consumption [1]. Furthermore, one single image can provide different types of information about the environment simultaneously. It is also possible to reduce costs by using cameras rather than other types of sensors [2].

Vision-based navigation approaches are already usual in navigation systems for structured or semi-structured environments [3][10][11]. These systems classify the image, with track segmentation for safe navigable area identification, resulting in reactive models for navigation control. Works such as ALVINN [13] and RALPH [14] were some of the first to apply neural networks for this reactive control in outdoor environments.

In [3], Shinzato developed a neural classifier composed by an ANN ensemble, able to detect and to segment the navigable areas of the road through image features analysis. Later, this classifier was adopted by Souza in [15] for a Template-Matching based reactive control.

Purely reactive models are not totally adequate for our autonomous navigation system development, since immediate reaction to sensors data is not enough to guarantee a correct control in complex environments. A more robust system should be implemented, providing sequence and context information that are absent in purely reactive models.

In robotics, FSM-based approaches [5] are very often used. FSMs are useful because the system can be easily described as a sequence of states (context changes), considering the inputs (sensors) and specific actions for each state. This way, for each detected state the robot can assume a different behavior. This work focuses on this main idea, with possible paths described as FSMs in which current state is detected after processing sensors data.

The use of Machine Learning techniques such Artificial Neural Networks is a very interesting way to process input data, identifying and classifying the states and transitions to determine the best actions to be performed [6]. ANNs are tolerant to noise and imprecision on input data, and able to detect the states and transitions between these states. ANNs are also very efficient to generalize knowledge (adjusting the outputs to many inputs, even the ones not explicitly taught to the net). So, this technique is very useful for state detection through path features recognition.

The association of ANNs with FSMs is being developed and evolved since the 1990´s [21][22][23][24], and recent researches were focused on applying this technique to robotics. In [12], an autonomous car parking system was developed, using recurrent neural networks for FSM learning. The sensors data and current state were used as ANN inputs, so the system could detect when a context change was needed. This work inspired the development of our FSM-based approach.

In [20], a Vision-based autonomous navigation system was implemented for indoor environments using a simple FSM control. In that work, Sinzato´s classifier was used to generate the navigability matrices from input frames. Then, an algorithm was developed to analyze interest areas of these matrices in order to determine the current state of a FSM (robot context) to navigate through a set of turns and straights. The main idea of this FSM-Control was evolved, so in [4] and [25] an autonomous navigation system was developed with an ANN as the control unit. A LIDAR sensor was adopted as the main sensor, so an ANN was trained to recognize the "laser signatures" associated to every possible state of a FSM. Each state was related to a specific part of an indoor environment. This way, the FSM states could be learned by the ANN, and the paths represented by a sequence of known states. These works introduced the Topological Navigation approach proposed in this paper.

The navigation system developed in this paper combines the visual (low-cost) navigation with the neural learning of FSMs in Topological Navigation approach, considering the best features obtained with these previous results. The developed system components are described in next section.

III. System Overview

The proposed system is composed by three main modules: "Navigability Map Generation", "State detection" and "Reactive Control". The Navigability Map generator is responsible to convert a captured frame into a navigability map, used as input for our Navigation Control System. The State Detection module is responsible to detect the current state and filter oscillations, avoiding unexpected state transitions (considering the path plan). The reactive control determines an appropriate steering angle according to the navigable area detection and current state detected. Figure 1 shows the full system flowchart.

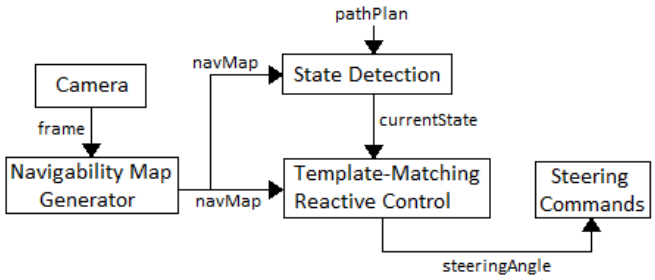


Figure 1 – Navigation System Flowchart

A. Navigability Map Generation

This step is performed using the neural classifier proposed by Shinzato in [3]. It is composed by an ANN ensemble with six ANNs; each one is responsible to classify the pixels into navigable (1) or non-navigable (0), based on different sets of features of the image such as RGB values, HSV values and entropy. So, the mean of these output values is the final classification value, ranging from 0 to 1.

The ANNs training is performed with a small set of representative frames. The supervisor must classify parts of these frames into navigable or non-navigable in a GUI, obtaining six trained ANNs.

The input frame has 320x240 pixels size, sliced in 10x10 block to produce a 32x24 navigability map. The neural classifier and the image features used for each ANN are fully described in [3]. Figure 2 shows some examples of processed frames and the resulting navigability maps. The segmented navigable area is used as input for the State detection and Reactive Control modules.

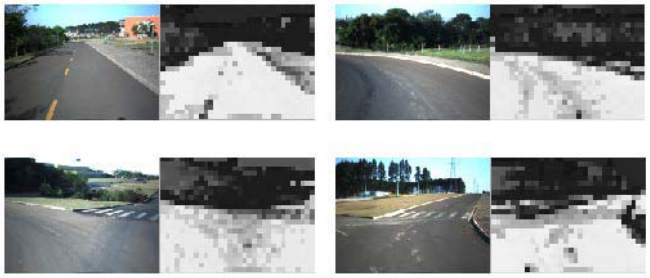


Figure 2 – Navigable area detection results on different situations (bright = navigable block)

B. State Detection

This module is responsible to recognize patterns on the input data, allowing the detection of environment features used to determine the current state (context). An ANN is used for this task.

The environment is mapped as a topological map in which each node is related to a specific part of the environment, described by its structural features such as straight path, turn or intersections. All possible states (environment features) are taught to the ANN, so possible paths on this environment can be described as a sequence of these learned states, being represented by FSMs. Figure 3 shows a part of an environment and its topological map.

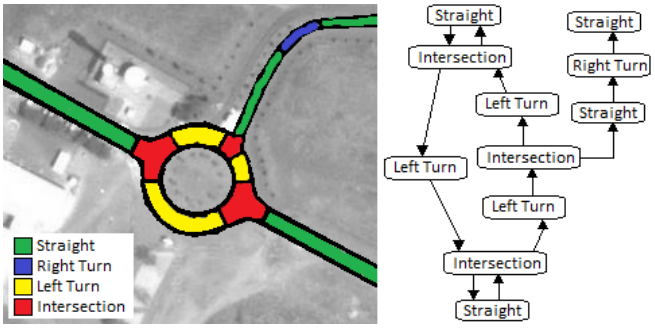


Figure 3 – Environment mapping example

The ANN input is the navigability matrix obtained at the previous step, and the output is the current state (environment feature) detected. Four different classes were created, each one related to a different track shape or condition (straight, left and right turns and intersection).

In order to begin the training process, a database must be generated collecting a set of frames for each possible situation. So, these frames may be processed by the first neural classifier to generate the navigability maps. As the learning process is supervised, a specialist must classify these navigability maps in one of the possible states before the final ANN training, creating a set of input/desired-output pairs. After finishing the ANN training, it should be able to recognize different possible paths at the environment.

Once the Topological Map of an environment is known, it is possible to establish a route between two points, manually or with a path planning algorithm. Every route can be seen as a sequence of steps (states), so it is trivial to generate a FSM (sub-graph) to represent a well-defined path.

For this system implementation, it is assumed that vehicle´s initial position is always known, as the topological map also. The current position is estimated based on current state detection, so it isn´t necessary to estimate robot´s exact position. Navigation and self-localization are performed together.

The desired path is also used as input to State Control unit, so the system can determine if a state transition is needed or if the vehicle still remains at the current state. The State Control unit also filters state detection oscillations, allowing a state transition only if the next expected state is observed by a certain amount of consecutive steps. If the observed state does not match the path plan, current state is kept, as shown on Figure 4.

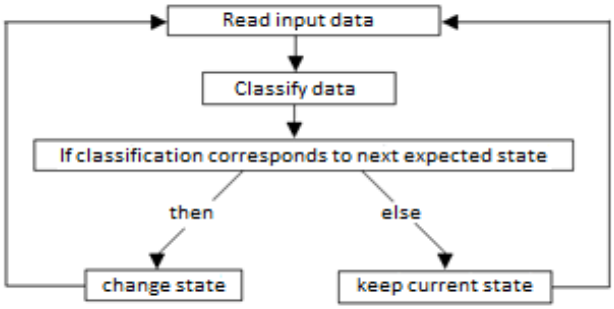


Figure 4 – State transition flowchart

This Topological Navigation approach allows the vehicle to follow its planned path and also know its approximate position, but does not control the motion "into" every situation (state). This task is performed by a reactive control unit, described in the next section.

C. Template-Matching Based Reactive Control

This module is responsible to react to road current conditions, determining the vehicle´s steering angle. As mentioned earlier, the motion control system must combine the deliberative control resulting from FSM-based topological navigation with reactive behaviors to guarantee a safe driving, avoiding leaving the track.

The implemented reactive control uses a template-matching approach for road track following, as proposed in [15]. The navigability maps are compared with five different templates, each one related to a specific steering angle for reaction, as shown on Figure 5. Each template receives a matching score, so the steering angle is defined after measuring the best score.

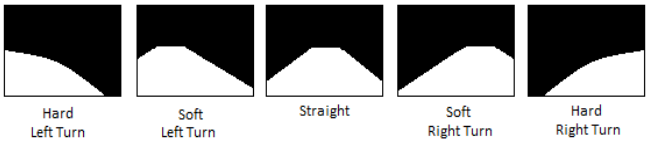


Figure 5 – Reactive Control Templates representation

The current state is also considered as input, so a bias is added to the score of the most suitable templates for each current condition. Straight path state adds a bias to straight and soft turns templates, avoiding abrupt movements; left and right turns adds a bias to its corresponding soft and hard turn templates; intersection state adds a bias to straight, hard and soft turns considering the path plan (left to keep in the roundabout, right to get out).

IV. Experiments and Results

The experiments were carried out in a real urban environment, using an autonomous vehicle equipped with a video camera as the only sensor. The environment was composed by straights, turns and intersections, so its properties could be represented with four states: "straight path", "left turn", "right turn", and "intersection". Some examples of input frames for these four states are shown on Figure 6.

Figure 6 – Example of input frames for each implemented state

A. System Setup

The camera used in our tests was a PointGrey Bumblebee2 Stereo Cam [26]. We do not deal with depth information in this paper, so only left image is taken as perception system input. The camera was placed above the car as shown on Figure 7, so viewing angle was adjusted in order to allow immediate reaction to each frame/navigability map processed.

Figure 7 – Camera position

The control platform was implemented on Robot Operating System (ROS) [27], a framework which provides libraries and tools for robot applications development. Each module was implemented as a ROS node, with message-passing between the modules, as shown on Figure 1. The full control system works in real-time, with one steering command per frame, at 30 fps frame rate.

The Navigability Map Generator was created after collecting and classifying data on 15 frames of the environment. So, the ANN ensemble was generated, trained and included into the embedded system.

B. State Detection Unit

The ANN used for state detection was implemented and trained with Stuttgart Neural Network Simulator (SNNS) software. ANN training database was generated collecting about 2040 frames (converted to navigability maps) for each class. So, the final database was composed by 8165 input/output pairs.

The training algorithm used was Resilient Propagation (R-Prop). This algorithm achievies great results for feed-forward networks in many applications due to its good training time and convergence. Training parameters used are shown on Table 1.

Table 1 – Training Parameters

Parameter	Value
Training Algorithm	R-Prop
δ0	0,1
δMax	50
α	4
cicles	500
aprox. total training time	2 hours

Due to the camera position, the first 288 elements on upper half of navigability matrices are most commonly related to elements above the horizon line, so they are not considered for state detection. The lower 96 elements (bottom 3 lines) are also discarded because they are related to the current navigable area, not an incoming situation. This way, a detection window with 12 lines (384 elements) is used for state detection. So, the ANN input layer is composed by 384 neurons only.

Some empirical tests based on previous work knowledge were performed in order to determine the best ANN topology. The best results were achieved by a feed-forward MLP, with the 384 input neurons, 384 neurons on hidden layer and 4 neurons on output layer (1 neuron per class), all neurons with activation function defined as "Act_Logistic" implemented on SNNS.

ANN validation was done with stratified 5-fold cross-validation method. This way, 5 train and test sets were generated, with 80-20 proportion on data (80% used for training and 20% for test, with same proportion of elements from the 4 classes on the datasets). The ANN accuracy on the five tests can be observed at Table 2, and the confusion matrices for the five test sets resulting from 5-fold cross validation are shown on Figure 8.

Table 2 – ANN Accuracy after 500 training cicles

	Test 1	Test 2	Test 3	Test 4	Test 5
ANN Accuracy	98.8%	99.0%	99.1%	99.3%	99.5%

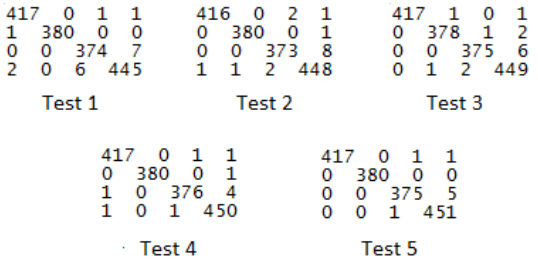


Figure 8 – Confusion matrices

A low error per class can be observed on confusion matrices. However, this only shows the network learned the database examples. If the system wrongly detects any situation as the next expected state, the reactive behavior will be affected, adding a bias to wrong templates and performing wrong steering actions. Likewise, if a new expected state is not detected in time, the system will keep the current behavior active, impairing self-localization and navigation tasks.

Therefore, state detection is a critical point. To ensure a correct navigation, the ANN training database must cover an enough amount of examples, avoiding misclassification on state detections. This problem can also be reduced if a minimum amount of consecutive new state detections is required for state transitions. The experiments carried out with this filter showed an adequate reliability level for state changes in developed applications.

C. Autonomous Navigation Tests

The environment in which the database was generated and the tests performed is shown next, on Figure 9. Our goal was to autonomously navigate following well-defined routes. The vehicle successfully accomplished the navigation task in a path with straights and a roundabout in autonomous mode, validating the proposed approach.

Figure 9 – Outdoor Environment used for tests

The reactive control was correctly affected by state detection, allowing a smooth behavior. Figure 10 shows the vehicle performing a left turn in a roundabout. A video with a successful path sequence and more information is available at http://www.icmc.usp.br/~fosorio/research/vicomor12.html.

Figure 10 – Autonomous vehicle in a roundabout situation

Some misclassification occurred when unexpected light conditions affected input data, resulting in noisy navigability maps. Other dynamic elements such as vehicles and obstacles were not considered as a new state, and also produced bad responses when crossing the state detection window area.

The minimum amount of consecutive new state detections was manually tuned, normalizing the chance of straight and turn behaviors to be activated, since turn states are detected for a shorter period of time.

We also considered using a binary navigability map instead of continuous probability maps in template-matching step, filtering uncertainty on road detection with a threshold. In some cases, this solution produced better results than continuous version, so the use of binary maps was implemented as a parameter to be set before navigation system launch, allowing the selection of the fittest solution.

V. CONCLUSION

The successful navigation results demonstrated the suitability of this approach for autonomous vehicles navigation, when an accurate state detection is possible. This way, the use of a camera as the main sensor can be an efficient and reliable solution, allowing the project of low-cost autonomous driving systems.

This work complements the range of applications proposed on our previous indoor works, showing Topological Navigation as a promising approach for autonomous vehicles navigation too.

The system can be re-trained to recognize new situations, settings and features, and also use and combine other sensorial systems, allowing its implementation in different scenarios. For future works, we consider using sensor fusion and other techniques to detect new features and landmarks useful for navigation control.

ACKNOWLEDGMENT

The author acknowledges FAPESP and CNPq for their support to INCT-SEC (National Institute of Science and Technology - Critical Embedded Systems - Brazil), processes 573963/2008-9 and 08/57870-9 and financial support to authors (master´s grant).

REFERENCES

[1] Zingg, S., Scaramuzza, D., Weiss, S., and Siegwart, R. MAV Navigation through Indoor Corridors Using Optical Flow , IEEE International Conference on Robotics and Automation (ICRA 2010), Anchorage, Alaska, May, 2010.

[2] Scaramuzza, D., Siegwart, R. Appearance Guided Monocular Omnidirectional Visual Odometry for Outdoor Ground Vehicles. IEEE Transactions on Robotics, vol. 24, issue 5, October 2008.

[3] Shinzato, P. Y, Wolf, D. F. Features Image Analysis for Road Following Algorithm Using Neural Networks. In: 7th IFAC Symposium on Intelligent Autonomous Vehicles, Lecce, 2010.

[4] Sales, D; Osório, F; Wolf, D. Topological Autonomous Navigation for Mobile Robots in Indoor Environments using ANN and FSM. In: Proceedings of the I Brazilian Conference on Critical Embedded Systems (CBSEC), São Carlos, Brazil, 2011.

[5] Hopcroft, J.E., Ullman, J.D. (1979) "Introduction to Automata Theory, Languages and Computation". Addison - Wesley, 1979.

[6] Marino, A.; Parker, L.; Antonelli, G. and Caccavale, F. Behavioral Control for Multi-Robot Perimeter Patrol: A Finite State Automata approach. In: ICRA, 2009.

[7] Thrun, S. et al. (2006) "Stanley: The Robot that Won the DARPA Grand Challenge," Journal of Field Robotics, Vol. 23, No. 9, June 2006, p.661-692.

[8] Urmson, Chris et al. (2008). "Autonomous driving in urban environments: Boss and the Urban Challenge". In: Journal of Field Robotics. Vol. 25 , Issue 8 (August 2008). Special Issue on the 2007 DARPA Urban Challenge, Part I. Pages 425-466.

[9] Buehler, Martin; Iagnemma, Karl; Singh, Sanjiv (Editors). The 2005 DARPA Grand Challenge: The Great Robot Race (Springer Tracts in Advanced Robotics). Springer; 1st. edition (October, 2007).

[10] Nefian, A.V.; Bradski, G.R. (2006) "Detection of Drivable Corridors for Off-Road Autonomous Navigation". ICIP-06: Proceedings of the IEEE International Conference on Image Processing. pp. 3025-3028.

[11] J.M. Álvarez, A. M. López, and R. Baldrich. (2008) "Illuminant Invariant Model-Based Road Segmentation". IEEE Intelligent Vehicles Symposium, Eindhoven, Netherlands, June 2008. http://www.cvc.uab.es/adas/index.php?section=publications.

[12] Heinen, Milton Roberto ; Osório, Fernando S. ; Heinen, Farlei ; Kelber, Christian . SEVA3D: Using Artificial Neural Networks to Autonomous Vehicle Parking Control.. In: IJCNN - IEEE Intenational Joint Conference on Neural Networks, 2006, Vancouver. Proceeding of the WCCI (World Congress on Computational Intelligence) - IJCNN. Vancouver, Canadá : IEEE Press, 2006. v. 1. p. 9454-9461.

[13] Pomerleau, D. ALVINN: An Autonomous Land Vehicle In a Neural Network. Advances in Neural Information Processing Systems 1, 1989.

[14] Pomerleau, D. RALPH: Rapidly Adapting Lateral Position Handler. IEEE Symposium on Intelligent Vehicles, September, 1995, pp. 506 - 511.

[15] Souza, J.; Sales, D. O.; Shinzato, P. Y.; Osório, F. S.; Wolf, D. F. Template-based autonomous navigation in urban environments. In: Proceedings of the 2011 ACM Symposium on Applied Computing, TaiChung, China, 2011.

[16] Sales, D. O.; Correa, D. S. O.; Osório, F. S.; Wolf, D. F. 3D Vision-based Autonomous Navigation System using ANN and Kinect Sensor. In: Conference Proceedings EANN 2012 – CCIS: Volume number 311., London, UK, 2012.

[17] Wolf, Denis F.; Osório, Fernando S.; Simões, Eduardo; Trindade Jr., Onofre. Robótica Inteligente: Da Simulação às Aplicações no Mundo Real. [Tutorial] In: André Ponce de Leon F. de Carvalho; Tomasz Kowaltowski. (Org.). JAI: Jornada de Atualização em Informática da SBC. Rio de Janeiro: SBC - Editora da PUC. RJ, 2009, v. 1, p. 279-330.

[18] Goebl, M.; Althoff, M.; Buss, M.; Farber, G.; Hecker, F.; Heissing, B.; Kraus, S.; Nagel, R.; Leon, F.P.; Rattei, F.; Russ, M.; Schweitzer, M.; Thuy, M.; Cheng Wang; Wuensche, H.J.; (2008) "Design and capabilities of the Munich Cognitive Automobile". IEEE Intelligent Vehicles Symposium, 2008. Page(s): 1101 – 1107.

[19] Haykin, S. Neural Networks: A Comprehensive Foundation. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1998.

[20] Sales, D; Shinzato, P; Pessin, G; Wolf, D; Osório, F. Vision-based Autonomous Navigation System using ANN and FSM Control. In: Proceedings of the IEEE Latin American Robotics Symposium (LARS), São Bernardo do Campo, Brazil, 2010.

[21] Giles, C. Lee; Horne, Bill G.; LIN, Tsungnan. Learning a class of large finite state machines with a recurrent neural network. Neural Networks 8(9): 1359-1365. 1995.

[22] Omlin, Christian W.; Giles, C. Lee. Constructing Deterministic Finite-State Automata in Recurrent Neural Networks. Jornal of the ACM 43(6): 937-972. 1996.

[23] Frasconi, Paolo; Gori, Marco; Maggini, Marco; Soda, Giovanni. Representation of finite state automata in Recurrent Radial Basis Function networks. Machine Learning 23:1, 5-32 1996.

[24] Cleeremans, Axel; Servan-Schreiber, David; McClelland, James L. Finite State Automata and Simple Recurrent Networks. Neural Computation, Vol. 1, No. 3, Pages 372- 381. 1989.

[25] Sales, D; Feitosa, D; Osório, F; Wolf, D. Multi-Agent Autonomous Patrolling System using ANN and FSM Control. In: Proceedings of the II Brazilian Conference on Critical Embedded Systems (CBSEC), Campinas, Brazil, 2012.

[26] PointGray Bumblebee2 Stereo Cam. Internet: http://www.ptgrey.com/products/stereo.asp [09/2012]

[27] Robot Operating System (ROS). Internet: http://www.ros.org/wiki/ [09/2012]

IROS Workshop on Visual Control of Mobile Robots (ViCoMoR 2012)
October 11th, 2012, Vilamoura, Algarve, Portugal

Accurate Figure Flying with a Quadrocopter Using Onboard Visual and Inertial Sensing

Jakob Engel, Jürgen Sturm, Daniel Cremers

***Abstract*— We present an approach that enables a low-cost quadrocopter to accurately fly various figures using vision as main sensor modality. Our approach consists of three components: a monocular SLAM system, an extended Kalman filter for data fusion and state estimation and a PID controller to generate steering commands. Our system is able to navigate in previously unknown indoor and outdoor environments at absolute scale without requiring artificial markers or external sensors. Next to a full description of our system, we introduce our scripting language and present several examples of accurate figure flying in the corresponding video submission.**

I. Introduction

In recent years, research interest in autonomous micro-aerial vehicles (MAVs) has grown rapidly. Significant progress has been made, recent examples include aggressive flight maneuvers [1, 2], ping-pong [3] and collaborative construction tasks [4]. However, all of these systems require external motion capture systems. Flying in unknown, GPS-denied environments is still an open research problem. The key challenges here are to localize the robot purely from its own sensor data and to robustly navigate it even under potential sensor loss. This requires both a solution to the so-called simultaneous localization and mapping (SLAM) problem as well as robust state estimation and control methods. These challenges are even more expressed on low-cost hardware with inaccurate actuators, noisy sensors, significant delays and limited onboard computation resources.

For solving the SLAM problem on MAVs, different types of sensors such laser range scanners [5], monocular cameras [6, 7], stereo cameras [8] and RGB-D sensors [9] have been explored in the past. In our point of view, monocular cameras provide two major advantages above other modalities: (1) the amount of information that can be acquired is immense compared to their low weight, power consumption, size and cost, which are unmatched by any other type of sensor and (2) in contrast to depth measuring devices, the range of a monocular camera is virtually unlimited – allowing a monocular SLAM system to operate both in small, confined and large, open environments. The drawback however is, that the scale of the environment cannot be determined from monocular vision alone, such that additional sensors (such as an IMU) are required.

The motivation behind our work is to showcase that robust, scale-aware visual navigation is feasible and safe on low-cost robotic hardware. As a platform, we use the Parrot AR.Drone which is available for $300 and, with a weight of only 420 g and a protective hull, safe to be used in public places. As the onboard computational resources are utterly limited, all computations are performed externally.

J. Engel, J. Sturm and D. Cremers are with the Department of Computer Science, Technical University of Munich, Germany {engelj,sturmju,cremers}@in.tum.de

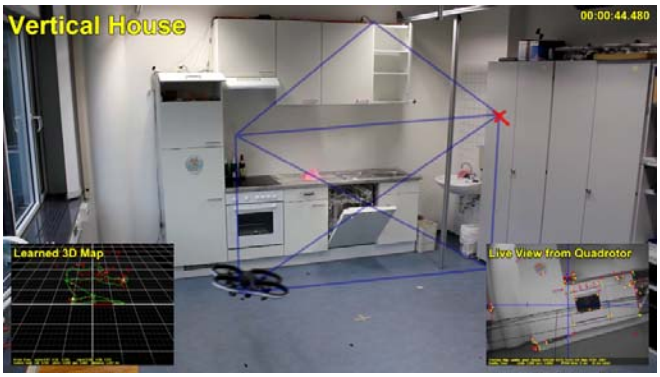


Fig. 1. Our approach enables a low-cost quadrocopter to accurately follow any given flight trajectory. We use the front-facing camera as the main sensor for localization based on PTAM [11]. **Middle:** Flight figure (blue lines) and current goal location (red cross). **Bottom left:** Learned 3D feature map. **Bottom right:** Visual features detected and the live-stream from the quadrocopter.

This paper is an extension of our recently published work [10]. In this work, we additionally describe the scripting language that enables our quadrocopter to complete complex flight patterns including take-off, autonomous map initialization, and landing. This paper comes with two videos, demonstrating the robustness of our approach, its ability to eliminate drift effectively and to follow pre-defined, absolute scale trajectories. They are available online at:

`http://youtu.be/tZxlDly7lno`
`http://youtu.be/eznMokFQmpc`

II. Related Work

Previous work on autonomous flight with quadrocopters can be categorized into different research areas. One part of the community focuses on accurate quadrocopter control and a number of impressive results have been published [12, 1, 3]. These works however rely on advanced external tracking systems, restricting their use to a lab environment. A similar approach is to distribute artificial markers in the environment, simplifying pose estimation [13]. Other approaches learn a map offline from a previously recorded, manual flight and thereby enable a quadrocopter to again fly the same trajectory [14]. For outdoor flights where GPS-based pose estimation is possible, complete solutions are available as commercial products [15].

In this work we focus on autonomous flight without previous knowledge about the environment nor GPS signals, while

using only onboard sensors. First results towards this goal have been presented using a lightweight laser scanner [5], a Kinect [9] or a stereo rig [8] mounted on a quadrocopter as primary sensor. While these sensors provide absolute scale of the environment, their drawback is a limited range and large weight, size and power consumption when compared to a monocular setup [16, 7].

In our work we therefore focus on a monocular camera for pose estimation. Stabilizing controllers based on optical flow were presented in [17], and similar methods are integrated in commercially available hardware [18]. Such systems however are subject to drift over time, and hence not suited for long-term navigation.

To eliminate drift, various monocular SLAM methods have been investigated on quadrocopters, both with off-board [16, 5] and on-board processing [7]. A particular challenge for monocular SLAM is, that the scale of the map needs to be estimated from additional metric sensors such as IMU or GPS, as it cannot be recovered from vision alone. This problem has been addressed in recent publications such as [19, 20]. The current state of the art is to estimate the scale using an extended Kalman filter (EKF), which contains scale and offset in its state. In contrast to this, we propose a novel approach which is based on direct computation: Using a statistical formulation, we derive a closed-form, consistent estimator for the scale of the visual map. Our method yields accurate results both in simulation and practice, and requires less computational resources than filtering. It can be used with any monocular SLAM algorithm and sensors providing metric position or velocity measurements, such as an ultrasonic or pressure altimeter or occasional GPS measurements.

In contrast to the systems presented in [16, 7], we deliberately refrain from using expensive, customized hardware: the only hardware required is the AR.Drone, which comes at a costs of merely $300 – a fraction of the cost of quadrocopters used in previous work. Released in 2010 and marketed as high-tech toy, it has been used and discussed in several research projects [21, 22, 23]. To our knowledge, we are the first to present a complete implementation of autonomous, camera-based flight in unknown, unstructured environments using the AR.Drone.

III. Hardware Platform

As platform we use the Parrot AR.Drone, a commercially available quadrocopter. Compared to other modern MAV's such as Ascending Technology's Pelican or Hummingbird quadrocopters, its main advantage is the very low price, its robustness to crashes and the fact that it can safely be used indoor and close to people. This however comes at the price of flexibility: Neither the hardware itself nor the software running onboard can easily be modified, and communication with the quadrocopter is only possible over wireless LAN. With battery and hull, the AR.Drone measures 53 cm × 52 cm and weights 420 g.

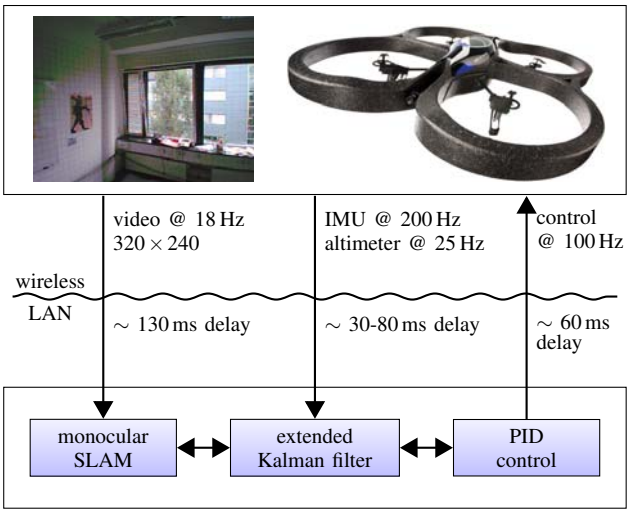


Fig. 2. Approach Outline: Our navigation system consists of three major components: a monocular SLAM implementation for visual tracking, an EKF for data fusion and prediction, and PID control for pose stabilization and navigation. All computations are performed offboard, which leads to significant, varying delays which our approach has to compensate.

A. Sensors

The AR.Drone is equipped with a 3-axis gyroscope and accelerometer, an ultrasound altimeter and two cameras. The first camera is aimed forward, covers a field of view of $73.5^\circ \times 58.5^\circ$, has a resolution of 320×240 and a rolling shutter with a delay of 40 ms between the first and the last line captured. The video of the first camera is streamed to a laptop at 18 fps, using lossy compression. The second camera aims downward, covers a field of view of $47.5^\circ \times 36.5^\circ$ and has a resolution of 176×144 at 60 fps. The onboard software uses the down-looking camera to estimate the horizontal velocity. The quadcopter sends gyroscope measurements and the estimated horizontal velocity at 200 Hz, the ultrasound measurements at 25 Hz to the laptop. The raw accelerometer data cannot be accessed directly.

B. Control

The onboard software uses these sensors to control the roll Φ and pitch Θ, the yaw rotational speed $\dot{\Psi}$ and the vertical velocity $\dot{z}$ of the quadrocopter according to an external reference value. This reference is set by sending a new control command $\mathbf{u} = (\bar{\Phi}, \bar{\Theta}, \bar{\dot{z}}, \bar{\dot{\Psi}}) \in [-1, 1]^4$ every 10 ms.

IV. Approach

Our approach consists of three major components running on a laptop connected to the quadrocopter via wireless LAN, an overview is given in Fig. 2.

1) **Monocular SLAM:** For monocular SLAM, our solution is based on Parallel Tracking and Mapping (PTAM) [11]. After map initialization, we rotate the visual map such that the *xy*-plane corresponds to the horizontal plane according to the accelerometer data, and scale it such that the average keypoint depth is 1. Throughout tracking, the scale of the map $\lambda \in \mathbb{R}$ is estimated using a novel method described in Section IV-A. Furthermore, we use the pose estimates from the EKF to identify and reject falsely tracked frames.

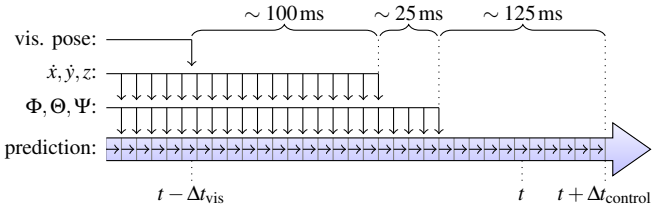


Fig. 3. Pose Prediction: Measurements and control commands arrive with significant delays. To compensate for these delays, we keep a history of observations and sent control commands between $t - \Delta t_{\text{vis}}$ and $t + \Delta t_{\text{control}}$ and re-calculate the EKF state when required. Note the large timespan with no or only partial odometry observations.

2) **Extended Kalman Filter:** In order to fuse all available data, we employ an extended Kalman filter (EKF). We derived and calibrated a full motion model of the quadrocopter's flight dynamics and reaction to control commands, which we will describe in more detail in Section IV-B. This EKF is also used to compensate for the different time delays in the system, arising from wireless LAN communication and computationally complex visual tracking.

We found that height and horizontal velocity measurements arrive with the same delay, which is slightly larger than the delay of attitude measurements. The delay of visual pose estimates Δt_{vis} is by far the largest. Furthermore we account for the time required by a new control command to reach the drone $\Delta t_{\text{control}}$. All timing values given subsequently are typical values for a good connection, the exact values depend on the wireless connection quality and are determined by a combination of regular ICMP echo requests sent to the quadrocopter and calibration experiments.

Our approach works as follows: first, we time-stamp all incoming data and store it in an observation buffer. Control commands are then calculated using a prediction for the quadrocopter's pose at $t + \Delta t_{\text{control}}$. For this prediction, we start with the saved state of the EKF at $t - \Delta t_{\text{vis}}$ (i.e., after the last visual observation/unsuccessfully tracked frame). Subsequently, we predict ahead up to $t + \Delta t_{\text{control}}$, using previously issued control commands and integrating stored sensor measurements as observations. This is illustrated in Fig. 3. With this approach, we are able to compensate for delayed and missing observations at the expense of recalculating the last cycles of the EKF.

3) **PID Control:** Based on the position and velocity estimates from the EKF at $t + \Delta t_{\text{control}}$, we apply PID control to steer the quadrocopter towards the desired goal location $\mathbf{p} = (\hat{x}, \hat{y}, \hat{z}, \hat{\Psi})^T \in \mathbb{R}^4$ in a global coordinate system. According to the state estimate, we rotate the generated control commands to the robot-centric coordinate system and send them to the quadrocopter. For each of the four degrees-of-freedom, we employ a separate PID controller for which we experimentally determined suitable controller gains.

A. Scale Estimation

One of the key contributions of this paper is a closed-form solution for estimating the scale $\lambda \in \mathbb{R}^+$ of a monocular SLAM system. For this, we assume that the robot is able to make noisy measurements of absolute distances or velocities from additional, metric sensors such as an ultrasound altimeter.

As a first step, the quadrocopter measures in regular intervals the d-dimensional distance traveled both using only the visual SLAM system (subtracting start and end position) and using only the metric sensors available (subtracting start and end position, or integrating over estimated speeds). Each interval gives a pair of samples $\mathbf{x}_i, \mathbf{y}_i \in \mathbb{R}^d$, where $\mathbf{x}_i$ is scaled according to the visual map and $\mathbf{y}_i$ is in metric units. As both $\mathbf{x}_i$ and $\mathbf{y}_i$ measure the motion of the quadrocopter, they are related according to $\mathbf{x}_i \approx \lambda \mathbf{y}_i$.

More specifically, if we assume Gaussian noise in the sensor measurements with constant variance[1], we obtain

$$\mathbf{x}_i \sim \mathcal{N}(\lambda \boldsymbol{\mu}_i, \sigma_x^2 \mathbf{I}_{3\times 3}) \tag{1}$$

$$\mathbf{y}_i \sim \mathcal{N}(\boldsymbol{\mu}_i, \sigma_y^2 \mathbf{I}_{3\times 3}) \tag{2}$$

where the $\boldsymbol{\mu}_i \in \mathbb{R}^d$ denote the true (unknown) distances covered and $\sigma_x^2, \sigma_y^2 \in \mathbb{R}^+$ the variances of the measurement errors. Note that the individual $\boldsymbol{\mu}_i$ are not constant but depend on the actual distances traveled by the quadrocopter in the measurement intervals.

One possibility to estimate λ is to minimize the sum of squared differences (SSD) between the re-scaled measurements, i.e., to compute one of the following:

$$\lambda_y^* := \arg\min_{\lambda} \sum_i \|\mathbf{x}_i - \lambda \mathbf{y}_i\|^2 = \frac{\sum_i \mathbf{x}_i^T \mathbf{y}_i}{\sum_i \mathbf{y}_i^T \mathbf{y}_i} \tag{3}$$

$$\lambda_x^* := \left(\arg\min_{\lambda} \sum_i \|\lambda \mathbf{x}_i - \mathbf{y}_i\|^2 \right)^{-1} = \frac{\sum_i \mathbf{x}_i^T \mathbf{x}_i}{\sum_i \mathbf{x}_i^T \mathbf{y}_i}. \tag{4}$$

The difference between these two lines is whether one aims at scaling the $\mathbf{x}_i$ to the $\mathbf{y}_i$ or vice versa. However, both approaches lead to different results, none of which converges to the true scale λ when adding more samples. To resolve this, we propose a maximum likelihood (ML) approach, that is estimating λ by minimizing the negative log-likelihood

$$\mathcal{L}(\boldsymbol{\mu}_1 \dots \boldsymbol{\mu}_n, \lambda) \propto \frac{1}{2} \sum_{i=1}^{n} \left(\frac{\|\mathbf{x}_i - \lambda \boldsymbol{\mu}_i\|^2}{\sigma_x^2} + \frac{\|\mathbf{y}_i - \boldsymbol{\mu}_i\|^2}{\sigma_y^2} \right) \tag{5}$$

By first minimizing over the $\boldsymbol{\mu}_i$ and then over λ, it can be shown analytically that (5) has a unique, global minimum at

$$\boldsymbol{\mu}_i^* = \frac{\lambda^* \sigma_y^2 \mathbf{x}_i + \sigma_x^2 \mathbf{y}_i}{\lambda^{*2} \sigma_y^2 + \sigma_x^2} \tag{6}$$

$$\lambda^* = \frac{s_{xx} - s_{yy} + \text{sign}(s_{xy}) \sqrt{(s_{xx} - s_{yy})^2 + 4 s_{xy}^2}}{2 \sigma_x^{-1} \sigma_y s_{xy}} \tag{7}$$

with $s_{xx} := \sigma_y^2 \sum_{i=1}^n \mathbf{x}_i^T \mathbf{x}_i$, $s_{yy} := \sigma_x^2 \sum_{i=1}^n \mathbf{y}_i^T \mathbf{y}_i$ and $s_{xy} := \sigma_y \sigma_x \sum_{i=1}^n \mathbf{x}_i^T \mathbf{y}_i$. Together, these equations give a closed-form solution for the ML estimator of λ, assuming the measurement error variances σ_x^2 and σ_y^2 are known.

[1]The noise in $\mathbf{x}_i$ does not depend on λ as it is proportional to the average keypoint depth, which is normalized to 1 for the first keyframe.

B. State Prediction and Observation

In this section, we describe the state space, the observation models and the motion model used in the EKF. The state space consists of a total of ten state variables

$$\mathbf{x}_t := (x_t, y_t, z_t, \dot{x}_t, \dot{y}_t, \dot{z}_t, \Phi_t, \Theta_t, \Psi_t, \dot{\Psi}_t)^T \in \mathbb{R}^{10}, \quad (8)$$

where (x_t, y_t, z_t) denotes the position of the quadrocopter in m and $(\dot{x}_t, \dot{y}_t, \dot{z}_t)$ the velocity in m/s, both in world coordinates. Further, the state contains the roll Φ_t, pitch Θ_t and yaw Ψ_t angle of the drone in deg, as well as the yaw-rotational speed $\dot{\Psi}_t$ in deg/s. In the following, we define for each sensor an observation function $h(\mathbf{x}_t)$ and describe how the respective observation vector $\mathbf{z}_t$ is composed from the sensor readings.

1) **Odometry Observation Model:** The quadrocopter measures its horizontal speed $\hat{v}_{x,t}$ and $\hat{v}_{y,t}$ in its local coordinate system, which we transform into the global frame $\dot{x}_t$ and $\dot{y}_t$. The roll and pitch angles $\hat{\Phi}_t$ and $\hat{\Theta}_t$ measured by the accelerometer are direct observations of Φ_t and Θ_t. To account for yaw-drift and uneven ground, we differentiate the height measurements $\hat{h}_t$ and yaw measurements $\hat{\Psi}_t$ and treat them as observations of the respective velocities. The resulting observation function $h_{\mathrm{I}}(\mathbf{x}_t)$ and measurement vector $\mathbf{z}_{\mathrm{I},t}$ is hence given by

$$h_{\mathrm{I}}(\mathbf{x}_t) := \begin{pmatrix} \dot{x}_t \cos\Psi_t - \dot{y}_t \sin\Psi_t \\ \dot{x}_t \sin\Psi_t + \dot{y}_t \cos\Psi_t \\ \dot{z}_t \\ \Phi_t \\ \Theta_t \\ \dot{\Psi}_t \end{pmatrix} \quad (9)$$

$$\mathbf{z}_{\mathrm{I},t} := (\hat{v}_{x,t}, \hat{v}_{y,t}, (\hat{h}_t - \hat{h}_{t-1}), \hat{\Phi}_t, \hat{\Theta}_t, (\hat{\Psi}_t - \hat{\Psi}_{t-1}))^T \quad (10)$$

2) **Visual Observation Model:** When PTAM successfully tracks a video frame, we scale the pose estimate by the current estimate for the scaling factor λ^* and transform it from the coordinate system of the front camera to the coordinate system of the quadrocopter, leading to a direct observation of the quadrocopter's pose given by

$$h_{\mathrm{P}}(\mathbf{x}_t) := (x_t, y_t, z_t, \Phi_t, \Theta_t, \Psi_t)^T \quad (11)$$

$$\mathbf{z}_{\mathrm{P},t} := f(\mathbf{E}_{\mathcal{DC}} \mathbf{E}_{\mathcal{C},t}) \quad (12)$$

where $\mathbf{E}_{\mathcal{C},t} \in \mathrm{SE}(3)$ is the estimated camera pose (scaled with λ), $\mathbf{E}_{\mathcal{DC}} \in \mathrm{SE}(3)$ the constant transformation from the camera to the quadrocopter coordinate system, and $f : \mathrm{SE}(3) \to \mathbb{R}^6$ the transformation from an element of SE(3) to our roll-pitch-yaw representation.

3) **Prediction Model:** The prediction model describes how the state vector $\mathbf{x}_t$ evolves from one time step to the next. In particular, we approximate the quadrocopter's horizontal acceleration $\ddot{x}, \ddot{y}$ based on its current state $\mathbf{x}_t$, and estimate its vertical acceleration $\ddot{z}$, yaw-rotational acceleration $\ddot{\Psi}$ and roll/pitch rotational speed $\dot{\Phi}, \dot{\Theta}$ based on the state $\mathbf{x}_t$ and the active control command $\mathbf{u}_t$.

The horizontal acceleration is proportional to the horizontal force acting upon the quadrocopter, which is given by

$$\begin{pmatrix} \ddot{x} \\ \ddot{y} \end{pmatrix} \propto \mathbf{f}_{\mathrm{acc}} - \mathbf{f}_{\mathrm{drag}} \quad (13)$$

where $\mathbf{f}_{\mathrm{drag}}$ denotes the drag and $\mathbf{f}_{\mathrm{acc}}$ denotes the accelerating force. The drag is approximately proportional to the horizontal velocity of the quadrocopter, while $\mathbf{f}_{\mathrm{acc}}$ depends on the tilt angle. We approximate it by projecting the quadrocopter's z-axis onto the horizontal plane, which leads to

$$\ddot{x}(\mathbf{x}_t) = c_1 (\cos\Psi_t \sin\Phi_t \cos\Theta_t - \sin\Psi_t \sin\Theta_t) - c_2 \dot{x}_t \quad (14)$$

$$\ddot{y}(\mathbf{x}_t) = c_1 (-\sin\Psi_t \sin\Phi_t \cos\Theta_t - \cos\Psi_t \sin\Theta_t) - c_2 \dot{y}_t \quad (15)$$

We estimated the proportionality coefficients c_1 and c_2 from data collected in a series of test flights. Note that this model assumes that the overall thrust generated by the four rotors is constant. Furthermore, we describe the influence of sent control commands $\mathbf{u}_t = (\bar{\Phi}_t, \bar{\Theta}_t, \bar{\dot{z}}_t, \bar{\dot{\Psi}}_t)$ by a linear model:

$$\dot{\Phi}(\mathbf{x}_t, \mathbf{u}_t) = c_3 \bar{\Phi}_t - c_4 \Phi_t \quad (16)$$

$$\dot{\Theta}(\mathbf{x}_t, \mathbf{u}_t) = c_3 \bar{\Theta}_t - c_4 \Theta_t \quad (17)$$

$$\ddot{\Psi}(\mathbf{x}_t, \mathbf{u}_t) = c_5 \bar{\dot{\Psi}}_t - c_6 \dot{\Psi}_t \quad (18)$$

$$\ddot{z}(\mathbf{x}_t, \mathbf{u}_t) = c_7 \bar{\dot{z}}_t - c_8 \dot{z}_t \quad (19)$$

Again, we estimated the coefficients $c_3, \ldots, c_8$ from test flight data. The overall state transition function is now given by

$$\begin{pmatrix} x_{t+1} \\ y_{t+1} \\ z_{t+1} \\ \dot{x}_{t+1} \\ \dot{y}_{t+1} \\ \dot{z}_{t+1} \\ \Phi_{t+1} \\ \Theta_{t+1} \\ \Psi_{t+1} \\ \dot{\Psi}_{t+1} \end{pmatrix} \leftarrow \begin{pmatrix} x_t \\ y_t \\ z_t \\ \dot{x}_t \\ \dot{y}_t \\ \dot{z}_t \\ \Phi_t \\ \Theta_t \\ \Psi_t \\ \dot{\Psi}_t \end{pmatrix} + \delta_t \begin{pmatrix} \dot{x}_t \\ \dot{y}_t \\ \dot{z}_t \\ \ddot{x}(\mathbf{x}_t) \\ \ddot{y}(\mathbf{x}_t) \\ \ddot{z}(\mathbf{x}_t, \mathbf{u}_t) \\ \dot{\Phi}(\mathbf{x}_t, \mathbf{u}_t) \\ \dot{\Theta}(\mathbf{x}_t, \mathbf{u}_t) \\ \dot{\Psi}_t \\ \ddot{\Psi}(\mathbf{x}_t, \mathbf{u}_t) \end{pmatrix} \quad (20)$$

using the model specified in (14) to (19). Note that, due to the many assumptions made, we do not claim the physical correctness of this model. It however performs very well in practice, which is mainly due to its completeness: the behavior of all state parameters and the effect of all control commands is approximated, allowing "blind" prediction, i.e., prediction without observations for a brief period of time (~ 125 ms in practice, see Fig. 3).

V. Experiments and Results

We conducted a series of real-world experiments to analyze the properties of the resulting system. The experiments were conducted in different environments, i.e., both indoor in rooms of varying size and visual appearance as well as outdoor under the influence of sunlight and (slight) wind. A selection of these environments is depicted in Fig. 4.

A. Scale Estimation Accuracy

To analyze the accuracy of the scale estimation method derived in IV-A, we instructed the quadrocopter to fly a fixed figure, while every second a new sample is taken and the scale re-estimated. In the first set of flights, the quadrocopter was commanded to move only vertically, such that the samples mostly consist of altitude measurements. In the second set, the quadrocopter was commanded to fly

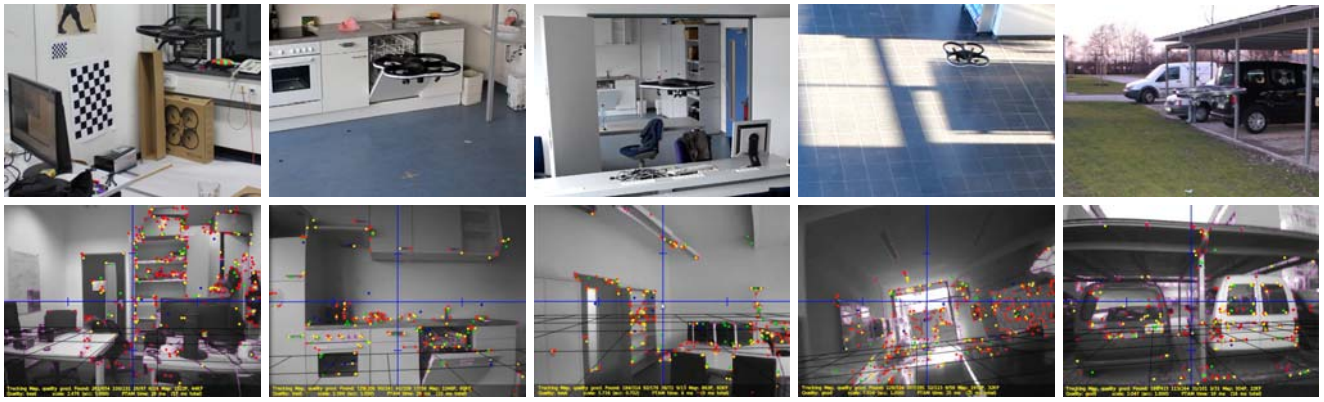


Fig. 4. Testing Environments: The top row shows an image of the quadrocopter flying, the bottom row the corresponding image from the quadrocopter's frontal camera. This shows that our system can operate robustly in different, real-world environments.

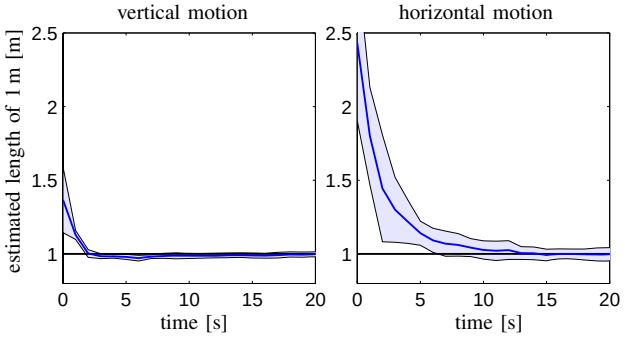


Fig. 5. Scale Estimation Accuracy: The plots show the mean and standard deviation of the the estimation error e, corresponding to the estimated length of 1 m, from horizontal and vertical motion. It can be seen that the scale can be estimated accurately in both cases, it is however more accurate and converges faster if the quadrocopter moves vertically.

a horizontal rectangle, such that primarily the IMU-based velocity information is used. After each flight, we measured the ground truth $\hat{\lambda}$ by manually placing the quadrocopter at two measurement points, and comparing the known distance between these points with the distance measured by the visual SLAM system. Note that due to the initial scale normalization, the values for $\hat{\lambda}$ roughly correspond to the mean feature depth in meters of the first keyframe, which in our experiments ranges from 2 m to 10 m. To provide better comparability, we analyze and visualize the estimation error $e := \frac{\lambda^*}{\hat{\lambda}}$, corresponding to the estimated length of 1 m.

Fig. 5 gives the mean error as well as the standard deviation spread over 10 flights. As can be seen, our method quickly and accurately estimates the scale from both types of motion. Due to the superior accuracy of the altimeter compared to the horizontal velocity estimates, the estimate converges faster and is more accurate if the quadrocopter moves vertically, i.e., convergence after 2 s versus 15 s, and to a final accuracy $\pm 1.7\,\%$ versus $\pm 5\,\%$. Note that in practice, we allow for (and recommend) arbitrary motions during scale estimation so that information from both sensor modalities can be used to improve convergence. Large, sudden changes in measured relative height can be attributed to uneven ground, and removed automatically from the data set.

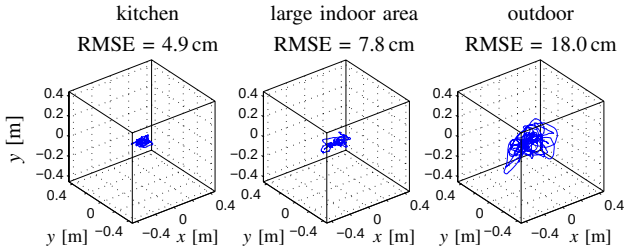


Fig. 6. Flight Stability: Path taken and RMSE of the quadrocopter when instructed to hold a target position for 60 s, in three of the environments depicted in Fig. 4. It can be seen that the quadrocopter can hold a position very accurately, even when perturbed by wind (right).

B. Positioning Accuracy

In this section, we evaluate the performance of the complete system in terms of position control. We instructed the quadrocopter to hold a target position over 60 s in different environments and measure the root mean square error (RMSE). The results are given in Fig. 6: the measured RMSE lies between 4.9 cm (indoor) and 18.0 cm (outdoor).

C. Drift Elimination

To verify that the incorporation of a visual SLAM system eliminates odometry drift, we compare the estimated trajectory with and without the visual SLAM system. Fig. 7 shows the resulting paths, both for flying a fixed figure (left) and for holding a target position while the quadrocopter is being pushed away (right). Both flights took approximately 35 s, and the quadrocopter landed no more than 15 cm away from its takeoff position. In contrast, the raw odometry accumulated an error of 2.1 m for the fixed figure and 6 m when being pushed away - which is largely due to the relative lack of texture on the floor. This experiment demonstrates that the visual SLAM system efficiently eliminates pose drift during maneuvering.

D. Figure Flying

Based on the accurate pose estimation and control, we implemented a simple scripting language for flying pre-specified maneuvers. Available commands in this scripting language include take-off, landing, automatic initialization

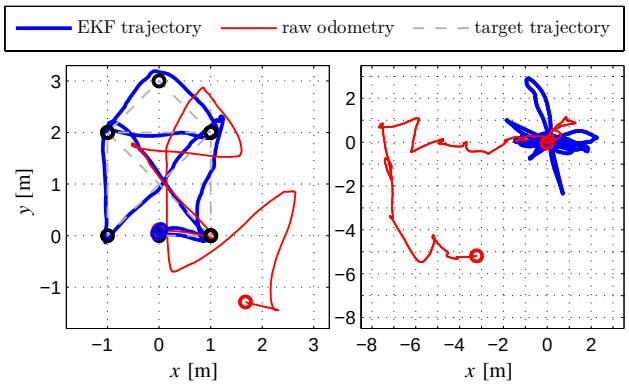


Fig. 7. Elimination of Odometry Drift: Horizontal path taken by the quadrocopter as estimated by the EKF compared to the raw odometry (i.e., the integrated velocity estimates). Left: when flying a figure; right: when being pushed away repeatedly from its target position. The odometry drift is clearly visible, in particular when the quadrocopter is being pushed away. When incorporating visual pose estimates, it is eliminated completely.

(take-off + PTAM map initialization) and approaching a waypoint, both in absolute coordinates (with the world origin at the take-off location) as well as relative to the current location. Various parameters can be set during the flight, for example to re-define the world origin, limit flight speed, and the parameters for approaching a waypoint. A waypoint has been "reached" after the quadrocopter reaches and remains within a certain distance (default: 0.5 m) for a certain time (default: 2.0 s). With this scripting language, we were able to let the quadrocopter autonomously complete a large variety of different figures including take-off and map initialization, for example a rectangle and the "Haus vom Nikolaus" as depicted in Fig. 1, both vertically and horizontally. Demonstrations of these flight patterns are also shown in the video accompanying this paper.

VI. Conclusion

In this paper, we presented a visual navigation system for autonomous figure flying. Our system enables the quadrocopter to visually navigate in unstructured, GPS-denied environments and does not require artificial landmarks nor prior knowledge about the environment. We tested our system in a set of extensive experiments in different real-world indoor and outdoor environments and with different flight patterns. We found in these experiments, that our system achieves an average positioning accuracy of 4.9 cm (indoor) to 18.0 cm (outdoor) and can robustly deal with communication delays of up to 400 ms. With these experiments, we demonstrated that accurate, robust and drift-free visual figure flights are possible.

We plan to release our software as open-source in the near future.

References

[1] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2011.

[2] S. Lupashin, A. Schöllig, M. Sherback, and R. D'Andrea, "A simple learning strategy for high-speed quadrocopter multi-flips." in *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2010.

[3] M. Müller, S. Lupashin, and R. D'Andrea, "Quadrocopter ball juggling," in *Proc. IEEE Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2011.

[4] Q. Lindsey, D. Mellinger, and V. Kumar, "Construction of cubic structures with quadrotor teams," in *Proceedings of Robotics: Science and Systems (RSS)*, Los Angeles, CA, USA, 2011.

[5] S. Grzonka, G. Grisetti, and W. Burgard, "Towards a navigation system for autonomous indoor flying," in *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2009.

[6] M. Blösch, S. Weiss, D. Scaramuzza, and R. Siegwart, "Vision based MAV navigation in unknown and unstructured environments," in *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2010.

[7] M. Achtelik, M. Achtelik, S. Weiss, and R. Siegwart, "Onboard IMU and monocular vision based control for MAVs in unknown in- and outdoor environments," in *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2011.

[8] M. Achtelik, A. Bachrach, R. He, S. Prentice, and N. Roy, "Stereo vision and laser odometry for autonomous helicopters in GPS-denied indoor environments," in *Proc. SPIE Unmanned Systems Technology XI*, 2009.

[9] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy, "Visual odometry and mapping for autonomous flight using an RGB-D camera," in *Proc. IEEE International Symposium of Robotics Research (ISRR)*, 2011.

[10] J. Engel, J. Sturm, and D. Cremers, "Camera-based navigation of a low-cost quadrocopter," in *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[11] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Proc. IEEE Intl. Symposium on Mixed and Augmented Reality (ISMAR)*, 2007.

[12] D. Mellinger, N. Michael, and V. Kumar, "Trajectory generation and control for precise aggressive maneuvers with quadrotors," in *Proceedings of the Intl. Symposium on Experimental Robotics*, Dec 2010.

[13] D. Eberli, D. Scaramuzza, S. Weiss, and R. Siegwart, "Vision based position control for MAVs using one single circular landmark," *Journal of Intelligent and Robotic Systems*, vol. 61, pp. 495–512, 2011.

[14] T. Krajník, V. Vonásek, D. Fišer, and J. Faigl, "AR-drone as a platform for robotic research and education," in *Proc. Research and Education in Robotics: EUROBOT 2011*, 2011.

[15] "Ascending technologies," 2012. [Online]: http://www.asctec.de/

[16] M. Blösch, S. Weiss, D. Scaramuzza, and R. Siegwart, "Vision based MAV navigation in unknown and unstructured environments," in *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2010.

[17] S. Zingg, D. Scaramuzza, S. Weiss, and R. Siegwart, "MAV navigation through indoor corridors using optical flow," in *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2010.

[18] "Parrot AR.Drone," 2012. [Online]: http://ardrone.parrot.com/

[19] S. Weiss and R. Siegwart, "Real-time metric state estimation for modular vision-inertial systems," in *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2011.

[20] G. Nützi, S. Weiss, D. Scaramuzza, and R. Siegwart, "Fusion of IMU and vision for absolute scale estimation in monocular SLAM," *Journal of Intelligent Robotic Systems*, vol. 61, pp. 287 – 299, 2010.

[21] C. Bills, J. Chen, and A. Saxena, "Autonomous MAV flight in indoor environments using single image perspective cues," in *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2011.

[22] T. Krajník, V. Vonásek, D. Fišer, and J. Faigl, "AR-drone as a platform for robotic research and education," in *Proc. Communications in Computer and Information Science (CCIS)*, 2011.

[23] W. S. Ng and E. Sharlin, "Collocated interaction with flying robots," in *Proc. IEEE Intl. Symposium on Robot and Human Interactive Communication*, 2011.