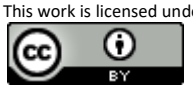
Available online at: https://ijact.in

Date of Submission	29/06/2020
Date of Acceptance	27/08/2020
Date of Publication	10/09/2020
Page numbers	3798-3809 (12 Pages)



ISSN:2320-0790

An International Journal of Advanced Computer Technology

NEURAL NETWORK VISUAL ODOMETRY BASED FRAMEWORK FOR UAV LOCALIZATION IN GPS DENIED ENVIRONMENT

[1]Mohamed Ali SEDRINE, [2]Wided SOUIDENE MSEDDI, [3]Rabah ATTIA
Tunisia Polytechnic School, SERCOM Lab, Tunisia Polytechnic School, Carthage University, La Marsa, Tunisia & L2TI, Paris 13 University, Paris, France
[1]sedrinedali@gmail.com, [2]wided.souidene@ept.u-carthage.tn, [3]rabah.attia@enit.rnu.tn

Abstract: This paper presents a vision-based localization framework based on visual odometry.Visual odometry is a classic approach to incrementally estimate robot motion even in GPS denied environment, by tracking features in successive images. As it is subject to drift, this paper proposes to call a convolutional neural netwok and visual memory to improve process accuracy.
In fact, our framework is made of two main steps. First, the robot builds its visual memory by annotating places with their ground truth positions. Dedicated data structures are made to store referenced images and their positions. Then, during navigation step, we use loop closure corrected visual odometry. A siamese convolutional neural network allows us to detect already visited positions. It takes as input current image and an already stored one. If the place is recognized, the drift is then quantified using the stored position. Drift correction is conducted by an original two levels correction process. The first level is directly applied to the estimation by substracting the error. The second level is applied to the graph itself using iterative closest point method, to match the estimated trajectory graph to the ground truth one.
Experiments showed that the proposed localization method has a centimetric accuracy.

*Keywords:*visual odometry, localization, loop-closure, deep learning, convolutional neural network

I. INTRODUCTION

Now a days autonomous vehicles are developed for various applications such as agriculture, transportation and security. In order to improve their performances and accuracy, communities focused their attention on autonomous vehicles. Despite these efforts, mobile robots location and orientation estimation is still a main issue. However, available commercial products depend on GPS to perform this task. Such sensors suffer from lack of precision, unavailability and can be jammed. In fact, commercial GPS systems generate errors of the order of meters and can't be reliable in critic tasks. GPS with better percision are expensive. Besides, positionning systems and satellites are goverments and states property which is a strategic matter.

Regarding these constraints, other alternatives could be adopted like vision based approaches. The main advantage of visual sensors is that they are inexpensive and available. Existing works adopt a wide variety camera based approaches. Monocular, stereo or omnidirectional cameras were used. Besides, visual odometry and visual Simultaneous Localization and Mapping (SLAM) are most common algorithms. While SLAM aims to build robot surroundings model and localize it locally, visual odometry calculates robot egomotion by incrementing relative rotations and translations estimations. These estimations are made by extracting features from incoming frames and matching them. This process leads to incremental drift growth. To decrease the drift, some methods consist in fusing visual odometry with other sensors like Inertial Measurment Units (IMUs) or gyros using filters. Other

approaches exploit loop closure detection method to estimate error and reduce it.

Namely, such techniques use loop closure as a constraint to improve future estimations. This constraint could be applied to the process itself or to generated graph. Loop closure detection could also be seen as place recognition. It is based on learning a database of georeferenced images then the unmanned vehicle has to match between current incoming frame and this database. Some works used visual Bag-of-Words to build the reference database, others used Deep Learning approaches.

In this paper, we propose a framework which aims to perform visual odometry correction based on loop closure detection. The loop closure detection is made using a pre-trained CNN, presented in a previous paper [19]. Then visual place recognition will be performed in order to estimate drift and correct it. Our idea is to perform two-level correction : graph-level and process-level.

II. RELATED WORKS

Visual Odometry methods were developed for decades and were applied for many purposes like augmented reality and autonomous vehicles. Recently, many works paid attention to this technique. [1] and [2] could be considered as important works as they explain different implementations and applications of visual odometry. A more recent work [3] detailed the evolution of visual odometry.

In fact, visual odometry is a method that estimates the vehicle pose using video stream input. It measures image features transformation between successive frames. Such technique is cost-efficient and versatile. Many methods have been developed depending on sensors types (monocular [4], stereo [5] and omnidirectional [6]) and on extracted features from frames (feature-based approaches as in [7] and apparance-based approaches as in [8]).

However, visual odometry suffers from increasing drift. These errors have many reasons. First of all, imaging conditions such as light, blur, textures lead to a low estimation accuracy, due to the lack of informations in the image and unaccurate pixel displacement measurement. Add to that, the calibration errors are accumulated at each iteration. Another limitation of visual odometry process, is its unability to estimate scale, particularly for monocular variant like it is noted in [9]. All these drawbacks and limitations led researchers to look for solutions to reduce the drift.

[2] explains drift propagation and presents some ways to reduce the errors. One way is to use pose optimization. It consists in using known successive transformations as constraints. Loop closure is a constraint that allows to estimate the drift when a place is revisited. Papers [10] and [11] use different approaches to detect loop closure, and different ways to estimate and to correct the drift. Another way to reduce the errors is to use filtering methods, in order to merge sensors datas, like in [12] and [13].

In our proposed method, we use loop closure constraint to reduce drift .We can classify loop closure detection methods in two categories. Appearance-based approaches like [10], [11], [14] and [15]. The second category is more recent and based on Convolution Neural Networks (CNN), like [16] and [17].

In [10], the correction is based on estimating the drift between the pose currently estimated by the visual odometry algorithm and the pose given when the place was previously visited. The main idea of [11] is to measure transformation between first and last frame. If it is equal to identity then there is loop closure. If not, the correction is made by back propagating error. In [17], a deep CNN is used (MobileNetV2) to extract incoming frame features. These features are coupled to SURF [18] features to gain invariance. A distance between CNN features is then calculated to select loop closure detection candidates. Then, using SURF features and hash function, these candidates are then filtered to generate final loop closure detection.

In our previous work [19], we compared different architectures in performing loop-closure detection. It shows that Siamese ResNet has a good accuracy, up to 94%.

In this paper we were inspired by [20] and [21]in the implemention of a visual memory. Yet, our method uses visual odometry as position estimation technique. Besides, our key frame selection criterion is based on the traveled distance. In fact, authors in [20] introduce a vision based navigation and guidance system. It uses the scenes appearances as descriptor and stores them. The system needs a learning stage that acquires a set of key frames to build visual reference path. The set of key frames is then used during a localization stage when the vehicle is in autonomous mode. In the other hand, [21] presents a vision navigation system based on three stages which are memory building, localization and servoing. It uses also a set of key frames to build a visual route. Athors used Harris corner detector [22] as a criterion to guide key frame selection. The memorized visual route is explored to localize the robot.

Finally, [23] presents a hybrid visual odometry method for Micro Aerial Vehicles (MAVs). Besides, it implements mapping task. In fact, the framework is made of two parallel threads, one for estimating the camera motion, and a second one for mapping, like in [24]. The camera motion is estimated through sparse model-based image alignment which seeks to find the transformation that minimizes the photometric error. It tracks patches brightness and gradient information.

Through this work, we seek to introduce an original and novel framework implementing monocular feature-based visual odometry, combined with CNN to achieve loop-closure detection. Our approach is structured in two stages, the first one for learning and the second one for localization. We don't focus on control step in this work. The next section is dedicated to detail our method.

III. PROPOSED APPROACH

A. Overview

Our approach is focused on feature-based monocular visual odometry which is improved using external sensors and constrained by loop closure detection. It is based on two main steps.

The first one is **Learning Step**, where the autonomous vehicle builds its visual memory. This memory is made of key frames and positions. It is needed to improve the second step which is the **Localization Step**.
The second step core is visual odometry (Fig.1). The visual odometry, here, has two main objectives : it estimates robot's 2D positions and altitude information from altimeter and turns2D position estimation into 3D position estimation. As monocular visual odometry is unable to guess the scale, i.e traveled distance between two successive images [1], we extract velocity information from speed sensor to ensure traveled distance estimation. This information allows to know the traveled distance between two captures. Besides, in order to reduce visual odometry drift, a loop closure detection module is called. This one is based on a CNN (Siamese ResNet), it has been presented in our previous paper [19]. In fact, during localization step, the loop closure detection system compares incoming frame with key frames stored in the already built visual memory. The comparison is supervised by the estimated position to decrease processing time. Once a matching is detected by the system, we apply a two stage correction. A drift correction is applied into visual odometry core, and the Localization estimation graph is then optimized using Iterative Closest Point (ICP) method.

Through this work, we propose not only to tackle visual odometry correction methods but also an original localization method for mobile robots in general and for UAVs in particular. The novelty in this work is the use of CNN in loop closure detection besides the two level correction. In the other hand, unlike [20] and [21], we make use of visual odometry as core process of localization step. Next subsections are dedicated to highlight each step of our proposed approach.

B. Loop Closure Detection Based On CNN

As presented in [19], we selected state-of-the-art CNNs which are AlexNet [25], GoogLeNet [26] and ResNet [27]. We chose these networks because they are ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners. Loop closure detection can be seen as binary classification. Such task could be performed by Siamese networks [28] and Multi-channel networks [29]. That's why we implemented for each network a Siamese and a Multi-channel version. Then we obtain six CNNs for which we compared the respective performances.

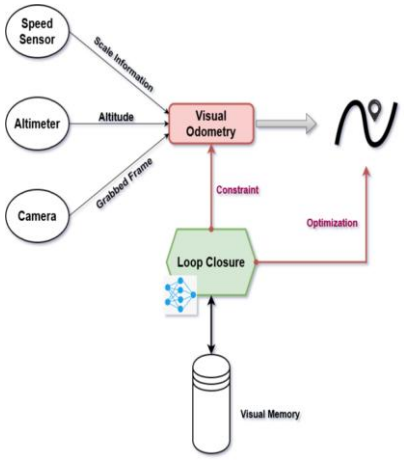


Figure 1 : Localization step architecture

In order to compare these six networks, we trained them using customized dataset made of 9000 image pairs, splitted into positive and negative pairs equally. Each network was trained for 30 epochs using 80% of the total dataset, i.e 7800 pairs.

The validation showed that Siamese ResNet outperforms other implemented CNNs in performing loop-closure detection task, as it is highlighted in Table I.

In order to use the Siamese ResNet in our proposed framework, we have to acquire images. These images constitute the robot's visual memory and serve as reference. That's why a learning step has to be undertaken.

C. Learning Step

First, we make the assumption that the camera frame is rigidly linked to the UAV frame. The camera optical axis has the same direction as the robot longitudinal axis. We suppose that the CNN used here is already trained as presented in [19].

Table I: TESTING RESULTS

Network	Accuracy	TPR	TNR
6-Chan AlexNet	87.7%	63.09%	96.2%
6-Chan GoogLeNet	82.38%	65.86%	96.65%
6-Chan ResNet	93.55%	84.64%	97.65%
Siamese AlexNet	93%	91.49%	94.53%
Siamese GoogLeNet	83.61%	88.83%	78.67%
Siamese ResNet	94.83%	96.42%	92.27%

Let the UAV follows a loop like path. This kind of trajectory includes the properties of both closed and opened loop trajectories.

While [20] and [21] used a matching difference threshold, we build visual memory using a fixed travel distance. In fact, the path is divided into equidistant points. Let $\boldsymbol{a}$ be the spatial distance separating points (Fig.2). Then, $\frac{1}{a}$ is the Learning Spatial Resolution (LSR).

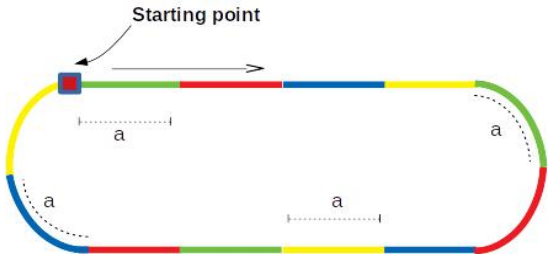


Figure 2 : Learning Spatial Resolution

During learning step we build two data structures to create the visual memory :

• **Referenced Images Dictionary (RID)** : it is dedicated to store images with their ground truth positions. In fact, we use a key-value data structure, where the key is the ground truth position and the value is the saved image.The first frame $\boldsymbol{I_{ref}}$ is grabbed and referenced with its position. It is useful for initialization and alignment. Besides, it is used to automatically disable learning mode if needed. Then, each time the vehicle travels a distance equal to $\boldsymbol{a}$, the grabbed frame is pushed into (RID) with its ground truth position.

• **Ground Truth Position List (GTPL)** : it is dedicated to store ground truth positions only. It is an array-like data structure where we store only the ground truth position for each grabbed frame during learning step.

The algorithm depicted in figure 3 implements the Learning stage. As these functions are not the object of our study, we assume that *getFrame()*, *getPosition()* and *getEndLearningEvent()* are predefined functions. They grab incoming frame, read current ground truth position and return end of learning event respectively. We suppose also that the camera and position sensor are synchronized.

D. Localization Step

The localization step is launched once the learning step is ended. The main goal of this step is to localize the vehicule using its camera stream. It is based on monocular feature-based visual odometry.

1) Monocular feature-based visual odometry

Visual odometry is the core process of our method. It was well explained in [1]. We implement SURF feature detector [18] as feature-based method. In fact, [7] showed that it outperforms SIFT detector [30]in visual odometry case.

In the other hand, we focus here on the scale factor acquisition. We use speed sensor as scale factor provider and we integrate speed information in our process.

We do this byassuming that instants $\boldsymbol{t_i}$ and $\boldsymbol{t_{i+1}}$ are too close. If $\boldsymbol{V_{t_i}}$ is the vehicule linear speed at $\boldsymbol{t_i}$, then the scale factor $\boldsymbol{\rho_{i+1}}$ between $\boldsymbol{t_i}$ and $\boldsymbol{t_{i+1}}$ is given by (1).

$$\rho_{i+1} = (t_{i+1} - t_i)V_{t_i} \quad (1)$$

As the visual odometry is the core process of the localization step, improving visual odometry means improving localization process.

2) Correction

As annouced previously, we propose in this article an original and new localization prcedure. Namely, we implement a two step correction in order to impove the localization performance. The first correction consists in correcting drift, directly into visual odometry process. The second correction step is graph based correction where we use Iterative Closest Point (ICP) process. In fact, ICP allows to minimize the error between estimation graph and ground truth graph. As far as we read, this is the first time that such a localization process is proposed in the litterature.

Algorithm 1: Learning Step Algorithm

Input : a
Output: RID,GTPL
end_learning ← *False* ;
i ← 0 ;
GTPL ← {};
RID ← {:};

while *end_learning* **is** *False* **do**
  current_frame ← *getFrame*();
  current_position ← *getPosition*();
  if *i* == *0* **then**
    Iref ← *current_frame* ;
    last_position ← *current_position* ;
  end
  traveled_distance ← ‖*current_position* − *last_position*‖ ;
  if *traveled_distance* ≥ *a* **then**
    traveled_distance ← 0 ;
    RID[*current_position*] ← *current_frame*;
  end
  GTPL[*i*] ← *current_position*;
  last_position ← *current_position*;
  end_learning ← *getEndLearningEvent*();
  i ← *i* + 1 ;
end

Figure 3 : Learning Step Algorithm

As we did during learning step where we introduced Learning Spatial Resolution (LSR), we define

here the Correction Spatial Resolution (CSR). Let $\frac{1}{b}$ be the CSR. The vehicle trajectory is divided into segments having the same length, which is the CSR as depicted in (Fig. 4). Besides, we assume that $b \geq a$ This condition guarantees the fact that between two correction attempts we have at least one ground truth information.

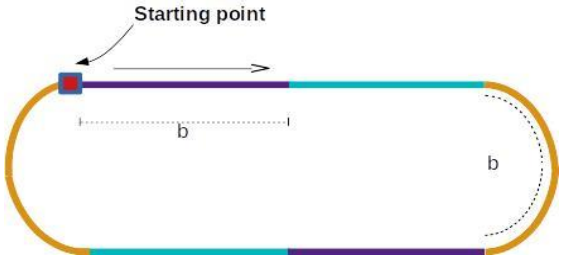


Figure 4 : Correction Spatial Resolution

The estimated positions coming from visual odometry process are stored in an array like data structure, called **Estimated Position List** (EPL).When the robot travels a distance equal to ***b***, the visual odometry starts to drift. So we have to undertake the **first correction stage**, which is directly applied into visual odometry process.

1) Let ***M*** be the EPL length. The last estimated position EP_M is projected on the ground truth path, given by GTPL. It is the closest position saved in GTPL to EPL_M.
Its projection on the ground truth path is given by equation (2) and figure 5.

$$GTPL_{min} = arg \min_{GTPL_i \in GTPL} (\| EPL_M - GTPL_i \|) \quad (2)$$

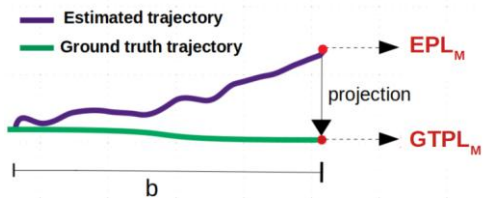


Figure 5 : Projection estimated position on ground truth

2) Then, we look for the closest saved image to the last grabbed frame. In fact, we search for the closest referenced image to $\boldsymbol{GTPL_{min}}$. This can be made using the key-value data structure RID, as the closest image to $\boldsymbol{GTPL_{min}}$ has the closest key.
Let DKL be the RID keys list. The closest key to $\boldsymbol{GTPL_{min}}$ is given by the following equation.

$$DKL_{min} = arg \min_{DKL_i \in DKL} (\| GTPL_{min} - DKL_i \|) \quad (3)$$

Thus, if $\boldsymbol{RI_{min}}$ is the closest reference image to $\boldsymbol{GTPL_{min}}$, then $RI_{min} = RID_{DKL_{min}}$. As $\boldsymbol{RI_{min}}$ is the closest reference image to $\boldsymbol{GTPL_{min}}$, and as $\boldsymbol{GTPL_{min}}$ is the closest ground truth position to $\boldsymbol{EPL_M}$, then $\boldsymbol{RI_{min}}$ is the closest to $\boldsymbol{EPL_M}$.

We note I_M the last grabbed frame, which coordinates are EPL_M.

3) This step can be seen as a local loop closure detection. In fact, the CNN is called to compare I_M with $\boldsymbol{RI_{min}}$. The CNN returns the similarity between $\boldsymbol{RI_{min}}$ and I_M. If the similarity is above a threshold, we can refer to $GTPL_{min}$ as reference location and assume that $GTPL_{min}$ is the correct location. So this allow us to quantify and estimate the drift (translation drift) in EPL_M. If the similarity is under the threshold, then we start a new iteration.
The detection threshold is defined by the user and kept constant during all the process. In our case, we chose the threshold defined during the CNN training.

The drift quantification is inspired by [10]. The error ω is given by the equation (4).

$$\omega = GTPL_{min} - EPL_M \quad (4)$$

4) Once the drift is estimated, it is used to correct visual odometry process. Here we propose to compensate the error directly in the visual odometry.
Visual odometry, at each iteration, concatenates relative translation and rotation to give absolute ones [1]. This can be explained by the following equations.

$$T_i = T_{i-1} + \rho_i \cdot R_{i-1} \cdot T_{(i-1)\to i} \quad (5)$$

$$R_i = R_{(i-1)\to i} \cdot R_{i-1} \quad (6)$$

With T_i is the absolute translation at the i^{th} iteration and $T_{(i-1)\to i}$ the relative estimated translation between frames $(i-1)$ and i.

Similarly, R_i is the absolute rotation after grabbing the frame i and $R_{(i-1)\to i}$ is the estimated rotation between frames $(i-1)$ and i.

Besides, ρ_i is the relative displacement between $(i-1)$ and i, and it is given equation (1).

As it is mentioned in the literature, the accumulated translation T_i contains the current estimated position. In fact, T_i components are equal to EPL_M ones. Using error estimation ω (equation (4)), our goal is to drop the error to 0. We apply this equation to correct visual odometry translation error.

$$T_{i_{corrected}} = T_i + \omega \quad (7)$$

The previously desrcibed procedure allows to locally cancel the error. In fact, theorically, if D is the trajectory length, then we should have $\frac{D}{b}$ points where the error is

close to 0 (Figure 6). Between these points, the odometry estimation error is random and trends to increase.

This shows that transaltion correction is unsufficient. That's why a second step has to be undertaken. The goal of this second step is to cancel the drift between correction points. Our idea is to optimize the segments between two correction points.

The second correction stage is applied on graph, i.e applied on the corrected visual odometry output. We want to locally minimize the deviation between the estimated trajectory and the ground truth points clouds.

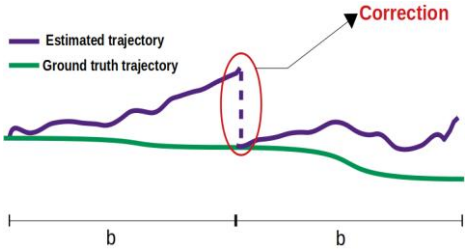


Figure 6 : Correction first step : process oriented correction

In our case, the ICP alogrithm takes a sub-array of GTPL and a sub-array of EPL. Although, the inputs must have the same size. However, GTPL has a constant size, but EPL size is growing every frame. Then, the main challenge in this step is to synchronize sub-arrays sizes. We have to find the starting and ending indexes allowing to extract convenient sub-arrays from EPL and GTPL.

Let lci be the last correction index. So the sub-array extracted from EPL, called $subEPL$ is defined as follows :

$$subEPL = \{EPL_i; lci \leq i < M\} \quad (8)$$

Where $\boldsymbol{M}$ is EPL size.

So $subEPL$ starting index is lci and ending index is $M-1$. Besides, its size is $M - lci$.

Now we have to extract a sub-array from GTPL, let us call it $subGTPL$. However, $subGTPL$ has to verify two constraints :

• The ending index has to not exceed $N-1$, where N is GTPL size.

• Its size has to be equal to $subEPL$ size, i.e $(M - lci)$.

The starting index is the index of the ground truth position that corrected EPL_{lci}. Let lmi be $subGTPL$ starting index. Thus, we have to adopt a circular indexation system. In fact, if the ending index exceeds $N-1$, we have to start again from 0. So, having the starting index of $subGTPL$ which is lmi, theorically the ending index is $lmi + (M - lci)$. But, in order to ensure circle-like indexation, we have two cases.

First case is where $\boldsymbol{lmi + (M - lci) < N}$. Then, the $subGTPL$ is defined as follows

$$subGTPL = \{GTPL_i; lmi \leq i < lmi + (M - lci)\} \quad (9)$$

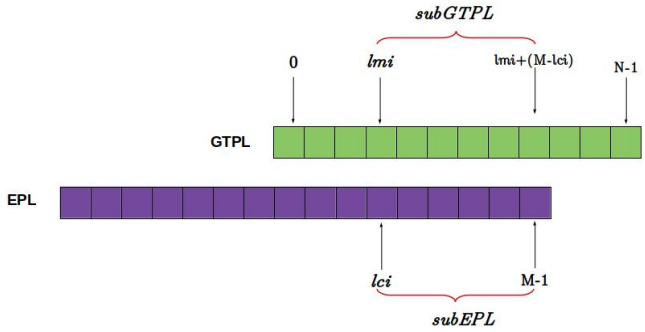


Figure 7 : Extracting subGTPL : first case

Second case is where $\boldsymbol{lmi + (M - lci) \geq N}$. Then, the $subGTPL$ is defined as follows:

$$subGTPL = \{GTPL\_i; lmi \leq i < N\} \cup \{GTPL\_i; 0 \leq i < (M - lci) - (N - lmi)\} \quad (10)$$

We can see from equations (9) and **Error! Reference source not found.** that we verifiy the ending index constraint. It doesn't exceed N-1. Besides, equations (9) and **Error! Reference source not found.** ensure that $subGTPL$ has the same size of $subEPL$, which is $(M - lci)$.

From equation (9) : length is $lmi + (M - lci) - lmi = M - lci$.

From equation **Error! Reference source not found.** : length is $(N - lmi) + (M - lci) - (N - lmi) = M - lci$.

This is the proof that in all cases ICP inputs have the same sizes. By applying ICP on two subsets of GTPL and corrected EPL, we undertook the second correction stage. Thus, the error between two subsequent translation corrections is minimized. Then, thanks to this second correction step, the drift won't increase. Estimated trajectory tries to clamp to the ground truth (Fig. 9).

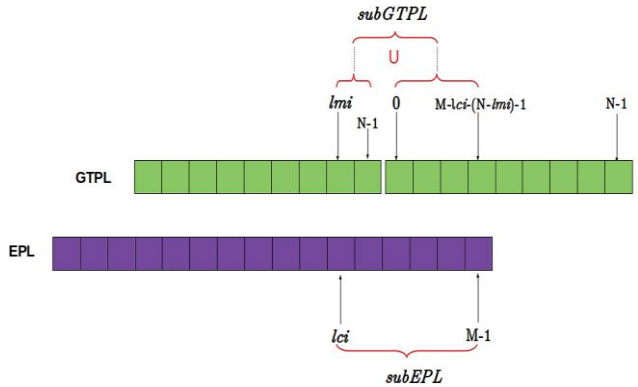


Figure 8 : Extracting sub GTPL : second case

In this section, we detailed our proposed approach. In summary, first, a visual memory is built. It is made of two data structures. One is dedicated to store all acquired ground truth positions and the second stores key frames referenced by their true positions. Then comes the localization/navigation step. It can be seen as a corrected monocular visual odometry process. The correction is called when the trained CNN detects a matching between current grabbed and a stored key frame. Besides, it is made of two stages. The first correction stage is a translation that brings back the error to zero. It is applied on visual odometry translation estimation. The second correction consists in ICP algorithm which applied a posteriori between two successive translations. It allows to minimize the error between the ground truth graph and estimated trajectory graph. Figure 18 shows the whole framework flowchart.

In the next section, we will discuss this approach performances through experiments.

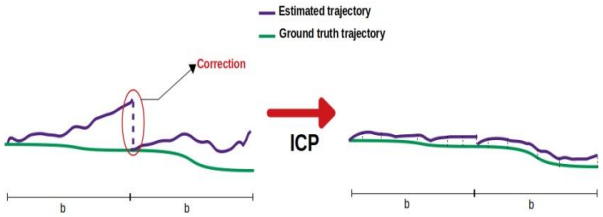


Figure 9 : Correction after applying ICP

IV. EXPERIMENTS

In this section, our framework experimental tests are presented. First, the tests platform will be presented. Then, we are going to highlight some results and test performances.

A. Hardware and software environment configuration

The tests are performed on a motherboard having an intel i3, 2.3GHz quadcore processor and 12 GB RAM.

Besides, the flying vector is simulated using V-REP Simulation Software. It has a distributed architecture based on network client-server concept. This simulator could virtualize and handle many types of robots, sensors and environments.

In our case, we use a quadrotor having a forward looking camera, an altimeter and a speed sensor. The camera used in this simulation has 512x512 resolution and 10fps framerate.

In the other hand, as we mentioned it in [19], our CNN was trained on Google Colaboratory platform, using Intel Xeon 2.20GHz processor, 12 GB RAM and NVidia Tesla K80 GPU. Its training dataset contains images taken from VREP and from Bonn University's Visual Place Recognition dataset[1].

In order to experiment our approach, and ensure its robustness, we generated different scenarios which we are going to present in the next section.

B. Experimentation Scenarios

We tested our method in different conditions. We chose four of them. Two scenarios have a closed loop trajectory. Besides, one of the chosen scenarios is in urban-like environment, and the other ones are in an indoor, office-like environment. Table II details these scenarios specs.

Table II: SCENARIOS PROPERTIES

Scenario	1	2	3	4
Trajectory shape	Loop	8-like	S-like	Straight line
Closed Loop	Yes	Yes	No	No
Length (m)	24	42.5	22	115
Indoor/Outdoor	Indoor	Indoor	Indoor	Outdoor
UAV speed (m/s)	.5	.5	.5	1
$\frac{1}{CSR} = a$(m)	.5	.5	.5	1
$\frac{1}{LSR} = b$(m)	2	2	2	4

During learning step we build navigation memory by creating wo data structures GTPL and RID. One way of evaluating our algorithm performances is to quantify its memory footprint which is presented in the next section.

C. Memory footprints

As detailed in Section 3.3, we append incoming datas to two different data structures. As they are growing incrementally, we have to ensure the growth limit and quantify its the memory.

First, GTPL is an array-like data structure. As a position is an array of three floats (x, y and z), GPTL is an array of arrays of floats. Then, RID is a map made of keys, which are positions (array of floats), and values, which are 512x512 RGB images.

The following table shows the size of GTPL and RID for each scenario after the end of learning loop.

Table III: GTPL AND RID SIZES

[1] http://www.ipb.uni-bonn.de/data/visual-place-recognition-datasets

Scenario	1	2	3	4
GTPL size in KB	5.367	8.813	3.227	2.477
RID size in KB	4.852	5.914	2.906	9.500

All the data amounts presented in table III are under 10KB, which is relatively low. If we consider the average memory footprint per meter, we find that it is around **0.35 KB/m** for scenarios 1, 2 and 3 with 0.5 m/s speed, and **0.105 KB/m** for the scene 4, having 1 m/s speed. Figures 10 to 13 show the memory footprint evolution for each datastructure of each scene. We mention that these evolutions can be approximated by linear functions

This demonstrates that not only our approach consumes low memory but also the memory footprint growth is linear to the path length. This allows us to estimate the global needed memory space.

In the next section, we are going to discuss our framework localization accuracy.

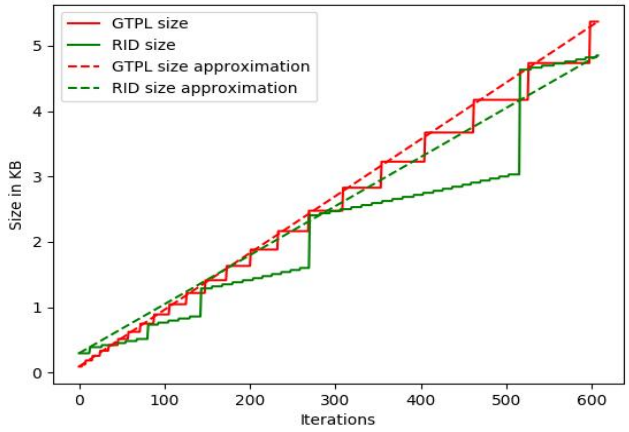


Figure 10 : Scene 1 memory footprint

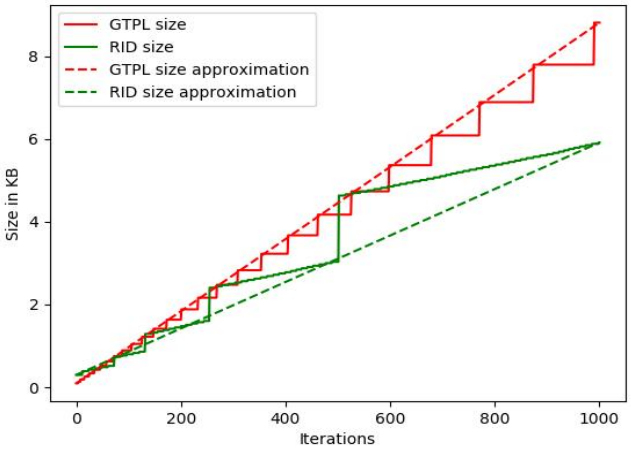


Figure 11: Scene 2 memory footprint

D. Errors Estimation

1) Estimated trajectories

Table IV presents ground-truth, estimated trajectory without ICP and estimated trajectory with ICP. Top view (2D) figures enhance the fact that the one level correction (blue graphs) is based on translation only. In fact, the error is growing and the drift is increasing progressively. Then, the translation corrects the drift and brings the estimated trajectory to the ground truth one. This gives to blue trajectories a discontinous and ticky aspect. However, the two level correction graphs (red graphs) are smoother and clamp to ground truth paths (green graphs). This shows that the ICP improves the estimation. Globally, two levels corrected trajectories fit more to ground truth, are smoother and seem to be more reliable.

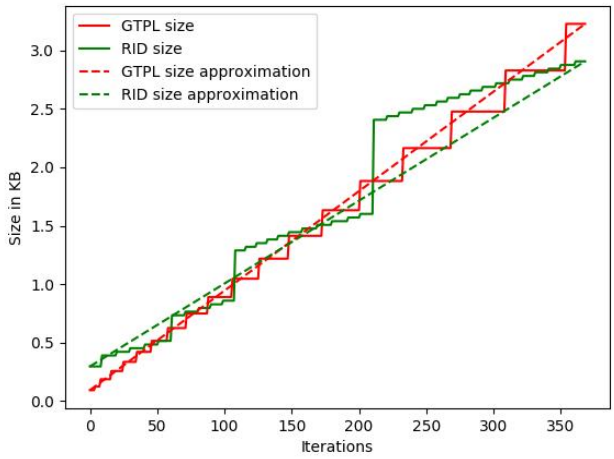


Figure 12 : Scene 3 memory footprint

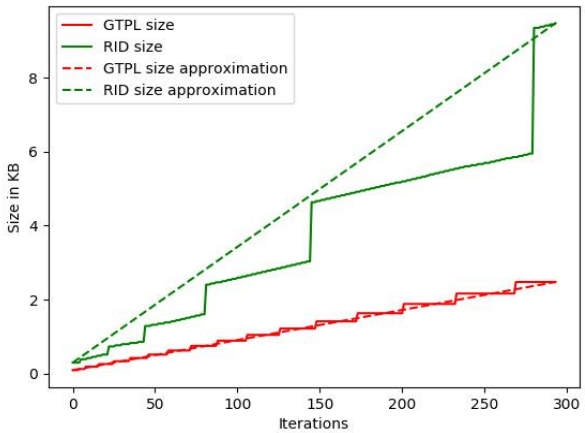


Figure 13 : Scene 4 memory footprint

In the other hand, 3D view images show that there is altitude changes which were purposly introduced. In fact, it serves to evaluate altitude estimation given by the altimeter. It allows also to know if the altitude changes interfere with 2D position estimation. First, in constant altitude sectors, 2D estimation keeps the same behaviour as during altitude changes. This means that altitude variations don't affect vision oriented localization and correction. In the other hand, altitude estimation based on the altimeter is reliable as it is close to the ground truth.

Finally, Scene 4 figures shows noisy estimations. This is due to the noise of visual odometry process. It is coming from a longer course, and the relatively difference in textures.

2) Drifts

In this section, we are going to evaluate drifts and errors. This quantifies how reliable our framework is. Figures 14, 15, 16 and 17 have two continous line graphs each. Red are for two level corrections and blue are for one level correction. Averages are given by dashed lines.

First, for all these figures, ICP corrected estimations have smaller drifts than one level corrected estimations. Then, errors are bounded in figures 14, 15 and 16. Errors never exceeded $\frac{1}{LSR} = b$. Besides, the translation correction is visible when the blue curve (correction without ICP) drops down instantaneously. This happens when distance b is traveled. Then, the CNN is called, the place is recognized, the error is estimated and the translation is applied.

In the other hand, between two successive corrections, ICP corrected trajectories error still approximately constant while one step correction gives highly increasing error. Generally, in our experiments, two level corrected estimations have errors that still below one level corrected ones, except some local points.

Quantitavely, table V gives errors averages for each scene. The lateral average error for the ICP corrected trajectories never exceeds **1.5%** of the total trajectory length. Finally, applying ICP helps to gain some centimeters of accuracy. Sure its a small gain quatitavely but ICP gives smoother trajectory estimation and more similar to ground truth.

Table V : ERRORS AVERAGE

Scenario	1	2	3	4
Error average with ICP (m)	0.118	0.198	0.208	1.63
Error average without ICP (m)	0.220	0.268	0.268	1.67

3) Discussions and Contributions

The experiments showed that the proposed method allows to accurately localize a mobile vector. This task is mainly based on monocular camera. A first step consists in building visual memory. As it is demonstrated by the experiments, it is not memory consuming and can be used in large scale environment. The second step is the localization. It is made of a translation to reduce local error, and ICP to optimize the estimated trajectory. Tests highlighted the fact that the error is low and that ICP allows to improve accuracy

Our work can be seen as a multidisciplinary paper. In fact, it tackles visual odometry correction methods and proposes a novel localization method for UAVs.

This can be seen in :

• Using visual odometry during localization step, unlike [20] and [21]

• Using traveled distance as key frame selection criterion

• Using CNN in loop closure detection and visual odometry correction

• Applying two stages correction, process based correction and graph based correction

• Exploiting speed sensor as scale factor provider

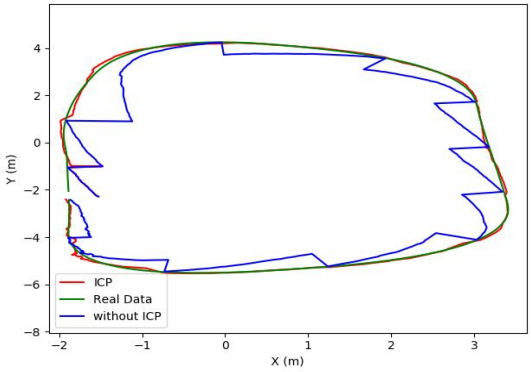
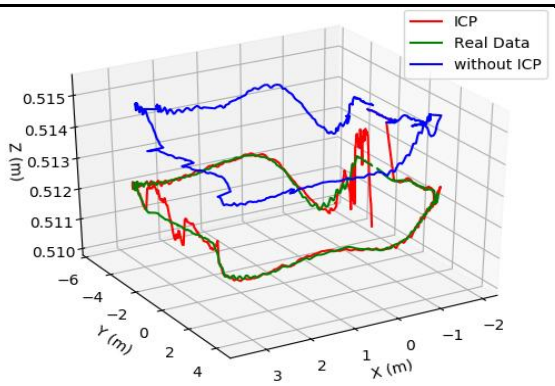
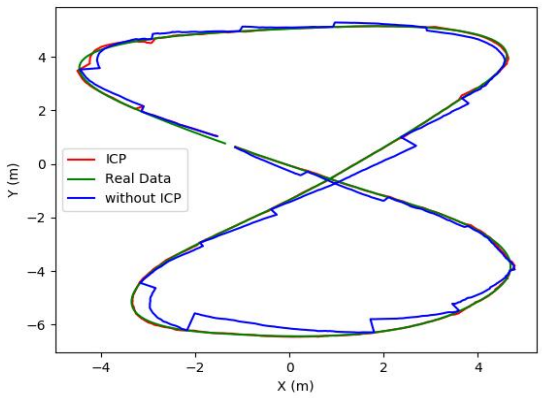
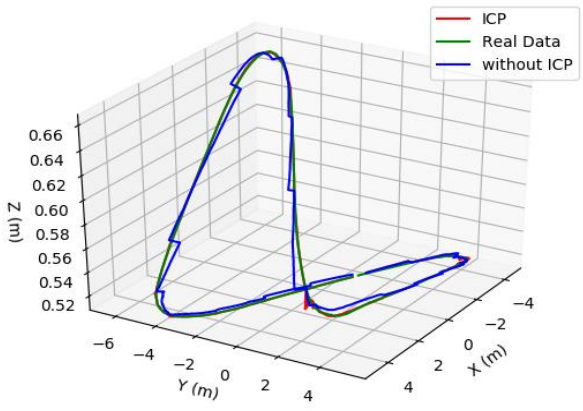
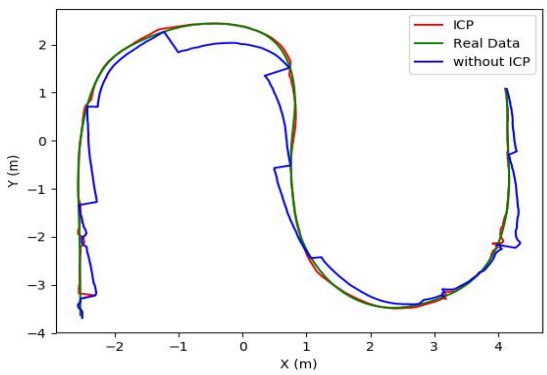
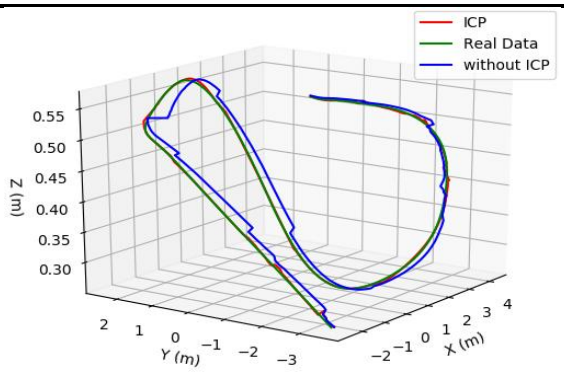
V. CONCLUSION

In this paper we proposed an original vision based method to localize mobile robots. This framework can be summed up in building visual memory which serves as reference during localization process. This one is based on monocular visual odometry. In order to correct visual odometry drift, we call a customized CNN to compare incoming images with the already built visual memory. Then, two level correction is applied. First, the quantified drift is compensated in visual odometry by introducing a translation. Second, the trajectory graph is optimized by using ICP.

Experiments gave satisfying results. Memory requirements are low, two level corrected estimation aspect is similar to ground truth and estimation error is small.

Future works will be devoted first to improve the estimation. Besides, we will tackle navigation and control tasks. In fact, we will focus on trajectory planning and visual control. This can be improved by working on obstacles detection and avoidance.

Table IV : 2D A 3D ESTIMATED AND GROUND-TRUTH TRAAJECTORIES VIEWS

Scene name	Top View (2D)	3D View
Scene 1		
Scene 2		
Scene 3		
		

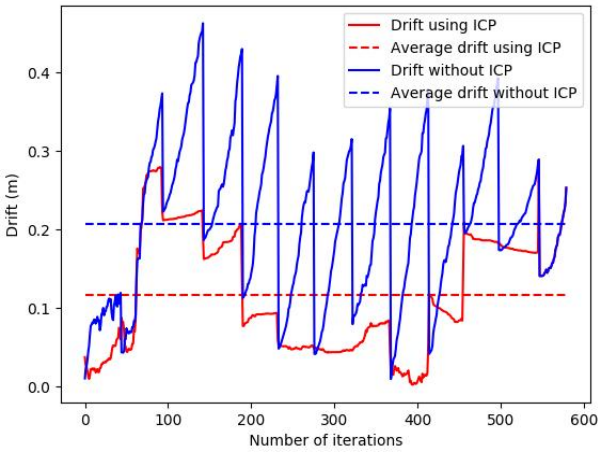


Figure 14 : Scenario 1 drift evaluation

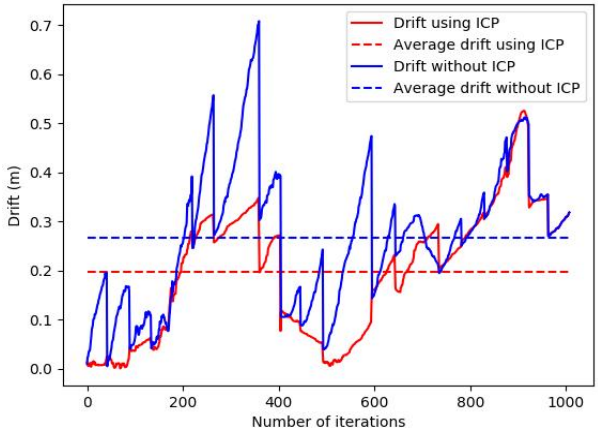


Figure 15 : Scenario 2 drift evaluation

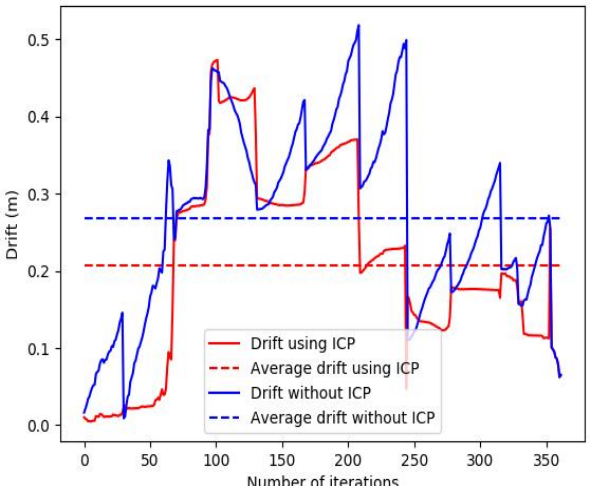


Figure 16 : Scenario 3 drift evaluation

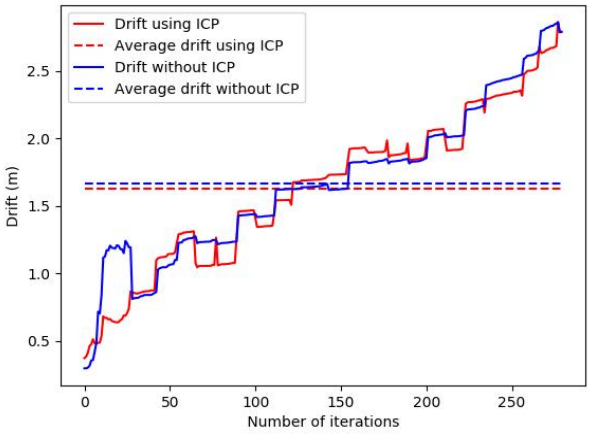


Figure 17 : Scenario 4 drift evaluation

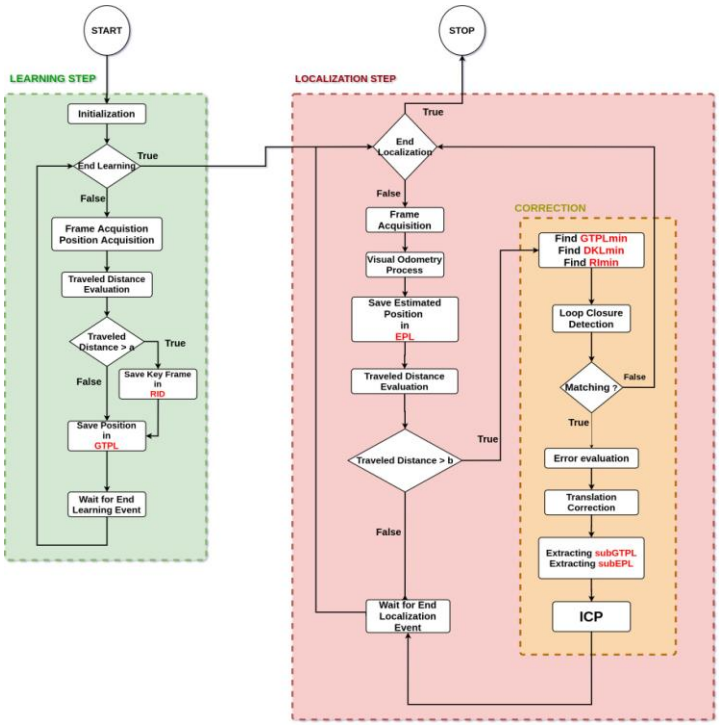


Figure 18 **: Framework Flow Chart**

VI. REFERENCES

[1]Scaramuzza, D. and Fraundorfer, F. 2011. "Visual Odometry [Tutorial]," in *IEEE Robotics & Automation Magazine*, vol. 18, no. 4, pp. 80-92.

[2]Fraundorfer, F. and Scaramuzza, D. 2012."Visual odometry: Part ii - matching, robustness, and applications," in IEEE Robotics & Automation Magazine - *IEEE Robotics & Automation Magazine*, vol. 19, pp. 78–90.

[3]Poddar, Shashi et al. (2018). "Evolution of Visual Odometry Techniques." [Online]. Available: https://arxiv.org/ftp/arxiv/papers/1804/1804.11142.pdf. [Accessed April 2020]

[4]Jiang, Y.; Xiong, G.; Chen, H. and Lee, D.-J.2014."Incorporating a Wheeled Vehicle Model in a New Monocular Visual Odometry Algorithm for Dynamic Outdoor Environments.", in Sensors, 14, pp. 16159-16180.

[5] Siddiqui,J. R. and Khatibi, S. 2014."Robust visual odometry estimation of road vehicle from dominant surfaces for large-scale mapping", in IET Intelligent Transport Systems 9.

[6]Scaramuzza, D. and Siegwart, R.2008."Monocular Omnidirectional Visual Odometry for Outdoor Ground Vehicles.",in: Gasteratos A., Vincze M., Tsotsos J.K. (eds) Computer Vision Systems. ICVS. Lecture Notes in Computer Science, vol 5008. Springer, Berlin, Heidelberg.

[7]Houssem Eddine Benseddik, Oualid Djekoune, and Mahmoud Belhocine. 2014. "SIFT and SURF Performance Evaluation for Mobile Robot-Monocular Visual Odometry," in Journal of Image and Graphics, Vol. 2, No. 1, pp. 70-76.

[8] Yu, Y., Pradalier, C., and Zong, G. 2011. "Appearance-based monocular visual odometry for ground vehicles", in IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM), pp. 862-867.

[9] Zhou, D., Dai, Y., and Li, H. 2016. "Reliable scale estimation and correction for monocular Visual Odometry.", in IEEE Intelligent Vehicles Symposium (IV), pp. 490-495.

[10]Caramazana, L.; Arroyo ,R. and Bergasa, L. M. 2016."Visual odometry correction based on loop closure detection", inOpen Conference on Future Trends in Robotics, pp. 97-104.

[11]Daneshmand, M.; Avots, E. and Anbarjafari, G. 2018."Proportional error back-propagation (peb): Real-time automatic loop closure correction for maintaining global consistency in 3d reconstruction with minimal computational cost,"in Measurement Science Review 18, pp. 86–93.

[12]Sirtkaya, S.; Seymen, B. and Alatan, A. A. 2013."Loosely coupled Kalman filtering for fusion of Visual Odometry and inertial navigation", in Proceedings of the 16th International Conference on Information Fusion, pp. 219-226.

[13]Li, M. and Mourikis, A. I. 2012."Vision-aided inertial navigation for resource-constrained systems", in IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1057-1063.

[14] Filliat, D. 2007."A visual bag of words method for interactive qualitative localization and mapping," in Proceedings ofIEEE International Conference on Robotics and Automation, Roma, pp. 3921-3926.

[15]Garcia-Fidalgo, E. and Ortiz, A. 2018."iBoW-LCD: An Appearance-Based Loop-Closure Detection Approach Using Incremental Bags of Binary Words," in IEEE Robotics and Automation Letters, vol.3, n 4, pp. 3051-3057.

[16] Ma, J.; Qian, K.; Ma, X. and Zhao, W. 2018. "Reliable Loop Closure Detection Using 2-channel Convolutional Neural Networks for Visual SLAM," in*37th Chinese Control Conference (CCC)*, pp. 5347-5352.

[17] An, S.; Che, G.; Zhou, F.; Liu, X.; Ma, X. andChen,Y. 2019. "Fast and incremental loop closure detection using proximity graphs".[Online]. Available: https://arxiv.org/pdf/1911.10752v1.pdf. [Accessed April 2020]

[18]Bay, H.; Ess, A.; Tuytelaars, T. and Van Gool, L. 2008. "Speeded-Up Robust Features (SURF)," inComputer Vision and Image Understing, pp. 346–359.

[19]Sedrine, M. A.; Souidène Mseddi, W.; Abdellatif, T. and Attia, R. 2019. "Loop Closure Detection for Monocular Visual Odometry: Deep-Learning Approaches Comparison." in 15th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS) (2019): 483-490.

[20]Sabatini, R.; Bartel, C.; Kaharkar,A. and Shaid, T. 2012. "Design and integration of vision based sensors for unmanned aerial vehicles navigation and guidance," in Optical Sensing and Detection II

[21] Courbon, J.; Mezouar, Y.; Guenard, N.; and Martinet, P. 2010. "Vision-based navigation of unmanned aerial vehicles," in Control Engineering Practice, pp. 789–799.

[22]Harris, C. and Stephens, M. 1988."A combined corner and edge detector," in Proceedings 4th Alvey Vision Conference, pp. 147–151.

[23]Forster,C.; Pizzoli,M.; and Scaramuzza,D. 2014. "SVO: Fast semi-direct monocular visual odometry," in IEEE International Conference on Robotics and Automation, pp. 15-22.

[24]Klein, G. and Murray, D. 2007."Parallel Tracking and Mapping for Small AR Workspaces," in6th IEEE and ACM International Symposium on Mixed and Augmented Reality, pp. 225-234.

[25]Krizhevsky, A.; Sutskever, I. and Hinton, G. E. 2012. "ImageNet classification with deep convolutional neural networks", in Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1, pp. 1097–1105.

[26]Szegedy, C.; Liu,W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V. and Rabinovich, A. 2015. "Going deeper with convolutions", in IEEE Conference on Computer Vision and Pattern Recognition, pp. 1-9.

[27]Szegedy, C.; Ioffe, S.; Vanhoucke, V. and Alemi, A. A. 2017. "Inception-v4, inception-ResNet and the impact of residual connections on learning.",in Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, pp. 4278–4284.

[28]Bromley, J.; Bentz, J. W.; Bottou, L.; Guyon, I.; LeCun, Y.; Moore,Y. et al. 1993. "Signature verification using a siamese time delay neural network", in International Journal for Pattern Recognition and Artificial Inteligence., vol. 7, no. 4, pp. 669-687.

[29]Zagoruyko, S. and Komodakis, N. 2015."Learning to compare image patches via convolutional neural networks," in IEEE Conference on Computer Vision and Pattern Recognition, pp. 4353-4361.

[30]Lowe, D.2004. "Distinctive image features from scale-invariant keypoints", in International Journal of Computer Vision 60, pp. 91–110.