Universidad Zaragoza

1542

# Trabajo Fin de Máster

## Visual SLAM and Scale Estimation from Omnidirectional Wearable Vision


Autor

Daniel Gutiérrez Gómez

Director

José Jesús Guerrero Campo

Escuela de Ingeniería y Arquitectura


2012

# Visual SLAM and Scale Estimation from Omnidirectional Wearable Vision

## RESUMEN

La resolución del problema de Localización y Mapeado Simultáneos (SLAM) con sistemas de visión permite reconstruir un mapa del entorno a partir de medidas extraídas de imágenes y, al mismo tiempo, estimar la trayectoria u odometría visual de la cámara. En los último años el SLAM visual ha sido uno de los problemas más tratados en el campo de la visión por computador y ha sido abordado tanto con sistemas estéreo como monoculares. Los sistemas estéreo tienen la característica de que conocida la distancia entre las cámaras se pueden triangular los puntos observados y por lo tanto, es posible obtener una estimación tridimensional completa de la posición de los mismos.

Por el contrario, los sistemas monoculares, al no poderse medir la profundidad a partir de una sola imagen, permiten solamente una reconstrucción tridimensional con una ambigüedad en la escala. Además, como es frecuente en la resolución del problema de SLAM, el uso de filtros probabilísticos que procesan las imágenes de forma secuencial, da lugar a otro problema más alla de una ambigüedad de escala. Se trata de la existencia de una deriva en la escala que hace que esta no sea constate durante en toda la reconstrucción, y que da lugar a una deformación gradual en la reconstrucción final a medida que el mapa crece.

Dado el interés en el uso de dichos sensores por su bajo coste, su universalidad y su facilidad de calibración existen varios trabajos que proponen resolver dicho problema; bien utilizando otros sensores de bajo coste como IMUs, [17, 22] o sensores de odometría disponibles en los vehículos con ruedas [5, 7, 26]; bien sin necesidad de sensores adicionales a partir de algún tipo de medida conocida a priori como la distancia de la cámara al suelo [16] o al eje de rotación del vehículo [25].

De entre los trabajos mencionados, la mayoría se centran en cámaras acopladas a vehículos con ruedas. Las técnicas descritas en los mismos son dificilmente aplicables a una cámara llevada por una persona, debido en primer lugar a la imposibilidad de obtener medidas de odometría, y en segundo lugar, por el modelo más complejo de movimiento.

En este TFM se recoge y se amplia el trabajo presentado en el artículo "Full Scaled 3D Visual Odometry From a Single Wearable Omnidirectional Camera" enviado y aceptado para su publicación en el próximo "IEEE International Conference on Intelligent Robots and Sytems (IROS)". En él se presenta un algoritmo para estimar la escala real de la odometría visual de una persona a partir de la estimación SLAM obtenida con una cámara omnidireccional catadióptrica portable y sin necesidad de usar sensores adicionales.

La información a priori para la estimación en la escala viene dada por una ley empírica que relaciona directamente la velocidad al caminar con la frecuencia de paso o, dicho de otra forma equivalente, define la longitud de zancada como una función de la frecuencia de paso [11]. Dicha ley está justificada en una tendencia de la persona a elegir una frecuencia de paso que minimiza el coste metabólico para una velocidad dada [29], [15].

La trayectoria obtenida por SLAM se divide en secciones, calculándose un factor de escala en cada sección. Para estimar dicho factor de escala, en primer lugar se estima la frecuencia de paso mediante análisis espectral de la señal correspondiente a la componente $z$ de los estados de la cámara de la sección actual. En segundo lugar se calcula la velocidad de paso mediante la relación empírica descrita anteriormente. Esta medida de velocidad real, así como el promedio de la velocidad absoluta de los estados contenidos en la sección, se incluyen dentro de un filtro de partículas para el cálculo final del factor de escala. Dicho factor de escala se aplica a la correspondiente sección mediante una fórmula recursiva que asegura la continuidad en posición y velocidad.

Sobre este algoritmo básico se han introducido mejoras para disminuir el retraso entre la actualización de secciones de la trayectoria, así como para ser capaces de descartar medidas erróneas de la frecuencia de paso y detectar zonas o situaciones, como la presencia de escaleras, donde el modelo empírico utilizado para estimar la velocidad de paso no sería aplicable. Además, dado que inicialmente se implementó el algoritmo en MATLAB, aplicándose *offline* a la estimación de trayectoria completa desde la aplicación SLAM, se ha realizado también su implementación en C++ como un módulo dentro de esta aplicación para trabajar en tiempo real conjuntamente con el algoritmo de SLAM principal.

Los experimentos se han llevado a cabo con secuencias tomadas tanto en exteriores como en interiores dentro del Campus Río Ebro de la Universida dde Zaragoza. En ellos se compara la estimación de la trayectoria a escala real obtenida mediante nuestro método con el Ground Truth obtenido de las imágenes por satélite de Google Maps. Los resultados de los experimentos muestran que se llega a alcanzar un error medio de hasta menos de 2 metros a lo largo de recorridos de 232 metros. Además se aprecia como es capaz de corregir una deriva de escala considerable en la estimación inicial de la trayectoria sin escalar.

El trabajo realizado en el presente TFM utiliza el realizado durante mi Proyecto de Fin de Carrera [13]

con una beca de Iniciación a la Investigación del I3A y defendido en septiembre de 2011. En dicho proyecto se adaptó una completa aplicación C++ de SLAM en tiempo real con cámaras convencionales, para ser usada con cámaras omnidireccionales de tipo catadióptrico. Para ello se realizaron modificaciones sobre dos aspectos básicos: el modelo de proyección y las transformaciones aplicadas a los descriptores de los puntos característicos. Fruto de ese trabajo se realizó una publicación [12] en el "$11^{th}$ OMNIVIS" celebrado dentro del ICCV 2011.

# Contents

# Chapter 1

# Introduction

The resolution of the problem of Simultaneous Localisation and Mapping (SLAM) with one camera allows to reconstruct a map of the environment from the measurements taken from the images and, at the same time, estimate the visual odometry of the sensor. This problem can be addressed using either global optimization techniques or probabilistic filters like the extended Kalman filter or the particle fitler. Focusing on visual SLAM, with global optimization, the map and the position of the cameras are estimated by minimizing the reprojection error of the points in the given images. When using probabilistic filters, odometry and map estimations are updated by processing the images sequentially. As a result, probabilistic filters are frequently used in real time SLAM applications where images are processed as they are delivered. However, they have the drawback of a decrease in the accuracy in the long term since the images are forgotten as they are processed and used to update the estimation.

In the last years, visual SLAM has become one of the most trending research fields in computer vision and has been addressed both by using stereo and monocular systems. The main feature of stereo systems is that, knowing the baseline of the cameras, detected landmarks of the scene can be triangulated and the visual odometry and landmark positions can be completely estimated. SLAM approaches using stereo systems have been presented in [19, 21, 23].

On the other hand, due to the impossibility to extract the depth of a landmark just from one single image, monocular systems only allow the camera motion and scene to be estimated up to an unknown scale. With this in mind, stereo systems may seem more appropriate than monocular ones to perform visual SLAM. However the use of single cameras for visual SLAM is still appealing since they are cheaper, more compact and easier to calibrate than stereo systems.

One of the most important and succesful works on monocular SLAM is the one developed by Davison *et al.* [6], which is based on the extended Kalman filter. As landmark depths cannot be estimated only from the first image, this approach uses a pattern of known size to initialise some feature locations allowing the SLAM to start. Thus the scale of the map is fixed by the size of this initial pattern. In a later work by Civera *et al.* [2], the inverse depth parametrization for the map points allowed the SLAM to start automatically without the need of using an initialisation pattern. In this case the scale is arbitrarily fixed by a depth prior of the map landmarks and an acceleration noise setup parameter.

Altough the scale can be initialised by a pattern of known size or some kind of prior, it is likely that scale drift arises between different portions of the scene as the size of the map gets larger. The reason why this drift occurs is the continuous lost and initialization of tracked landmarks, which act as anchor for the scale, due to the sequential processing of the images. This drift acts as a source of incremental error in the SLAM estimation, which leads to a deformation of the final map even after applying conventional loop closing techniques by identifying revisited parts of the map. In [27], Strasdat *et al.* propose a loop closing method which corrects the map deformation due to scale drift.

Visual SLAM using omnidirectional cameras has been proposed in [4, 18, 28]. Due to the $360^{o}$ field of view (FoV) of omnidirectional cameras, features last longer on the image than in the case of conventional cameras, specially during big camera rotations. The increased lifespan of the features on the image translates in a better estimation of the position of the features on the map, a lower need to initialise new features and an increased robustness [24].

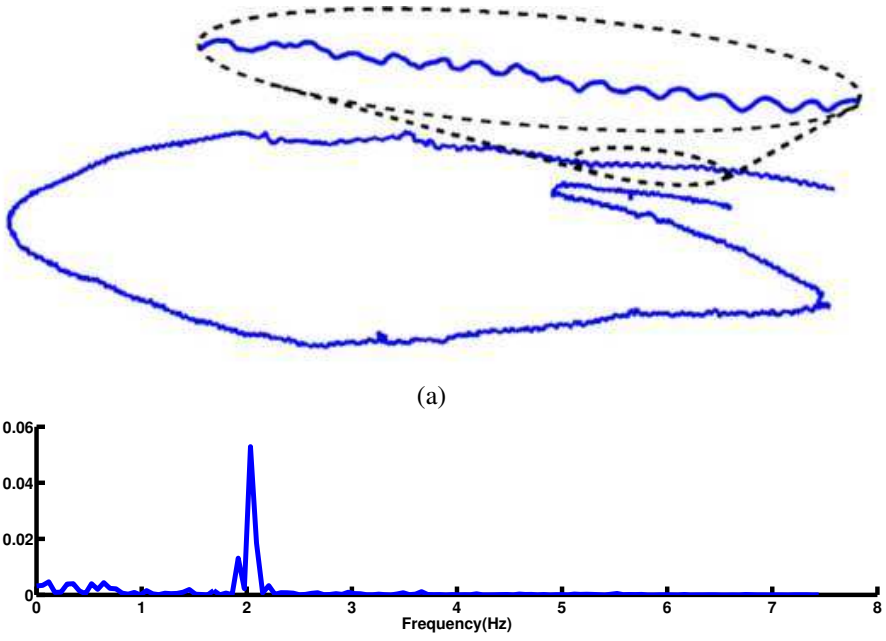
In this work we extend the SLAM approach for catadioptric cameras developed in our previous work

[12] which was presented as final degree project and submitted for the $11th$ OMNIVIS Workshop in 2011. This approach derived from state of the art real time EKF monocular SLAM for conventional cameras [3] and is used in this work to compute the visual SLAM estimation from sequences of images acquired with a catadioptric camera mounted on a helmet which is carried by an operator (Fig. 1.1).

(a) (b)

Figure 1.1: (a) Hemlet-camera device used in our experiments. (b) Omnidirectional image captured with our device.

An induced effect of human walking is a head vertical oscillation whose frequency matches up with the step frequency [14]. Under the premise that the 6 d.o.f. visual SLAM is accurate enough, this vertical oscillatory motion of the head should be visible. Fig. 1.2a depicts an example of this behaviour, where the camera trajectory was obtained by performing a visual SLAM algorithm. Hence the step frequency of the camera carrier can be measured by estimating the power spectra of the vertical component of the camera trajectory (Fig. 1.2b).




(a)

(b)

Figure 1.2: (a) Trajectory estimation of Visual SLAM from a head-mounted catadioptric camera. (b) Power spectra of the vertical component

Walking speed is strictly calculated as the product of step frequency and stride length. However, there exist biomedical studies like the one lead by Grieve [11], which show an empirical relation between step frequency and the walking speed with no dependence on the stride length (or equivalently, a dependence

of the stride length on the step frequency). Further studies explain this relation as the result of a human tendence to choose a step frequency that minimizes metabolic cost of locomotion at a given walking speed [15, 29].

Based on this, we propose an approach to calculate the scale of the visual odometry from a single omnidirectional camera carried on the head of a person. This is done by first performing spectral analysis on short sections of the trajectory to extract the step frequency. Then we compute the estimated walking speed using the relation between step frequency and walking speed, and finally this estimation is integrated into a particle filter which recursively computes the scale factor.

This work gave rise to a research article which has been recently accepted for the International Conference on Intelligent Robots and Systems, to be held between 7-12 October in Vilamoura(Portugal). In addition to the presentation of the work developed in this publication, in this Final Master Project we also improve the initial algorithm. Firstly, since this algorithm has been initially programmed in MATLAB and performed offline on the final SLAM reconstruction, we have implemented it in a module inside the C++ monoSLAM real time application and thus being able to obtain a real time scaled estimation. In the line of real time performance we also introduce changes in the algorithm to reduce the delay in the update of the scaled trajectory. Besides this, we have also included a condition to check the truth of the estimated step frequency and thus, being able to detect situations or zones, like stairs, where the used walking empirical model is not correct.

This memory is structured as follows. In Section 2 we discuss the Related Work on the determination of the scale of the visual odometry with monocular vision. In Section 3 we detail the visual SLAM algorithm for catadioptric cameras. In Section 4 we introduce our scaling algorithm. The experimental evaluation of our algorithm is presented in Section 5. Finally, in Section 6 we extract the conclusions and discuss the future work.

# Chapter 2

# Related Work

The problem of scale estimation in monocular SLAM has been adressed either by using additional sensors which provide any measurement with length dimensions or by taking some prior of an spatial dimension.

Regarding the literature on scale estimation with additional sensors, in [3], Civera *et al.* use GPS information to align and scale the SLAM estimation by a rigid transformation which minimizes the distance between corresponding trajectory points. However this approach was designed for a benchmarking purpose and its practical utility is very limited for two reasons. Firstly, because in outdoor environments GPS itself provides a very precise estimation of the location without need of additional sensors. And secondly because one of the features which makes visual SLAM appealing is the ability to operate in GPS denied environments (indoors).

Lupton *et al.* [17] and Nützi *et al.* [22] propose the use of an IMU to resolve the scale. The former aims to make the true map scale observable by integrating the visual data and the IMU data within an information filter. This allows the computation of the true map and trajectory estimations with no bias due to acceleration noise and feature depth priors. The latter fuses the SLAM estimation and IMU data in an EKF framework to compute the scale factor. Nevertheless both IMU based approaches present the drawback of the need of the numerical integration of the acceleration measurements and thus also making an initial assumption on the velocity.

In the works by Cumani *et al.* [5] and Eudes *et al.* [7] it is suggested the combination of the wheel odometry and the visual information to obtain the scaled map. In [5] odometry is used to provide a prior estimation of the true scaled motion between two consecutive frames which is refined by the update from camera measurements. In [7], the measurement of distance between two camera poses from the odometry is used to compute an scale factor, which is applied to the displacement estimated from the camera measurements. In a similar way Scaramuzza *et al.* [26] use the vehicle speed measurement to compute the distance between the last two frames and recover the 3D structure by triangulation of the common image points.

In other works the scale of the scene is estimated without additional sensors using a prior of any spatial dimension. As mentioned in the Introduction, in the initial work by Davison *et al.* [6] scale was fixed with the size of the pattern used to initialise the landmarks needed to start the SLAM. However it does not avoid scale drift as the map gets larger, since initial landmarks which anchor the true scale are eventually lost sooner or later. Thus, to overcome scale drift the scale factor must be updated periodically. Loethe *et al.* [16] use the prior knowledge of the distance from the camera to the ground plane to compute the scale factor of the scene, which is well suited for camera mounted on vehicles. The main challenge of this approach is to locate and identify the points on the ground, which are needed to compute the ground plane. This is a difficult task due to the flatness of the road and the presence of other dominant planes in the scene. Under the assumption of planar motion, Scaramuzza *et al.* [25] exploit non-holonomic motion constraints of wheeled vehicles to resolve the absolute scale. Taking the offset distance between the camera location and the rear-axis of the vehicle they develop and expression which allows to compute the true distance between two camera poses given that the vehicle is turning.

In this work we present a method to compute the scale in visual SLAM performed with a head-mounted omnidirectional camera and without need of additional sensors. One of the appeals of our method is that, although there exist plenty of methods to estimate the scale in visual SLAM with on-road vehicles, to

the best of our knowledge there do not exist works on scale estimation in monocular SLAM with human mounted cameras. Moreover, since it is not possible to obtain odometry measurements of human motion, we believe that our method may suppose one practical and reliable solution to estimate the scale in such cases. Also, although we evaluate our approach in the specific case of a head-mounted omnidirectional camera, it could be generalized to any case when the walking oscillatory motion can be observed in the visual odometry.

# Chapter 3

# Visual SLAM with catadioptric systems

The V-SLAM approach used in this work is based on the Extended Kalman Filter (EKF) which is divided in two parts. In the first part, *Prediction*, the new state of the system is estimated from the previous time step state through the motion model. The second part of the algorithm, *Update*, uses the measurements of the environment to improve the new state prediction. The full state vector, composed of both the map and last camera location, is modelled as a multidimensional Gaussian distribution coded by its mean vector and covariance matrix. For a detailed explanation of the prediction and update equations of the EKF, refer to Appendix A.

The state of the system is given by the state vector $\mathbf{x}$

$$\mathbf{x} = (\underbrace{\mathbf{r},\ \mathbf{q},\ \mathbf{V},\ \omega}_{\text{Camera state}}, \underbrace{x_i, y_i, z_i, \theta_i, \phi_i, \rho_i, ...}_{\text{3D points (IDP)}}) \tag{3.1}$$

where $\mathbf{r}_{(3\times1)}$ is the camera pose, $\mathbf{q}_{(4\times1)}$ is the quaternion of its orientation and $\mathbf{V}_{(3\times1)}$ and $\omega_{(3\times1)}$ are its linear and angular velocities, respectively.

Landmarks are characterised by a descriptor and their 3D location. The descriptor of the landmark is taken as the image patch around its projection when it is initialised. The 3D locations are parameterised in inverse depth parametrisation (IDP) [2]. As the depth of the landmarks cannot be measured from one single image, landmarks observed by first time are initialised with an arbitrary inverse depth prior $\rho_{0i}$ with large uncertainty. This prior is gradually refined in succesive observations.

The inability to measure the initial depth of the features involves the unobservability of the absolute scale of the scene. Thus, the scale of the SLAM reconstruction is biased due to the difference between the arbitrary depth prior and the true depth of the first measured landmarks. Moreover, scale is liable to drift due to the gradual lost of old landmarks and the initialisation of new ones.

To update the state estimation, the position of the tracked landmarks on the image has to be measured by a matching process. This is done by an active search algorithm. Firstly, an elliptical search region is defined for each visible landmark around its predicted image projection by the projection model. The size of the search region depends both on the motion and the uncertainty of the corresponding landmark 3D position. For each search region, the pixel scoring the highest correlation with the landmark descriptor is selected as a putative match. Secondly, outliers are rejected by checking the scene joint rigidity of every putative matches with a 1-point-RANSAC algorithm [3]. Matches which are compatible with the most voted hypothesis are taken as the measurements to be used in the update step of the EKF.

Due to the distinct characteristics of the catadioptric projection, the visual SLAM approach for conventional cameras must undergo a series of modifications for its use with catadioptric cameras. One of these modifications is related with the projection model encapsulated by the measurement function of the EKF. The conventional pin-hole camera model must be substituted by a more complex projection model which is able to model the projection of a point reflected on a parabolic or an hyperbolic mirror. One of the most used is the Spherical Camera Model proposed by Geyer and Danillidis [9] and extended by Barreto and Araujo [1]. The definition and equations of the Sphere Projection Model as well as its integration in an EKF-SLAM scheme [24] are explained in detail in the Appendix B.

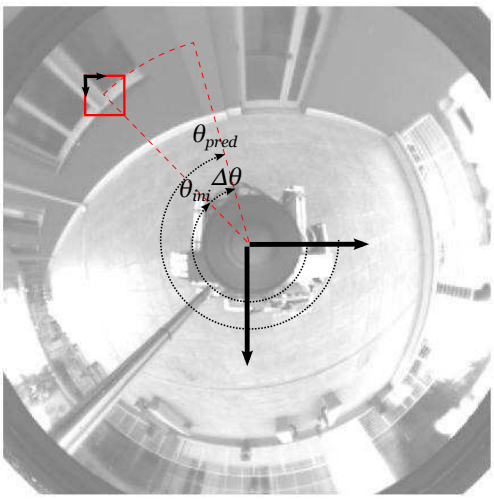



Figure 3.1: A landmark is first detected in a position with polar angle $\theta_{ini}$, initialized and the patch around it saved as its descriptor. Lets suposse that in a future frame, this landmark is predicted to be in the position with polar angle $\theta_{pred}$. Thus, to improve the search in the region around this position, the descriptor must be rotated by the difference of the polar angle $\Delta\theta$.

Omnidirectional images not only involve a more complex projection model, but also an important image deformation, distortion, and variable scale in the image. Generally, during the active search of a landmarks in the image the true matching point must resemble as much as posible to the descriptor patch associated with that landmark. Since the appearance of one landmark in the image changes as the view point varies with the camera movement, it is desirable to predict this change. This is achieved by applying 2D affine transformations to the patches.

The patches to which these transformations are applied are not the own descriptor patches used during the matching process, but bigger patches extracted when the feature is initialised. Prior to the matching process, the big patches are warped by the propper 2D transformation and then, the descriptor patches for correlation are extracted from the center of the warped patches.

In the case of omnidirectional cameras these transformations must encapsulate on the one hand, the deformation due to the rotation of the camera around the vertical axis, and on the other hand the change on the scale of the landmark in the image. This change of scale has two distinct components. The first one, which is common to every vision system, is related with the change of the true depth of the landmark (*i.e.*, objects near to the camera look bigger and objects too far away look smaller). The second component, however, is due to the projective properties of the catadioptric cameras which make the size of a projected object vary with the radial distance of the projection from the principal point of the image.

To prevent the deformation associated to the rotation of the camera, the descriptor patch of the landmark must be rotated by the variation of its polar angle in the image respect to the initialization instant (Fig. 3.1).

For the deformation due to the change in scale we use an expression for the scale factor developed in [12]. This expression is achieved as follows (Fig. 3.2):

1) Let us take an sphere on radius $r$ at a distance $D >> r$ from the camera and parameterise it by the quadric form $\mathsf{Q}$.

$$\mathsf{Q}_{(4\times4)} = \begin{bmatrix} \mathtt{I} & \mathbf{0} \\ \mathbf{0}^\mathsf{T} & -r^2 \end{bmatrix} \tag{3.2}$$

2) Being $\theta$ the elevation angle and taking an azimuth angle of $\phi = 0$ without loss of generality, the 3D position of the center of sphere is given by $\mathbf{X_0} = (D\cos\theta,\ 0,\ D\sin\theta,\ 1)^\mathsf{T}$ which, according to the sphere projection model, is projected in the point $\mathbf{p_0} = (\frac{\gamma\cos\theta}{\xi+sin\theta},\ 0,\ 1)^\mathsf{T}$ in the image plane. The distance from the

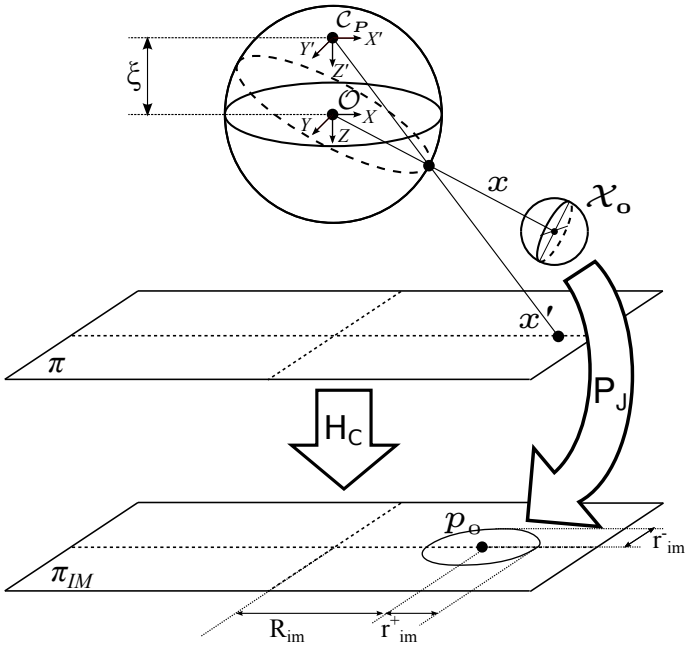



Figure 3.2: Projection of a sphere from the scene to the image plane by the jacobian computed on its centre $\mathbf{X_o}$.

principal point is then:

$$R_{im} = \|\mathbf{p_o}\| = \frac{\gamma \cos\theta}{\xi + sin\theta} \tag{3.3}$$

3) The projection function is linearized by computing its jacobian $\mathsf{J}$ at the center of the sphere and from this Jacobian we build the affine projection matrix $\mathsf{P_J}$.

$$\mathsf{J}_{\mathbf{X}=\mathbf{X_o}} = \frac{\gamma}{D(\xi+S_\theta)^2}\begin{bmatrix} S_\theta(1+\xi S_\theta) & 0 & -C_\theta(1+\xi S_\theta) \\ 0 & \xi+S_\theta & 0 \end{bmatrix} \tag{3.4}$$

$$\mathsf{P}_{\mathsf{J}(3\times4)} = \begin{bmatrix} \mathsf{J}_{\mathbf{X}=\mathbf{X_o}} & \mathbf{0} \\ \mathbf{0}^\mathsf{T} & 1 \end{bmatrix} \tag{3.5}$$

4) By this projection matrix the sphere $\mathsf{Q}$ is projected into a ellipse defined by the conic $\mathsf{C}$:

$$\mathsf{C} = (\mathsf{P_J Q^{-1} P_J^T})^{-1} = \begin{bmatrix} \frac{\gamma^2(1+\xi S_\theta)^2}{D^2(\xi+S_\theta)^4} & 0 & 0 \\ 0 & \frac{\gamma^2}{D^2(\xi+S_\theta)^2} & 0 \\ 0 & 0 & \frac{-1}{r^2} \end{bmatrix} \tag{3.6}$$

5) The resulting ellipse has two semiaxis $r^+_{im}$ (radial direction) and $r^-_{im}$ (polar direction) As we want to apply a uniform scale we choose one of the semiaxis to compute the scale factor. We choose the minor semiaxis, which corresponds to the polar direction and thus it is less afected by the radial distortion induced by the lens, which has not been considered in this derivation:

$$r^-_{im} = \gamma \frac{r}{D}\frac{1}{\xi + \sin\theta} \tag{3.7}$$

where, from (3.3), $\sin\theta$ can be substituted by:

$$S_\theta = f(\xi, \frac{R_{im}}{\gamma}) = \frac{\sqrt{1+(\frac{R_{im}}{\gamma})^2(1-\xi^2)} - \xi(\frac{R_{im}}{\gamma})^2}{1+(\frac{R_{im}}{\gamma})^2} \tag{3.8}$$

Given the resulting expression we can conclude that the size of an object in a catadioptric image depends on the real size of the object $r$, its distance from the camera $D$, the camera-mirror parameters $\xi$ and $\gamma$ and the distance $R_{im}$ of the projection from the principal point. The change in scale of one patch between two frames can be computed as the quotient of its sizes in the two frames:

$$k = \frac{r^-_{im2}}{r^-_{im1}} = \underbrace{\frac{D_1}{D_2}}_{k_1} \underbrace{\frac{\xi + f(\xi, \frac{R_{im1}}{\gamma})}{\xi + f(\xi, \frac{R_{im2}}{\gamma})}}_{k_2} \tag{3.9}$$

where we can note that the dependence on the real size is removed. Also, note that the quotient $k_1$ is the contribution of the change of true depth (which common to every cameras), and the quotient $k_2$ is the contribution of the mirror, which is a particular characteristic of catadioptric cameras.

At this point it has been shown how the change on rotation and scale of the patches are predicted. The computed values are included as the parameters of an affine tranformation $\mathsf{H_S}$ to be applied to the patches prior to the active search:

$$\mathsf{H_S} = \begin{bmatrix} k\cos(\Delta\theta) & -k\sin(\Delta\theta) & 0 \\ k\sin(\Delta\theta) & k\cos(\Delta\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.10}$$

Concerning the implementation details it must be noted that the patch to be warped by the affine transformation has to be bigger than the patch used as descriptor, being the later extracted from the middle of the bigger one. To ensure that the extraction is not done beyond the limits of the warped patch, the scale factor is down limited by the following expression:

$$k_{lim} = \sqrt{2}\frac{h_P}{h_{BP}}\cos(\frac{\pi}{4} - mod(\Delta\theta, \frac{\pi}{2})) \tag{3.11}$$

where $h_{BP}$ is the size of the big patch and $h_P$ the size of the descriptor patch.

In Fig. 3.3 we show an example of the performance of the described SLAM approach for omnidirectional cameras from our previous work in [12]. This trajectory was obtained from a sequence of omnidirectional images along a path of $340$ m taken from the database of the Rawseeds Project [1]. The trajectory was uniformly scaled and compared with the synchronized GPS Ground Truth following the benchmarking method of [3], yielding a mean error of $3.44$ m ($1\%$ mean error over the trajectory).

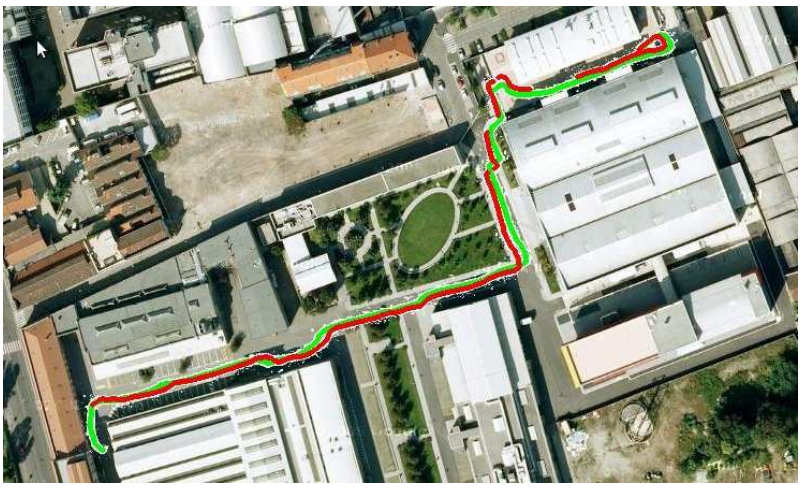


Figure 3.3: GPS trajectory (red) and SLAM trajectory (green) superposed on the satellite image of the Campus of Bovisa (Milan) where the sequences were adquired.

[1] http://www.rawseeds.org/home/

# Chapter 4

# Scaling of the visual odometry

Up to here we have introduced and explained the basic visual SLAM algorithm for omnidirectional cameras. However, one problem of using a monocular system is that it is only able to provide an estimation of the scene reconstruction up to a scale factor. Moreover, this drawback is linked to a more critical one. Since in every Visual SLAM approach all the tracked points are lost sooner or later, the scale of the scene is not anchored and shifts along time as old points are lost and new points are initialised. This phenomena, known as scale drift, makes the scale problem go beyond a simple scale ambiguity which can be solved by applying a uniform scale factor. Indeed, variation of the scale involves a great deformation of the final reconstruction in larger scenes. For this reason it is neccesary to provide a method to compute the scale factor periodically along the motion estimation.

To solve the scale problem, in this work we propose a method which is performed iteratively on sections of the trajectory estimated by the EKF visual SLAM approach. The final output of our method is a full scaled estimation of the visual odometry.

The main assumption and the core of our method is that the SLAM estimation of the visual odometry must register the oscillatory motion of the head during human walking. Thus, although our experiments are focused in SLAM with omnidirectional cameras, its use can be extended to any kind of camera or sensor as long as the unscaled visual odometry estimation registers any oscillatory motion of a part of the human body linked to the step frequency.

Despite this is only applicable on humans, we claim its wide utility, since internal odometry measurements, which are very reliable to provide scale information in SLAM with vehicles, are not available in the case of human walking.

## 4.1 Description of the basic scaling algorithm

The basic algorithm of our method to determine the scale can be divided in four steps, which will be treated in detail in next subsections:

- Spectral analysis on the SLAM visual odometry for the estimation of the step frequency.
- Empirical estimation of the walking speed from the step frequency.
- Integration of the walking speed in a particle filter for a recursive estimation of the scale factor.
- Scaling of the final visual odometry.

### 4.1.1 Spectral analysis on SLAM visual odometry

In the case of our omnidirectonal camera, the camera frame is oriented with its $z$-axis pointing approximately to the direction of the normal to the ground plane, so the head vertical oscillation is given by the $z$-component of the camera position vector. If we split the visual odometry in sections of $N$ camera poses, spectral analysis is carried on the data sequence $(z_{k,1}, z_{k,2}, ..., z_{k,N})$, where $z_{k,n}$ is the $z$-component of the $n$-th camera pose in the section $k$.

The power spectral density $\Gamma_d$ is calculated by applying the Discrete Fourier Transform (DFT) to the data sequence as follows:

$$\Gamma_{d,k}(f_m) = \frac{1}{F_s N} \| \sum_{n=1}^{N} z_{k,n} \exp\left( -j \frac{2\pi f_m (n-1)}{F_s} \right) \|^2 \tag{4.1}$$

$$f_m = \frac{m F_s}{N} \qquad m = -\frac{N}{2}, ..., -1, 0, 1, ..., \frac{N}{2} \tag{4.2}$$

where $F_s$ is the sampling frequency, which in our case is the number of frames per second (fps) of the camera, and $f_m$ are the frequencies for which the spectrogram is sampled.

The computation of the DFT of discrete signals involves a series of issues which have to be adressed. The most inmediate one is related to the sampling frequency $F_s$ of the camera. As we are interested in extracting an step frequency, $F_s$ has to be large enough to avoid aliasing in the case when the highest admisible step frequency occurs. In the acquired sequences, the sampling frequency of the camera was set to 15 frames per second (*i.e.*, $F_s = 15$ Hz), which is greater enough than the $f_{st}^+ = 3$ Hz taken as the upper limit for a feasible human step frequency.

The choice of the number of samples $N$ is also important for the computation of the spectrogram. From de definition of the DFT in 4.1 and 4.2, one can see that the spectrogram is discretized in $N$ frequency bins ranging from $-\frac{F_s}{2}$ to $\frac{F_s}{2}$, so a higher $N$ involves an increased resolution. Also, in any case, $N$ has a lower limit given by the minimum number of samples needed to observe at least one oscillation in the less favourable case of the minimum admissible step frequency (taken as $f_{st}^- = 1$ Hz):

$$N^{min} = \frac{F_s}{f_{st}^-} = 15 \tag{4.3}$$

Note that although for DFT computation purposes $N$ has to be as highest as possible, from the global point of view of trajectory scaling, a high $N$ involve a less frequent update of the scale factor and a reduced ability to detect changes in the step frequency. This can result in a decreasing accuracy in the computation of the scale factor. Moreover, if interested in real time operation, the time delay to update the scaled trajectory grows linearly with $N$, since before scaling one section we need to get the new $N$ unscaled camera poses from the SLAM algorithm. Thus, in summary, for the choice of $N$ we must reach a compromise between the resolution of the DFT and the frequency with which the scale factor is updated.

Another problem of computing the DFT which may not seem very evident is the creation of new low frequency components which did not exist in the original signal. This phenomena is known as spectral leakage and arises when we work with finite signals. When applying the DFT, the input signal is considered to be one period of an infinite signal. Thus, discontinuties are likely to occur and these discontinuities end up by producing non-sinusoidal components with a low frequency and a high amplitude (Fig. 4.1a). The harmonics in which these components are decomposed are spreaded along all the spectrogram and might end up by masking the searched step frequency (Fig. 4.1c). To solve this problem we preproccess the data sequence $(z_{k,1}, z_{k,2}, ..., z_{k,N})$ by substracting the first element $z_{k,1}$ from all the elements and filtering with a second order digital filter with a cutoff frequency of $f_c = 0.3$ Hz (Fig. 4.1b). This way, the power peak at the real step frequency becomes clearly visible in the spectrogram (Fig. 4.1d).

Once the spectrogram of the signal is computed, we extract the maximum peak in the interval of feasible human step frequencies, which are assumed to fall in the range between $f_{st}^- = 1$ Hz and $f_{st}^+ = 3$ Hz. Given the spectrogram $\Gamma_{d,k}(f_m)$, the estimated step frequency $f_{st,k}$ is computed as:

$$f_{st,k} = \underset{f_m \in [f_{st}^-, f_{st}^+]}{\arg\max} \; \Gamma_{d,k}(f_m) \tag{4.4}$$

### 4.1.2 Walking speed estimation

To estimate the walking speed, we consider the biomedical work by Grieve *et al.* [11] where a relation between the step frequency ($f_{st,k}$) and the walking speed ($V_{walk,k}$ ) normalized with height ($H$) is presented:

$$V_{walk,k} = \alpha f_{st,k}^{\beta} H \tag{4.5}$$

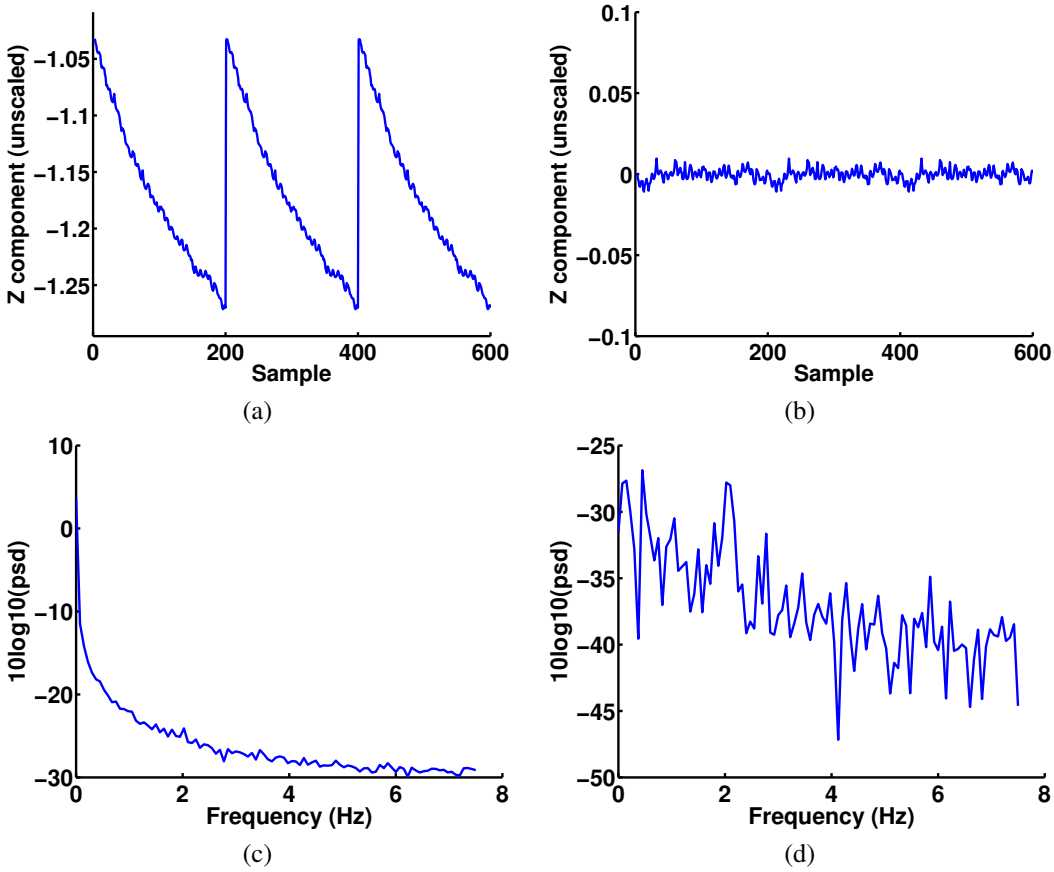



Figure 4.1: Z-component signal segment (top) and corresponding power spectra in logarithmic scale (bottom) of two instances from the same visual odometry section: (a,c) without preprocessing the input signal and (b,d) with offset elimination and filtering of the input signal. Note how in (b) the power peak at the step frequency (2 Hz) is observable and the highest in the interval of feasible step frequencies. Signal segments have been copied three times to make visible the difference in the discontinuty between the two instances.

where $V_{walk,k}$ is in m/s, $f_{st,k}$ in Hz, $H$ in m, and $\alpha$ and $\beta$ are characteristic parameters which differ from one individual to another.

Further studies have proved that this direct relationship between walking speed and step frequency responds to a tendence to minimize the metabolic cost of walking [15, 29].

In the work by Grieve *et al.*, values of $\alpha$ and $\beta$ parameters are presented as dependent on the characteristics of each individual and they provide a group equation with the means of the values obtained for the subjects participating in their experiments. For higher accuracy, in this work we have computed our own $\alpha$ and $\beta$ parameters for the camera operator. We measured the time $t_i$ it took the operator to walk a distance $s = 100$ m at the times per step $\Delta T_i$ given by a metronome ranging from $0.45$ to $0.80$ seconds in intervals of $0.05$ seconds (see Table 4.1). The height of the operator is $H = 1.88$ m.

Normalized walking speeds $V_i'$ and step frequencies $f_i$ were computed from the raw experimental data. Then a power fitting was applied to obtain the values of $\alpha = 0.329$ and $\beta = 1.534$ (Fig. 4.2).

### 4.1.3 Particle Filter for scale factor tracking

Having the walking speed estimate, the scale factor for section $k$ could be straighforwardly computed by $d_k = \frac{V_{walk,k}}{\mu_{V,k}}$, where $\mu_{V,k}$ is the average adimensional speed of the camera poses in section $k$.

However, given the empirical method for the walking speed estimation and the possible high variability of the SLAM velocity along $N$ frames, we decide to use a probabilistic filter for the computation of the scale factor. This allows us to introduce an uncertainty to the scale factor and at the same time decrease the

Table 4.1: Experimental data used to compute the empirical *Step frequency-Walking speed* relationship for the camera operator.

| $\Delta T_i\,[s]$ | $t_i\,[s]$ | $f_i = \frac{1}{T_i}\,[Hz]$ | $V_i{}' = \frac{s}{t_i H}\,\left[\frac{1}{s}\right]$ |
|---|---|---|---|
| 0.45 | 48.18 | 2.22 | 2.08 |
| 0.50 | 55.60 | 2 | 1.80 |
| 0.55 | 61.63 | 1.82 | 1.62 |
| 0.60 | 74.54 | 1.67 | 1.34 |
| 0.65 | 84.42 | 1.54 | 1.19 |
| 0.70 | 94.63 | 1.43 | 1.06 |
| 0.75 | 104.42 | 1.33 | 0.96 |
| 0.80 | 116.06 | 1.25 | 0.86 |

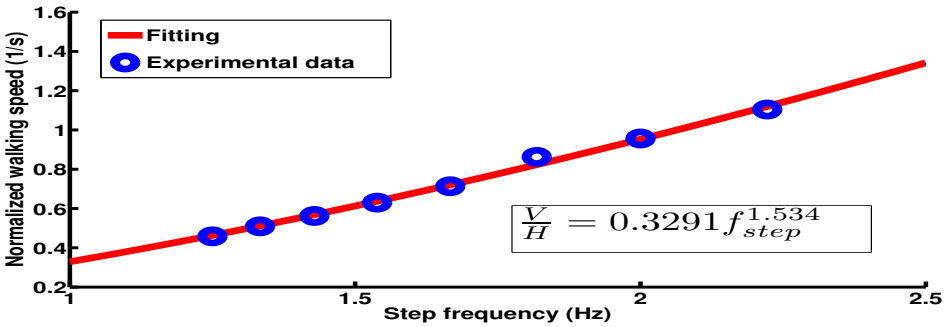



Figure 4.2: Power fitting of the experimental data to compute the relation between walking speed and step frequency ($\mu_{err} = 0.018$, $max_{err} = 0.04$).

effect of spurious estimations of the walking speed in the computation of the scale factor.

For the design of the probabilistic filter we consider a dynamic system whose state $\mathbf{x}_k$ is composed by the magnitude of the SLAM velocity $V_{SLAM,k}$ and the decimal logarithm of the scale factor $\lambda_k = \log_{10}(d_k)$.

$$\mathbf{x}_k^{(L)} = \begin{bmatrix} V_{SLAM,k}^{(L)} \\ \lambda_k^{(L)} \end{bmatrix} \tag{4.6}$$

As it will be detailed in further reasoning in this section, there exist a variety of good reasons to take the logarithm instead of the scale factor directly:

- It allows to restrict the scale factor to positive values by using simply a gaussian distribution to model the uncertainty.
- Uncertainty is encoded in orders of magnitude, which is more realistic than taking an interval in $d$ with the same upper and lower limits. For example, with no prior knowledge of the scale factor, the chances of it falling between $0.1$ and $1$ should be equal to the chances of falling between $1$ and $10$.
- Every uncertainties in the model can be modeled by additive gaussian noise.
- All the non-linearities of the model are encapsulated in the measurement function.

To track the scale factor, a particle filter with Sampling Importance Resampling is designed [10]. We use a particle filter rather than an extended Kalman filter (EKF) so that it can deal with high uncertainty priors of the scale factor which would involve a large linearization error in an EKF approach.

Hence the state of the system in each section $k$ is approximated by a set of particles:

$$S_k = \left\{ (\mathbf{x}_k^{(L)}, w_k^{(L)}) \mid L = 1, 2, ..., P \right\} \tag{4.7}$$

where $P$ is the number of particles and $\mathbf{x}_k^{(L)}$ and $w_k^{(L)}$ are respectively the state vector and the resampling weight of particle $L$.

The particles are initialised such that the initial values of $\lambda_0^{(L)}$ are drawn from a Gaussian distribution $\lambda_0 \sim \mathcal{N}(0, \sigma_0)$, where $\sigma_0$ is a parameter related to the orders of magnitude being scoped out.

In the first step of the particle filter, particles are sampled down by a proposal distribution $p(\mathbf{x}_k|\mathbf{x}_{k-1})$:

$$\mathbf{x}_k^{(L)} \sim p(\mathbf{x}_k|\mathbf{x}_{k-1}^{(L)}) \tag{4.8}$$

In our system the sampling of the proposal distribution includes both the update of the SLAM velocity, which is taken as a control input coming from the visual odometry, and the possible drift in the scale. This is encoded in the following equations:

$$V_{SLAM,k}^{(L)} = \mu_{V,k} + \nu^{(L)} \tag{4.9}$$

$$\lambda_k^{(L)} = \lambda_{k-1}^{(L)} + \alpha^{(L)} \tag{4.10}$$

with $\nu^{(L)} \sim \mathcal{N}(0, \sigma_{V,k})$ and $\alpha^{(L)} \sim \mathcal{N}(0, \sigma_{drift})$, and where $\mu_{V,k}$ and $\sigma_{V,k}$ are the averaged speed and the corresponding standard deviation of the last set of $N$ SLAM camera poses used for spectral analysis, and $\sigma_{drift}$ is the standard deviation prior of the scale drift between two consecutive sections, which is modelled as Gaussian noise.

This initial sampling by the proposal distribution responds to an initial prediction of the state in the step $k$. After this prediction, the uncertainty of the estimation is reduced by integrating the measurement of the real walking speed $V_{walk,k}$ from the spectral analysis routine. To do this, firstly the particles are weighted as follows:

$$w_k^{(L)} = p(V_{walk,k}|\mathbf{x}_k^{(L)}) \tag{4.11}$$

where $p(V_{walk,k}|\mathbf{x}_k^{(L)})$ is the probability density function defined by the measurement model $h(\mathbf{x}_k)$ and the statistics of the sensor noise. Intuitivelly, this expression means that particles for which the measurement function yields a walking speed consistent with the walking speed measurement will get higher weights. Assuming that the speed estimation is affected by Gaussian noise of zero mean and standard deviation $\sigma_{Vwalk}$ to be set up empirically, weights are computed as:

$$\omega_k^{(L)} = p(V_{walk,k}|\ \mathbf{x}_k^{(L)}) = \phi\left(\frac{V_{walk,k} - h(\mathbf{x}_k^{(L)})}{\sigma_{Vwalk}}\right) \tag{4.12}$$

where $\phi(z)$ is the probability density function of the standard normal distribution and the measurement function $h(\mathbf{x}_k^{(L)})$ is given by:

$$h(\mathbf{x}_k^{(L)}) = V_{SLAM,k}^{(L)} 10^{\lambda_k^{(L)}} \tag{4.13}$$

Then weights have to be normalized as follows:

$$\hat{\omega}_k^{(L)} = \frac{\omega_k^{(L)}}{\sum_{M=1}^{P} \omega_k^{(M)}} \tag{4.14}$$

Finally, the set of particles $S_k$ is resampled by drawing $P$ particles from a multinomial distribution $Mult(P, \hat{\omega}^{(1)}, ..., \hat{\omega}^{(P)})$ where the probability of drawing a particle $(L)$ is given by its corresponding weight $\hat{\omega}^{(L)}$.

### 4.1.4 Scaling of the trajectory

The scale factor to be applied to the camera poses of each section $k$ is obtained by averaging the logarithmic scale values of the particle set $S_k$ and undoing the logarithmic change as follows:

$$\bar{\lambda}_k = \frac{\sum_{i=1}^{P} \lambda_k^{(i)}}{P} \tag{4.15}$$

$$d_k = 10^{\bar{\lambda}_k} \tag{4.16}$$

This scale factor must be applied to the position and velocity of the $N$ camera states of section $k$. To simplify the notation we define a vector $\mathbf{C}_k(n)$ which encapsulates all the variables to be scaled:

$$\mathbf{C}_k(n) = (r_x^n,\ r_y^n,\ r_z^n,\ v_x^n,\ v_y^n,\ v_z^n) \quad n = 1, 2, ..., N \tag{4.17}$$

To ensure the continuity in position and velocity, the offset in the initial unscaled pose of the section is eliminated by substracting the last unscaled pose of the previous section from each vector $\mathbf{C}_k(n)$. Then the scale factor is applied and the offset is recovered by adding the last scaled point of the previous section. This is encoded by the following recursive equation:

$$\hat{\mathbf{C}}_k(n) = \hat{\mathbf{C}}_{k-1}(N) + d_k[\mathbf{C}_k(n) - \mathbf{C}_{k-1}(N)] \quad k = 2, 3, ... \tag{4.18}$$

$$\hat{\mathbf{C}}_1(n) = d_1\mathbf{C}_1(n) \tag{4.19}$$

where $\hat{\mathbf{C}}_k(n)$ is the vector which includes the scaled position and velocity of the camera poses contained in section $k$.

## 4.2 Implementation within a real time monoSLAM framework

The original state of the art monoSLAM C++ application used in this work uses two threads. The main thread executes the monoSLAM algorithm iteratively from the incoming frames. The second thread is devoted to update the two graphical outputs: the *real* display, where the original image with the tracked landmarks is shown, and the *virtual* display, where the estimated map and the camera trajectory are displayed. Drawing functions executed by the second thread are triggered from the main monoSLAM thread. To avoid simultaneous use of shared variables (SLAM state variables are needed also by the drawer to update the displays) a mutual exclusion variable is used.

The implementation of our scaling algorithm has been done in a new thread. This way monoSLAM can go on working while the last section of camera poses is being scaled. After each iteration, the main thread stores the last state variables of the camera in a shared buffer. When this buffer is filled (*i.e.*, it contains the states corresponding to the $N$ camera poses needed for the espectral analysis), the main thread sends a signal which triggers the scaling thread, which loads the buffer into a variable exclusive for this thread. After executing the scaling algorithm described in the previous sections of this chapter, the scaled trajectory is updated by adding the recently scaled camera poses. To avoid conflicts between the new thread and the original ones, two new mutual exclusion variables have been added: one for the buffer containing the camera state variables, shared by the monoSLAM and the scaling threads, and another one for the scaled trajectory which is shared by the scaling and the drawing threads.

As it was breafly introuced in Sec. 4.1.1, one drawback of the described approach is that, although it is able to operate in real time, there will always exist a delay in the update of the scaled estimation. This delay is linked to the time it takes to fill the buffer with the $N$ states needed to perform the DFT. For example, given the camera frame rate of 15 fps and assuming $N = 200$ as the minimum number of camera poses needed to get an accurate estimation of the step frequency, the minimum delay in the update of the scaled visual odometry would be:

$$t_{delay}^{min} = \frac{N}{F_s} = \frac{100\ f}{15\ f/s} = 13.33s \tag{4.20}$$

We propose to decrease this delay by updating only one fraction of the buffer instead of renewing it completely for each iteration of the scaling algorithm. Thus, the number of poses of each scaled section (except for the first section which has to be $N$ compulsorily) will be:

$$N_f = ceil\,(\alpha N) \tag{4.21}$$

with $0 < \alpha \leq 1$. The only restriction in the choice of $\alpha$ is the time to scale the $N_f$ camera states to be lower than the time taken to acquire $N_f$ frames.

This way the number of camera poses used for the spectral analysis will remain $N$ (by reusing poses from previous sections), while the amount of scaled camera states per section is reduced to $N_f$.

## 4.3 Check of the spectral power consistency

One issue of the estimation of the step frequency by spectral analysis is the possibility of getting false estimations due to the presence of other dominant frequencies in the spectrogram.

To try to reject these false estimations we propose improving the basic algorithm by checking that the spectral power of the frequency taken as step frequency is consistent with the typical range of amplitudes of the head oscillation during walking. To do this, first we develop the following general formulation:

Let us take a continuous sinusoidal signal:

$$z\left(t\right) = A_z \sin\left(2\pi f_p t\right) \tag{4.22}$$

The power of this signal is computed as:

$$\bar{P} = \frac{1}{T_p}\int_0^{T_p} z\left(t\right)^2 dt = \frac{1}{T_p}\int_0^{T_p} A_z^2 \sin^2\left(\frac{2\pi t}{T_p}\right) dt = \frac{1}{2\pi}\int_0^{2\pi} A_z^2 \sin^2 x \; dx = \frac{A_z^2}{2} \tag{4.23}$$

Now let us sample the continuos signal $z(t)$ into a finite time series $z_n = z\left(\frac{n}{F_s}\right)$ with $1 \leq n < N = \frac{F_s}{f_p}$. Then, by applying 4.1 and 4.2, we obtain the power spectra $\Gamma\left(f_m\right)$ of the signal, which will be zero for every $f_m$ except for $f_m = \pm f_p$.

The power spectra is related to the power of the original signal through the Parseval's theorem, which states that the energy of a signal is preserved in the frequency domain:

**Theorem (Parseval)**: Let $\Gamma\left(f\right)$ be the power spectral density function of one signal $z\left(t\right)$. Then we have:

$$\int_{-\infty}^{\infty} \Gamma\left(f\right) df = \frac{1}{T}\int_0^T z\left(t\right)^2 dt \tag{4.24}$$

For discrete time signals the Parseval's theorem becomes:

$$\frac{F_s}{N}\sum_{m=-\frac{N}{2}}^{\frac{N}{2}} \Gamma_d\left(f_m\right) = \frac{1}{N}\sum_{n=1}^{N} z_n^2 \tag{4.25}$$

where $F_s$ is the sampling frequency and $N$, the number of samples. Applying this theorem to our signal and substituting the right term by the result of 4.23 we obtain:

$$2\frac{F_s}{N}\Gamma_d\left(f_p\right) = \bar{P} = \frac{A_z^2}{2} \tag{4.26}$$

which encodes a relation between the power of one sinusoidal signal and its spectral power density.

Now lets move back to our real problem of the estimation of the step frequency. After estimating the step frequency $f_{st,k}$ with 4.4, the power of the component associated with the head oscillation can be approximated as:

$$\bar{P}\left(f_{st,k}\right) = 2\frac{F_s}{N}\Gamma_d\left(f_{st,k}\right) \tag{4.27}$$

However, since the signal is not perfect and due to discretization error the power of the head oscillation may be spreaded along the near frequencies. Thus we propose to reformulate the previous equation as:

$$\bar{P}\left(f_{st,k}\right) = 2\int_{f_{st}-\Delta f}^{f_{st}+\Delta f} \Gamma\left(f\right) df = 2\frac{F_s}{N}\sum_{m=m^-}^{m^+} \Gamma_d\left(f_{m,k}\right) \tag{4.28}$$

with $m^- = \text{round}\left(N\frac{f_{st}-\Delta f}{F_s}\right)$ and $m^+ = \text{round}\left(N\frac{f_{st}+\Delta f}{F_s}\right)$

To validate $f_{st,k}$ as a feasible step frequency we have to check that $\bar{P}\left(f_{st,k}\right)$ is consistent with the typical range of amplitudes of the head oscillation movement during walking. Thus, first we need some knowledge about the maximum $A_z^+$ and a minimum $A_z^-$ reachable values for $A_z$. Basing on [14], one conservative estimation of such values would be $A_z^+ = 40$ mm and $A_z^- = 7.5$ mm.

Also note that, since the power spectral density is computed for the unscaled $z$-component of the visual odometry, the computed power must be scaled by multiplying it by the square of the current scale factor $d_k$. Thus the condition for the spectral power consistency of the step frequency remains:

$$\frac{1}{2}A_z^{-2} \leq d_k^2 \bar{P}\left(f_{st,k}\right) \leq \frac{1}{2}A_z^{+2} \tag{4.29}$$

If this condition is not filled the strategy would be to keep the current scale factor $d_k$ and skip the weighting and resampling steps.

Finally, the complete scaling algorithm presented in this section is sumarized in Algorithm 1 at the end of the chapter.

**Algorithm 1** Complete Visual Odometry Scaling algorithm

**Require:** $C_{k,1..N}$, $S_{k-1}$
**Ensure:** $\hat{C}_{k,1..N_f}$, $S_k$
  //Notation
  $C_{k,n} =$ $n^{th}$ unscaled camera state
  $\hat{C}_{k,n} =$ $n^{th}$ scaled camera state
  $N =$ # input camera states
  $N_f =$ # output/new camera states
  $S_k =$ Set of particles for the particle filter
  ////////////////////////////////////////////////////
  //Algorithm
  $k = 0$
  $[S_0]$ = Initialize particles ()
  **while** Not end of sequence **do**
    $k = k + 1$
    Wait for new $C_{k,1..N}$ from monoSLAM
    $[z_{k,1..N},\ \mu_{V,k},\ \sigma_{V,k}]$ = Extract $z$-component and mean speed $(C_{k,1..N})$
    $[z_{k,1..N}]$ = High Pass Filter $(z_{k,1..N})$
    $[f_m,\ \Gamma_{d,k}]$ = Spectrogram $(z_{k,1..N})$
    $[f_{st,k},\ \Gamma_{d,k}\left(f_{st,k}\right)]$ = Estimate Step Frequency $(f_m,\ \Gamma_{d,k})$
    $[S_k]$ = Sample Proposal Distribution $(S_{k-1},\ \mu_{V,k},\ \sigma_{V,k})$
    **if** Step frequency power is consistent $(S_k,\ \Gamma_d\left(f_{st}\right))$ **then**
      $[V_{walk,k}]$ = Walking speed model $(f_{st,k})$
      $[S_k]$ = Weighting and Resampling $(S_k,\ V_{walk,k})$
      $[d_k]$ = Compute mean scale factor $(S_k)$
    **else**
      $d_k = d_{k-1}$
    **end if**
    **if** k=1 **then**
      $\left[\hat{C}_{1,1..N}\right]$ = Scale Trajectory Section $(d_1,\ C_{1,1..N})$
    **else**
      $\left[\hat{C}_{k,1..N_f}\right]$ = Scale Trajectory Section $\left(d_k,\ C_{k,(N-N_f+1)..N}\right)$
    **end if**
  **end while**

# Chapter 5

# Experiments

We use a catadioptric omnidirectional camera with a resolution of $1024$x$768$ and a frame rate of $15$ fps. This camera is mounted on a helmet carried by a human operator. The dataset used for the experiments contains,firstly, $3$ outdoor image sequences along the same path of $232$ m and taken at three different step frequencies. The Ground Truth step frequency was fixed by a metronome with with $0.01$ seconds of resolution. It was set up to $0.70$, $0.60$ and $0.50$ seconds per beat for each sequence, which translates in step frequencies of $1.43$ Hz, $1.67$ Hz and $2$ Hz, respectively. Secondly, we acquired an indoor sequence whithout metronome to evaluate the scaling of the trajectory under a normal gait condition.

The experiments are divided in two parts. In the first part we focus only in the analysis of the accuracy in the estimation of the step frequency and select an optimal length of the data sequence with which the DFT is feeded. In the second part we evaluate the global scaling algorithm and compare the performance with different setups of the tunning variables.

## 5.1 Spectral analysis for step frequency estimation

First, we evaluate the feasability of using spectral analysis to measure the step frequency. As stated in Sec. 4.1.1, visual odometry is divided in sections of $N$ camera poses and the DFT is carried out on each section.

To compute the DFT we use the FFTW (Fast Fourier Transform West) C library [8]. We compare different section dimensions of $N = 100$ and $N = 200$. As the routines of this library perform faster when the length of the data sequence is a power of 2, data sequences are padded with zeros to a length of $N_p$ to fill this condition. A greater padding involves an increased resolution of the spectrogram, but it should not provide any improvement in the accuracy of the estimation since no new information is added. Thus, to check this fact, we also compare two zero-padding instances ZP1 and ZP2. ZP1 corresponds to a padding being $N_p$ the power of 2 closest to $N$. ZP2 corresponds to a padding with $N_p = 1024$.

In Fig. 5.1 we show the results of the measured step frequency of the three trajectories with four different DFT setups resulting from the combination of the possible choices of $N$ and $N_p$.

It can be observed that taking $N = 200$ provides more accurate estimations. This is done at the expense of increasing the interval between two consecutive estimations. As expected, it is also shown that a greater zero-padding does not provide any improvement in accuracy. Thus we select a setup of $N = 200$ data points and the ZP1 padding instance to compute the DFT for spectral analysis.

## 5.2 Scaling of the Visual Odometry

### 5.2.1 The Ground Truth

First of all, to evaluate the performance of our algorithm, we need a Ground Truth with which we can compare the results. We have obtained it from the Google Maps satellite view in the following steps:

- Build the walked path in Google Maps with the distance Measurement Tool, saving an image capture of the built path and taking note of the total distance $d_{GMaps}$ in m.

- Load the captured image in MATLAB and build a $N_{pt} \times 2$ matrix $\mathtt{t}\,[px] = [\mathbf{t}_x,\ \mathbf{t}_y]$ of 2D points by consecutively clicking on key points of the trajectory.
- Points of this trajectory are expressed in pixel coordinates. To convert them to meters and obtain the final Ground Truth we apply:

$$\mathtt{t}_{GT}\,[m] = \mathtt{t}\,[px]\,\frac{d_{GMaps}}{\sum_{i=2}^{N_{pt}} \sqrt{(t_{x,i} - t_{x,i-1})^2 + (t_{y,i} - t_{y,i-1})^2}} \tag{5.1}$$

To be able to numerically compare the Ground Truth with the visual odometry estimations provided by our algorithm, we need to establish a pointwise mapping between each point in the estimated trajectory and the Ground Truth. However, in our case this is a difficult task since our Ground Truth points lack from synchronized timestamps to relate them with point of the trajectory. To solve this issue we propose the following method:

- Since Ground Truth has been defined by segments, first we split these segments in points to obtain a fine discretization.

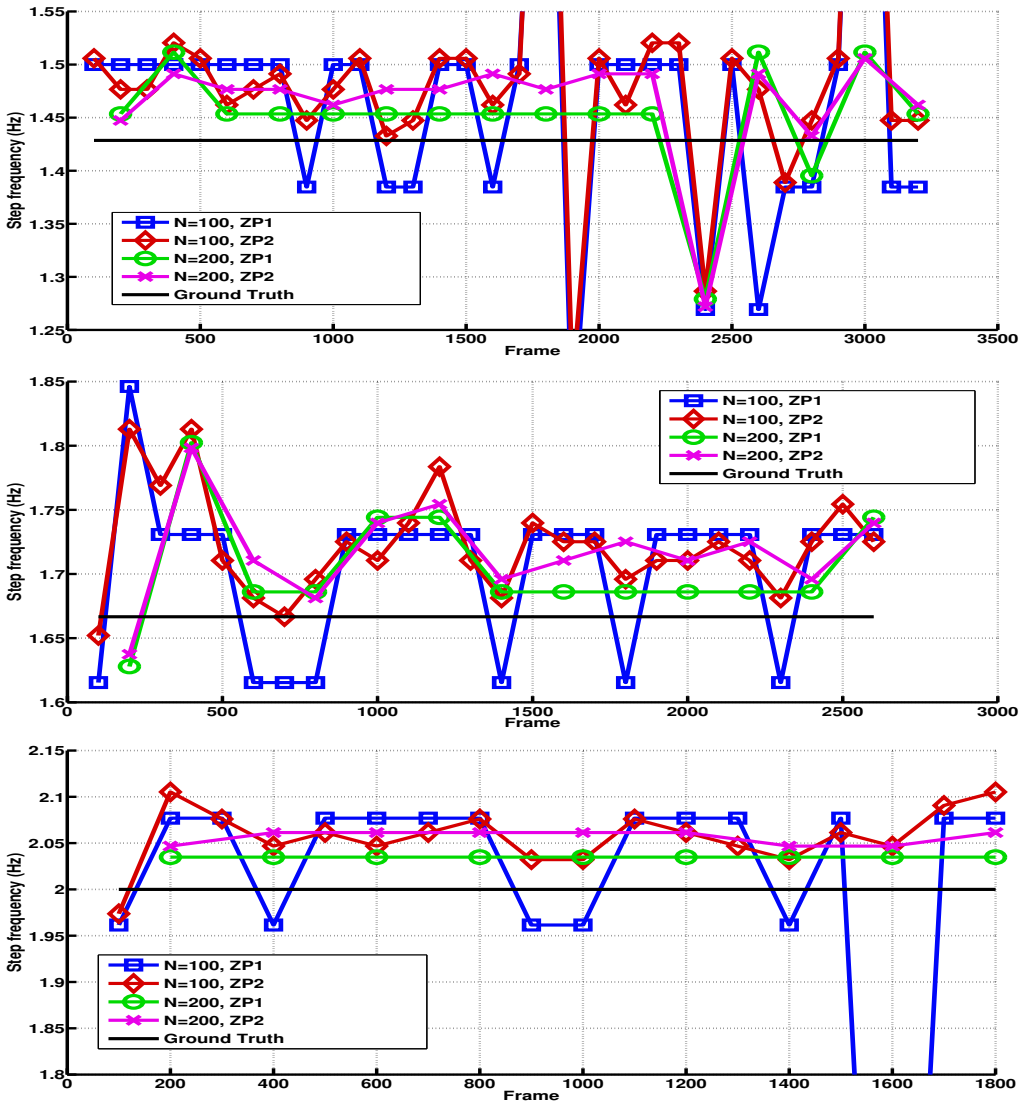



Figure 5.1: Spectral analysis along the same path at the three step frequencies of 1.43 (top), 1.67 (center) and 2 Hz (bottom) with different setups for the computation of the DFT.

- We define a parameter $\alpha$ which is computed for every point of the trajectory as the quotient between the accumulated covered distance and the total distance, ranging from 0 (start) to 1 (end). The same process is applied to the scaled estimations from our algorithm.

- Thus, to compute the estimation error for each point $\mathbf{t}_{VO}^{(i)}$ of the scaled visual odometry estimation we do:

$$\alpha_{VO}^{(i)} = \alpha(\mathbf{t}_{VO}^{(i)}) \tag{5.2}$$

$$\mathbf{t}_{GT}^{err} = \arg\min_{\mathbf{t}_{GT}} \|\alpha_{VO}^{(i)} - \alpha(\mathbf{t}_{GT})\| \tag{5.3}$$

$$error^{(i)} = \mathrm{d}(\mathbf{t}_{VO}^{(i)}, \mathbf{t}_{GT}^{err}) \tag{5.4}$$

### 5.2.2 Setup of the parameters

The scaling algorithm has a series of parameters which have to be adjusted. To this setup we will only consider one of the 3 outdoor sequences (concretely the one taken at $0.70$ seconds per beat). The number of particles used in the particle filter is fixed to $P = 5000$, which is considered to be enough to fill the probability distribution of the 2-dimensional state vector. The standard deviation of the distribution modelling the initial logarithmic scale factor is set to $\sigma_0 = 1$ for all the experiments. This setup allows us to consider an initial uncertainty interval for the scale factor between $10^{-2}$ and $10^2$ with a $95\%$ of confidence. The number of input states from the SLAM algorithm is set to $N = 200$ from Sec. 5.1, and the number of new states to be scaled at each iteration is initially set to the maximum value of $N_f = N = 200$.

The setup of the standard deviations of the distributions modelling the scale drift $\sigma_{drift}$ and the measurement noise $\sigma_{Vwalk}$ of the walking speed estimation is done empirically by testing different values. We have taken 2 posible values both for $\sigma_{drift}$ and $\sigma_{Vwalk}$ and we have considered the four posible combinations of these values. Fig. 5.2a shows that high $\sigma_{drift}$ and low $\sigma_{Vwalk}$ values produce sharper variations of the scale factor, which means that the system is more confident on the estimations of $V_{walk,k}$ from the spectral analysis. On the contrary, a low $\sigma_{drift}$ and a high $\sigma_{Vwalk}$ imply that the estimation of $V_{walk,k}$ is less reliable and thus the new scale factor is more dependent on the previus estimation yielding a moderate curve.

Table 5.1: Estimation error for different configurations of $\sigma_{drift}$ and $\sigma_{Vwalk}$ .

| Configuration | Mean error[m] | Maximum error[m] | Relative mean error |
|---|---|---|---|
| $\sigma_{drift} = 0.05$, $\sigma_{Vwalk} = 0.1$ | 2.82 | 6.68 | 1.22% |
| $\sigma_{drift} = 0.05$, $\sigma_{Vwalk} = 0.2$ | 1.68 | 4.33 | 0.72% |
| $\sigma_{drift} = 0.1$, $\sigma_{Vwalk} = 0.1$ | 3.66 | 6.75 | 1.57% |
| $\sigma_{drift} = 0.1$, $\sigma_{Vwalk} = 0.2$ | 1.63 | 5.10 | 0.70% |

Table 5.1 and visual inspection of Fig. 5.2b show that the configuration $\sigma_{drift} = 0.1$ and $\sigma_{Vwalk} = 0.2$ m/s provides an slightly better scaled visual odometry estimation than the others. So, we select these values to set up the particle filter.

To prove the usefulness of including a particle filter in the estimation of the scale factor, we compare the results obtained using the particle filter with the ones obtained by simply computing the scale factor as $d_k = \frac{V_{walk,k}}{\mu_{V,k}}$. Note that this last approach is equivalent to taking a particle filter where both the walking speed $V_{walk,k}$ and the mean speed from SLAM $\mu_{V,k}$ are considered as perfect measurements (*i.e.*, $\sigma_{Vwalk} = 0$ and $\sigma_{V,k} = 0$). Table 5.2 and Fig. 5.3 show that the softening of the scale factor curve induced by the particle filter, gives raise to a great improvement on the scaled visual odometry estimation.

One proposed improvement on the basic algorithm to reduce the delay between visual odometry updates during real time operation, was to modify the number of states to be scaled at each iteration while maintaining an optimal number $N$ of input states for an accurate computation of the DFT. This is done by implementing FIFO (First In, First Out) routine, taking at each iteration the $N_f$ more recent unscaled

Table 5.2: Estimation error with and without particle filter.

| Particle filter | Mean error[m] | Maximum error[m] | Relative mean error |
|---|---|---|---|
| Yes | 1.70 | 5.12 | 0.73% |
| No | 4.35 | 6.53 | 1.88% |

camera states and eliminating the $N_f$ oldest ones from the input list of states. As it is shown in Table 5.3 and Fig. 5.4 we have tested our approach with different values of $N_f$. We finally choose $N_f = 50$ since it involves a great reduction of the time between updates while producing a rather accurate estimation.

Table 5.3: Estimation error for different values of $N_f$.

| $N_f$ | Mean error[m] | Maximum error[m] | Relative mean error |
|---|---|---|---|
| 200 | 1.77 | 5.33 | 0.76% |
| 100 | 1.45 | 4.34 | 0.62% |
| 50 | 1.66 | 3.85 | 0.71% |
| 20 | 2.52 | 6.19 | 1.09% |

### 5.2.3 Scaling of the trajectories

Having set up the parameters of the scaling algorithm, now we apply it to the acquired trajectories to check the overall accuracy. To recap the used parameters are: $P = 5000$, $\sigma_0 = 1$, $\sigma_{drift} = 0.1$, $\sigma_{Vwalk} = 0.2$ m/s, $N = 200$ and $N_f = 50$.

Firstly, we tested our approach on the three outdoor sequences with different fixed step frequencies. Fig. 5.5 shows the final reconstruction of the trajectory compared to the Ground Truth over a satellite view from Google Maps. It can be observed the great improvement respect to the raw visual odometry estimation from the SLAM algorithm. Notice also that our approach provides a better estimation than applying a uniform scale factor. The reason is that the dynamic estimation of the scale factor every $N_f$ frames allows the correction the scale drift of the raw visual odometry. Nevertheless, note also that the accuracy slightly decreases for the sequences taken at the step frequencies of 1.67 Hz and 2 Hz, which have not been considered during the tunning of the algorithm. Thus optimal set up parameters may somewhat vary on each particular case.

Table 5.4: Estimation error for the three different step frequencies considered.

| Step frequency [Hz] | Mean error[m] | Maximum error[m] | Relative mean error |
|---|---|---|---|
| 1.43 | 1.72 | 3.63 | 0.74% |
| 1.67 | 3.61 | 6.27 | 1.56% |
| 2 | 5.47 | 10.35 | 2.36% |

We have also tested our approach in an indoor environment with normal gait not set by a metronome [20]. In Fig. 5.6 we note that trajectory greatly deviates from the Ground Truth, due to the inaccurate estimation of the camera rotation. The reason of this error is the lack of points located at the infinity in indoor environments. As can be noted in the previous experiments and the example in Sec. 3, in outdoor environments, the presence of points at infinity greatly improve the estimation of turns since they only change their position in the image with the rotation of the camera.

Given the great deviation from the Ground Truth, a numerical evaluation of our approach by analysing the error is not possible. However, qualitatively, it can be observed that the raw visual odometry contains a great amount of scale drift which has been corrected in the scaled estimation.

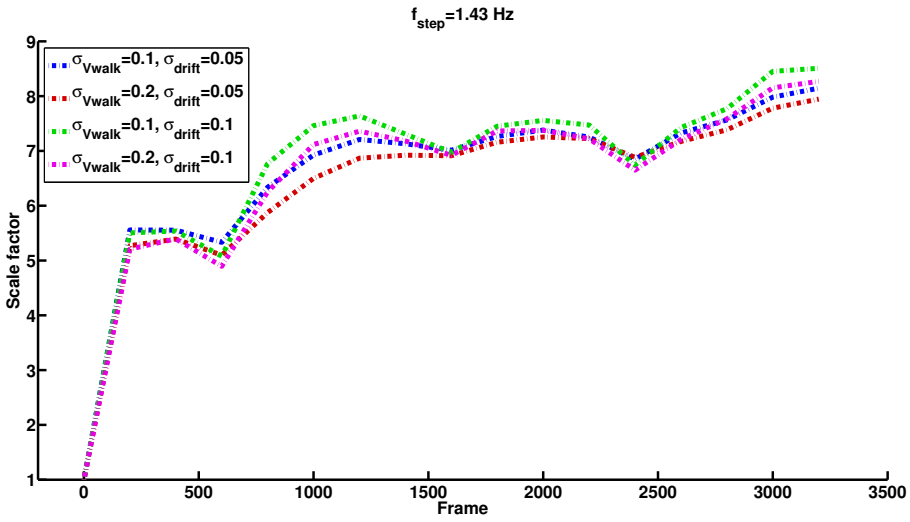
## 5.3 Analysis of the computational cost

Prior to its real-time implementation we analyzed the computational cost of the batch algorithm implemented in MATLAB. In Fig. 5.7 we show the computation time to perform the whole algorithm

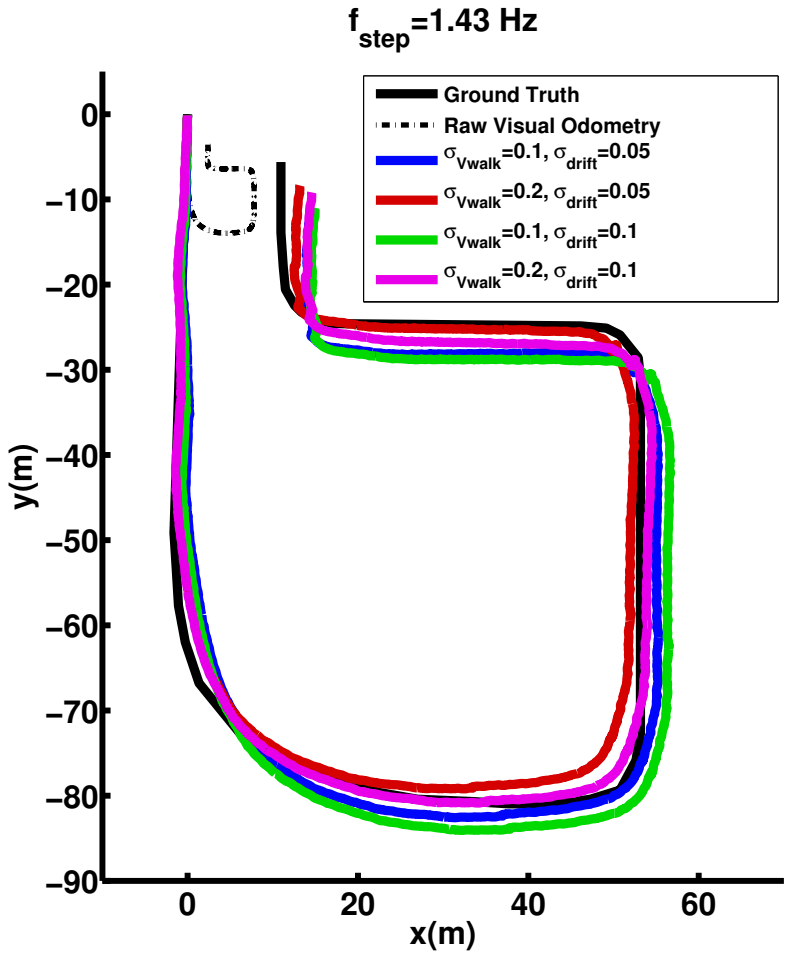
to scale each section of the visual odometry. After a cost of almost $0.2$ second, probably due to the memory allocation of the new variables, the computational cost stabilizes around $0.01$ seconds. Direct sequential implementation in the monoSLAM application could suppose a problem, since our algorithm would take a significant fraction of the time between frames ($\Delta_t = \frac{1}{15} = 0.067$ seconds). However, a parallelized implementation allows the visual SLAM algorithm to make use of all the available time between frames, while our approach runs within the time window that takes to renew the list of $N$ states for the next iteration.

## 5.4 Analysis of the power consistency condition

The evaluate the performance of the algorithm with the check of the power consistency we take the indoor sequence, where there exists a part of the trajectory which includes stairs. This part corresponds aproximately to the frames between $3500$ and $4000$. If we observe the estimation of the step frequency along time in Fig. 5.8, for the frames within this interval, the estimated step frequency suddenly decreases while its spectral power increases. Though it does not reach the top limit proposed in Sec. 4.3, an inconsistency is likely to have ocured since normally, a lower step frequency implies a lower amplitude of the head vertical oscillation [14]. Thus to make the consistency test fail in this part we have empirically set up a new top limit. Fig. 5.9 shows the results of our algorithm including the consistency test. It can be observed that the fail of the consistency test makes the previous scale factor to be mantained instead of updating it with a value obtained from a walking speed model which is not proved to be valid in the case going up/down stairs.

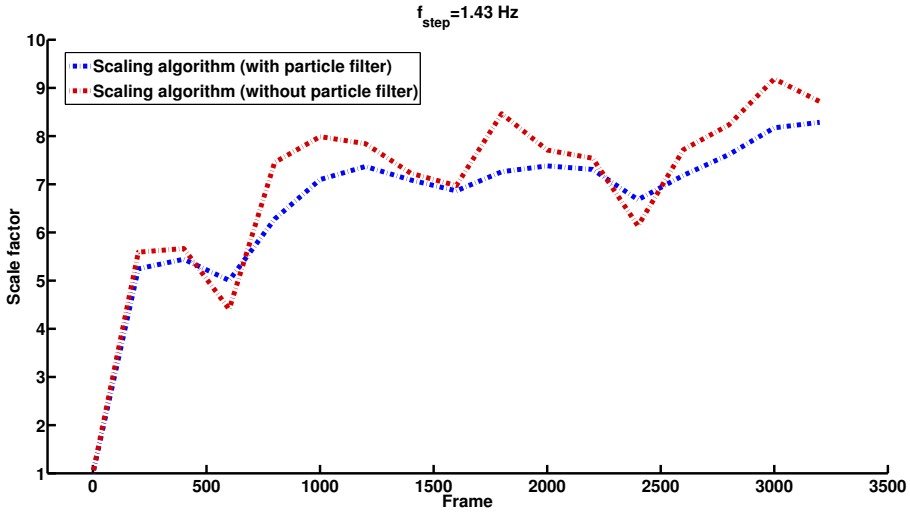



(a)

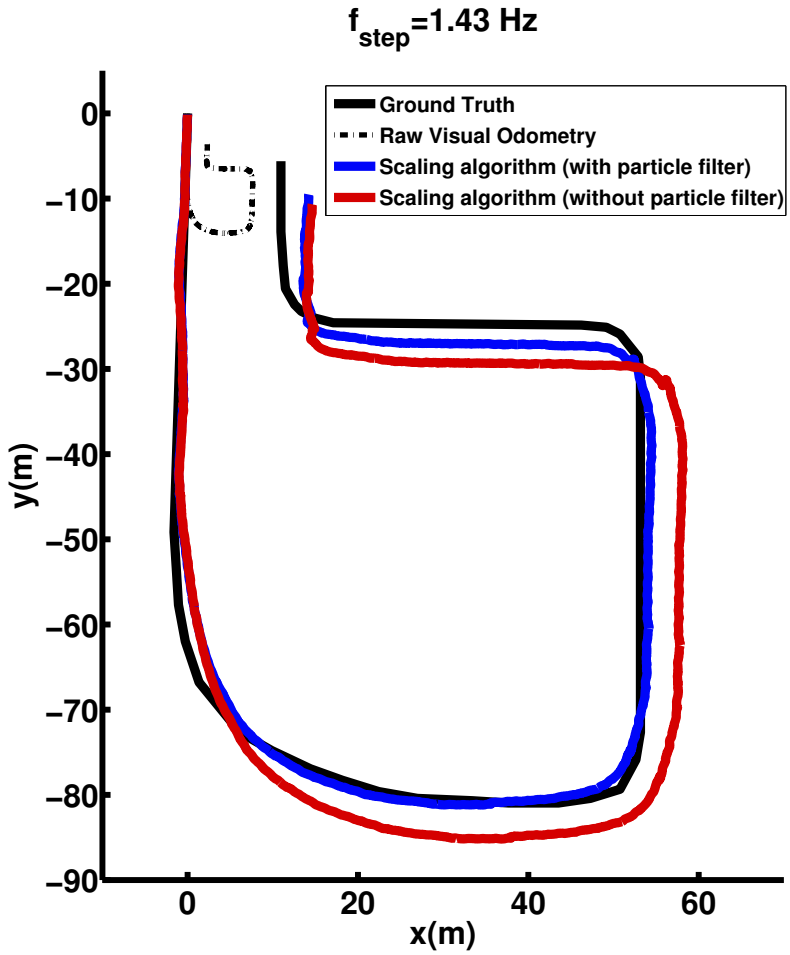



(b)

Figure 5.2: (a) Scale factor and (b) scaled visual odometry for different setups of the particle filter.

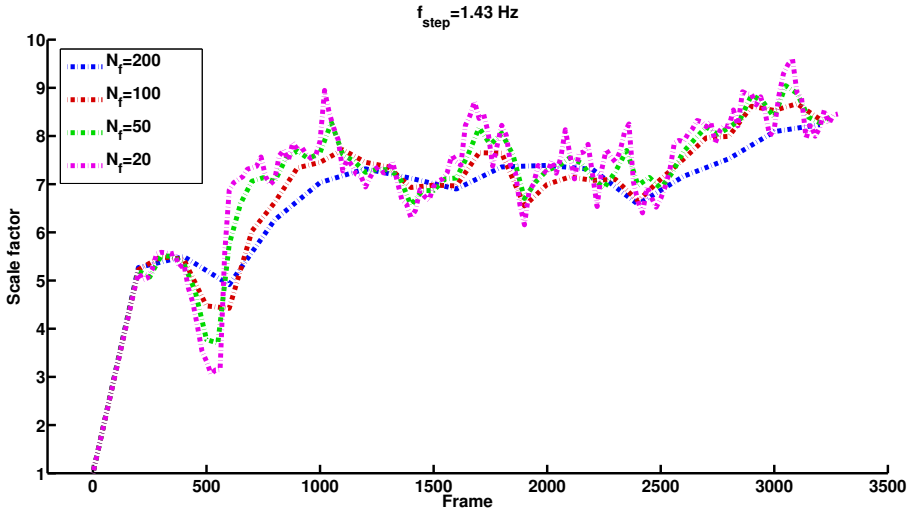



(a)

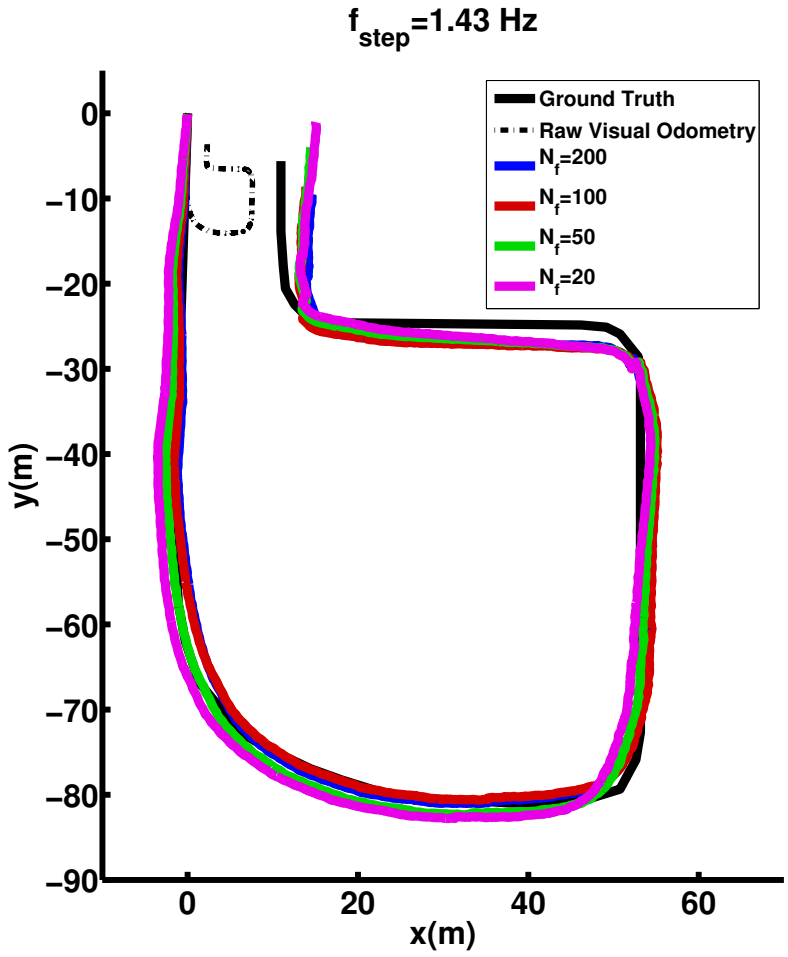



(b)

Figure 5.3: (a) Scale factor and (b) scaled visual odometry with(blue) and without(red) particle filter.

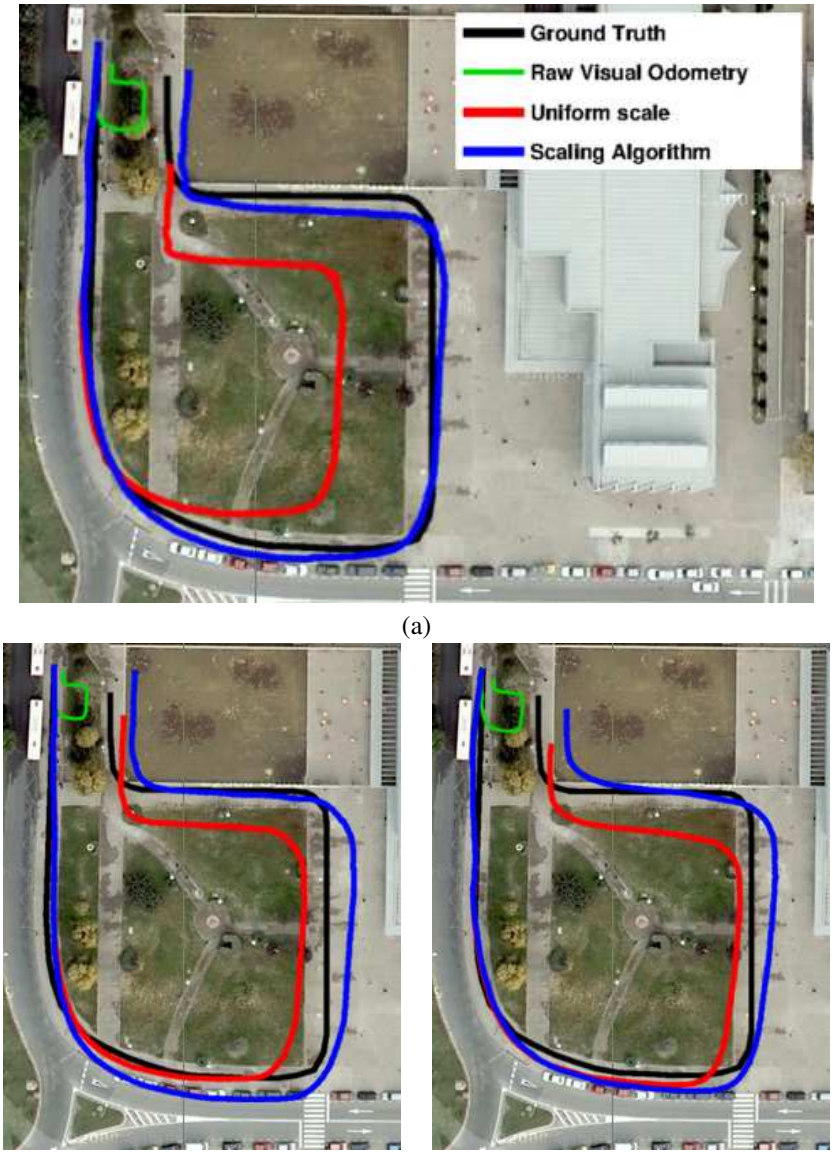



(a)




(b)

Figure 5.4: (a) Scale factor and (b) scaled visual odometry for different choices of $N_f$.




(a)

(b) (c)

Figure 5.5: Visual odometry estimations using different approaches on the three trajectories walked at different step frequencies of about 1.43 Hz (a), 1.67 Hz (b) and 2 Hz (c).

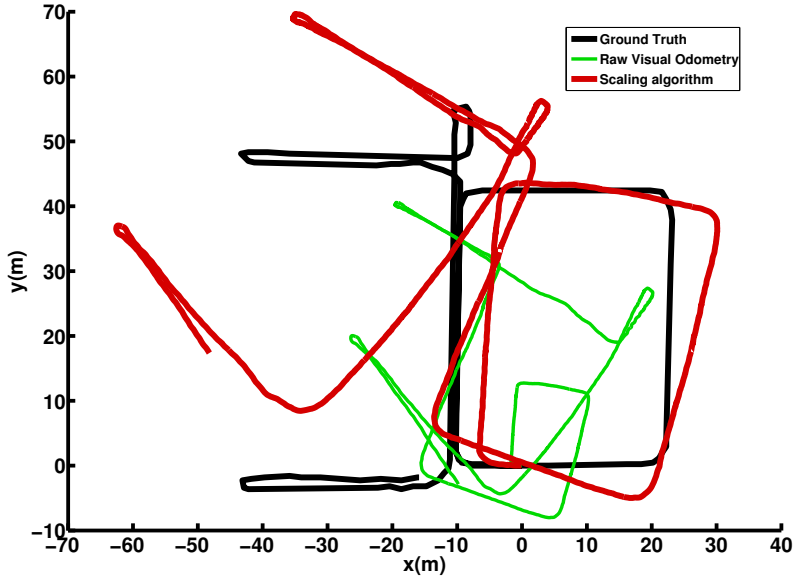



Figure 5.6: Scaled visual odometry estimation in an indoor environment with normal gait compared to the raw SLAM estimation. Note that the squared section in the middle has partly recovered its shape, practically inobservable in the raw estimation. Also, the great drift which can be appreciated on the top left branch of the trajectory has been totally corrected.

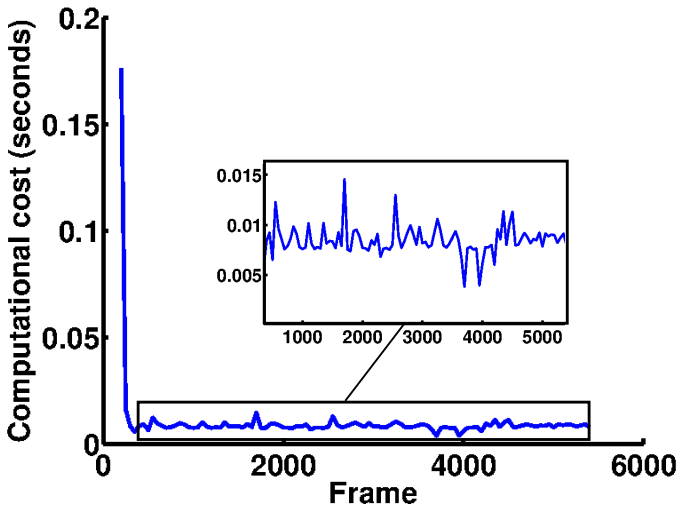



Figure 5.7: Computation time used to perform our approach for the different sections of the indoor sequence.

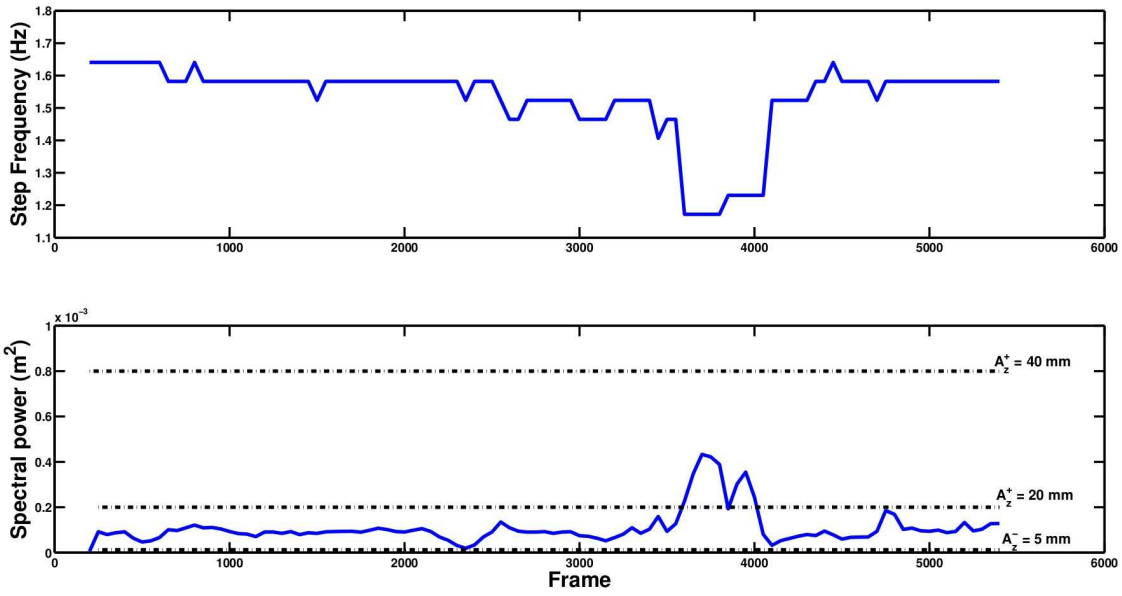
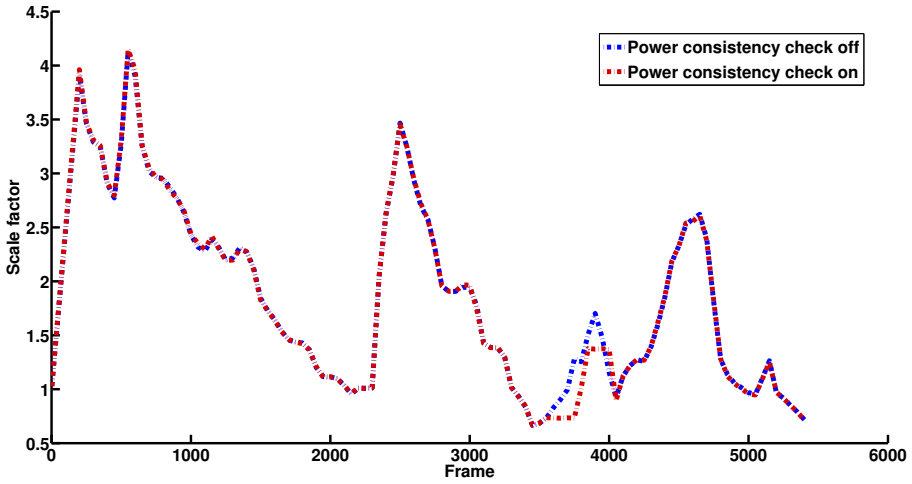
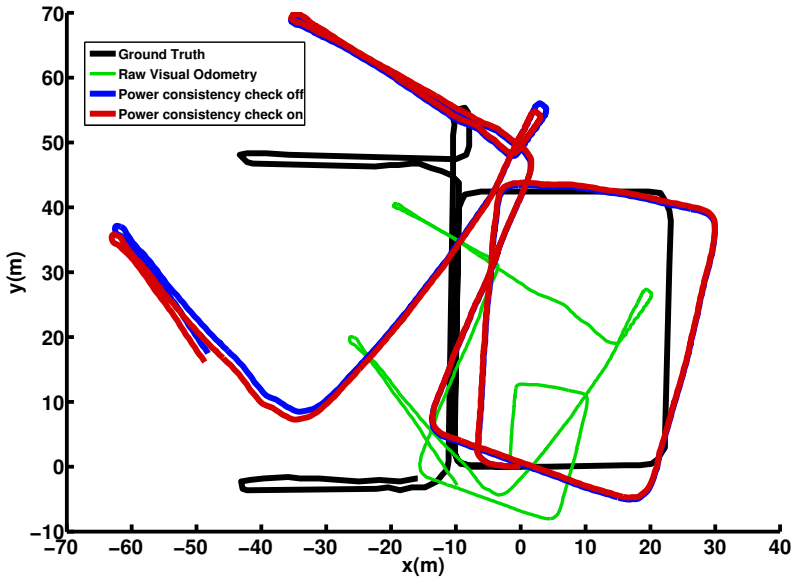



Figure 5.8: Estimated step frequency (top) and power of the head oscillation (bottom) along time for the indoor sequence of images. Frames between 3500 and 4000 correspond to a zone with stairs. Note how the estimated step frequency suddenly drops while the associated power increases. This anormal change in the power can be used to detect a situation where the walking model cannot be applied, and follow to another strategy.




(a)




(b)

Figure 5.9: (a) Scale factor and (b) scaled visual odometry with(red) and without(blue) checking the consistency of the step frequency.

# Chapter 6

# Conclusions and Future Work

In this work we have presented a novel approach to estimate the true scaled visual odometry of a head-mounted omnidirectional camera without need of additional sensors. The general idea behind our method is to take advantage of the head vertical movement registered in the unscaled visual odometry from the SLAM algorithm to obtain the step frequency. Given the step frequency and assuming a human walking model we can compute a real estimation of the walking speed, from which finally we get an scale factor. To improve the accuracy and correct the scale drift of the raw visual odometry, the scale factor is computed for sections of camera states using a particle filter to reliably update it.

The algorithm has been validated experimentally obtaining very satisfactory results. The improvement respect to the raw estimation is clearly noticeable and it has proved to be able to correct a large amount of scale drift present in the visual odometry estimation from an indoor environment. Also its low computational cost and its capability to be executed concurrently without interfering with the main SLAM algorithm made it possible its implementation in the framework of a real-time monoSLAM application.

From our point of view the main contribution of this approach is that, while there exist methods which can accurately determine the scale for cameras mounted on wheeled vehicles, to the best of our knowledge there does not exist any method which does so with wearable cameras.

As future work we will explore the open possibility of making more use of the information provided by the power spectrum of the head oscillation to detect special situations such as stopping, sudden speed variation, stairs and develop strategies to cope with them. For that purpose we will acquire new sequences of images where these situations arise.

# Appendix A

# The Extended Kalman Filter

The Extended Kalman Filter is a recursive estimator based on dynamic systems discretized in the time domain. The EKF aims to estimate the internal state of a system or process given only a sequence of noisy observations and, optionally, control inputs. To perform this estimation we have to define a state transition model and a measurement model.

The state transition model describes the evolution of the system from time $k-1$ to time $k$, and it is defined by the following equation:

$$\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_k) + \mathbf{w}_k \tag{A.1}$$

where $\mathbf{f}(\cdot)$ is the state transition function , $\mathbf{x}_{k-1}$ is the past state of the system, $\mathbf{u}_k$ is the control input and $\mathbf{w}_k$ is the process noise modeled as zero mean uncorrelated gaussian noise with covariance $\mathsf{Q}_k$.

The measurement model is defined as follows:

$$\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k) + \mathbf{v}_k \tag{A.2}$$

where $\mathbf{h}(\cdot)$ is the measurement function, $\mathbf{x}_k$ is the state of the system and $\mathbf{v}_k$ is the additive observation error modeled as zero mean uncorrelated gaussian noise with covariance $\mathsf{R}_k$

In an Extended Kalman Filter the state of the system is represented by a mean vector $\hat{\mathbf{x}}_k$ and a covariance matrix $\mathsf{P}_k$.

At each iteration the next state estimate is performed in two steps: Prediction and Update. In the prediction, a state estimate in the current timestep is produced from the state estimate in the previous time step and a control input by using the state transition model. This is encoded in the following equations:

$$\hat{\mathbf{x}}_{k|k-1} = f(\hat{\mathbf{x}}_{k-1|k-1},\ \mathbf{u}_k) \tag{A.3}$$

$$\mathsf{P}_{k|k-1} = \mathsf{F}_{k-1}\mathsf{P}_{k-1|k-1}{\mathsf{F}_{k-1}}^{\mathsf{T}} + \mathsf{Q}_k \tag{A.4}$$

where $f$ is the state transition function, $\hat{\mathbf{x}}_{k|k-1}$ and $\mathsf{P}_{k|k-1}$ are the state mean and covariance estimates at timestep $k$ from measurements until timestep $k-1$ and $\mathsf{F}_{k-1}$ is the jacobian of the state transition function:

$$\mathsf{F}_{k-1} = \frac{\partial f(\mathbf{x}, \mathbf{u})}{\partial \mathbf{x}}|_{\mathbf{x}_{k-1|k-1}}$$

In the update step the initial prediction is refined by including the measurements $\mathbf{z}_k$ taken in the current timestep. To do that, first it is computed the innovation of the measurement as the difference between the real measurements $\mathbf{z}_k$ provided by the sensors and the prediction of the measurements $h(\mathbf{x}_{k|k-1})$ given by the measurement model. This innovation has an associated covariance $\mathsf{S}_k$ which encodes both the propagation of the state uncertainty through the measurement model and the possible measurement errors.

$$\nu_k = \mathbf{z}_k - h(\hat{\mathbf{x}}_{k|k-1}) \tag{A.5}$$

$$\mathsf{S}_k = \mathsf{H}_k\mathsf{P}_{k|k-1}{\mathsf{H}_k}^{\mathsf{T}} + \mathsf{R}_k \tag{A.6}$$

where $\mathsf{H}_k$ is the jacobian of the measurement function:

$$\mathsf{H}_k = \frac{\partial h(\mathbf{x})}{\partial \mathbf{x}}|_{\mathbf{x}_{k|k-1}}$$

Next it is computed the Kalman gain $\mathsf{W}_k$, which intutivelly speaking points how much we can trust in the new measurements to update the initial state prediction. This gain is used to weight the innovation when computing the final state mean and covariance in timestep $k$.

$$\mathsf{W}_k = \mathsf{P}_{k|k-1}\mathsf{H}_k{}^{\mathsf{T}}\mathsf{S}_k{}^{-1} \tag{A.7}$$

$$\mathbf{x}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathsf{W}_k\nu_k \tag{A.8}$$

$$\mathsf{P}_{k|k} = \mathsf{P}_{k|k-1} - \mathsf{W}_k\mathsf{S}_k\mathsf{W}_k{}^{\mathsf{T}} \tag{A.9}$$

# Appendix B

# The Sphere Camera Model

The Sphere Camera Model is a unified projection model valid for every central catadioptric system, *i.e.* a system with a unique projection center. This model was developed by Geyer *et al.* [9] and extended by Barreto *et al.* [1].

The model takes as the origin of the reference system $\mathbf{O}$, the origin of the central system which is modeled (one focus of the hyperbola/parabola in the case of hiper/para-catadioptric systems or the optical center of the camera in the case of a perspective conventional camera). Then they define a unit sphere $S$ centered on the origin of the reference system and a point $\mathbf{C_P} = (0,\ 0,\ -\xi)^{\mathsf{T}}$ known as virtual projection center.

The information about the mirror is encapsulated in the characteristic parameters $\xi$ and $\psi$. The parameter $\xi$ is defined as the distance between $\mathbf{O}$ and $\mathbf{C_P}$ and it encodes the kind of system being modeled and its geometry. So, $\xi = 0$ for perspective cameras, $\xi = 1$ for para-catadioptric systems and $0 < \xi < 1$ for hiper-catadioptric systems. Table B.1 shows the values of $\xi$ and $\psi$ for every kind of system as a function of the the distance between the focus $d$ and the latus rectum $4p$.

Taking a 3D point expressed in homogeneus coordinates $\mathbf{X_w} = [x, y, z, 1]$, its projection on the image is divided in the following steps (Fig. B.1 and Fig. B.2):

1) Point $\mathbf{X_w}$ is mapped into a projective ray $\mathbf{x}$ in the camera reference frame. This is done by $\mathsf{P}$, a conventional projection matrix $\mathbf{x} = \mathsf{P}\mathbf{X_w}$.

2) The ray $\mathbf{x}$ is projected onto the unit sphere centered in the origin $\mathbf{O}$. The intersection point is projected to a virtual projection plane $\pi$ through the virtual projection center $\mathbf{C_P}$ yielding the point $\mathbf{x}'$ . This step is coded by the non-linear function $\hbar$:

$$\mathbf{x}' = \hbar(\mathbf{x}) = \begin{pmatrix} x \\ y \\ z + \xi\sqrt{x^2 + y^2 + z^2} \end{pmatrix} \tag{B.1}$$

3) The virtual plane $\pi$ is transformed in the image plane $\pi_{IM}$ through a homographic transformation $\mathsf{H_c}$

$$\mathbf{x}'' = \mathsf{H_c}\mathbf{x}' \tag{B.2}$$

$$\mathsf{H_c} = \mathsf{K_c}\mathsf{R}\mathsf{M_c} \tag{B.3}$$

$$\mathsf{K_C} = \begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{B.4}$$

$$\mathsf{M_C} = \begin{bmatrix} \psi - \xi & 0 & 0 \\ 0 & \xi - \psi & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -\eta & 0 & 0 \\ 0 & \eta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{B.5}$$

| | $\xi$ | $\psi$ |
|---|---|---|
| Espejo parabólico | 1 | $1+2p$ |
| Espejo hiperbólico | $\frac{d}{\sqrt{d^2+4p^2}}$ | $\frac{d+2p}{\sqrt{d^2+4p^2}}$ |
| Espejo elíptico | $\frac{d}{\sqrt{d^2+4p^2}}$ | $\frac{d-2p}{\sqrt{d^2+4p^2}}$ |
| Cámara perspectiva | 0 | 1 |

Table B.1: Characteristic parameters of the spherical camera Model [1]

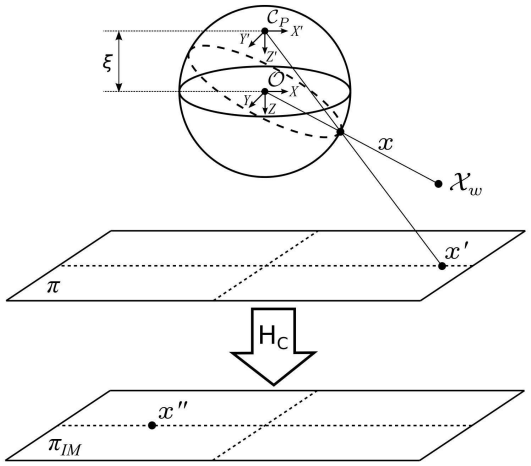



Figure B.1: Projection of a 3D $\mathbf{X}_w$ point onto the image plane with the spherical camera model

where $\mathsf{K_C}$ includes the camera intrinsic parameters, $\mathsf{M_C}$ includes the mirror parameters [9] and $\mathsf{R}$ is the rotation matrix between camera and mirror. By assuming a pin-hole camera model and $\mathsf{R} = \mathsf{I}$, the transformation $\mathsf{H_C}$ yields:

$$\mathsf{H_C} = \begin{bmatrix} \eta f & 0 & u_0 \\ 0 & \eta f & v_0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \gamma & 0 & u_0 \\ 0 & \gamma & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{B.6}$$

where $\gamma = \eta f$ is the generalized focal lenght of the camera-mirror system with $\eta$ a mirror parameter and $f$ the focal length of the camera.

4) Finally image coordinates are calculated by dividing $\mathbf{x}''$ by its $z''$ coordinate:

$$\mathbf{p} = \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = f_u(\mathbf{x}'') = \begin{pmatrix} \frac{x''}{z''} \\ \frac{y''}{z''} \\ \frac{z''}{z''} \end{pmatrix} \tag{B.7}$$

With this model it is also possible to estimate the 3D ray from where the image point comes. That projection is named the inverse projection model. It starts with the point in image coordinates $\mathbf{p} = (u,\ v)^\mathsf{T}$, being $\mathbf{x}'' = (u,\ v,\ 1)^\mathsf{T}$. The equations of the inverse projection model are:

$$\mathbf{x}' = \mathsf{H_c}^{-1}\mathbf{x}'' \tag{B.8}$$

$$\mathbf{x} = \hbar^{-1}(\mathbf{x}') = \begin{pmatrix} x' \\ y' \\ z' - \frac{\xi(x'^2+y'^2+z'^2)}{\xi z'^2+\chi} \end{pmatrix} \tag{B.9}$$

where $\chi = \sqrt{(1-\xi^2)(x'^2+y'^2+z'^2)}$

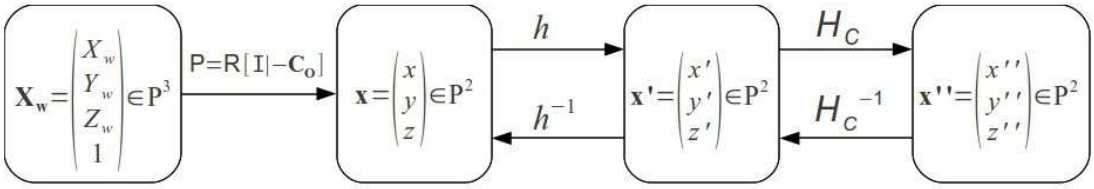



Figure B.2: Steps of spherical camera model projection

## B.1 The Spherical Camera Model for the EKF

The EKF algorithm requires the jacobian $\mathsf{H}_k$ of the measurement function $\mathbf{h}(\mathbf{x}_k)$. In the case of a catadioptric camera, the measurement function $\mathbf{h}(\mathbf{x}_k)$ corresponds to the projection function of the spherical camera model, and thus $\mathsf{H}_k$ corresponds to the jacobian of this function, which is computed as follows [24]:

$$\mathsf{H}_k = \mathsf{J}_{\mathsf{SC}} = \mathsf{J}_{\mathsf{fu}}\mathsf{H}_{\mathsf{C}}\mathsf{J}_{\hbar} \tag{B.10}$$

$$\mathsf{J}_{\mathsf{fu}} = \begin{bmatrix} \frac{1}{z''} & 0 & -\frac{x''}{z''^2} \\ 0 & \frac{1}{z''} & -\frac{y''}{z''^2} \end{bmatrix} \tag{B.11}$$

$$\mathsf{J}_{\hbar} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \frac{\xi x}{\rho} & \frac{\xi y}{\rho} & 1+\frac{\xi z}{\rho} \end{bmatrix} \tag{B.12}$$

where $\rho = \sqrt{x^2+y^2+z^2}$
To initialize new features, the inverse jacobian of the model is also required:

$$\mathsf{J_{SC}}^{-1} = \mathsf{J}_{\hbar^{-1}}\mathsf{H_C}^{-1} \tag{B.13}$$

$$\mathsf{J}_{\hbar^{-1}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -\frac{\xi x'}{\chi} & -\frac{\xi y'}{\chi} & 1-\frac{\xi(z'-\xi\frac{x'^2+y'^2+z'^2}{\xi z'+\chi})}{\chi} \end{bmatrix} \tag{B.14}$$

where $\chi = \sqrt{(1-\xi^2)(x'^2+y'^2+z'^2)}$

# List of Figures

# List of Tables

# Bibliography


[1] J. Barreto and H. Araujo. Issues on the geometry of central catadioptric image formation. In: *Computer Vision and Pattern Recognition (CVPR), pp. 422–427*, 2001.

[2] J. Civera, A. J. Davison, and J. M. M. Montiel. Inverse depth parametrization for monocular slam. *IEEE Transactions on Robotics*, 24(5):932–945, 2008.

[3] J. Civera, O. G. Grasa, A. J. Davison, and J. M. M. Montiel. 1-Point RANSAC for EKF Filtering: application to real-time structure from motion and visual odometry. *Journal of Field Robotics*, 27(5):609–631, 2010.

[4] P. Corke, D. Strelow, and S. Singh. Omnidirectional visual odometry for a planetary rover. In: *Intelligent Robots and Systems (IROS)*, 4:pp. 4007-4012, 2004.

[5] S. Cumani, A. Denasi, A. Guiducci, and G. Quaglia. Integrating monocular vision and odometry for slam. *WSEAS Transactions on Computers*, 3:625–630, 2004.

[6] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. Monoslam: Real-time single camera slam. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29:1052–1067, 2007.

[7] A. Eudes, M. Lhuillier, S. Naudet-Collette, and M. Dhome. Fast odometry integration in local bundle adjustment-based visual slam. In: *International Conference on Pattern Recognition (ICPR)*, volume 0, pages 290–293, 2010.

[8] M. Frigo and S. G. Johnson. The design and implementation of FFTW3. *Proceedings of the IEEE*, 93(2):216–231, 2005. Special issue on "Program Generation, Optimization, and Platform Adaptation".

[9] C. Geyer and K. Daniilidis. A unifying theory for central panoramic systems and practical applications. In *European Conference on Conputer Vision (ECCV) (2)*, pp. 445–461, 2000.

[10] N. J. Gordon, D. J. Salmond, and A. F. M. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *Radar and Signal Processing, IEE Proceedings F* , 140(2):107–113, Apr. 1993.

[11] D. Grieve and R. J. Gear. The relationships between length of stride, step frequency, time of swing and speed of walking for children and adults. *Ergonomics*, 5(9):379–399, 1966.

[12] D. Gutierrez, A. Rituerto, J. M. M. Montiel, and J. J. Guerrero. Adapting a real-time monocular visual slam from conventional to omnidirectional cameras. In *11th OMNIVIS, held with International Conference on Computer Vision (ICCV)*, 2011.

[13] D. Gutiérrez-Gómez. *Localización por visión omnidireccional para asistencia personal*. Proyecto Fin de Carrera, Universidad de Zaragoza, 2011.

[14] E. Hirasaki, S. T. Moore, T. Raphan, and B. Cohen. Effects of walking velocity on vertical head and body movements during locomotion. *Experimental Brain Research*, 127(2):117–130, 1999.

[15] A. D. Kuo. A simple model of bipedal walking predicts the preferred speed-step length relationship. *Journal of Biomechanical Engineering*, 123:264–269, 2001.

[16] P. Lothe, S. Bourgeois, E. Royer, M. Dhome, and S. Naudet-Collette. Real-time vehicle global localisation with a single camera in dense urban areas: Exploitation of coarse 3d city models. In: *Computer Vision and Pattern Recognition (CVPR)*, pp. 863–870, 2010.

[17] T. Lupton and S. Sukkarieh. Removing scale biases and ambiguity from 6dof monocular slam using inertial. In: *International Conference on Robotics and Automation (ICRA)*, pp. 3698–3703. IEEE, 2008.

[18] C. Mei. *Laser-Augmented Omnidirectional Vision for 3D Localisation and Mapping*. PhD thesis, INRIA Sophia Antipolis, Project-team ARobAS, 2007.

[19] C. Mei, G. Sibley, M. Cummins, P. Newman, and I. Reid. Rslam: A system for large-scale mapping in constant-time using stereo. *International Journal of Computer Vision*, 94(2):198–214, 2010.

[20] A. C. Murillo, D. Gutiérrez-Gómez, A. Rituerto, L. Puig, and J. J. Guerrero. Wearable omnidirectional vision system for personal localization and guidance. In: *2nd IEEE Workshop on Egocentric (First-Person) Vision, held with CVPR*, 2012.

[21] D. Nistér, O. Naroditsky, and J. Bergen. Visual odometry for ground vehicle applications. *Journal of Field Robotics*, 23:3–20, 2006.

[22] 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*, 61(1-4):287–299, 2010.

[23] L. M. Paz, P. Piniés, J. D. Tardós, and J. Neira. Large scale 6dof slam with stereo-in-hand. *IEEE Transactions on Robotics*, 24(5):946–957, 2008.

[24] A. Rituerto, L. Puig, and J. J. Guerrero. Visual slam with an omnidirectional camera. In: *International Conference on Pattern Recognition (ICPR)*, pp. 348–351, 2010.

[25] D. Scaramuzza, F. Fraundorfer, M. Pollefeys, and R. Siegwart. Absolute scale in structure from motion from a single vehicle mounted camera by exploiting nonholonomic constraints. *International Conference on Computer Vision (ICCV)*, pp. 1413–1419, 2009.

[26] D. Scaramuzza, F. Fraundorfer, and R. Siegwart. Real-time monocular visual odometry for on- road vehicles with 1-point ransac. In: *International Conference on Robotics and Automation (ICRA)*, pp. 4293–4299, 2009.

[27] H. Strasdat, J. M. M. Montiel, and A. Davison. Scale drift-aware large scale monocular slam. In: *Robotics: Science and Systems (RSS)*, 2010.

[28] J.-P. Tardif, Y. Pavlidis, and K. Daniilidis. Monocular visual odometry in urban environments using an omnidirectional camera. In: *Intelligent Robots and Systems (IROS)*, pp. 2531-2538, 2008.

[29] M. Zarrugh, F. Todd, and H. Ralston. Optimization of energy expenditure during level walking. *European Journal of Applied Physiology and Occupational Physiology*, 33:293–306, 1974.