Visual Terrain Estimation For Legged Robots

Frederike Dümbgen

Master's Thesis
Ecole polytechnique fédérale de Lausanne
Department of Mechanical Engineering
Supervisors: Alireza Karimi (EPFL)
Michael Blösch (ETHZ)
Philipp Krüsi (ETHZ)
Dominik Schindler (ETHZ)

Zurich, 29 July 2016

1

Notation and Basics

1.1 Geometry

1.1.1 Vectors and Coordinate Frames

The vector and coordinate frame notations used throughout this report follow the conventions introduced in [**?**]. If $\mathcal{M}$ is the coordinate frame associated with the map, and A, B are two 3D points, then ${}_A\boldsymbol{r}_{AB}$ contains the coefficients of the vector pointing from A to B expressed in the coordinate system $\mathcal{A}$. In the same manner, the rotation matrix $\boldsymbol{C}_{BA}$ is the (passive) matrix linking the coordinates in a frame $\mathcal{A}$ to the coorindates in $\mathcal{B}$ like

$$ {}_B\boldsymbol{r}_{AB} = \boldsymbol{C}_{BA} \left({}_A\boldsymbol{r}_{AB} \right) \ . \tag{1.1} $$

1.1.2 Rotation Representations

For convenient handling of rotations, i.e. their derivatives and updates, the rotation vectors $\boldsymbol{\Phi}_{C_kM}$, $\boldsymbol{\Phi}_{SM} \in \mathbb{R}^3$ are introduced as recommended in [**?**] to represent the relative orientations of the stereo and camera frames with respect to the map frame, respectively. The corresponding updates in rotation are denoted by $\boldsymbol{\varphi}_{C_kM}$ and $\boldsymbol{\varphi}_{SM}$.

The new rotation $\boldsymbol{\Phi}^{k+1}_{C_kM}$, resulting from a relatively small change in rotation $\boldsymbol{\varphi}_{C_kM}$, can be conveniently expressed using the $\boxplus$-operator as

$$ \begin{aligned} \boldsymbol{\Phi}^{k+1}_{C_kM} &= \boldsymbol{\Phi}^{k}_{C_kM} \boxplus \boldsymbol{\varphi}_{C_kM} && (1.2) \\ &= \exp(\boldsymbol{\varphi}_{C_kM}) \circ \boldsymbol{\Phi}^{k}_{C_kM} && (1.3) \\ &= \boldsymbol{C}(\boldsymbol{\varphi}_{C_kM}) \boldsymbol{C}(\boldsymbol{\Phi}^{k}_{C_kM}) && (1.4) \end{aligned} $$

The mapping $\boldsymbol{C} : SO(3) \implies \mathbb{R}^{3\times 3}$ applied to the orientation vector $\boldsymbol{\Phi}_{C_kM}$ corresponds to the rotation matrix $\boldsymbol{C}_{C_k M}$since it is defined as

$$\boldsymbol{\Phi}_{C_k M}(\boldsymbol{r}) := \boldsymbol{C}(\boldsymbol{\Phi}_{C_k M})\boldsymbol{r} \tag{1.5}$$
$$= \boldsymbol{C}_{C_k M}(\boldsymbol{r}) \ . \tag{1.6}$$

The transformation defined in 1.6 can be derived with respect to the orientation vector $\boldsymbol{\Phi}_{C_k M}$and the vector $\boldsymbol{r}$, yielding

$$\frac{\partial \boldsymbol{\Phi}_{C_k M}(\boldsymbol{r})}{\partial \boldsymbol{r}} = \boldsymbol{C}(\boldsymbol{\Phi}_{C_k M}) \ , \quad \frac{\partial \boldsymbol{\Phi}_{C_k M}(\boldsymbol{r})}{\partial \boldsymbol{\Phi}} = -(\boldsymbol{\Phi}_{C_k M}(\boldsymbol{r}))^{\times} \ , \tag{1.7}$$

and

$$\frac{\partial (\boldsymbol{\Phi}_{C_k M} \circ \boldsymbol{\Phi}_{SM})(\boldsymbol{r})}{\partial \boldsymbol{\Phi}_{SM}} = \boldsymbol{C}(\boldsymbol{\Phi}_{C_k M}) \ . \tag{1.8}$$

1.2 Camera

1.2.1 Camera Model

Having introduced the orientation vectors, the camera poses can be described by

$$\boldsymbol{\xi}_{C_k} := \begin{bmatrix} {}_M\boldsymbol{r}_{MC_k} & \boldsymbol{\varphi}_{C_k M} \end{bmatrix}^T$$
$$\boldsymbol{\xi}_{S} := \begin{bmatrix} {}_M\boldsymbol{r}_{MS} & \boldsymbol{\varphi}_{SM} \end{bmatrix}^T \ .$$

The transformation from object space coordinates ${}_M\boldsymbol{r}_{MX_j}$to pixel coordinates $\boldsymbol{u}_{k,j}$ can be split up in the transformation

$${}_{C_k}\boldsymbol{r}_{C_k X_j} = \boldsymbol{C}_{C_k M}({}_M\boldsymbol{r}_{MX_j} -_M \boldsymbol{r}_{MC_k}),$$

which is essentially a coordinate transformation from the map frame to the camera frame using the camera's extrinsic parameters, and

$$\boldsymbol{u}_{k,j} = \boldsymbol{K}_k D_k(\boldsymbol{\pi}({}_{C_k}\boldsymbol{r}_{C_k X_j})), \tag{1.9}$$

which assigns a pixel value to each camera coordinate using the camera model and its intrinsic parameters.

The model used to approximate the real camera is a pinhole model suspect to radial - tangential distortion. A pinhole model assigns to each point ${}_{C_k}\boldsymbol{r}_{C_k X_j}$its projection

$$\tilde{\boldsymbol{u}} = \boldsymbol{\pi}({}_{C_k}\boldsymbol{r}_{C_k X_j}) = \begin{bmatrix} x/z & y/z & 1 \end{bmatrix}^T \tag{1.10}$$

The obtained normalized coordinates $\tilde{\boldsymbol{u}}$ are then distorted using the distortion model D_k defined by [?], which can be scaled and shifted using the camera matrix $\boldsymbol{K}_k$ in order to give pixels the final pixels $\begin{bmatrix} u, v \end{bmatrix}$ lying in $\begin{bmatrix} 0 \ldots w, 0 \ldots h \end{bmatrix}$.

1.2.2 Stereo Parameters

The same considerations as above can be made when unrectified instead of rectified images are available. The camera calibration in general provides us with information on the orientations of the cameras with respect to the rectified image plane $\boldsymbol{R}_k$ and their baseline b [?] [?, p. 523f], therefore

$$\boldsymbol{C}_{SC_k} = \boldsymbol{R}_k \quad \text{and} \quad {}_S\boldsymbol{r}_{C_1 C_2} = b \ . \tag{1.11}$$

are known. The distortion coefficients D_k are zero in the rectified image, but the normalized points need to be rotated to lie in the same rectified image plane. The camera projection 1.9 for the rectified case therefore becomes

$$\tilde{\boldsymbol{w}}_{k,j} = \boldsymbol{K}'_k \boldsymbol{R}_k (\boldsymbol{\pi}({}_{C_k}\boldsymbol{r}_{C_k X_j})),$$

where $\boldsymbol{K}'_k$ is a modified camera matrix, which can be used to scale the image to avoid non-defined image borders resulting from image undistortion and rectification.

For a given set of rectified and unrectified images of a well calibrated camera, there are strong correspondances between the pixels obtaiend from projections of 3D points to the rectified and the unrectified images, respectively. This property was exploited to test the correctness of the above transformations and the provided camera paramteres, as shown in Appendix **??**.

1.2.3 Interpolation

The image has to be interpolated around the obtained floating-precision pixel locations using any common interpolation kernel $h(i, j)$ as such:

$$I(x, y) = \sum_{i=1}^{S} \sum_{j=1}^{S} I(x_i, y_j) h(i, j) \tag{1.12}$$

$$= \sum_{i=1}^{S} \sum_{j=1}^{S} I(x_i, y_j) h(i) h(j) \ . \tag{1.13}$$

The indices x_i and y_i correspond to the location in the image where the interpolation function $h(i,j)$ takes its maximum value.

In this project, the Keys cubic convolution interpolation kernel [?], was applied, defined by

$$h(x) = \begin{cases} 0.5(x-2)^3 + 2.5(x-2)^2 + 4(x-2) + 2 & 0 \leq x < 1 \\ -1.5(x-2)^3 - 2.5(x-2)^2 + 1 & 1 \leq x < 2 \\ 1.5(x-2)^3 - 2.5(x-2)^2 + 1 & 2 \leq x < 3 \\ -0.5(x-2)^3 + 2.5(x-2)^2 - 4(x-2) + 2.0 & 3 \leq x < 4 \end{cases} \tag{1.14}$$

Other choices of interpolation schemes, using bilinear or quadratic interpolation kernels, could have been considered but no further investigation was done since the interpolation kernel did not show any significant effect on the results in terms of accuracy or converge in terms of accuracy or convergence.

1.2.4 Disparity Images

In order to quantify the error between the obtained 3D map and the reference map, disparity images of both maps were created and their difference was evaluated. This approach is preferred over other approaches such as the sum over all height differences, since at sharp edges a little lateral offset can induce very high differences, but the disparity map's error stays relatively constrained because of the limiting factor of the pixel values. Furthermore, even before quantitative analysis, the disparity map offers an intermediate visualization that gives a very intuitive first impression of the accuracy of the obtained point cloud. Finally, the disparity being simply defined as $d = bf_x/z$, with z the depth of the point in the camera frame $\mathcal{C}$, it can be very easily calculated, using Algorithm **??**).

Algorithm 1: Create a disparity map from 3D coordinates

Input : Reduced camera projection Proj and map coordinates $\boldsymbol{x}$
Output: Disparity map
$Disparities[1\dots h, 1\dots w]$;
$Counter[1\dots h, 1\dots w]$;
for $i = 1$ *to* N **do**
  $[u,v,z] = \mathrm{Proj}(\boldsymbol{x}(i))$;
  $d = bf_x/z$;
  $Disparities[v,u] += d$;
  $Counter[v,u]++$;
end
return $Disparities/\ Counter$;

It can not be assured that every pixel in the obtained disparity map has been assigned a disparity value after running Algorithm 1. By reducing the resolution of the disparity image by a sufficiently high factor with respect to the original images, however, the number of non-defined pixels can be limited considerably. In the present case, a good choice of this factor was shown to be 2-4 and the few non-defined pixels left were assigned the average value of the surrounding non-zero pixels, leading to smooth disparity images.

1.3 B-Splines for surface representation

Splines are piecewise polynomial functions of a degree $\leq D$ whose parameters are allowed to change at so-called knot points, allowing to impose certain behaviour suh as smoothness and derivability. B-Splines got their name from the fact that they are commonly used as (B)asis functions for representing more complex behaviours or geometries. In this project, for instance, they can be used to represent realistic and relatively smooth surfaces with adjustable amount detail using only a limited number of parameters. B-Splines can be obtained using the *Cox de Boor* recursion formula. They are continuous at the knots and their deriavtives are also continuous up to $D-1$.

1.3.1 Interpolation

The naming and indexing conventions for the B-splines formulations are inspired by [**?**]. Let us denote by (x_i, y_i) the knot point "responsible" for the point at which the spline is evaluated, $(x_{i,k}, y_{i,k})$. and let $(\delta_x(k), \delta_y(k))$ be the offset of the evaluation point with respect to the knot point (i.e. $\delta_x(k) = x_{i,k} - x_i$, idem for y). The height at the point (x_j, y_j) for a B-spline representation of support S, is then given by

$$h(x_j, y_j) = h(x_{i,k}, y_{i,k}) = \sum_{\mu=1}^{S} \sum_{\nu=1}^{S} a_{i+\mu, j+\nu} \gamma_{k,\mu,\nu} \ , \tag{1.15}$$

where $\gamma_{k,\mu,\nu}$ is the pre-evaluated tensor product basis, corresponding to the value of the spline function at (μ, ν), calculated using

$$\gamma_{k,\mu,\nu} = h_D(\mu + \delta_x(k)) h_D(\nu + \delta_y(k)) \ . \tag{1.16}$$

where $h_D(x)$ is the interpolation function of the spline of degree D, for example for cubic splines

$$h_3(x) = \begin{cases} 1/6x^3 & 0 \leq x < 1 \\ -0.5x^3 + 2x^2 - 2x + \ 2/3 & 1 \leq x < 2 \\ 0.5x^3 - 4x^2 + 10x - 22/3 & 2 \leq x < 3 \\ -1/6x^3 + 2x^2 - 8x + 32/3 & 3 \leq x < 4 \end{cases} \tag{1.17}$$

The coordinates of grid point j are then given by

$$ {}_{M}\boldsymbol{r}_{MX_j} = \left[x_j, y_j, h(x_j, y_j)(\boldsymbol{a})\right]^T \ , $$

which can be repeated for all points ($j = 1 \dots N$) to yield the map coordinates.

1.3.2 Bending energy

$$ B(h) = \iint_{\mathbb{R}^2} \left(\frac{\partial^2 h}{\partial x^2}(x, y)\right)^2 + \left(\frac{\partial^2 h}{\partial x \partial y}(x, y)\right)^2 + \left(\frac{\partial^2 h}{\partial y^2}(x, y)\right)^2 \mathrm{d}x\mathrm{d}y \tag{1.18} $$

1.3.3 Gradient energy

$$ G(h) = \iint_{\mathbb{R}^2} \left(\frac{\partial h}{\partial x}(x, y)\right)^2 + \left(\frac{\partial h}{\partial y}(x, y)\right)^2 \mathrm{d}x\mathrm{d}y \tag{1.19} $$