Project Report: Direct Visual Odometry for RGB-D Images

BOQIAN YU, 03708925

Due to its better usage of information provided in the input images, direct methods in the field of visual odometry have been receiving increasing focus in the past years. This report summarizes the practical project based on the robust direct visual odometry algorithm for RGB-D images, including its mathematical derivation and the manual implementation. Results based on the TUM-dataset are compared with those provided in the original paper to measure the performance of the implementation.

1 INTRODUCTION

Simultaneous localization and mapping (SLAM) has wide applications in many fields, including autonomous driving[12], geological survey [10], underground construction [16] [17] etc. One of the most significant techniques of SLAM is the estimation and maintaining of the knowledge of robot poses and motions through real-time data typically either as point clouds (from laser scanners) or as image frames (from cameras), of which the latter is refered to as visual odometry and has been among the most popular topics in the past years.

Visual odometry algorithms can be categorized into two classes:

- **Feature-based methods.** The idea of feature-based methods is to estimate the relative motion between two frames by only registering some feature observations extracted from the images. These features are usually computed using descriptors such as SIFT [13], SURF [1], ORB [18], BRIEF [3] etc. with various criteria. However, feature-based methods discard information in the images that are not considered as "significant" by the chosen descriptor, although such information can also contribute to the motion estimation.
- **Direct methods.** Direct methods, on the other hand, overcome the disadvantage mentioned above by optimizing the motion estimation directly on the image intensities and thus enabling the use of all information contained in the images. In this way, higher accuracy can be achieved within less time (not necessary to select and match feature descriptors). The idea of direct methods for visual odometry have been widely expanded for different applications, such as in [19], [20], [15], [6], [5] etc.

This report discusses the theory and implementation of the robust direct visual odometry algorithm proposed in [9] and is formulated as follow: section 2 will discuss the detailed mathematical derivation of the algorithm, including the warping function, the optimization goal and methods and means to enhance robustness; section 3 will discuss the manual implementation that is completed in this project; section 4 will then show some results of this implementation based on the TUM-dataset and compare its performance with the reference results given in the original paper.

2 MATHEMATICAL DERIVATION

The basic assumption of the robust DVO algorithm is that the same world point $\mathbf{p}$ observed in a successive pair of RGB-D frames $(\mathcal{I}_1, \mathcal{Z}_1)$ and $(\mathcal{I}_2, \mathcal{Z}_2)$ should yield identical intensity, i.e.

$$\mathcal{I}_1(\mathbf{x}) = \mathcal{I}_2(\tau(\boldsymbol{\xi}, \mathbf{x}))$$

where $\tau(\boldsymbol{\xi}, \mathbf{x})$ is the warping function that enables the pixel coordinate transformation between these two camera frames under a certain camera motion $\boldsymbol{\xi}$ ($\boldsymbol{\xi} \in \mathbb{R}^6$ in form of twist coordinates). The residual of a pixel $\mathbf{x}_i$ is defined as:

$$r_i(\boldsymbol{\xi}) := \mathcal{I}_2\left(\tau\left(\boldsymbol{\xi}, \mathbf{x}_i\right)\right) - \mathcal{I}_1\left(\mathbf{x}_i\right) \tag{1}$$

By computing the MAP estimation of the overall residuals and minimizing it w.r.t $\boldsymbol{\xi}$, the relative motion estimation problem is then transformed into a weighted least square problem and can be solved by standard methods.

Beyond this, a motion prior based on the camera motion of the previous pair and a weight function based on the observed distribution of residuals are introduced to enhance the robustness of the algorithm against noises and outliers; a image pyramid is also suggested in the paper to support the validity of local linearization in the optimization steps.

2.1 Warping Function

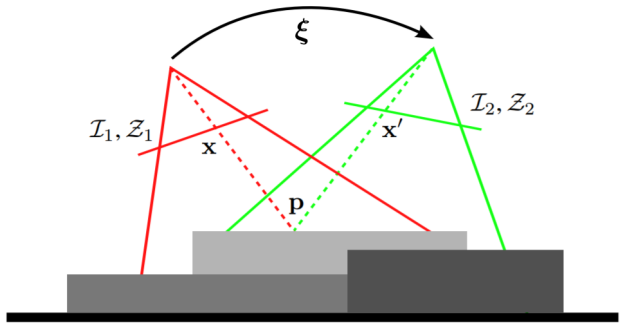


Fig. 1. The warping function $\tau(\boldsymbol{\xi}, \mathbf{x})$ enables coordinate transformation $\mathbf{x}' = \tau(\boldsymbol{\xi}, \mathbf{x})$ between two frames under a certain relative motion $\boldsymbol{\xi}$ [9].

Consider a pinhole camera model with focal lengths f_x, f_y and optical center (c_x, c_y), the unprojection of a pixel $\mathbf{x} = (u, v)^\top$ in the frame of the first camera to its corresponding point $\mathbf{p}$ in the same frame can be written as:

$$\mathbf{p} = \pi^{-1}\left(\mathbf{x}, \mathcal{Z}_1(\mathbf{x})\right) = \mathcal{Z}_1(\mathbf{x}) \left(\frac{u + c_x}{f_x}, \frac{v + c_y}{f_y}, 1\right)^\top \tag{2}$$

where $\mathcal{Z}_1(\mathbf{x})$ refers to the depth value of this pixel in the first frame.

In order to transpose $\mathbf{p}$ into the frame of the second camera, a rigid body motion $g(\boldsymbol{\xi}) \in \mathrm{SE}(3)$ is applied to $\mathbf{p}$, which includes a rotation represented as a 3×3 orthogonal matrix $R \in \mathrm{SO}(3)$ and a translation represented as a 3×1 vector $\mathbf{t} \in \mathbb{R}^3$ and can be obtained from the twist $\boldsymbol{\xi}$ with $g(\boldsymbol{\xi}) = \exp(\hat{\boldsymbol{\xi}})$.

Apply the rigid body motion:

$$T(g(\boldsymbol{\xi}), \mathbf{p}) = R\mathbf{p} + \mathbf{t} \tag{3}$$

and then project the transformed point $(X, Y, Z)^\top = T(g(\boldsymbol{\xi}), \mathbf{p})$ into the second frame:

$$\pi(T(g, \mathbf{p})) = \left(\frac{f_x X}{Z} - c_x, \frac{f_y Y}{Z} - c_y \right)^\top \tag{4}$$

By summing up (2) (3) (4), the warping function between the two frames can be written as:

$$\tau(\boldsymbol{\xi}, \mathbf{x}) = \pi(T(g(\boldsymbol{\xi}), \mathbf{p})) = \pi\left(T\left(g(\boldsymbol{\xi}), \pi^{-1}\left(\mathbf{x}, \mathcal{Z}_1(\mathbf{x})\right)\right)\right) \tag{5}$$

Although the procedures above are discussed merely with a pinhole camera model, they can actually be adapted to any camera model as long as the model complies with the intrinsic parameters specified by the dataset.

2.2 MAP Estimation

From the residual of each pixel defined in (1), a residual image between two frames can be formed by illustrating the residual of a pixel as brightness value, as shown in the example in Fig. 2c.

(a) (b) (c)

Fig. 2. Two frames and the corresponding residual image. (a) RGB image of frame 1 (b) RGB image of frame 2 (c) residual image between the two frames

By assuming the independence and identical distribution of all pixels and stacking the residuals as $\mathbf{r} = (r_1, ..., r_n)^\top$, the overall likelihood can then be written as:

$$p(\mathbf{r}|\boldsymbol{\xi}) = \prod_i p\left(r_i|\boldsymbol{\xi}\right)$$

According to Bayes' law, the a posteriori likelihood of a relative camera motion $\boldsymbol{\xi}$ given the residual image can be formulated as:

$$p(\boldsymbol{\xi}|\mathbf{r}) = \frac{p(\mathbf{r}|\boldsymbol{\xi})p(\boldsymbol{\xi})}{p(\mathbf{r})} \tag{6}$$

where $p(\boldsymbol{\xi})$ denotes the prior distribution over camera motion and $p(\mathbf{r})$ the distribution of residuals. These two parameters enable the integration of the prior knowledge of motion and residual distribution into the solution to achieve faster convergence and better robustness [8], which will be discussed in later sections.

The item $p(\mathbf{r})$ can be neglected as it is not dependent on $\boldsymbol{\xi}$. The desired camera motion between the two frames can then be formulated as:

$$\begin{aligned} \boldsymbol{\xi}_{\text{MAP}} &= \arg\max_{\boldsymbol{\xi}} p(\mathbf{r}|\boldsymbol{\xi})p(\boldsymbol{\xi}) \\ &= \arg\max_{\boldsymbol{\xi}} \prod_i p\left(r_i|\boldsymbol{\xi}\right) \cdot p(\boldsymbol{\xi}) \\ &= \arg\min_{\boldsymbol{\xi}} \sum_i -\log p\left(r_i|\boldsymbol{\xi}\right) - \log p(\boldsymbol{\xi}) \end{aligned} \tag{7}$$

If the camera motion is assumed to be uniformly distributed, (7) can be further simplified as:

$$\boldsymbol{\xi}_{\text{MAP}} = \arg\min_{\boldsymbol{\xi}} \sum_i^n -\log p\left(r_i|\boldsymbol{\xi}\right) \tag{8}$$

which can be solved by setting the derivative of the negative log-likelihood w.r.t. $\boldsymbol{\xi}$ to zero:

$$\sum_i \frac{\partial \log p\left(r_i|\boldsymbol{\xi}\right)}{\partial \boldsymbol{\xi}} = \sum_i \frac{\partial \log p\left(r_i\right)}{\partial r_i} \frac{\partial r_i}{\partial \boldsymbol{\xi}} = 0 \tag{9}$$

By introducing a new symbol $w\left(r_i\right) = \frac{\log p(r_i)}{\partial r_i} \frac{1}{r_i}$, (8) and (9) yields:

$$\boldsymbol{\xi}_{\text{MAP}} = \arg\min_{\boldsymbol{\xi}} \sum_i^n \omega(r_i)(r_i(\boldsymbol{\xi}))^2 \tag{10}$$

(10) formulates the problem of solving the MAP estimation of the optimal camera motion for a pair of frames as a weighted least square problem, where $\omega(r_i)$ is the weighting function that decides how much a certain residual will contribute to the overall loss. Furthermore, if the residuals are normally distributed, i.e. $p\left(r_i\right) = \mathcal{N}\left(r_i, 0, \sigma^2\right)$, then all $\omega(r_i)$ are identical (and equal to $-1/\sigma^2$), leading to a regular least square problem.

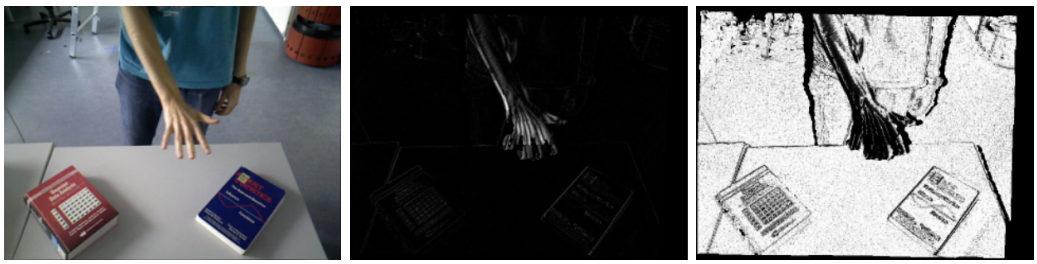
The above derivation assumes a uniform distribution for the camera motion, i.e. all possible sets of motion parameters are equally likely. By otherwise assuming a normal distribution for the camera motion, i.e. the motion prior $p(\boldsymbol{\xi}) = \mathcal{N}\left(\boldsymbol{\xi}, \boldsymbol{\xi}_0, \Sigma_{\text{prior}}\right)$, a solution similar to (10) can be derived from (7):

$$\boldsymbol{\xi}_{\mathrm{MAP}} = \arg\min_{\boldsymbol{\xi}} \sum_{i}^{n} \omega(r_i)(r_i(\boldsymbol{\xi}))^2 + \Sigma_{\mathrm{prior}}^{-1} (\boldsymbol{\xi} - \boldsymbol{\xi}_0)^2 \tag{11}$$

Both (10) and (11) can be solved using standard methods for least square problems, as will be discussed in section 2.4.

2.3 Residual Weighting

As stated in section 2.2, the assumption of the normal distribution of residuals can be taken to reduce (10) to a regular least square problem. This formulation, however, results in one drawback: very large residuals will have extremely strong influence on the overall loss due to the quadratic term, and thus reduce the robustness of the algorithm. These large residuals are usually caused by outliers violating the constant-intensity assumption or by dynamic environment disturbance (e.g. Fig. 3a, 3b). To enhance robustness, several robust error functions and underlying distribution models of the residuals are investigated in the paper to adjust the influence of large residuals and reduce the impact of outliers pixels.



(a) (b) (c)

Fig. 3. Example of residual weighting: (a) a scene with a hand (outlier pixels) moving through; (b) the residuals calculated from (1), outlier pixels have much larger residuals than other pixels; (c) the weights of the pixels, which greatly reduced the influence of the outlier pixels. [9]

The Tukey function proposed in [22] is among the earliest error-weighting functions used for robust estimation [23]. It is computed using the median absolute deviation of the dataset, and is defined as follows:

$$\omega_{\mathrm{Tukey}}(x) = \begin{cases} \left(1 - \frac{x^2}{b^2}\right)^2 & |x| \le b \\ 0 & \text{else} \end{cases} \tag{12}$$

and for each residual the weight function is computed as:

$$w_i = \omega_{\mathrm{Tukey}} \left(\frac{r_i}{\sigma_{\mathrm{r}}} \right) \tag{13}$$

where σ_r is the median absolute deviation of r, and the parameter b should be selected in accordance with the underlying outlier-free distribution of the data (e.g.

$b = 4.6851$ for 95% asymptotic efficiency on standard normal distribution [24]). The Tukey function proves to be highly robust w.r.t both global and local robustness criteria [4].

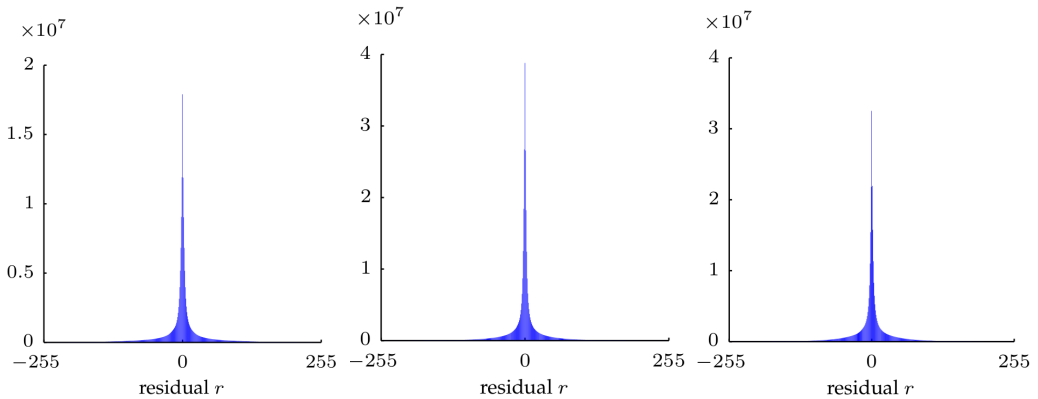
Apart from (12), the Huber function proposed in [7] has also been widely used for robustification of noisy dataset. It suppresses the influence of large residuals by adjusting their terms to linear instead of quadratic:

$$\omega_{\text{Huber}}(x) = \begin{cases} 1 & |x| \leq k \\ \frac{k}{|x|} & \text{else} \end{cases} \tag{14}$$

where the parameter k should again be selected according to the underlying distribution of the data (e.g. $k = 1.345$ for 95% asymptotic efficiency on standard normal distribution [24]). The effectiveness of Huber function in suppressing very large residuals is worse than Tukey function, but one advantage of Huber function is its convexity, which keeps the local minima of the original function without bringing about new interference for optimization.

Despite the theoretical effect of the two robust error functions, it is stated in the paper that the performance is not satisfying enough in the preliminary experiments. So an explicit modeling of the underlying distribution of the residuals is carried out to further enhance the robustness of the algorithm.

The distribution of all frame pairs are investigated on several datasets from the TUM-Dataset [21]. The accumulated histograms of the residuals of all successive frame pairs for some datasets are shown in Fig. 4. As can be seen from the histograms and Fig. 5, a normal distribution or one with robust error function can not properly reflect the underlying distribution of residuals, leading to inappropriate weighting functions. On the other hand, a t-distribution [11] fits the observed histograms nicely, as can be seen in Fig. 5.



(a) fr1/desk (b) fr3/sitting_halfsphere (c) fr3/walking_halfsphere

Fig. 4. Accumulated histogram of the residuals of all successive frame pairs from different datasets [8] (a) fr1/desk (b) fr3/sitting_halfsphere (c) fr3/walking_halfsphere

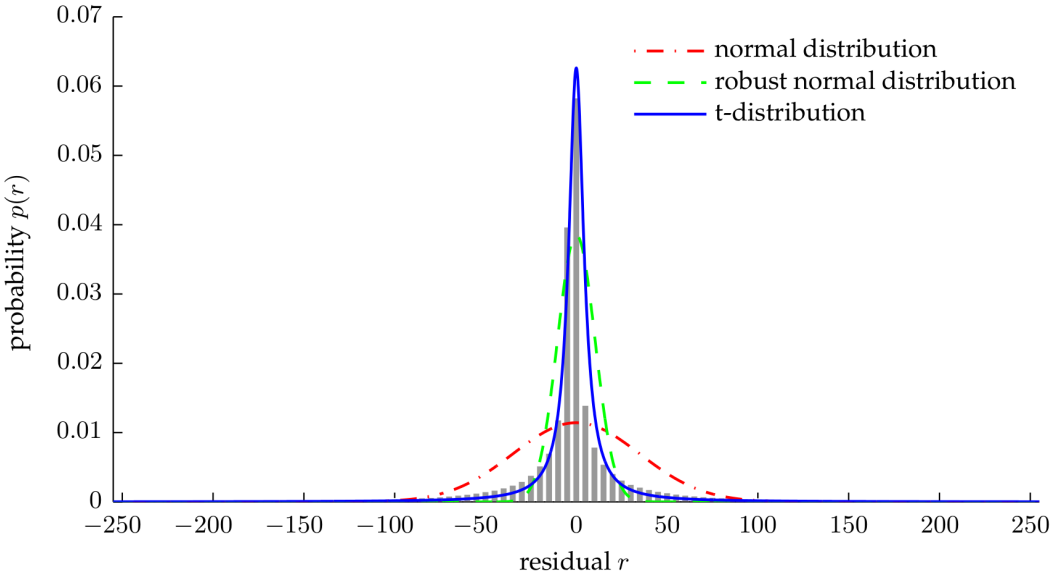


Fig. 5. Modeling the distribution of residuals with various kind of models. [8]

A t-distribution is defined as:

$$p(x|\mu,\sigma,\nu) = \frac{\Gamma\left(\frac{v+1}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)\sqrt{\pi\nu\sigma^2}}\left(1+\frac{1}{\nu}\frac{(x-\mu)^2}{\sigma^2}\right)^{-\frac{v+1}{2}} \tag{15}$$

where μ is the mean, σ^2 the variance, ν the degree of freedom of the data and $\Gamma(x) = (x-1)!$ the gamma function. A t-distribution can be interpreted as a Gauss Mixture Model with infinite many centroids [2], which is more flexible compared to the i.i.d. assumption in section 2.2. Each residual is weighted with its inverse variance considering (9) and (10), and large residuals (outliers) are generated by centroids with high variances. These residuals will accordingly get low weights and thus suppressed.

The weight function $\omega(r_i)$ can be calculated from (9) and (15) as:

$$\omega_{\mathrm{t}}(r_i) = \frac{\nu+1}{\nu+\left(\frac{r_i}{\sigma_{\mathrm{t}}}\right)^2} \tag{16}$$

where the mean is set as $\mu = 0$ and the variance σ_{t}^2 has to be iteratively solved using maximum likelihood estimation based on the current residuals:

$$\sigma^2_{\mathrm{t},k+1} = \frac{1}{n}\sum_{i}^{n}\frac{\nu+1}{\nu+r_i^2/\sigma^2_{\mathrm{t},k}}r_i^2 \tag{17}$$

which will converge in few iterations.

An example is shown in Fig. 3c, illustrating the weights of the residuals computed from the t-distribution. As can be discovered in the figure, the large residuals caused by the moving hand are weighted extremely low, which corresponds to an outlier-suppression behavior.

2.4 Optimization Methods

Conventionally, a (weighted) least square optimization problem can be solved by many standard methods such as gradient descent, Newton method, Gauss-Newton method, Levenberg-Marquardt method etc. Considering the convergence rate and the computational complexity, only Gauss-Newton method and Levenberg-Marquardt method will be discussed in this report.

The warping function (5) implies that the residual is not linear w.r.t camera motion $\boldsymbol{\xi}$, making (10) a non-linear optimization problem. Thus, a local linearization around $\boldsymbol{\xi} = \mathbf{0}$ using Taylor expansion is fisrt applied to enable the use of standard methods:

$$\begin{aligned} r_{\text{lin}}\left(\boldsymbol{\xi}, \mathbf{x}_i\right) &= r\left(\mathbf{0}, \mathbf{x}_i\right) + \left.\frac{\partial r\left(\tau\left(\boldsymbol{\xi}, \mathbf{x}_i\right)\right)}{\partial \boldsymbol{\xi}}\right|_{\boldsymbol{\xi}=\mathbf{0}} \Delta\boldsymbol{\xi} \\ &= r\left(\mathbf{0}, \mathbf{x}_i\right) + J_i \Delta\boldsymbol{\xi} \end{aligned} \tag{18}$$

where $J_i \in \mathbb{R}^{1\times 6}$ is the Jacobian of the i-th pixel w.r.t. the 6-DOF camera motion in twist coordinate evaluated at $\boldsymbol{\xi} = \mathbf{0}$. Combining (1) and (5) and applying the chain rule, the Jacobian can be calculated as:

$$\begin{aligned} J_i = \left.\frac{\partial r\left(\tau\left(\boldsymbol{\xi}, \mathbf{x}_i\right)\right)}{\partial \boldsymbol{\xi}}\right|_{\boldsymbol{\xi}=\mathbf{0}} = \left.\frac{\partial I_2(\mathbf{y})}{\partial \mathbf{y}}\right|_{\mathbf{y}=\pi(\mathbf{p})} \cdot \left.\frac{\partial \pi(\mathbf{p})}{\partial \mathbf{p}}\right|_{\mathbf{p}=T(g(\mathbf{0}),\pi^{-1}(\mathbf{x}_i,Z_1(\mathbf{x}_i)))} \\ \cdot \left.\frac{\partial T\left(g(\boldsymbol{\xi}), \pi^{-1}\left(\mathbf{x}_i, Z_1\left(\mathbf{x}_i\right)\right)\right)}{\partial \boldsymbol{\xi}}\right|_{\boldsymbol{\xi}=\mathbf{0}} \end{aligned} \tag{19}$$

which further yields

$$J_i = \begin{pmatrix} \nabla I_{2,x} & \nabla I_{2,y} \end{pmatrix} \begin{pmatrix} f_x \frac{1}{z} & 0 & -f_x \frac{x}{z^2} & -f_x \frac{x\cdot y}{z^2} & f_x\left(1+\frac{x^2}{z^2}\right) & -f_x \frac{y}{z} \\ 0 & f_y \frac{1}{z} & -f_y \frac{y}{z^2} & -f_y\left(1+\frac{y^2}{z^2}\right) & f_y \frac{x\cdot y}{z^2} & f_y \frac{x}{z} \end{pmatrix} \tag{20}$$

Note that (19) and (20) are only valid for pinhole camera model due to the projection and unprojection function π and π^{-1} used for computing derivatives.

As can be observed in (10) and (11), the weights only affect the scales of the corresponding residuals, and therefore can be incorporated into a diagonal matrix W with $w_{ii} = \omega(r_i)$. Plugging (18) into (9) and formulating in matrix notation yield:

$$J^\top W J \Delta\boldsymbol{\xi} = -J^\top W \mathbf{r}(\mathbf{0}) \tag{21}$$

where $J \in \mathbb{R}^{n\times 6}$ is the stacked matrix of all J_i. In the k-th iteration, an increment $\Delta\boldsymbol{\xi}$ is solved and integrated into the current camera motion with $\boldsymbol{\xi}_{k+1} = \log\left(\exp(\Delta\boldsymbol{\xi}) \cdot \exp(\boldsymbol{\xi}_k)\right)$.

Similarly, the equation for motion increment using Levenberg-Marquardt method can be derived:

$$\left(J^\top W J + \lambda \operatorname{diag}\left(J^\top W J\right)\right) \Delta\boldsymbol{\xi} = -J^\top W \mathbf{r}(\mathbf{0}) \tag{22}$$

When the motion prior described in (11) is also taken into account, (21) becomes:

$$\left(J^\top W J + \Sigma^{-1}\right) \Delta\boldsymbol{\xi} = -J^\top W \mathbf{r}(0) + \Sigma^{-1}\left(\boldsymbol{\xi}_0 - \boldsymbol{\xi}_k\right) \tag{23}$$

where the mean $\boldsymbol{\xi}_0$ can be set as the camera motion calculated from the last pair of frames by assuming the smoothness of motion, and the covariance matrix Σ_{prior} need to be tuned.

2.5 Image Pyramid

The equations of estimating the camera motion iteratively given in the previous section are actually not valid when the camera motion is rather large, because the linearization is carried out around $\boldsymbol{\xi} = \mathbf{0}$. Therefore, a coarse-to-fine scheme is adopted to support the estimation of large $\boldsymbol{\xi}$: an image pyramid is built with image resolutions being halved at each layer (Fig. 6). The motion estimation is first executed on the top layer with lowest resolution, and the estimation is iteratively propagated downwards, serving as initialization for the next layer. In this way, even large camera motions can be properly estimated.

Fig. 6. Image pyramid built to enable larger camera motion with valid linearization. In each layer, the image resolution is halved and the corresponding intrinsic parameters are modified.

While halving the image resolution, the corresponding intrinsic parameters should also be properly adapted. This can be achieved by comparing the pixel coordinates between two neighboring layers k and $k+1$:

$$2\mathbf{x}^{(k+1)} + \frac{1}{2} = \mathbf{x}^{(k)} \tag{24}$$

plugging into the projection equation $\mathbf{x} = \frac{1}{Z}K\mathbf{X}$ yields:

$$f_x^{(l+1)} = \frac{1}{2}f_x^{(l)}, \quad f_y^{(l+1)} = \frac{1}{2}f_y^{(l)}, \quad c_x^{(l+1)} = \frac{1}{2}c_x^{(l)} - \frac{1}{4}, \quad c_y^{(l+1)} = \frac{1}{2}c_y^{(l)} - \frac{1}{4} \tag{25}$$

Note that the $(0, 0)$ coordinate is define at the center of the top-left pixel.

3 IMPLEMENTATION

The implementation of the described algorithm can be inspected in the gitlab repository `RGBD_DVO` under https://gitlab.vision.in.tum.de/s0010/rgbd_dvo and mainly contains two classes:

- The `DirectOdometry` class implements the scheme of the algorithm discussed in section 2. The complete workflow is described in the pseudo-code 1:

Algorithm 1: Direct Visual Odometry

Require: RGB-D frame stream
Ensure: Camera motion of all successive frame pairs

```
for all succesive frame pairs do
    for layer in image pyramid (top-down) do
        while not converged do
            Compute residual image r(ξ_k)
            Compute weight matrix W from (16)
            Compute Jacobians J from (20)
            Compute motion increment Δξ from (21)
            Update camera motion ξ_{k+1} = log(exp(Δξ) · exp(ξ_k))
        end while
    end for
end for
```

- The `Frame` class provides support for image frames and pyramids, including the storage of a image frame (RGB image and depth image) and the construction of image pyramid discussed in section 2.5. As frame pairs are taken successively, the image pyramid is calculated only once for each frame.

For the optimization step, both methods discussed in section 2.4 are implemented in the `DirectOdometry` class, and the Gauss-Newton method is used for testing, as it requires no fine-tuning of hyper parameter (comparing to the λ parameter in Levenberg-Marquardt method).

The user interface shows both the current RGB image and the final residual image of the current frame pair. The trajectory generated from odometry and the ground-truth are shown in different color in real time below the images (Fig. 7). After the whole process finishes, an alignment between the true and the estimated trajectories is executed to reduce the influence of possible local errors on the global trajectory.

4 PERFORMANCE MEASURE AND POSSIBLE IMPROVEMENT

The implementation of the algorithm described above is tested on the TUM-Dataset [21]. The absolute trajectory error (ATE) is used to measure the performance of odometry and to compare with the reference test results given in [9]. The comparison of results are given in Table 1.

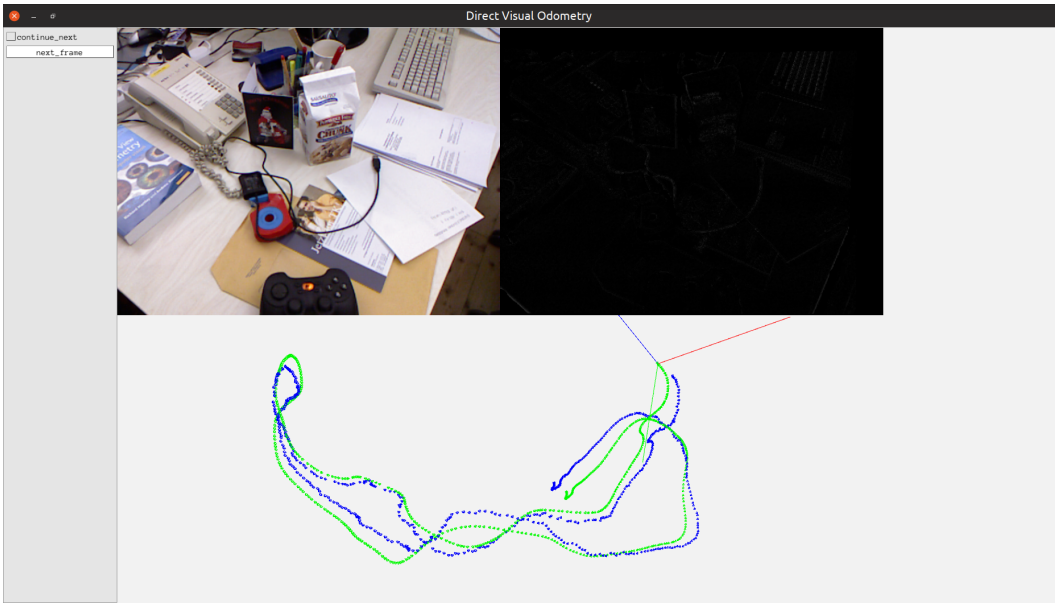


Fig. 7. The user interface and one sample result of visual odometry from the manual implementation in the project running on the TUM-Dataset fr1/desk. The estimated trajectory is illustrated in blue, and the ground-truth in green.

Table 1. Comparison of results from manual implementation and original paper

Datasets	Manual implementation	Results given in [9]
fr1/desk	0.0957	0.0687
fr1/desk2	0.0725	0.0491
fr2/desk	0.0312	0.0188
fr2/person	0.0497	0.0345

As can be see from the table, the trajectory estimation given by the manual implementation is not as good as the results in the paper, but the difference also lies in an acceptable range. By fine-tuning some hyper parameters (depth of image pyramid, convergence condition of Gauss-Newton method etc.) and applying the following improvements, the performance can still be further enhanced:

- Using Levenberg-Marquardt method instead of Gauss-Newton method. Although implemented in this project, Levenberg-Marquardt method is not used in the performance tests due to the parameter λ that requires tuning. Levenberg-Marquardt method combines the advantages of gradient descent and Gauss-Newton method, and can usually yield faster convergence and better robustness [14].
- Adjusting covariance matrix of motion prior. In the manual implementation, the covariance matrix of the motion prior is always set to identity, indicating that the motion prior is identical to the estimation from the last frame pair. Setting the covariance matrix to appropriate values can smoothen the generated trajectory and result in higher performance.
- Taking keyframes. The smoothness of the path can also be improved by using keyframing instead of simply estimating camera motion between neighboring frames. The camera motion will then be estimated between the current frame and its closest keyframe. The criteria of taking new keyframes should also be properly selected to avoid loss of image information during projection.

REFERENCES

[1] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. 2006. Surf: Speeded up robust features. In *European conference on computer vision*. Springer, 404–417.

[2] Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning*. Springer.

[3] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. 2010. Brief: Binary robust independent elementary features. In *European conference on computer vision*. Springer, 778–792.

[4] Zhiqiang Chen and David E. Tyler. 2002. The Influence Function and Maximum Bias of Tukey's Median. *Annals of Statistics* 30, 6 (2002), 1737–1759.

[5] Jakob Engel, Vladlen Koltun, and Daniel Cremers. 2017. Direct sparse odometry. *IEEE transactions on pattern analysis and machine intelligence* 40, 3 (2017), 611–625.

[6] Jakob Engel, Thomas Schöps, and Daniel Cremers. 2014. LSD-SLAM: Large-scale direct monocular SLAM. In *European conference on computer vision*. Springer, 834–849.

[7] Peter J. Huber. 2004. *Robust Statistics*. John Wiley & Sons.

[8] Christian Kerl. 2012. *Odometry from rgb-d cameras for autonomous quadrocopters*. Master's thesis. Technical University of Munich.

[9] Christian Kerl, Jürgen Sturm, and Daniel Cremers. 2013. Robust odometry estimation for RGB-D cameras. In *2013 IEEE International Conference on Robotics and Automation*. IEEE, 3748–3754.

[10] Karl Kraus and Norbert Pfeifer. 1998. Determination of terrain models in wooded areas with airborne laser scanner data. *ISPRS Journal of Photogrammetry and remote Sensing* 53, 4 (1998), 193–203.

[11] Kenneth L. Lange, Roderick J. A. Little, and Jeremy M. G. Taylor. 1989. Robust Statistical Modeling Using the t Distribution. *Publications of the American Statistical Association* 84, 408 (1989), 881–896.

[12] Jesse Sol Levinson. 2011. *Automatic laser calibration, mapping, and localization for autonomous vehicles*. Stanford University.

[13] David G Lowe et al. 1999. Object recognition from local scale-invariant features.. In *iccv*, Vol. 99. 1150–1157.

[14] Kaj Madson, Hans B. Nielsen, and Ole Tingleff. 2004. *Methods for Non-Linear Least Squares Problems (2nd ed.)*.

[15] Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison. 2011. DTAM: Dense tracking and mapping in real-time. In *2011 international conference on computer vision*. IEEE, 2320–2327.

[16] Andress Nüchter, Hartmut Surmann, Kai Lingemann, Joachim Hertzberg, and Sebastian Thrun. 2004. 6D SLAM with an application in autonomous mine mapping. In *Proceedings 2011 IEEE International Conference on Robotics and Automation*, Vol. 2. IEEE, 1998–2003.

[17] Marko Pejić. 2013. Design and optimisation of laser scanning for tunnels geometry inspection. *Tunnelling and underground space technology* 37 (2013), 199–206.

[18] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary R Bradski. 2011. ORB: An efficient alternative to SIFT or SURF.. In *ICCV*, Vol. 11. Citeseer, 2.

[19] Frank Steinbrücker, Jürgen Sturm, and Daniel Cremers. 2011. Real-time visual odometry from dense RGB-D images. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*. IEEE, 719–722.

[20] Jan Stühmer, Stefan Gumhold, and Daniel Cremers. 2010. Real-time dense geometry from a handheld camera. In *Joint Pattern Recognition Symposium*. Springer, 11–20.

[21] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. 2012. A Benchmark for the Evaluation of RGB-D SLAM Systems. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*.

[22] John W Tukey. 1975. Mathematics and the picturing of data. *Proc.inter.cong.math.vancouver* (1975), 523–531.

[23] Tommi Tykkala, Cédric Audras, and Andrew I. Comport. 2011. Direct Iterative Closest Point for real-time visual odometry.. In *ICCV Workshops*. IEEE, 2050–2056.

[24] Zhengyou Zhang. 1997. Parameter estimation techniques: a tutorial with application to conic fitting. *Water Research* 39, 15 (1997), 3686–3696.