Supplementary Material for
MBA-VO: Motion Blur Aware Visual Odometry

Peidong Liu[1] Xingxing Zuo[1] Viktor Larsson[1] Marc Pollefeys[1,2]
[1]Department of Computer Science, ETH Zürich
[2]Microsoft Mixed Reality and AI Lab, Zürich

1. Introduction

In this supplementary material, we present the details on the Jacobian derivations necessary for the implementation of the direct image alignment algorithm with blurry images. We also include the sample videos of the proposed motion blur robust VO benchmarking dataset. Additional discussion on the computational complexity as well as experimental results are also presented.

2. Notations

For the ease of illustration, we define following notations. We denote scalar with lower case letter (e.g. λ); we denote vector with bold lower case letter (e.g. $\mathbf{x}$); we denote matrix with bold upper case letter (e.g. $\mathbf{T}$); we denote a point $\mathbf{p}$ in coordinate frame $\mathcal{F}_a$ with $\mathbf{p}^a$; the transformation matrix $\mathbf{T}_a^b \in \mathbf{SE}(3)$ transforms a 3D point $\mathbf{p}^a$ in coordinate frame $\mathcal{F}_a$ to coordinate frame $\mathcal{F}_b$; we further decompose $\mathbf{T}_a^b$ with $\mathbf{R}_a^b \in \mathrm{SO}(3)$ and $\mathbf{t}_a^b \in \mathbb{R}^3$, such that

$$\mathbf{p}^b = \mathbf{T}_a^b \cdot \mathbf{p}^a = \mathbf{R}_a^b \cdot \mathbf{p}^a + \mathbf{t}_a^b, \tag{1}$$

where $\mathbf{t}_a^b$ and $\mathbf{R}_a^b \in \mathrm{SO}(3)$ represents the translation and orientation respectively; we use unit quaternion to represent the orientation $\mathbf{R}_a^b$, i.e. $\bar{\mathbf{q}}_a^b = \begin{bmatrix} q_x & q_y & q_z & q_w \end{bmatrix}^T$, where T represents the transpose operator.

3. Background

Before we proceed, we review several background knowledge that will be used in the following sections.

Exponential map: the exponential map operator connects the Lie algebra (e.g. $\mathbf{so}(3)$) to Lie group (e.g. $\mathbf{SO}(3)$). We can formally define the exponential map as $\exp : \mathbb{R}^3 \rightarrow \mathbb{R}^{3\times3}$ for $\mathbf{SO}(3)$. The rotation matrix is parameterized by unit quaternion representation. If we further denote the tangent vector $\mathbf{r} = \begin{bmatrix} r_x & r_y & r_z \end{bmatrix}^T \in \mathbf{so}(3)$ and use the unit quaterion $\bar{\mathbf{q}} = \begin{bmatrix} q_x & q_y & q_z & q_w \end{bmatrix}^T$ to represent the rotation matrix $\mathbf{R} \in \mathbf{SO}(3)$, we can have

$$\bar{\mathbf{q}} = \begin{bmatrix} q_x \\ q_y \\ q_z \\ q_w \end{bmatrix} = \exp(\mathbf{r}) = \begin{bmatrix} \lambda r_x \\ \lambda r_y \\ \lambda r_z \\ \cos(\theta) \end{bmatrix}, \tag{2}$$

where $\lambda = \frac{\sin(\theta)}{2\theta}$ and $\theta = \frac{1}{2}\sqrt{r_x^2 + r_y^2 + r_z^2}$ [3]. We therefore can derive the Jacobian $\frac{\partial \bar{\mathbf{q}}}{\partial \mathbf{r}} \in \mathbb{R}^{4\times3}$ as follows.

$$\frac{\partial \bar{\mathbf{q}}}{\partial \mathbf{r}} = \begin{bmatrix} \frac{\partial q_x}{\partial r_x} & \frac{\partial q_x}{\partial r_y} & \frac{\partial q_x}{\partial r_z} \\ \frac{\partial q_y}{\partial r_x} & \frac{\partial q_y}{\partial r_y} & \frac{\partial q_y}{\partial r_z} \\ \frac{\partial q_z}{\partial r_x} & \frac{\partial q_z}{\partial r_y} & \frac{\partial q_z}{\partial r_z} \\ \frac{\partial q_w}{\partial r_x} & \frac{\partial q_w}{\partial r_y} & \frac{\partial q_w}{\partial r_z} \end{bmatrix}, \tag{3}$$

where

$$\frac{\partial \theta}{\partial r_x} = \frac{r_x}{2\theta}, \quad \frac{\partial \theta}{\partial r_y} = \frac{r_y}{2\theta}, \tag{4}$$

$$\frac{\partial \theta}{\partial r_z} = \frac{r_z}{2\theta}, \quad \frac{\partial \lambda}{\partial \theta} = \frac{\cos(\theta) - 2\lambda}{2\theta}, \tag{5}$$

$$\frac{\partial q_x}{\partial r_x} = \frac{\partial \lambda}{\partial \theta}\frac{\partial \theta}{\partial r_x} r_x + \lambda, \quad \frac{\partial q_x}{\partial r_y} = \frac{\partial \lambda}{\partial \theta}\frac{\partial \theta}{\partial r_y} r_x, \tag{6}$$

$$\frac{\partial q_x}{\partial r_z} = \frac{\partial \lambda}{\partial \theta}\frac{\partial \theta}{\partial r_z} r_x, \quad \frac{\partial q_y}{\partial r_x} = \frac{\partial \lambda}{\partial \theta}\frac{\partial \theta}{\partial r_x} r_y, \tag{7}$$

$$\frac{\partial q_y}{\partial r_y} = \frac{\partial \lambda}{\partial \theta}\frac{\partial \theta}{\partial r_y} r_y + \lambda, \quad \frac{\partial q_y}{\partial r_z} = \frac{\partial \lambda}{\partial \theta}\frac{\partial \theta}{\partial r_z} r_y, \tag{8}$$

$$\frac{\partial q_z}{\partial r_x} = \frac{\partial \lambda}{\partial \theta}\frac{\partial \theta}{\partial r_x} r_z, \quad \frac{\partial q_z}{\partial r_y} = \frac{\partial \lambda}{\partial \theta}\frac{\partial \theta}{\partial r_y} r_z, \tag{9}$$

$$\frac{\partial q_z}{\partial r_z} = \frac{\partial \lambda}{\partial \theta}\frac{\partial \theta}{\partial r_z} r_z + \lambda, \quad \frac{\partial q_w}{\partial r_x} = -\sin(\theta) \cdot \frac{\partial \theta}{\partial r_x}, \tag{10}$$

$$\frac{\partial q_w}{\partial r_y} = -\sin(\theta) \cdot \frac{\partial \theta}{\partial r_y}, \quad \frac{\partial q_w}{\partial r_z} = -\sin(\theta) \cdot \frac{\partial \theta}{\partial r_z}. \tag{11}$$

It can be seen that $\lambda = \frac{\sin(\theta)}{2\theta}$ is not well defined if $\theta \rightarrow 0$. It is special handled by Taylor series expansion as follows

$$q_x = (\frac{1}{2} - \frac{1}{12}\theta^2 + \frac{1}{240}\theta^4)r_x, \tag{12}$$

$$q_y = (\frac{1}{2} - \frac{1}{12}\theta^2 + \frac{1}{240}\theta^4)r_y, \tag{13}$$

$$q_z = (\frac{1}{2} - \frac{1}{12}\theta^2 + \frac{1}{240}\theta^4)r_z. \tag{14}$$

$$q_w = 1 - \frac{1}{2}\theta^2 + \frac{1}{24}\theta^4. \tag{15}$$

Since $\theta \to 0$, the corresponding Jacobian is then derived as

$$\frac{\partial q_x}{\partial r_x} = 0.5 \qquad \frac{\partial q_x}{\partial r_y} = 0 \qquad \frac{\partial q_x}{\partial r_z} = 0 \tag{16}$$

$$\frac{\partial q_y}{\partial r_x} = 0 \qquad \frac{\partial q_y}{\partial r_y} = 0.5 \qquad \frac{\partial q_y}{\partial r_z} = 0 \tag{17}$$

$$\frac{\partial q_z}{\partial r_x} = 0 \qquad \frac{\partial q_z}{\partial r_y} = 0 \qquad \frac{\partial q_z}{\partial r_z} = 0.5 \tag{18}$$

$$\frac{\partial q_w}{\partial r_x} = 0 \qquad \frac{\partial q_w}{\partial r_y} = 0 \qquad \frac{\partial q_w}{\partial r_z} = 0 \tag{19}$$

Logarithm map: the logarithm map operator is the inverse of the exponential map operator. It can be formally defined as $\log : \mathbb{R}^{3\times3} \to \mathbb{R}^3$ for rotation matrix $\mathbf{R} \in \mathrm{SO}(3)$. We can have the tangent vector $\mathbf{r} \in \mathbf{so}(3)$ if we represent $\mathbf{R}$ with unit quaternion representation $\bar{\mathbf{q}}$,

$$\mathbf{r} = \log(\bar{\mathbf{q}}) = \lambda \begin{bmatrix} q_x \\ q_y \\ q_z \end{bmatrix}, \tag{20}$$

where $\lambda = 2\frac{\arctan(\frac{\theta}{q_w})}{\theta}$ and $\theta = \sqrt{q_x^2 + q_y^2 + q_z^2}$. We therefore can derive the Jacobian $\frac{\partial \mathbf{r}}{\partial \bar{\mathbf{q}}} \in \mathbb{R}^{3\times4}$ as follows:

$$\frac{\partial \mathbf{r}}{\partial \bar{\mathbf{q}}} = \begin{bmatrix} \frac{\partial r_x}{\partial q_x} & \frac{\partial r_x}{\partial q_y} & \frac{\partial r_x}{\partial q_z} & \frac{\partial r_x}{\partial q_w} \\ \frac{\partial r_y}{\partial q_x} & \frac{\partial r_y}{\partial q_y} & \frac{\partial r_y}{\partial q_z} & \frac{\partial r_y}{\partial q_w} \\ \frac{\partial r_z}{\partial q_x} & \frac{\partial r_z}{\partial q_y} & \frac{\partial r_z}{\partial q_z} & \frac{\partial r_z}{\partial q_w} \end{bmatrix}, \tag{21}$$

where

$$\frac{\partial r_x}{\partial q_x} = \frac{\partial \lambda}{\partial q_x}q_x + \lambda \qquad \frac{\partial r_x}{\partial q_y} = \frac{\partial \lambda}{\partial q_y}q_x \tag{22}$$

$$\frac{\partial r_x}{\partial q_z} = \frac{\partial \lambda}{\partial q_z}q_x \qquad \frac{\partial r_x}{\partial q_w} = \frac{\partial \lambda}{\partial q_w}q_x \tag{23}$$

$$\frac{\partial r_y}{\partial q_x} = \frac{\partial \lambda}{\partial q_x}q_y \qquad \frac{\partial r_y}{\partial q_y} = \frac{\partial \lambda}{\partial q_y}q_y + \lambda \tag{24}$$

$$\frac{\partial r_y}{\partial q_z} = \frac{\partial \lambda}{\partial q_z}q_y \qquad \frac{\partial r_y}{\partial q_w} = \frac{\partial \lambda}{\partial q_w}q_y \tag{25}$$

$$\frac{\partial r_z}{\partial q_x} = \frac{\partial \lambda}{\partial q_x}q_z \qquad \frac{\partial r_z}{\partial q_y} = \frac{\partial \lambda}{\partial q_y}q_z \tag{26}$$

$$\frac{\partial r_z}{\partial q_z} = \frac{\partial \lambda}{\partial q_z}q_z + \lambda \qquad \frac{\partial r_z}{\partial q_w} = \frac{\partial \lambda}{\partial q_w}q_z \tag{27}$$

$$\frac{\partial \lambda}{\partial q_x} = \frac{2q_w - \lambda}{\theta^2}q_x \qquad \frac{\partial \lambda}{\partial q_y} = \frac{2q_w - \lambda}{\theta^2}q_y \tag{28}$$

$$\frac{\partial \lambda}{\partial q_z} = \frac{2q_w - \lambda}{\theta^2}q_z \qquad \frac{\partial \lambda}{\partial q_w} = -2 \tag{29}$$

It can be seen that $\lambda = 2\frac{\arctan(\frac{\theta}{q_w})}{\theta}$ is not well defined if either $\theta \to 0$ or $q_w \to 0$. They are thus special handled as follows.

- If $\theta \to 0$, we can approximate λ with $\lambda = \frac{2}{q_w} - \frac{2\theta^2}{3q_w^3}$. The correspondng Jacobian can be obtained as

$$\frac{\partial \lambda}{\partial q_x} = 2\frac{1}{q_w} - \frac{4q_x}{3q_w^3}, \qquad \frac{\partial \lambda}{\partial q_y} = 2\frac{1}{q_w} - \frac{4q_y}{3q_w^3}, \tag{30}$$

$$\frac{\partial \lambda}{\partial q_z} = 2\frac{1}{q_w} - \frac{4q_z}{3q_w^3}, \qquad \frac{\partial \lambda}{\partial q_w} = -2\frac{1}{q_w^2} + 2\frac{\theta^2}{q_w^4}. \tag{31}$$

- If $q_w \to 0$ and $q_w > 0$, we can have $\lambda = \frac{\pi}{\theta}$. The corresponding Jacobian can be obtained as

$$\frac{\partial \lambda}{\partial q_x} = -\frac{\lambda}{\theta^2}q_x, \qquad \frac{\partial \lambda}{\partial q_y} = -\frac{\lambda}{\theta^2}q_y, \tag{32}$$

$$\frac{\partial \lambda}{\partial q_z} = -\frac{\lambda}{\theta^2}q_z, \qquad \frac{\partial \lambda}{\partial q_w} = 0. \tag{33}$$

- If $q_w \to 0$ and $q_w < 0$, we can have $\lambda = -\frac{\pi}{\theta}$. The corresponding Jacobian can be obtained as

$$\frac{\partial \lambda}{\partial q_x} = \frac{\lambda}{\theta^2}q_x, \qquad \frac{\partial \lambda}{\partial q_y} = \frac{\lambda}{\theta^2}q_y, \tag{34}$$

$$\frac{\partial \lambda}{\partial q_z} = \frac{\lambda}{\theta^2}q_z, \qquad \frac{\partial \lambda}{\partial q_w} = 0. \tag{35}$$

Quaternion multiplication: Given two unit quaternions, i.e. $\bar{\mathbf{q}}_0 = \begin{bmatrix} q_{x0} & q_{y0} & q_{z0} & q_{w0} \end{bmatrix}^T$ and $\bar{\mathbf{q}}_1 = \begin{bmatrix} q_{x1} & q_{y1} & q_{z1} & q_{w1} \end{bmatrix}^T$, we can compute their product as

$$\bar{\mathbf{q}} = \bar{\mathbf{q}}_0 \otimes \bar{\mathbf{q}}_1 = \mathbf{Q}(\bar{\mathbf{q}}_0) \cdot \bar{\mathbf{q}}_1 = \hat{\mathbf{Q}}(\bar{\mathbf{q}}_1) \cdot \bar{\mathbf{q}}_0, \tag{36}$$

where $\otimes$ is the product operator for quaternions, and $\mathbf{Q}(\bar{\mathbf{q}}_0)$ and $\hat{\mathbf{Q}}(\bar{\mathbf{q}}_1)$ can be defined as

$$\mathbf{Q}(\bar{\mathbf{q}}_0) = \begin{bmatrix} q_{w0} & -q_{z0} & q_{y0} & q_{x0} \\ q_{z0} & q_{w0} & -q_{x0} & q_{y0} \\ -q_{y0} & q_{x0} & q_{w0} & q_{z0} \\ -q_{x0} & -q_{y0} & -q_{z0} & q_{w0} \end{bmatrix}, \tag{37}$$

$$\hat{\mathbf{Q}}(\bar{\mathbf{q}}_1) = \begin{bmatrix} q_{w1} & q_{z1} & -q_{y1} & q_{x1} \\ -q_{z1} & q_{w1} & q_{x1} & q_{y1} \\ q_{y1} & -q_{x1} & q_{w1} & q_{z1} \\ -q_{x1} & -q_{y1} & -q_{z1} & q_{w1} \end{bmatrix}. \tag{38}$$

The Jacobian can be derived as

$$\frac{\partial \bar{\mathbf{q}}}{\partial \bar{\mathbf{q}}_0} = \bar{\mathbf{Q}}(\bar{\mathbf{q}}_1), \qquad \frac{\partial \bar{\mathbf{q}}}{\partial \bar{\mathbf{q}}_1} = \mathbf{Q}(\bar{\mathbf{q}}_0). \tag{39}$$

Inverse of unit quaternion: Given a unit quaternion, i.e. $\bar{\mathbf{q}} = \begin{bmatrix} q_x & q_y & q_z & q_w \end{bmatrix}^T$, its inverse is simply the conjugate $\bar{\mathbf{q}}^{-1}$ and can be defined as

$$\bar{\mathbf{q}}^{-1} = \begin{bmatrix} -q_x \\ -q_y \\ -q_z \\ q_w \end{bmatrix}. \tag{40}$$

The Jacobian can be derived as

$$\frac{\partial \bar{\mathbf{q}}^{-1}}{\partial \bar{\mathbf{q}}} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{41}$$

4. Motion trajectory modeling

We model the virtual camera pose at time $t \in [0, \tau]$ as

$$\mathbf{T}_t^w = \mathbf{T}_0^w \cdot \exp(\frac{t}{\tau} \cdot \log((\mathbf{T}_0^w)^{-1} \cdot \mathbf{T}_\tau^w)), \tag{42}$$

where τ is the exposure time, $\mathbf{T}_0^w \in \mathrm{SE}(3)$ and $\mathbf{T}_\tau^w \in \mathrm{SE}(3)$ are the poses corresponding to virtual sharp images captured at the beginning and end of the exposure time respectively. For efficiency, we decompose Eq. (42) as

$$\bar{\mathbf{q}}_t^w = \bar{\mathbf{q}}_0^w \otimes \exp(\frac{t}{\tau} \cdot \log((\bar{\mathbf{q}}_0^w)^{-1} \otimes \bar{\mathbf{q}}_\tau^w)), \tag{43}$$

$$\mathbf{t}_t^w = \mathbf{t}_0^w + \frac{t}{\tau}(\mathbf{t}_\tau^w - \mathbf{t}_0^w), \tag{44}$$

where $\mathbf{T}_*^w = [\mathbf{R}_*^w | \mathbf{t}_*^w] \in \mathrm{SE}(3)$, $\mathbf{R}_*^w \in \mathrm{SO}(3)$ and $\mathbf{t}_*^w \in \mathbb{R}^3$. We represent the rotation matrix $\mathbf{R}_*^w$ with unit quaternion $\bar{\mathbf{q}}_*^w$.

Local parameterization of rotation: For the real implementation, we use the local parameterization for the update of the rotation. The plus operation for unit quaternion $\bar{\mathbf{q}}$ is defined as

$$\bar{\mathbf{q}}' = \bar{\mathbf{q}} \otimes \Delta\bar{\mathbf{q}}, \tag{45}$$

where $\Delta\bar{\mathbf{q}} = \exp(\Delta\mathbf{r})$, $\Delta\mathbf{r} = \begin{bmatrix} \Delta r_x & \Delta r_y & \Delta r_z \end{bmatrix}^T$ and $\Delta r_x \to 0$, $\Delta r_y \to 0$, $\Delta r_z \to 0$. The Jacobian with respect to $\Delta\mathbf{r}$ can thus be derived as

$$\frac{\partial \bar{\mathbf{q}}'}{\partial \Delta\mathbf{r}} = \mathbf{Q}(\bar{\mathbf{q}}) \cdot \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.5 \\ 0 & 0 & 0 \end{bmatrix}. \tag{46}$$

Jacobian related to translation: We can simplify Eq. (44) as

$$\mathbf{t}_t^w = \frac{\tau - t}{\tau}\mathbf{t}_0^w + \frac{t}{\tau}\mathbf{t}_\tau^w. \tag{47}$$

The Jacobians, i.e. $\frac{\partial \mathbf{t}_t^w}{\partial \mathbf{t}_0^w} \in \mathbb{R}^{3\times3}$ and $\frac{\partial \mathbf{t}_t^w}{\partial \mathbf{t}_\tau^w} \in \mathbb{R}^{3\times3}$ can thus be derived as

$$\frac{\partial \mathbf{t}_t^w}{\partial \mathbf{t}_0^w} = \begin{bmatrix} \frac{\tau-t}{\tau} & 0 & 0 \\ 0 & \frac{\tau-t}{\tau} & 0 \\ 0 & 0 & \frac{\tau-t}{\tau} \end{bmatrix}, \tag{48}$$

$$\frac{\partial \mathbf{t}_t^w}{\partial \mathbf{t}_\tau^w} = \begin{bmatrix} \frac{t}{\tau} & 0 & 0 \\ 0 & \frac{t}{\tau} & 0 \\ 0 & 0 & \frac{t}{\tau} \end{bmatrix}. \tag{49}$$

Jacobian related to rotation: We decompose Eq. (43) as

$$\bar{\mathbf{q}}_\tau^0 = (\bar{\mathbf{q}}_0^w)^{-1} \otimes \bar{\mathbf{q}}_\tau^w, \tag{50}$$

$$\mathbf{r} = \frac{t}{\tau} \cdot \log(\bar{\mathbf{q}}_\tau^0), \tag{51}$$

$$\bar{\mathbf{q}}_t^0 = \exp(\mathbf{r}), \tag{52}$$

$$\bar{\mathbf{q}}_t^w = \bar{\mathbf{q}}_0^w \otimes \bar{\mathbf{q}}_t^0. \tag{53}$$

We can rewrite both Eq. (50) and Eq. (53) as

$$\bar{\mathbf{q}}_\tau^0 = \mathbf{Q}((\bar{\mathbf{q}}_0^w)^{-1}) \cdot \bar{\mathbf{q}}_\tau^w = \hat{\mathbf{Q}}(\bar{\mathbf{q}}_\tau^w) \cdot (\bar{\mathbf{q}}_0^w)^{-1}, \tag{54}$$

$$\bar{\mathbf{q}}_t^w = \mathbf{Q}(\bar{\mathbf{q}}_0^w) \cdot \bar{\mathbf{q}}_t^0 = \hat{\mathbf{Q}}(\bar{\mathbf{q}}_t^0) \cdot \bar{\mathbf{q}}_0^w. \tag{55}$$

The Jacobian $\frac{\partial \bar{\mathbf{q}}_t^w}{\partial \bar{\mathbf{q}}_0^w} \in \mathbb{R}^{4\times4}$ can thus be derived as

$$\frac{\partial \bar{\mathbf{q}}_t^w}{\partial \bar{\mathbf{q}}_0^w} = \hat{\mathbf{Q}}(\bar{\mathbf{q}}_t^0) + \mathbf{Q}(\bar{\mathbf{q}}_0^w) \cdot \frac{\partial \bar{\mathbf{q}}_t^0}{\partial \bar{\mathbf{q}}_0^w}, \tag{56}$$

$$\frac{\partial \bar{\mathbf{q}}_t^0}{\partial \bar{\mathbf{q}}_0^w} = \frac{\partial \bar{\mathbf{q}}_t^0}{\partial \mathbf{r}} \cdot \frac{\partial \mathbf{r}}{\partial \bar{\mathbf{q}}_\tau^0} \cdot \hat{\mathbf{Q}}(\bar{\mathbf{q}}_\tau^w) \cdot \frac{\partial (\bar{\mathbf{q}}_0^w)^{-1}}{\partial \bar{\mathbf{q}}_0^w}. \tag{57}$$

Similarly for the Jacobian $\frac{\partial \bar{\mathbf{q}}_t^w}{\partial \bar{\mathbf{q}}_\tau^w} \in \mathbb{R}^{4\times4}$, we can derive it as

$$\frac{\partial \bar{\mathbf{q}}_t^w}{\partial \bar{\mathbf{q}}_\tau^w} = \mathbf{Q}(\bar{\mathbf{q}}_0^w) \cdot \frac{\partial \bar{\mathbf{q}}_t^0}{\partial \mathbf{r}} \cdot \frac{\partial \mathbf{r}}{\partial \bar{\mathbf{q}}_\tau^0} \cdot \mathbf{Q}((\bar{\mathbf{q}}_0^w)^{-1}). \tag{58}$$

Note that both $\frac{\partial \bar{\mathbf{q}}_t^0}{\partial \mathbf{r}}$ and $\frac{\partial \mathbf{r}}{\partial \bar{\mathbf{q}}_\tau^0}$ are the Jacobians related to the exponential mapping and logarithm mapping respectively. $\frac{\partial (\bar{\mathbf{q}}_0^w)^{-1}}{\partial \bar{\mathbf{q}}_0^w}$ is the Jacobian related to the inverse of quaternion.

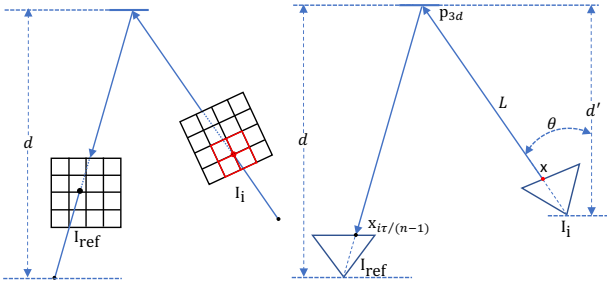

Figure 1. Geometric relationship between $\mathbf{x} \in \mathbb{R}^2$ (i.e. the red pixel) of the virtual sharp image $\mathbf{I}_i$ and $\mathbf{x}_{\frac{i\tau}{n-i}} \in \mathbb{R}^2$ (i.e. the black pixel) of the reference image $\mathbf{I}_{\text{ref}}$.

The Jacobians with respect to the local parameterization can then be computed as

$$\frac{\partial \bar{\mathbf{q}}_t^w}{\partial \Delta \mathbf{r}_0^w} = \frac{\partial \bar{\mathbf{q}}_t^w}{\partial \bar{\mathbf{q}}_0^w} \cdot \mathbf{Q}(\bar{\mathbf{q}}_0^w) \cdot \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.5 \\ 0 & 0 & 0 \end{bmatrix}, \tag{59}$$

$$\frac{\partial \bar{\mathbf{q}}_t^w}{\partial \Delta \mathbf{r}_\tau^w} = \frac{\partial \bar{\mathbf{q}}_t^w}{\partial \bar{\mathbf{q}}_\tau^w} \cdot \mathbf{Q}(\bar{\mathbf{q}}_\tau^w) \cdot \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.5 \\ 0 & 0 & 0 \end{bmatrix}. \tag{60}$$

5. Details on the pixel point transfer

In this section, we derive the geometric relationship between the reference keyframe and the i^{th} virtual sharp image, which we denote the corresponding pose with $\mathbf{T}_i^{ref} \in \mathbf{SE}(3)$. We simplify Figure 3 in the main text (i.e. left figure of Fig. 1) with its 2D top-down view (i.e. right figure of Fig. 1).

We denote the translation vector $\mathbf{p}_i^{ref}$ of $\mathbf{T}_i^{ref}$ with $[p_x, p_y, p_z]^T$ and represent the rotation $\mathbf{R}_i^{ref}$ with unit quaternion, i.e. $\bar{\mathbf{q}} = [q_x, q_y, q_z, q_w]^T$. We denote d as the depth of the frontal-parallel plane with respect to the reference key-frame. We can then compute the distance d' between the camera center of the i^{th} virtual camera to the frontal-parallel plane as

$$d' = d - p_z. \tag{61}$$

The unitary ray of pixel $\mathbf{x} \in \mathbb{R}^2$ in the i^{th} image $\mathbf{I}_i$ can be computed by the back-projection function $\pi^{-1} : \mathbb{R}^2 \rightarrow \mathbb{R}^3$ as

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \pi^{-1}(\mathbf{x}), \tag{62}$$

where $x^2 + y^2 + z^2 = 1$. We can then compute cosine function of the angle (i.e. θ) between the unitary ray and the plane normal of the frontal parallel plane as

$$\lambda = \cos(\theta) = (\mathbf{R}_i^{ref} \cdot \pi^{-1}(\mathbf{x}))^T \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \tag{63}$$

from which we can further simply it as

$$\lambda = 2x(q_x q_z - q_w q_y) + 2y(q_x q_w + q_y q_z) + z(q_w^2 - q_x^2 - q_y^2 + q_z^2). \tag{64}$$

The length of the line segment L, which goes through pixel point $\mathbf{x}$ from camera center of the i^{th} camera and intersects with the frontal-parallel plane, can then be simply computed as

$$|L| = \frac{d'}{\lambda} = \frac{d - p_z}{\lambda}. \tag{65}$$

The 3D intersection point $\mathbf{p}_{3d}$ between the line segment L and the frontal parallel plane can thus be computed as

$$\mathbf{p}_{3d} = |L| \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \frac{d - p_z}{\lambda} \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \tag{66}$$

where the $\mathbf{p}_{3d}$ is represented in the coordinate frame of the i^{th} camera. To compute the corresponding pixel point $\mathbf{x}_{\frac{i\tau}{n-1}}$ in the reference image $\mathbf{I}_{\text{ref}}$, we need transform the 3D point $\mathbf{p}_{3d}$ to the reference camera coordinate frame and then project it to the image plane. It can be formally defined as

$$\mathbf{p}'_{3d} = \mathbf{T}_i^{ref} \cdot \mathbf{p}_{3d}, \tag{67}$$

$$\mathbf{x}_{\frac{i\tau}{n-1}} = \pi(\mathbf{p}'_{3d}), \tag{68}$$

where $\mathbf{p}'_{3d}$ is the 3D point $\mathbf{p}_{3d}$ represented in the reference camera, $\pi : \mathbb{R}^3 \rightarrow \mathbb{R}^2$ is the camera projection function.

Jacobian derivations: The pose of the i^{th} virtual camera, i.e. $\mathbf{T}_i^{ref}$, relates to $\mathbf{T}_0^{ref}$ and $\mathbf{T}_\tau^{ref}$ via Eq. (42). To estimate both $\mathbf{T}_0^{ref}$ and $\mathbf{T}_\tau^{ref}$, we need to have the Jacobian of $\mathbf{x}_{\frac{i\tau}{n-1}}$ with respect to $\mathbf{T}_i^{ref}$. Since the relationship between $\mathbf{x}_{\frac{i\tau}{n-1}}$ and $\mathbf{T}_i^{ref}$ is complex, as derived above, we use the Mathematica Symbolic Toolbox[1] for the ease of Jacobian derivations. The details are as follows.

$$\alpha_0 = q_x x + q_y y + q_z z, \quad \alpha_1 = q_y x - q_w z - q_x y, \tag{69}$$

$$\alpha_2 = q_w y - q_x z + q_z x, \quad \alpha_3 = q_w x + q_y z - q_z y, \tag{70}$$

$$\alpha_4 = q_w z + q_x y - q_y x, \quad \beta_0 = -2(q_w q_z - q_x q_y), \tag{71}$$

$$\beta_1 = 2(q_w q_y + q_x q_z), \quad \beta_2 = 2(q_w q_z + q_x q_y), \tag{72}$$

$$\beta_3 = -2(q_w q_x - q_y q_z), \quad \beta_4 = -2(q_w q_y - q_x q_z), \tag{73}$$

$$\beta_5 = 2(q_w q_x + q_y q_z), \tag{74}$$

[1] https://www.wolfram.com/mathematica/

$$\gamma_0 = x(q_w^2 + q_x^2 - q_y^2 - q_z^2) + y\beta_0 + z\beta_1, \quad (75)$$
$$\gamma_1 = x\beta_2 + y(q_w^2 - q_x^2 + q_y^2 - q_z^2) + z\beta_3, \quad (76)$$
$$\gamma_2 = x\beta_4 + y\beta_5 + z(q_w^2 - q_x^2 - q_y^2 + q_z^2), \quad (77)$$

$$\frac{\partial \mathbf{p}'_{3d}}{\partial p_x} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad (78)$$

$$\frac{\partial \mathbf{p}'_{3d}}{\partial p_y} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad (79)$$

$$\frac{\partial \mathbf{p}'_{3d}}{\partial p_z} = \begin{bmatrix} -\gamma_0/\lambda \\ -\gamma_1/\lambda \\ 1 - \gamma_2/\lambda \end{bmatrix}, \quad (80)$$

$$\frac{\partial \mathbf{p}'_{3d}}{\partial q_x} = 2\frac{d - p_z}{\lambda} \begin{bmatrix} \alpha_0 - \alpha_2\gamma_0/\lambda \\ \alpha_1 - \alpha_2\gamma_1/\lambda \\ \alpha_2 - \alpha_2\gamma_2/\lambda \end{bmatrix}, \quad (81)$$

$$\frac{\partial \mathbf{p}'_{3d}}{\partial q_y} = 2\frac{d - p_z}{\lambda} \begin{bmatrix} \alpha_4 + \alpha_3\gamma_0/\lambda \\ \alpha_0 + \alpha_3\gamma_1/\lambda \\ -\alpha_3 + \alpha_3\gamma_2/\lambda \end{bmatrix}, \quad (82)$$

$$\frac{\partial \mathbf{p}'_{3d}}{\partial q_z} = 2\frac{d - p_z}{\lambda} \begin{bmatrix} -\alpha_2 + \alpha_0\gamma_0/\lambda \\ \alpha_3 - \alpha_0\gamma_1/\lambda \\ \alpha_0 - \alpha_0\gamma_2/\lambda \end{bmatrix}, \quad (83)$$

$$\frac{\partial \mathbf{p}'_{3d}}{\partial q_w} = 2\frac{d - p_z}{\lambda} \begin{bmatrix} \alpha_3 - \alpha_4\gamma_0/\lambda \\ \alpha_2 - \alpha_4\gamma_1/\lambda \\ \alpha_4 - \alpha_4\gamma_2/\lambda \end{bmatrix}. \quad (84)$$

The Jacobian $\frac{\partial \mathbf{x}_{\frac{i\tau}{n-1}}}{\partial \mathbf{p}'_{3d}} \in \mathbb{R}^{2\times 3}$ is related to the camera projection function. For a pinhole camera model with the intrinsic parameters f_x, f_y, c_x, c_y, it can be derived as

$$\frac{\partial \mathbf{x}_{\frac{i\tau}{n-1}}}{\partial \mathbf{p}'_{3d}} = \begin{bmatrix} \frac{f_x}{\mathbf{p}'_{3d_z}} & 0 & -\frac{f_x \mathbf{p}'_{3d_x}}{(\mathbf{p}'_{3d_z})^2} \\ 0 & \frac{f_y}{\mathbf{p}'_{3d_z}} & -\frac{f_y \mathbf{p}_{3d_y}}{(\mathbf{p}'_{3d_z})^2} \end{bmatrix}, \quad (85)$$

where $\mathbf{p}'_{3d} = [\mathbf{p}'_{3d_x}, \mathbf{p}'_{3d_y}, \mathbf{p}'_{3d_z}]^T$.

6. Computational complexity

To compensate the effect of motion blur, we need to sample multiple discrete virtual frames to synthesize the blurry image (Eq.6 in the main text). The number of discrete frames should be larger than the blur kernel size. Currently, we use $n = 64$ for our experiments. If $n = 1$, it reduces to the problem of assuming the images are sharp. The accuracy would thus be affected for motion blurred images. If we raise n to 2 or higher, the number of pose parameters would not be affected (i.e. 12 variables). However, the number of pixel transfers would be increased (e.g. doubled for 2 virtual frames). Larger n thus requires more computational resources. To further improve the efficiency, we can dynamically change the number of discrete frames (i.e. we can use a smaller n for less blurry image and a larger n for more blurry image). The back-end is the same as the original DSO [2] and it is running on CPU. Note that the back-end only relies on the deblurred keyframes when optimizing the keyframe poses and structure, thus we can use the original DSO implementation.

7. Additional experimental results with our real-world dataset

Table 1 presents additional experimental results with our real-world dataset. It demonstrates that ORB-SLAM [4] suffers from significant frame drops if the images are motion blurred. It is reasonable since ORB-SLAM is a sparse feature based approach. If the images are severely motion blurred, the sparse feature detector would have difficulties to detect enough good features for motion estimation. In contrast, DSO [2] is more robust to motion blur (i.e. no frame drops). However, since pairs of motion blurred images usually violate the photometric-consistency assumption, the accuracy of DSO is degraded. Our proposed motion blur aware visual odometry (i.e. MBA-VO) models the motion blur for direct image alignment algorithm, so that the photometric-consistency assumption would still hold even the images are motion blurred. It thus achieves better accuracy and robustness compared to DSO and ORB-SLAM.

8. Performance in the absence of motion-blur

To further demonstrate the performance of our proposed method with images in good quality, we run an experiment comparing the proposed approach with DSO on the EuRoC dataset [1]. Table 2 demonstrates that MBA-VO performs slightly worse than DSO on MH_02_easy and MH_03_medium. However, it performs better than DSO on MH_01_easy, MH_04_difficult and MH_05_difficult. In general, we observed similar performance as DSO for good images. The VICON room sequences in the EuRoC dataset are very challenging. Both DSO and MBA-VO fail even

	ORB-SLAM [4]		DSO [2]		MBA-VO	
	ATE (m)	FD (%)	ATE (m)	FD (%)	ATE (m)	FD (%)
Seq5	0.1931	73.2	0.3241	0	0.2667	0
Seq6	0.0743	25.4	0.4968	0	0.3321	0
Seq7	0.1872	47.9	0.3857	0	0.2718	0
Seq8	0.5861	31.9	0.7906	0	0.3915	0
Seq9	0.3791	26.6	0.8538	0	0.2838	0
Seq10	0.1708	33.6	0.4800	0	0.4319	0
Seq11	0.1378	39.1	x	x	0.4003	0
Seq12	x	x	0.5031	0	0.3632	0
Seq13	x	x	0.4501	0	0.3043	0
Seq14	x	x	x	x	0.4516	0
Seq15	x	x	x	x	0.3687	0
Seq16	x	x	x	x	0.3765	0
Seq17	x	x	x	x	0.3299	0

Table 1. *The performance of MBA-VO on our real-world dataset.* x denotes the corresponding algorithm fails on that particular sequence. It demonstrates that ORB-SLAM suffers from significant frame drops if the images are motion blurred, although it usually has more accurate motion estimations. MBA-VO improves the accuracy of DSO, while being robust to motion blur with no frame drops.

with sharp images (either with very large errors or complete tracking failure). This might be caused by the lack of good texture for some of the frames. The sequences also contain degenerate motions (e.g. close to pure rotation). Without any relocalization module (e.g. as is used in ORB-SLAM), it is hard for both DSO and MBA-VO to recover once the pipeline breaks or drifts significantly.

	ORB-SLAM	DSO	MBA-VO
MH_01_easy	**0.030**	0.050	0.035
MH_02_easy	**0.022**	0.077	0.101
MH_03_medium	**0.049**	0.178	0.239
MH_04_difficult	2.472	1.181	**0.476**
MH_05_difficult	4.386	1.261	**0.265**

Table 2. **EuRoC dataset:** Comparison in terms of ATE RMSE error metric (m). Note that we did not discard any images (e.g. the first few shaky images which are used to initialize IMU) from the EuRoC dataset for our evaluation. Therefore, the resulted accuracy for ORB-SLAM and DSO might be a bit different from prior reported results.

9. Potential integration with IMU measurements

We think the performance would be further improved if we integrate the measurements from IMU. However, we generally cannot rely on IMU solely for odometry due to drift, and it needs to be combined with vision. This means that we still need to track for blurry images to reduce drift. Note that IMU measurements can naturally enter our pipeline by providing initial estimates for the local motion trajectory, as well as helping the keyframe deblurring. In the paper, we decided to focus on the pure VO problem first, but we think the pipeline can be easily extended to VIO in the future. Another promising future direction would be to also model rolling shutter effects during the reblurring.

References

[1] Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus W Achtelik, and Roland Siegwart. The euroc micro aerial vehicle datasets. *The International Journal of Robotics Research*, 2016. 5

[2] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *Trans. Pattern Analysis and Machine Intelligence (PAMI)*, 40(3):611–625, 2017. 5, 6

[3] Sebastian Grassia. Practical Parameterization of Rotations Using the Exponential Map. *Journal of Graphics Tools*, 1998. 1

[4] Raul Mur-Artal and Juan D Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017. 5, 6