Five-Point Algorithms for Estimating Pose and Velocity

Nicholas Gans
Kaveh Fathian
Yujie Zhang
Cody Lundberg
Mohammadreza Davoodi
Bardia Mojra

Sunday, August 8, 2021

CCTA 2021 Workshop
The Confluence of Vision and Control - Part II

UTARI UNIVERSITY OF TEXAS AT ARLINGTON RESEARCH INSTITUTE

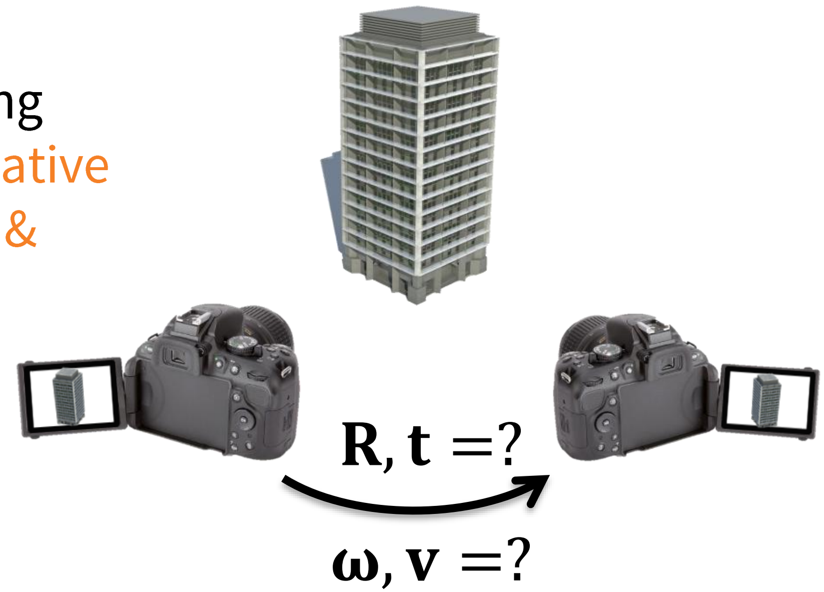
Outline

- Vision-Based Motion Estimation
- QuEst – Quaternion-Based Pose Estimation
- VEst – Velocity Estimation
- Combining Quest and Vest (current work, needs a catchy acronym)

Camera Relative Motion Estimation

Objective:

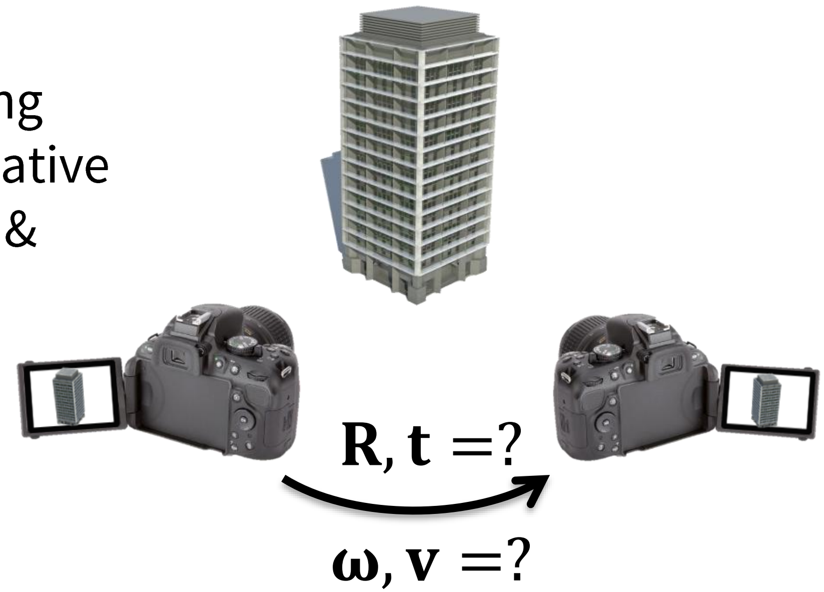
Given images from two cameras (or one moving camera) of the same scene/object, find the relative rotation and translation (i.e., pose) and linear & angular velocity between the two cameras



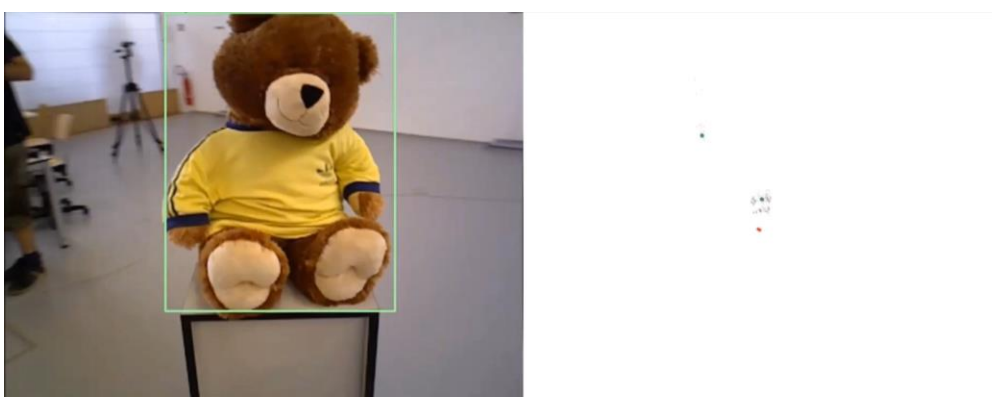
Camera Relative Motion Estimation

Objective:

Given images from two cameras (or one moving camera) of the same scene/object, find the relative rotation and translation (i.e., pose) and linear & angular velocity between the two cameras



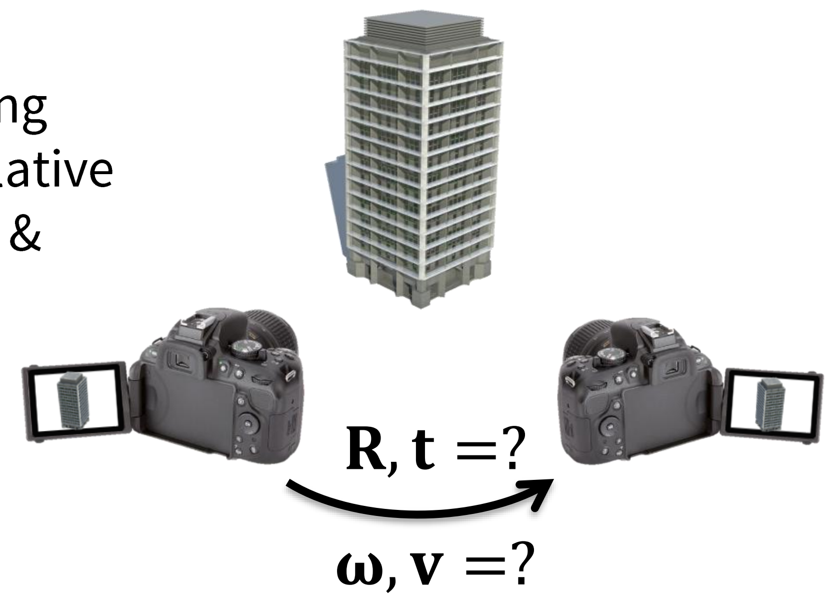
Applications:



Camera Relative Motion Estimation

Objective:

Given images from two cameras (or one moving camera) of the same scene/object, find the relative rotation and translation (i.e., pose) and linear & angular velocity between the two cameras



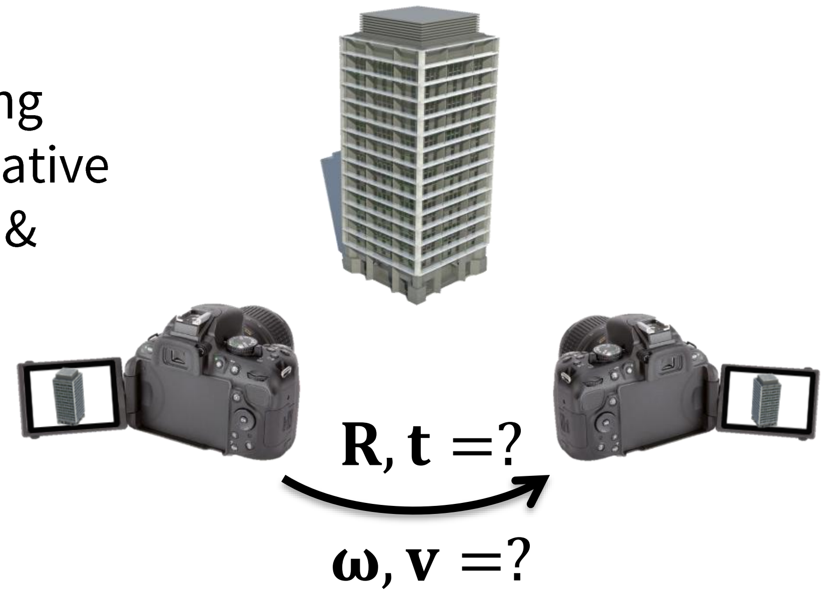
Existing solutions:

Longuet-Higgins	Nature 1981	(8pt algorithm)
Hartley	ECCV, 1994	(4pt homography)

Camera Relative Motion Estimation

Objective:

Given images from two cameras (or one moving camera) of the same scene/object, find the relative rotation and translation (i.e., pose) and linear & angular velocity between the two cameras



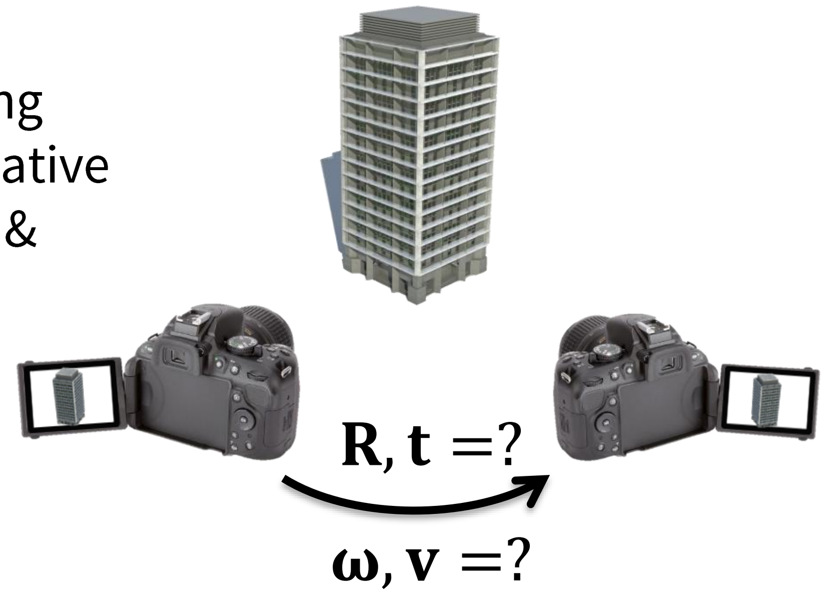
Existing Pose solutions:

Longuet-Higgins	Nature 1981	(8pt algorithm)
Hartley	ECCV, 1994	(4pt homography)
Nister	CVPR 2003	(5pt algorithm)
Stewenius et al.	JPRS 2006	(Gröbner basis)
Li, Hartley	ICPR 2006	(Speeded up 5pt)

Camera Relative Motion Estimation

Objective:

Given images from two cameras (or one moving camera) of the same scene/object, find the relative rotation and translation (i.e., pose) and linear & angular velocity between the two cameras



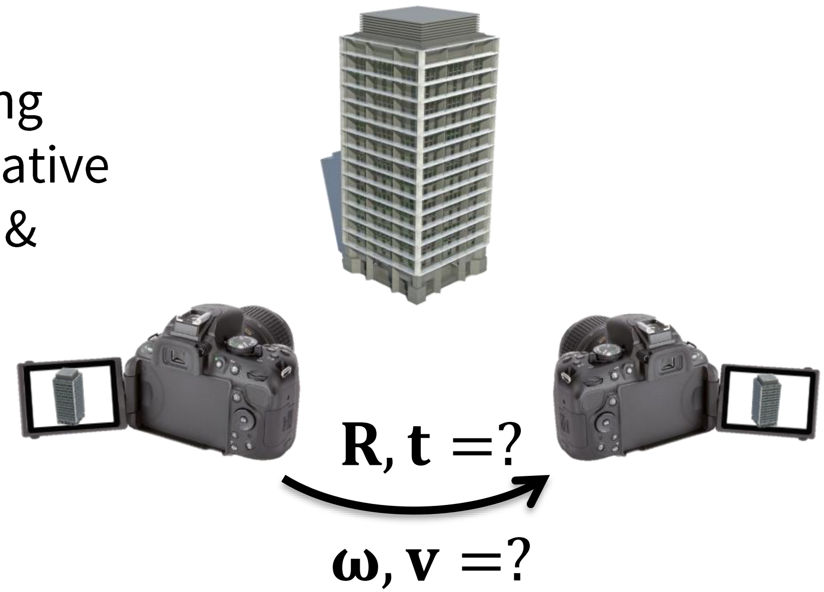
Existing velocity solutions:

Ma, et al	Invitation to 3D Vision, 1994	(8pt *continuous* essential)
		(4pt *continuous* homography)
Soatto & Perona	IJCV 1997	(EKF + Jacobean)
Dani, et al	ITAC 2012	(4pt Homography & nonlinear estimators)
Tick, et al.	ITCyb	(4pt continuous and discrete homography + EKF)

Camera Relative Motion Estimation

Objective:

Given images from two cameras (or one moving camera) of the same scene/object, find the relative rotation and translation (i.e., pose) and linear & angular velocity between the two cameras



In our work

- A direct solution to the pose and velocity estimation problem
- It eschews the essential and homography matrices
- It uses 5 matched/tracked points (the provable minimum)
- 5 points can be in an configuration (planar or nonplanar)
- It works in cases of 0 translation or 0 rotation
- There is no necessary a priori knowledge
    - We can only recover up to an unknown scale factor without additional knowledge)

The Rigid Motion Constraints

Depths of the 3D point

$$u\,\mathbf{R}\,\mathbf{m} + \mathbf{t} = v\,\mathbf{n}$$

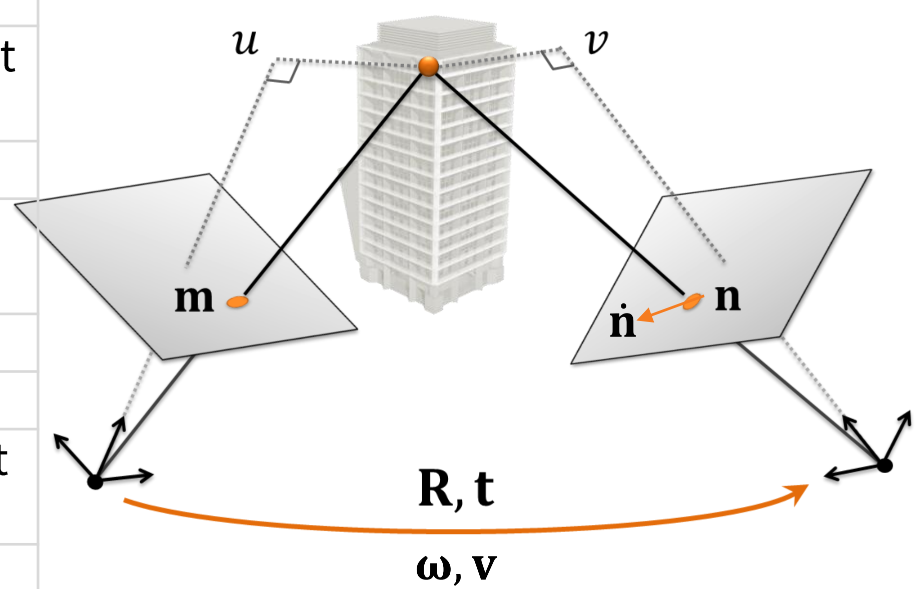
Homogeneous feature point coordinates

Depths of the 3D point

$$v\,\boldsymbol{\omega}_\times\,\mathbf{n} + \mathbf{v} = v\,\dot{\mathbf{n}} + \dot{v}\,\mathbf{n}$$

Homogeneous feature point coordinates and optical flow

u	depth of point m at time 0
$\mathbf{R}$	Rotation between camera frame at current time and time 0
$\mathbf{m}$	feature coordinates at time 0
$\mathbf{t}$	Translation between camera frame at current time and time 0
v	depth of point n at current time
$\mathbf{n}$	feature coordinates at current time
$\boldsymbol{\omega}_\times$	skew symmetric angular velocity matrix at current time
$\mathbf{v}$	linear velocity of camera at current time



The Rigid Motion Constraints

Example:

$$u_1\,\mathbf{R}\begin{bmatrix}-0.3\\0.22\\1\end{bmatrix}+\mathbf{t}\;=v_1\begin{bmatrix}-0.35\\0.23\\1\end{bmatrix}$$

$$u_2\,\mathbf{R}\begin{bmatrix}-0.07\\0.19\\1\end{bmatrix}+\mathbf{t}\;=v_2\begin{bmatrix}-0.1\\0.2\\1\end{bmatrix}$$

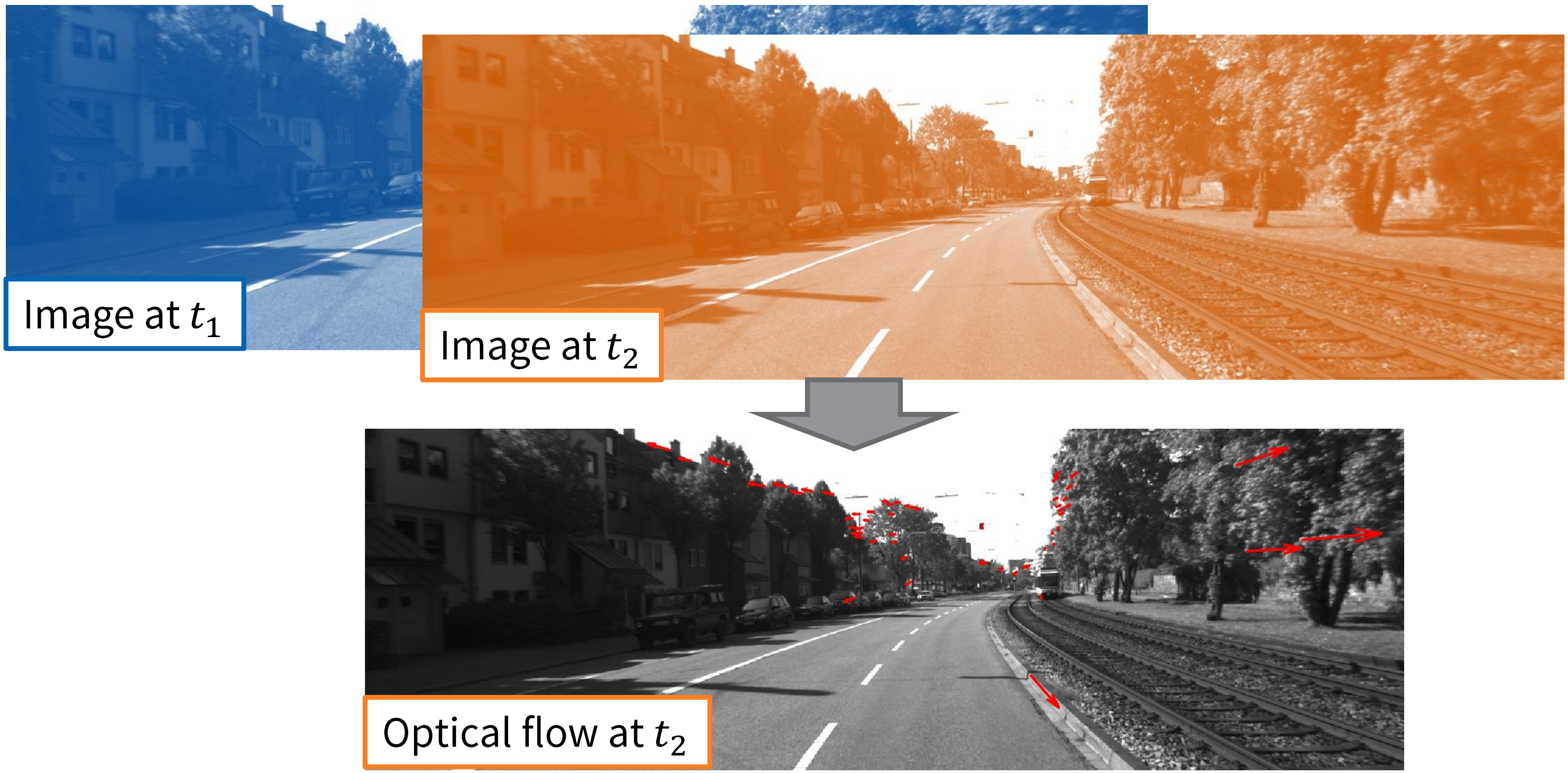
etc.

The Rigid Motion Constraints

Example:

$$u_2 \boldsymbol{\omega}_\times \begin{bmatrix} -0.6535 \\ -0.0010 \\ 1 \end{bmatrix} + \mathbf{v} = u_2 \begin{bmatrix} -0.0193 \\ -0.0041 \\ 0 \end{bmatrix} + \dot{u}_2 \begin{bmatrix} -0.6535 \\ -0.0010 \\ 1 \end{bmatrix}$$

$$u_3 \boldsymbol{\omega}_\times \begin{bmatrix} -0.1230 \\ -0.0561 \\ 1 \end{bmatrix} + \mathbf{v} = u_3 \begin{bmatrix} 0 \\ 0.0020 \\ 0 \end{bmatrix} + \dot{u}_3 \begin{bmatrix} -0.1230 \\ -0.0561 \\ 1 \end{bmatrix}$$



The QuEst Algorithm

Quaternions

$$u\,\mathbf{R}\,\mathbf{m} + \mathbf{t} = v\;\;\mathbf{n}$$

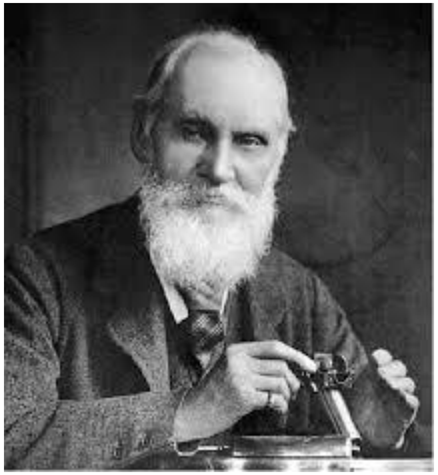
Rotation matrices can be expressed as a function of four elements $\{w, x, y, z\}$ are known as quaternions

$$R = \begin{bmatrix} w^2 + x^2 - y^2 - z^2 & 2(xy - wz) & 2(xz + wx) \\ 2(xy + wz) & w^2 - x^2 + y^2 - z^2 & 2(yz - wy) \\ 2(xz - wx) & 2(yz + wy) & w^2 - x^2 - y^2 + z^2 \end{bmatrix}$$

$$w^2 + x^2 + y^2 + z^2 = 1$$

“Quaternions came from Hamilton after his really good work had been done; and though beautifully ingenious, have been an unmixed evil to those who have touched them in any way, including Maxwell.“

- Lord Kelvin



The QuEst Algorithm

Quaternions

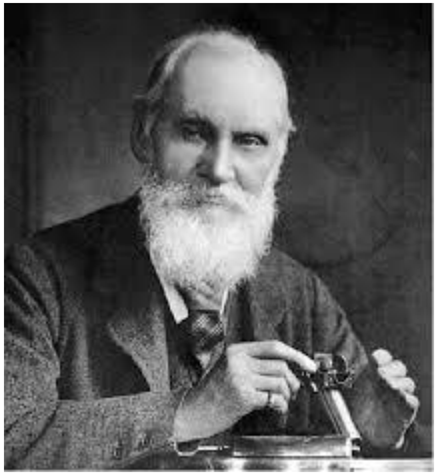
$$u\,\mathbf{R}\,\mathbf{m} + \mathbf{t} = v\ \mathbf{n}$$

Rotation matrices can be expressed as a function of four elements $\{w, x, y, z\}$ are known as quaternions

$$R = \begin{bmatrix} w^2 + x^2 - y^2 - z^2 & 2(xy - wz) & 2(xz + wx) \\ 2(xy + wz) & w^2 - x^2 + y^2 - z^2 & 2(yz - wy) \\ 2(xz - wx) & 2(yz + wy) & w^2 - x^2 - y^2 + z^2 \end{bmatrix}$$

$$w^2 + x^2 + y^2 + z^2 = 1$$

“Symmetrical equations are good in their place, but 'vector' is a useless survival, or offshoot from quaternions, and has never been of the slightest use to any creature.”
- Lord Kelvin



The QuEst Algorithm

[Fathian et al., RAL 2018]

Step 1: Subtract constraints

$$u\,\mathbf{R}\,\mathbf{m} + \mathbf{t} = v\,\mathbf{n}$$

for any 2 feature points to eliminate **t**.

Example:

$$\left.\begin{aligned} u_1\,\mathbf{R}\begin{bmatrix}-0.3\\0.22\\1\end{bmatrix} + \mathbf{t} &= v_1\begin{bmatrix}-0.35\\0.23\\1\end{bmatrix} \\ u_2\,\mathbf{R}\begin{bmatrix}-0.07\\0.19\\1\end{bmatrix} + \mathbf{t} &= v_2\begin{bmatrix}-0.1\\0.2\\1\end{bmatrix}\end{aligned}\right\} \xRightarrow{\text{subtract}}$$

$$u_1\,\mathbf{R}\begin{bmatrix}-0.3\\0.22\\1\end{bmatrix} - v_1\begin{bmatrix}-0.35\\0.23\\1\end{bmatrix} - u_2\,\mathbf{R}\begin{bmatrix}-0.07\\0.19\\1\end{bmatrix} + v_2\begin{bmatrix}-0.1\\0.2\\1\end{bmatrix} = 0$$

The QuEst Algorithm

Step 2: Eliminate depths using 3 feature points

$$u_1\,\mathbf{R}\begin{bmatrix}-0.3\\0.22\\1\end{bmatrix} - v_1\begin{bmatrix}-0.35\\0.23\\1\end{bmatrix} - u_2\,\mathbf{R}\begin{bmatrix}-0.07\\0.19\\1\end{bmatrix} + v_2\begin{bmatrix}-0.1\\0.2\\1\end{bmatrix} = 0$$

$$u_1\,\mathbf{R}\begin{bmatrix}-0.3\\0.22\\1\end{bmatrix} - v_1\begin{bmatrix}-0.35\\0.23\\1\end{bmatrix} - u_3\,\mathbf{R}\begin{bmatrix}-0.06\\-0.26\\1\end{bmatrix} + v_3\begin{bmatrix}-0.15\\-0.25\\1\end{bmatrix} = 0$$

Previous operation performed for points 1&2 and 1&3

$$\Longrightarrow \underbrace{\begin{bmatrix} \mathbf{R}\begin{bmatrix}-0.3\\0.22\\1\end{bmatrix} & \begin{bmatrix}0.35\\-0.23\\-1\end{bmatrix} & \mathbf{R}\begin{bmatrix}0.07\\-0.19\\-1\end{bmatrix} & \begin{bmatrix}-0.1\\0.2\\1\end{bmatrix} & \mathbf{0} & \mathbf{0} \\ \mathbf{R}\begin{bmatrix}-0.3\\0.22\\1\end{bmatrix} & \begin{bmatrix}0.35\\-0.23\\-1\end{bmatrix} & \mathbf{0} & \mathbf{0} & \mathbf{R}\begin{bmatrix}0.06\\0.26\\-1\end{bmatrix} & \begin{bmatrix}-0.15\\-0.25\\1\end{bmatrix} \end{bmatrix}}_{\mathbf{M}} \begin{bmatrix}u_1\\v_1\\u_2\\v_2\\u_3\\v_3\end{bmatrix} = \mathbf{0} \qquad (2)$$

$$\begin{bmatrix} w^2+x^2-y^2-z^2 & 2(xy-wz) & 2(xz+wx) \\ 2(xy+wz) & w^2-x^2+y^2-z^2 & 2(yz-wy) \\ 2(xz-wx) & 2(yz+wy) & w^2-x^2-y^2+z^2 \end{bmatrix}$$

$$w^2+x^2+y^2+z^2 = 1$$

Note that $\mathbf{M}$ is not full rank. The determinant of $\mathbf{M}$ **must** be zero

$$\det(\mathbf{M}) = 0 \Longrightarrow -0.1\,w^4 + 0.4\,w^3x + 5.2\,x^2y^2 + \cdots + 0.3\,z^4 = 0 \qquad (3)$$

The QuEst Algorithm

- Every set of 3 feature points generate a polynomial equation.
- 5 points generate $\binom{5}{3} = 10$ equations.

Example:

$$\underbrace{\left.\begin{aligned} -0.1\,w^4 + 0.4\,w^3x + 5.2\,x^2y^2 + \cdots + \ 0.3\,z^4 &= 0 \\ 1.4\,w^4 + 2.2\,w^3x - 1.7\,x^2y^2 + \cdots + \ 0.6\,z^4 &= 0 \\ &\vdots \\ 2.3\,w^4 + 2.7\,w^3x + 4.3\,x^2y^2 + \cdots - \ 8.1\,z^4 &= 0 \\ w^2 + x^2 + y^2 + z^2 &= 1 \end{aligned}\right\}\text{11 equations}}_{\text{4 unknown variables: } w,x,y,z}$$

- Rotation $q =< w, x, y, x >$ is recovered by solving the polynomial system.
- This is not a simple problem in itself, but multiple ways to do it
  - Relinearization, Grobner Bases
  - Our novel approach (Fathian et al, RAL 2018)

The QuEst Algorithm

- With $\mathbf{q}$ recovered, we recover $\mathbf{R}$ and then can recover $\mathbf{t}$ and depths u_i and $v_i, \forall i$
- Using the rigid body motion equation $u_i\ \mathbf{R}\,\mathbf{m} + \mathbf{t}\ = v_i\ \ \mathbf{n}$

$$\underbrace{\begin{bmatrix} \mathbf{I} & \mathbf{R}\mathbf{m}_1 & -\mathbf{n}_1 & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{R}\mathbf{m}_2 & -\mathbf{n}_2 & & \mathbf{0} & \mathbf{0} \\ \vdots & & \vdots & & & \ddots & & \vdots \\ \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{R}\mathbf{m}_k & -\mathbf{n}_k \end{bmatrix}}_{\mathbf{C} \in \mathbb{R}^{15\times 13} \quad \text{all terms are measured or estimated}} \left.\begin{bmatrix} \mathbf{t} \\ u_1 \\ v_1 \\ u_2 \\ v_2 \\ \vdots \\ u_k \\ v_k \end{bmatrix} = \mathbf{0} \right\} \begin{matrix} \text{13 unknown} \\ \text{3 translation} \\ \text{10 depths} \end{matrix}$$

- Note that 15x13 would generally have no right nullspace

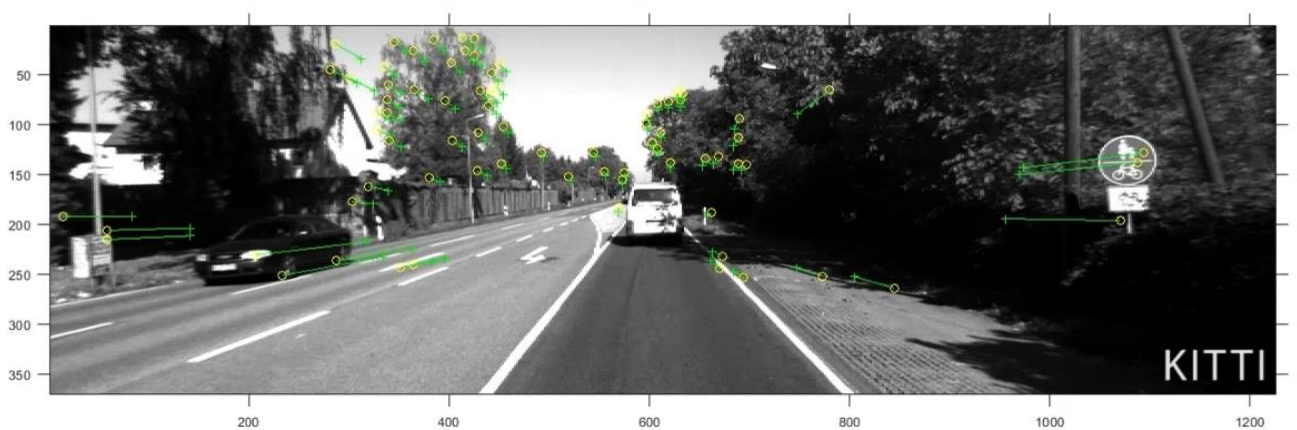
- Take the right singular vector corresponding to the smallest singular value

Benchmarks

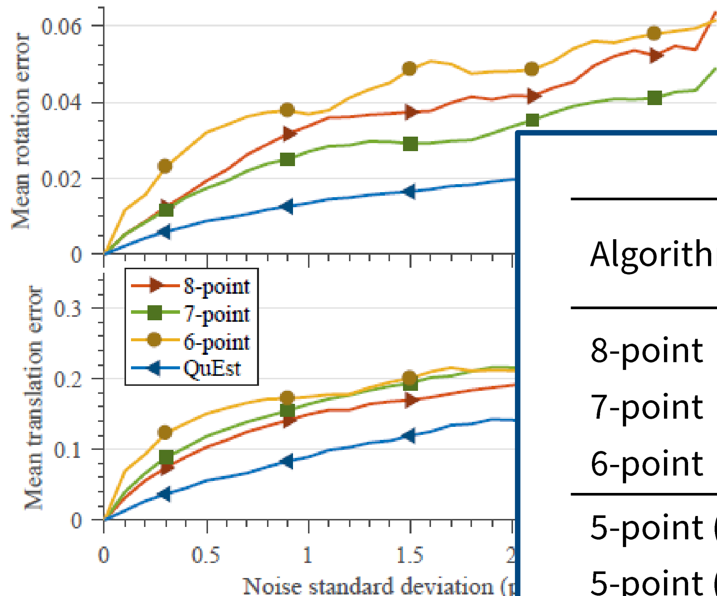
		Rotation error* × 100						Translation error* × 10				
		8-pt	Nister	Kneip	Kukel.	Stew.	**QuEst**	8-pt	Nister	Kukel.	Stew.	**QuEst**
KITTI	Med	0.4697	0.3197	0.898	0.3197	0.2933	**0.2155**	0.0669	2.4635	2.4635	0.084	**0.0596**
	Q1	0.2378	0.1801	0.4321	0.1801	0.1724	**0.1318**	0.0383	1.1119	1.1119	0.0443	**0.0358**
	Q3	0.9129	0.647	1.906	0.647	0.4913	**0.3452**	0.1256	4.09	4.09	0.1899	**0.1063**
TUM	Med	1.8289	1.3069	2.0601	1.3069	1.0415	**1.0099**	2.3068	2.9565	2.9565	2.1682	**1.455**
	Q1	1.0999	0.8096	1.2689	0.8096	0.6559	**0.6371**	1.2453	1.5268	1.5268	1.0206	**0.7945**
	Q3	3.012	2.1642	3.3846	2.1642	1.6797	**1.5916**	3.4994	4.083	4.083	3.4727	**2.5377**
ICL	Med	1.0092	0.8155	1.1879	0.8155	0.6545	**0.6202**	2.2591	2.5963	2.5963	2.0429	**1.4507**
	Q1	0.6093	0.5135	0.7424	0.5135	0.4049	**0.3937**	1.3115	1.3794	1.3794	1.0151	**0.7085**
	Q3	1.6024	1.2928	1.9744	1.2928	0.9601	**0.9356**	3.3328	3.7566	3.7566	3.043	**2.3593**
NAIST	Med	0.8191	1.0008	0.8487	1.0008	0.748	**0.4759**	0.2827	2.3717	2.3717	0.5727	**0.27**
	Q1	0.4885	0.4454	0.5012	0.4454	0.3941	**0.2607**	0.1773	1.0819	1.0819	0.2681	**0.1555**
	Q3	1.3768	2.0418	1.6832	2.0418	1.2932	**0.8338**	**0.4212**	4.0515	4.0515	1.4121	0.4614

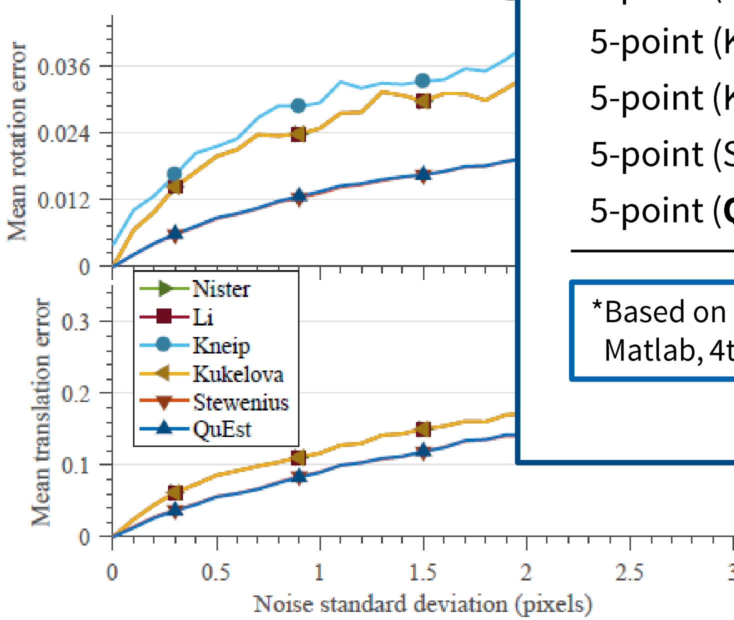
*Error metric: $\rho(q,\bar{q}) \coloneqq \int \frac{1}{\pi}\arccos(\langle q,\bar{q}\rangle) \in [0,1]$

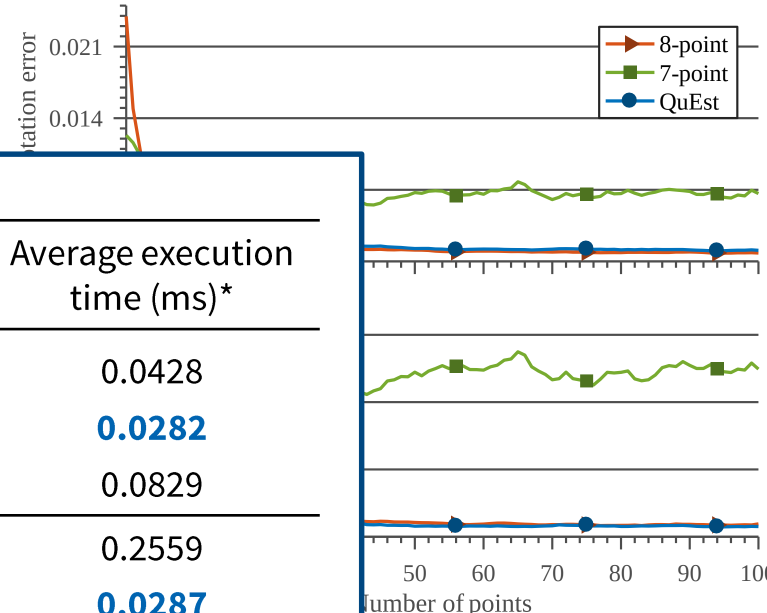
*Error metric: $\rho(t,\bar{t}) \coloneqq \int \frac{1}{\pi}\arccos\left(\left\langle \frac{t}{||t||}, \frac{\bar{t}}{||\bar{t}||}\right\rangle\right) \in [0,1]$

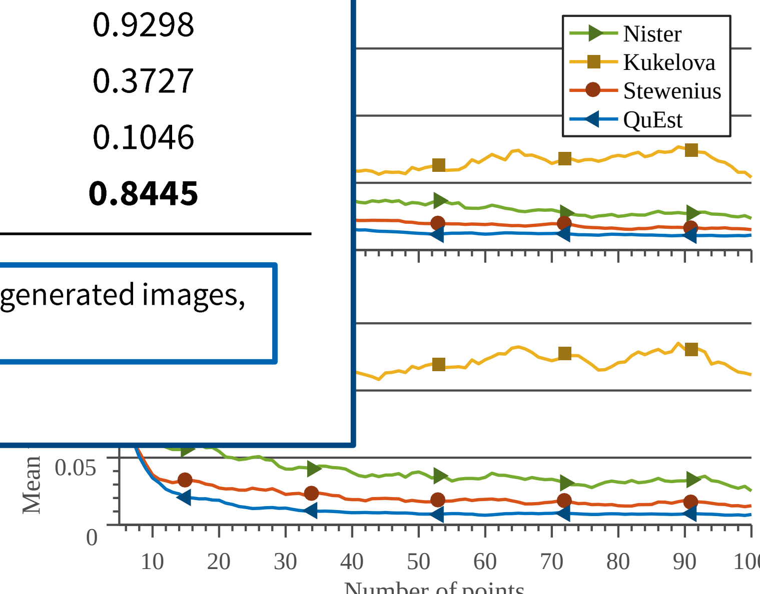


Benchmarks









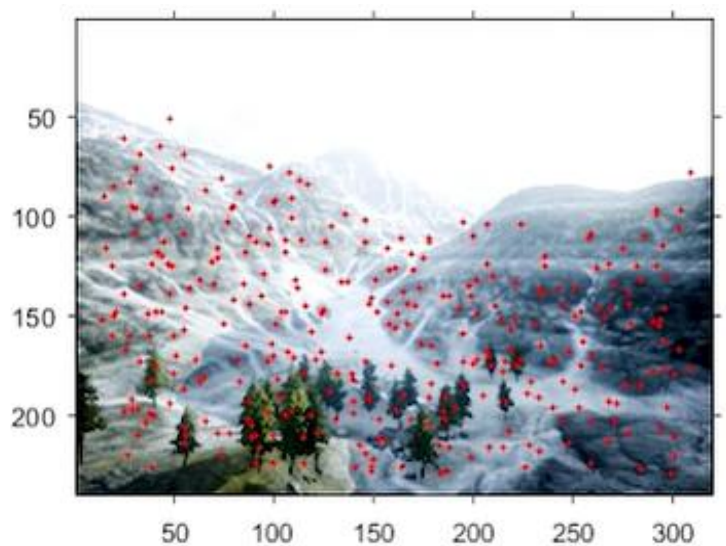
Algorithm	Average execution time (ms)*
8-point	0.0428
7-point	**0.0282**
6-point	0.0829
5-point (Nister)	0.2559
5-point (Li)	**0.0287**
5-point (Kneip)	0.9298
5-point (Kukelova)	0.3727
5-point (Stewenius)	0.1046
5-point (**QuEst**)	**0.8445**

*Based on 1000 randomly generated images, Matlab, 4th Gen i-7 CPU

QuEst in Vision Based Control

Our approach:

Step 1: UAVs extract feature points of onboard camera images



QuEst in Vision Based Control

Step 2: Feature point *data* are communicated to neighboring UAVs

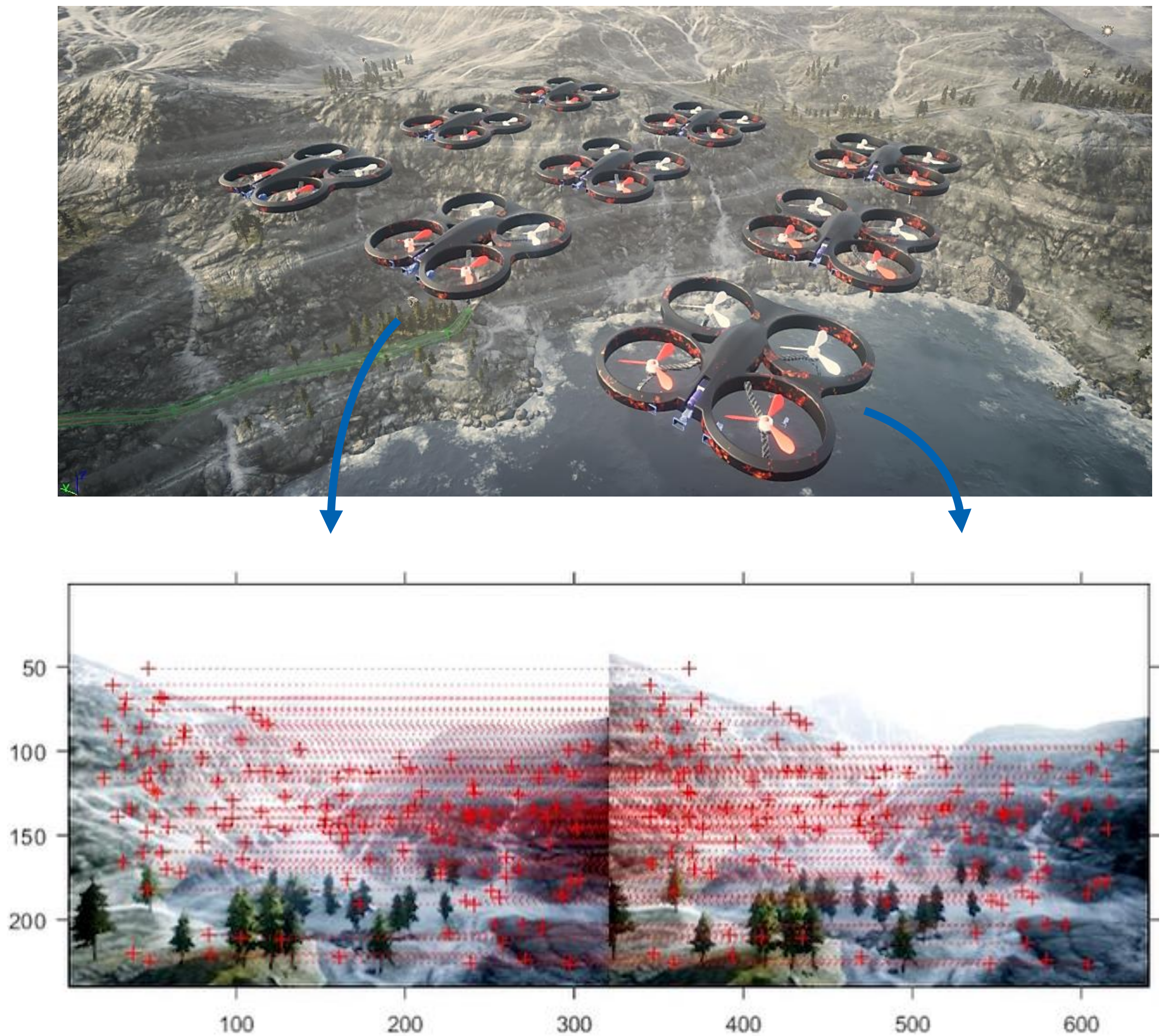
Communication bandwidth for each neighbor*: **5.2 KB/s**

*Based on

Pixel coordinates:	2 single precision numbers
Descriptor:	64 single precision numbers
Number of features:	500 pts
Feature extraction time:	50ms

QuEst in Vision Based Control

Step 3: Points are matched and used in QuEst to estimate the relative pose



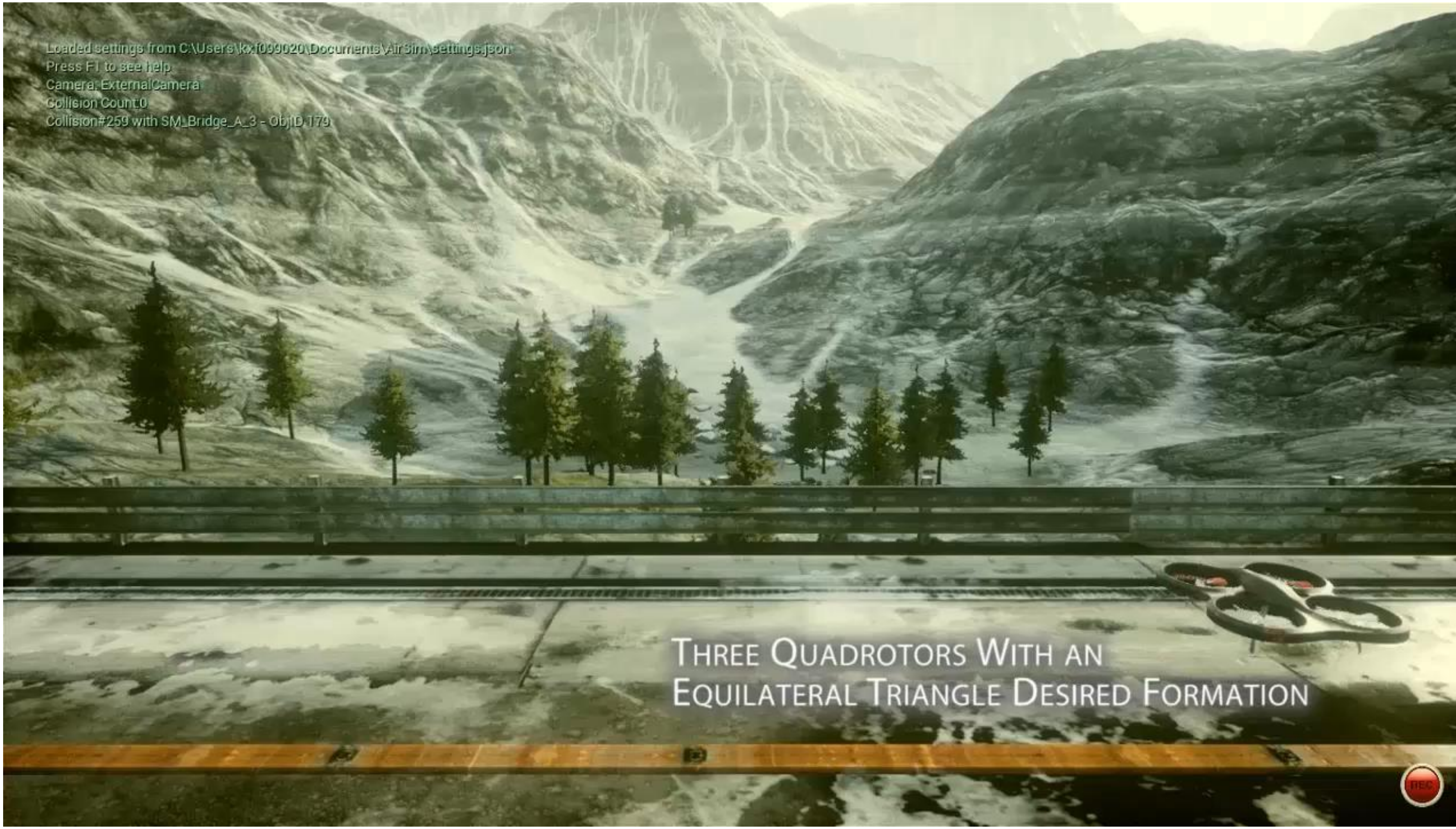
QuEst in Vision Based Control

Step 4: Recovered pose is used in the formation control strategy

Notes:

- Scaled translation does not affect the convergence

Simulation Results



The VEst Algorithm

Can we perform the same procedure based on the continuous rigid body motion equations to estimate angular and linear velocity?

$$v\, \boldsymbol{\omega}_{\times}\, \mathbf{n} + \mathbf{v} = v\, \dot{\mathbf{n}} + \dot{v}\, \mathbf{n}$$

Yes!

In fact it is a bit simpler since $\boldsymbol{\omega}$ has only 3 unknowns

The steps are exactly the same…

The VEst Algorithm

[Zhang et al., ACC 2020]

Step 1: Subtract constraints

$$v\,\boldsymbol{\omega}_\times \mathbf{n} + \mathbf{v} = v\,\dot{\mathbf{n}} + \dot{v}\,\mathbf{n}$$

for any 2 feature points to eliminate $\mathbf{v}$.

Example:

$$\left.\begin{aligned} u_2\boldsymbol{\omega}_\times \begin{bmatrix}-0.6535\\-0.0010\\1\end{bmatrix} + \mathbf{v} = u_2\begin{bmatrix}-0.0193\\-0.0041\\0\end{bmatrix} + \dot{u}_2\begin{bmatrix}-0.6535\\-0.0010\\1\end{bmatrix} \\ u_3\boldsymbol{\omega}_\times \begin{bmatrix}-0.1230\\-0.0561\\1\end{bmatrix} + \mathbf{v} = u_3\begin{bmatrix}0\\0.0020\\0\end{bmatrix} + \dot{u}_3\begin{bmatrix}-0.1230\\-0.0561\\1\end{bmatrix}\end{aligned}\right\} \xRightarrow{\text{subtract}}$$

$$\begin{aligned}&\left[\boldsymbol{\omega}_\times\begin{bmatrix}-0.4177\\-0.0775\\1\end{bmatrix} - \begin{bmatrix}-0.0051\\-0.0010\\0\end{bmatrix}\right]u_1 - \begin{bmatrix}-0.4177\\-0.0775\\0\end{bmatrix}\dot{u}_1 \quad - \\ &\left[\boldsymbol{\omega}_\times\begin{bmatrix}-0.6535\\-0.0010\\1\end{bmatrix} - \begin{bmatrix}-0.0193\\-0.0041\\1\end{bmatrix}\right]u_2 + \begin{bmatrix}-0.6535\\-0.0010\\1\end{bmatrix}\dot{u}_2 \quad = 0\end{aligned}$$

The VEst Algorithm

Step 2: Eliminate depths using 3 feature points

$$v_1\,\boldsymbol{\omega}_\times \boldsymbol{n_1} - v_2\,\boldsymbol{\omega}_\times \boldsymbol{n_2} = v_1\,\dot{\boldsymbol{n}}_1 + \dot{v}_1\boldsymbol{n_1} - v_2\,\dot{\boldsymbol{n}}_2 - \dot{v}_2\boldsymbol{n_2}$$
$$v_1\,\boldsymbol{\omega}_\times \boldsymbol{n_1} - v_3\,\boldsymbol{\omega}_\times \boldsymbol{n_3} = v_1\,\dot{\boldsymbol{n}}_1 + \dot{v}_1\boldsymbol{n_1} - v_3\,\dot{\boldsymbol{n}}_3 - \dot{v}_3\boldsymbol{n_3}$$

Previous operation
For points 1,2 &1,3

Rewrite in matrix form

$$\underbrace{\begin{bmatrix} \boldsymbol{\omega}_\times\boldsymbol{n_1} - \dot{\boldsymbol{n}}_1 & -\boldsymbol{n_1} & -\boldsymbol{\omega}_\times\boldsymbol{n_2} + \dot{\boldsymbol{n}}_2 & -\boldsymbol{n_2} & \mathbf{0} & \mathbf{0} \\ \boldsymbol{\omega}_\times\boldsymbol{n_1} - \dot{\boldsymbol{n}}_1 & -\boldsymbol{n_1} & \mathbf{0} & \mathbf{0} & -\boldsymbol{\omega}_\times\boldsymbol{n_3} + \dot{\boldsymbol{n}}_3 & -\boldsymbol{n_3} \end{bmatrix}}_{\mathbf{M}} \begin{bmatrix} v_1 \\ \dot{v}_1 \\ v_2 \\ \dot{v}_2 \\ v_3 \\ \dot{v}_3 \end{bmatrix} = \mathbf{0}$$

$$\begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}$$

Note that $\mathbf{M}$ is not full rank. The determinant of $\mathbf{M}$ **must** be zero

$$\det(\mathbf{M}) = 0 \Longrightarrow a\,\omega_x^3 + b\,\omega_y^3 + c\,\omega_z^3 + d\,\omega_x^2\omega_y + e\,\omega_x^2\omega_z + \cdots + s\omega_z + t = 0$$

The VEst Algorithm

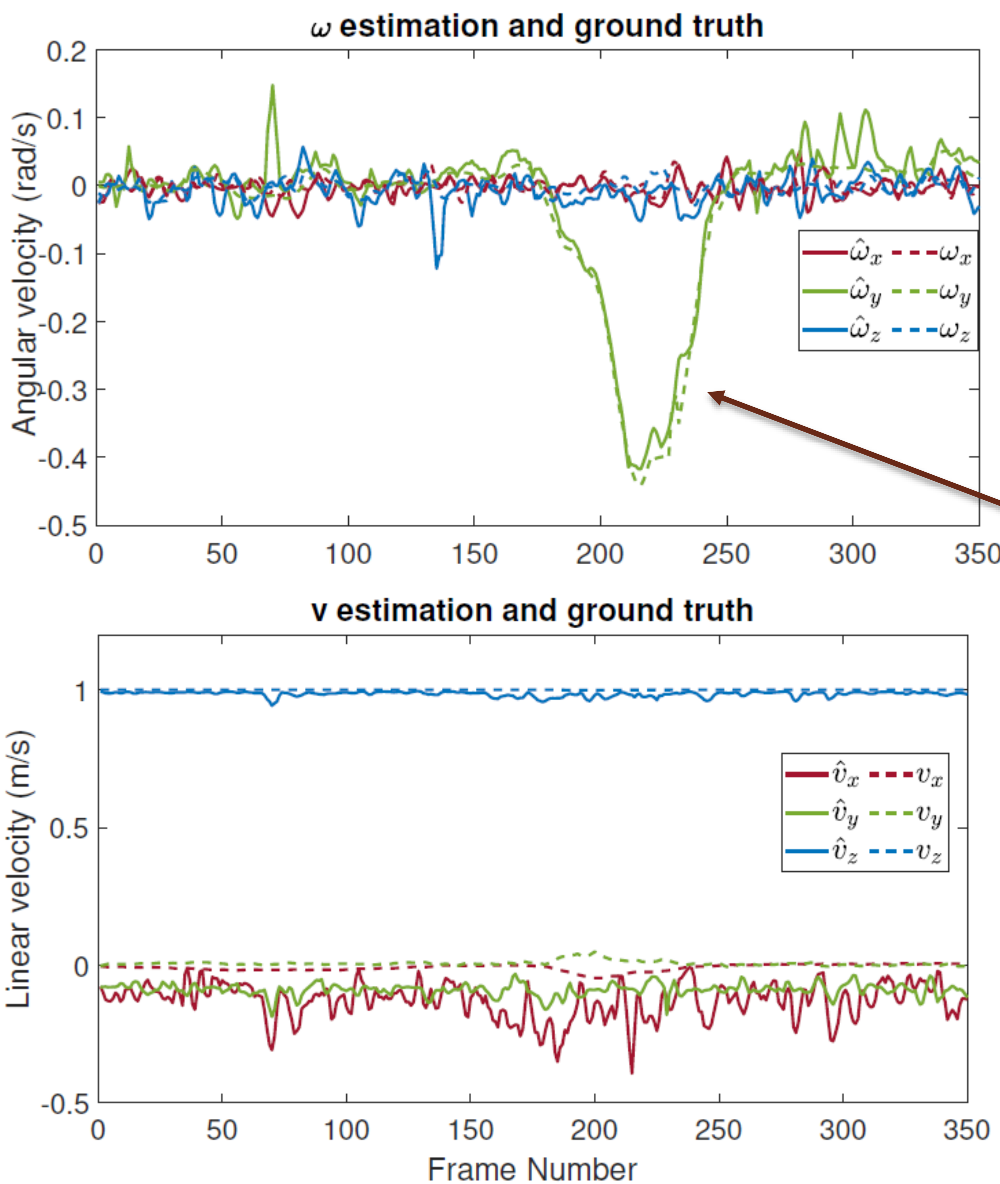
- Every set of 3 feature points generate a polynomial equation.

$$a\,\omega_x^3 + b\,\omega_y^3 + c\,\omega_z^3 + d\,\omega_x^2\omega_y + e\,\omega_x^2\omega_z + \cdots + s\omega_z + t = 0$$

- 5 points generate $\binom{5}{3} = 10$ equations.
- Rotation $\boldsymbol{\omega} = [\omega_x, \omega_y, \omega_z]^\top$ is recovered by solving the polynomial system.
- Can be solved in any number of ways – See ACC 2020 paper to see our method
- From the rigid velocity equation, use SVD to solve for linear velocity $\mathbf{v}$ and depths v_i

$$\begin{bmatrix} \boldsymbol{I} & \boldsymbol{\omega}_\times \boldsymbol{n}_1 - \dot{\boldsymbol{n}}_1 & -\boldsymbol{n}_1 & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \boldsymbol{I} & \mathbf{0} & \mathbf{0} & \boldsymbol{\omega}_\times \boldsymbol{n}_2 - \dot{\boldsymbol{n}}_2 & -\boldsymbol{n}_2 & \cdots & \mathbf{0} & \mathbf{0} \\ \boldsymbol{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \boldsymbol{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \ddots & \mathbf{0} & \mathbf{0} \\ \boldsymbol{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \boldsymbol{\omega}_{skew} \boldsymbol{n}_5 - \dot{\boldsymbol{n}}_5 & -\boldsymbol{n}_5 \end{bmatrix} \begin{bmatrix} \mathbf{v} \\ v_1 \\ \dot{v}_1 \\ v_2 \\ \dot{v}_2 \\ v_3 \\ \dot{v}_3 \\ v_4 \\ \dot{v}_4 \\ v_5 \\ \dot{v}_5 \end{bmatrix} = \mathbf{0}$$

Tests



Tested in the KITTI dataset

Vehicle turns left

QuEst and VEst Fusion

- Solve for $\mathbf{R}$ and $\boldsymbol{\omega}$ independently
- Solve the translation, velocity and depths **together**
- This helps ensure the scale factor between QuEst and Vest is the same

$$\begin{bmatrix} \mathbf{I} & \mathbf{R}\mathbf{m}_1 & -\mathbf{n}_1 & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{R}\mathbf{m}_2 & -\mathbf{n}_2 & & \mathbf{0} & \mathbf{0} \\ \vdots & & \vdots & & & \ddots & & \vdots \\ \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{R}\mathbf{m}_k & -\mathbf{n}_k \end{bmatrix} \begin{bmatrix} \mathbf{t} \\ u_1 \\ v_1 \\ u_2 \\ v_2 \\ \vdots \\ u_k \\ v_k \end{bmatrix} = \mathbf{0}$$

$+$

$$\begin{bmatrix} \mathbf{I} & \boldsymbol{\omega}_{\mathbf{skew}}\mathbf{m}_1 - \dot{\mathbf{m}}_1 & -\mathbf{m}_1 & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{I} & \mathbf{0} & \mathbf{0} & \boldsymbol{\omega}_{\mathbf{skew}}\mathbf{m}_2 - \dot{\mathbf{m}}_2 & -\mathbf{m}_2 & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \ddots & \mathbf{0} & \mathbf{0} \\ \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \boldsymbol{\omega}_{\mathbf{skew}}\mathbf{m}_5 - \dot{\mathbf{m}}_5 & -\mathbf{m}_5 \end{bmatrix} \begin{bmatrix} \mathbf{v} \\ v_1 \\ \dot{v}_1 \\ v_2 \\ \dot{v}_2 \\ v_3 \\ \dot{v}_3 \\ v_4 \\ \dot{v}_4 \\ v_5 \\ \dot{v}_5 \end{bmatrix} = \mathbf{0}$$

Note that $\boldsymbol{v_i}$ are the same in both QuEst and VEst

$=$

$$\underbrace{\begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{R}\mathbf{m}_1 & -\mathbf{n}_1 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{I} & \boldsymbol{\omega}_\times\mathbf{m}_1 - \dot{\mathbf{m}}_1 & \mathbf{0} & -\mathbf{m}_1 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots \\ \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{R}\mathbf{m}_2 & -\mathbf{n}_2 & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{\omega}_\times\mathbf{m}_2 - \dot{\mathbf{m}}_2 & \mathbf{0} & -\mathbf{m}_2 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}}_{30 \times 21} \underbrace{\begin{bmatrix} \mathbf{t} \\ \mathbf{v} \\ u_1 \\ v_1 \\ \dot{v}_1 \\ u_2 \\ v_2 \\ \dot{v}_2 \\ \vdots \end{bmatrix}}_{21} = 0$$

QuEst and VEst Fusion

- $\mathbf{q}$ and $\boldsymbol{\omega}$ should "agree" over time
- $\mathbf{t}$ and $\mathbf{v}$ should also "agree" over time
  - E.g. if you integrate $\mathbf{v}$, you should get $\mathbf{t}$
- We explored the use of extended Kalman filters to enforce ODE constraints between $\mathbf{q}$, $\boldsymbol{\omega}$, $\mathbf{t}$, and $\mathbf{v}$
- Results were mixed
  - Addition / subtraction of quaternions is not a group action
  - Need innovations should be the Lie Algebra
  - Multiplication of rotations is not commutative
  - Non commutative algebra cannot be represented by vectors as in EKF

$$\boldsymbol{e} = \boldsymbol{q}_1 - \boldsymbol{q}_2 \qquad \boldsymbol{e} = \boldsymbol{q}_1 \otimes \boldsymbol{q}_2^{-1}$$

$$\boldsymbol{w} = k\boldsymbol{e} \qquad \boldsymbol{w} = \log \boldsymbol{e}$$

- For systems on Lie groups, the Invariant EKF (IEKF) was introduced in Bonnabel "Left-invariant extended Kalman filter and attitude estimation" 2007 CDC
- A specific IEKF for quaternions (QEKF) was introduced in Bloesch et al. , "State estimation for legged robots consistent fusion of leg kinematics and IMU," Robotics 2013
- Many other interesting related papers in IEKF and QEKF

QuEst and VEst Fusion

- We adapted a method by Rotella et al., "State estimation for a humanoid robot" IROS 2014
- Prediction step - τ is step time.

$$\mathbf{t}_{k+1}^{-} = \mathbf{t}_{k}^{+} + \tau \mathbf{v}_{k}^{+}$$

$$\mathbf{v}_{k+1}^{-} = \mathbf{v}_{k}^{+}$$

$$\boldsymbol{q}_{k+1}^{-} = \exp(\tau {\omega_{\times}}_{k}^{+}) \otimes \boldsymbol{q}_{k}^{+}$$

$\otimes$–quaternion multiplication, exp()-exponential map

- Update step

Innovation /error

$$\boldsymbol{e}_q = \log(\boldsymbol{q}_{quest} \otimes (\boldsymbol{q}_{k+1}^{-})^{-1}),$$

log()-logorithm map

$$\boldsymbol{e}_t = \mathbf{t}_{quest} - \mathbf{t}_{k+1}^{-}, \quad \boldsymbol{e}_v = \mathbf{v}_{vest} - \mathbf{v}_{k+1}^{-}, \quad \boldsymbol{e}_\omega = \boldsymbol{\omega}_{vest} - \boldsymbol{\omega}_{k+1}^{-}$$

$$\boldsymbol{e} = \left[\boldsymbol{e}_q^T, \ \boldsymbol{e}_t^T, \ \boldsymbol{e}_v^T, \ \boldsymbol{e}_w^T\right]^T$$

correction

$$\Delta \boldsymbol{x} = \boldsymbol{K}\boldsymbol{e}$$

Kalman gain $\boldsymbol{K}$ is calculated the usual way. Linearized state update and measurement matrices are rather complicated

$$\mathbf{t}_{k+1}^{+} = \mathbf{t}_{k+1}^{-} + \Delta \boldsymbol{x}_{\mathrm{t}}, \quad \mathbf{v}_{k+1}^{+} = \mathbf{v}_{k+1}^{-} + \Delta \boldsymbol{x}_{\mathrm{v}}, \quad \boldsymbol{\omega}_{k+1}^{+} = \boldsymbol{\omega}_{k+1}^{-} + \Delta \boldsymbol{x}_{\omega},$$

$$\boldsymbol{q}_{k+1}^{+} = \exp(\Delta\phi) \otimes \boldsymbol{q}_{k}^{+}$$

$\Delta\phi$ is a function of $\boldsymbol{e}_q$

Results

Papers and Code

- K. Fathian, J. Jin, S. Wee, D. Lee, Y. Kim, N. R. Gans, “Camera Relative Pose Estimation for Visual Servoing using Quaternions” Elsevier Robotics and Autonomous Systems, 2018.

- K. Fathian, J. P. Ramirez, E. A. Doucette, J. W. Curtis, N. R. Gans., “QuEst: A Quaternion-Based Approach for Camera Motion Estimation from Minimal Feature Points,” IEEE Robotics and Automation Letters

- Y. Zhang, K. Fathian, N.R. Gans, “An Efficient Solution to the Camera Velocity Estimationfrom Minimal Feature Points,” Proc. American Controls Conference, 2020, Paper ThC12.6, 17:40-18:00

- https://sites.google.com/view/kavehfathian/code
  - MATLAB code available
  - C++ code exists and being standardized to OpenMVG
  - ROS packages being written