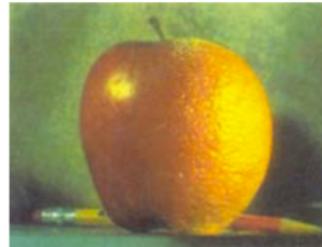


2. Image Formation



3. Image Processing



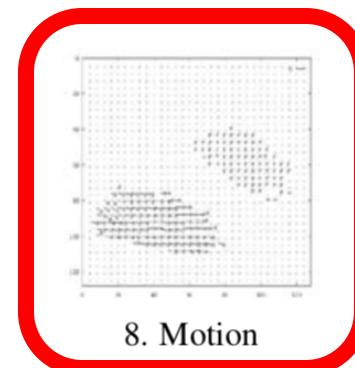
4. Features



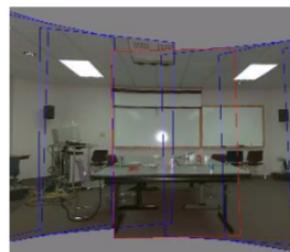
5. Segmentation



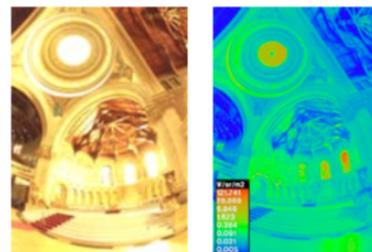
6-7. Structure from Motion



8. Motion



9. Stitching



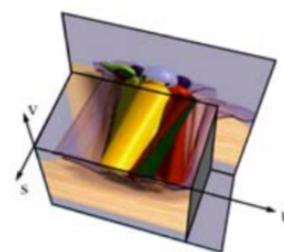
10. Computational Photography



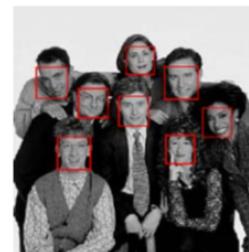
11. Stereo



12. 3D Shape



13. Image-based Rendering



14. Recognition

Credits

- Images and formulas from Szeliski
- Second half from CVPR talk by Zhaoyang Lv

Taking a Deeper Look at the Inverse Compositional Algorithm, Zhaoyang Lv , Frank Dellaert, James M. Rehg, Andreas Geiger, CVPR 2019

Motivating problem: Video Stabilization

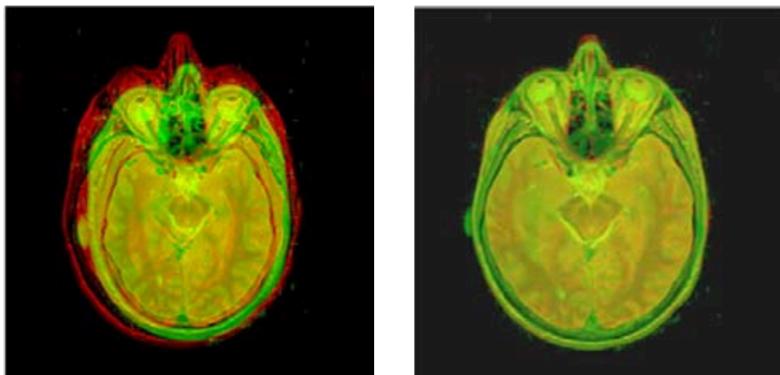


Original

Stabilized



Dense Motion Estimation



- Widely used!
 - Aligning images
 - Motion in video
 - Video Stabilization
- We need:
 - Error metric
 - Search technique
 - Full search
 - Hierarchical
 - Incremental

Outline

- Error metric/full search
- Hierarchical search
- Incremental refinement
 - Parametric Motion
- Deep Learning Approach (CVPR19)

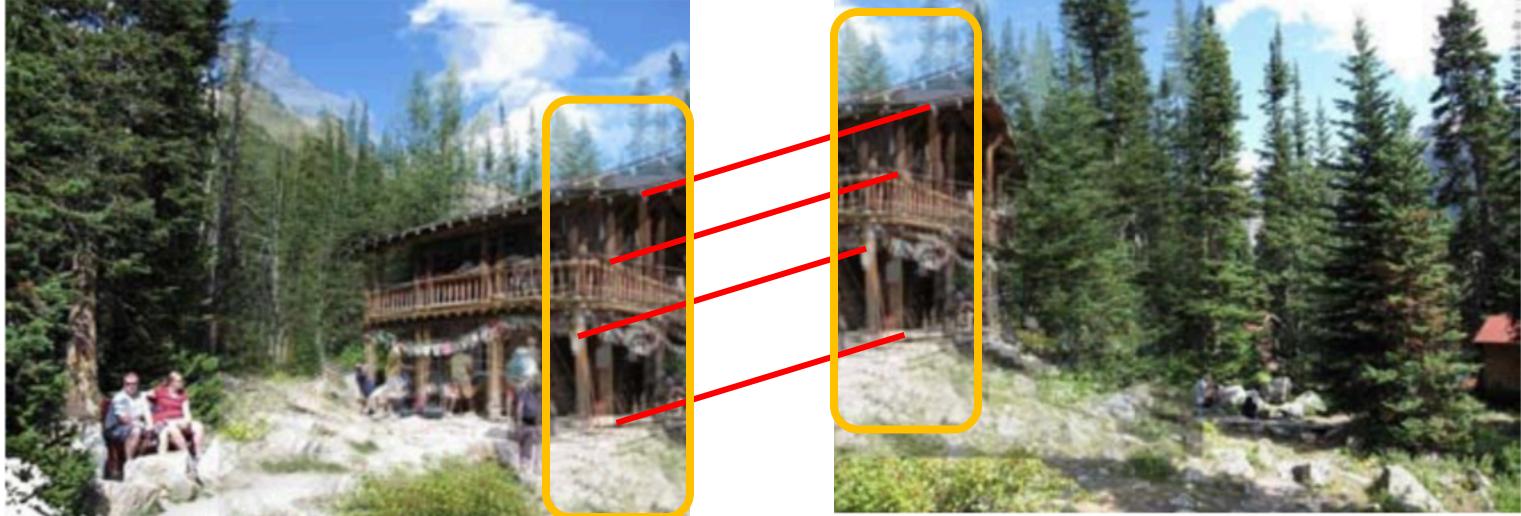
Translational Alignment



- Shift image I_1 with respect to template I_0
- Before, **feature-based** error:

$$E_{\text{LS}} = \sum_i \|r_i\|^2 = \sum_i \|\mathbf{f}(\mathbf{x}_i; \mathbf{p}) - \mathbf{x}'_i\|^2,$$

Translational Alignment



- Shift image I_1 with respect to template I_0
- Before, **feature-based** error:

$$E_{\text{LS}} = \sum \|r_i\|^2 = \sum \|f(x_i; p) - x'_i\|^2,$$

- Now, **image-based** error:

$$E_{\text{SSD}}(\mathbf{u}) = \sum_i [I_1(\mathbf{x}_i + \mathbf{u}) - I_0(\mathbf{x}_i)]^2 = \sum_i e_i^2,$$

SSD



$$E_{\text{SSD}}(\mathbf{u}) = \sum_i [I_1(\mathbf{x}_i + \mathbf{u}) - I_0(\mathbf{x}_i)]^2 = \sum_i e_i^2,$$

- Sum of Squared Differences
- Assumes: brightness constancy
- If \mathbf{u} fractional: interpolation needed
 - Bilinear (fast, good)
 - Bicubic (slower, slightly better)

Robust Error Metrics



$$E_{\text{SAD}}(\mathbf{u}) = \sum_i |I_1(\mathbf{x}_i + \mathbf{u}) - I_0(\mathbf{x}_i)| = \sum_i |e_i|.$$

- Quadratic error is unforgiving!
- Absolute error (SAD): allows for outliers
- Differentiable robust error metrics exist

Dealing with Boundary Conditions

- Should not count pixels outside
- Add two “window” functions
- Windowed SSD metric:

$$E_{\text{WSSD}}(\mathbf{u}) = \sum_i w_0(\mathbf{x}_i)w_1(\mathbf{x}_i + \mathbf{u})[I_1(\mathbf{x}_i + \mathbf{u}) - I_0(\mathbf{x}_i)]^2,$$

- Invariant to overlap: root mean square:

$$A = \sum_i w_0(\mathbf{x}_i)w_1(\mathbf{x}_i + \mathbf{u}) \quad RMS = \sqrt{E_{\text{WSSD}}/A}$$

Violations of Brightness Constancy

- Estimate Bias and Gain

$$I_1(\mathbf{x} + \mathbf{u}) = (1 + \alpha)I_0(\mathbf{x}) + \beta,$$

$$E_{\text{BG}}(\mathbf{u}) = \sum_i [I_1(\mathbf{x}_i + \mathbf{u}) - (1 + \alpha)I_0(\mathbf{x}_i) - \beta]^2$$

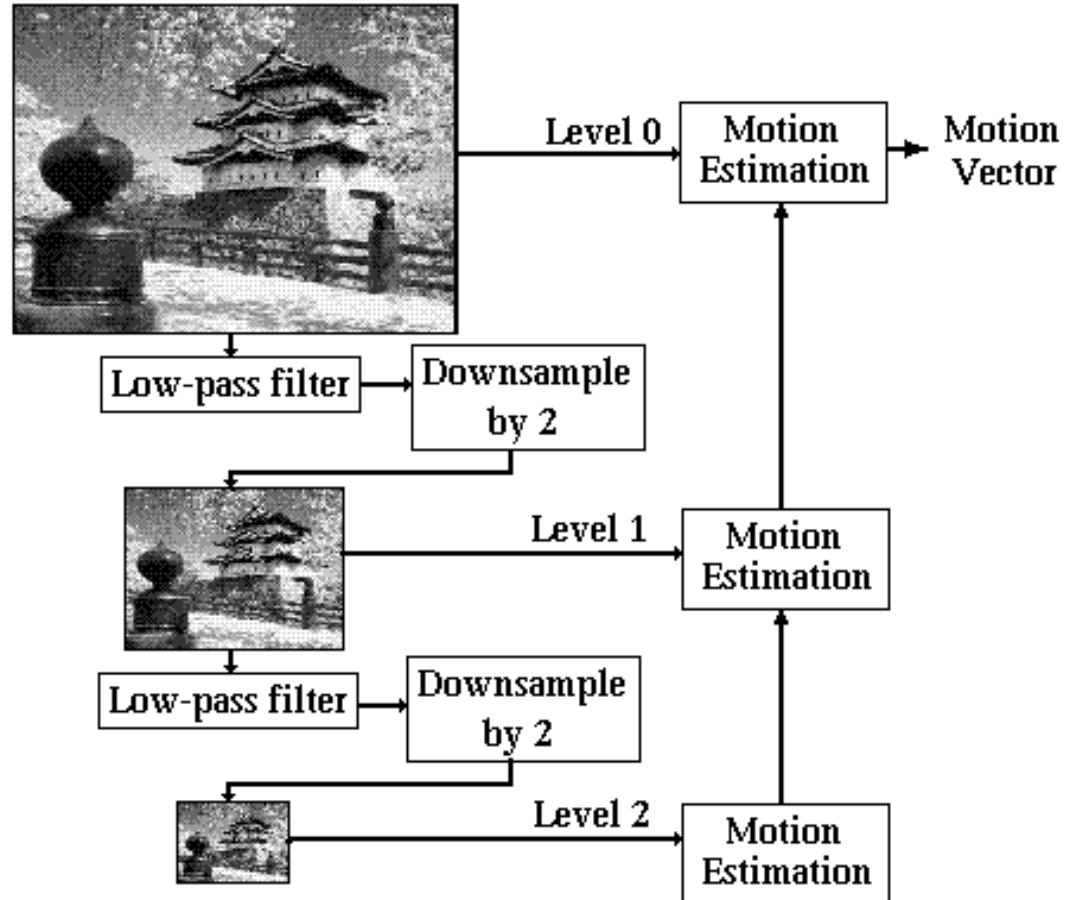
- Normalized Cross-Correlation

$$E_{\text{CC}}(\mathbf{u}) = \sum_i I_0(\mathbf{x}_i)I_1(\mathbf{x}_i + \mathbf{u}).$$

$$E_{\text{NCC}}(\mathbf{u}) = \frac{\sum_i [I_0(\mathbf{x}_i) - \bar{I}_0] [I_1(\mathbf{x}_i + \mathbf{u}) - \bar{I}_1]}{\sqrt{\sum_i [I_0(\mathbf{x}_i) - \bar{I}_0]^2} \sqrt{\sum_i [I_1(\mathbf{x}_i + \mathbf{u}) - \bar{I}_1]^2}}$$

Hierarchical Motion Estimation

- Build an image pyramid:
 - Low-pass
 - Decimate
- Recursively estimate motion:
 - Estimate motion at highest level
 - Use result as initial estimate at lower level



Sub-pixel Refinement

- Taylor expansion of SSD in sub-pixel update $\Delta\mathbf{u}$:

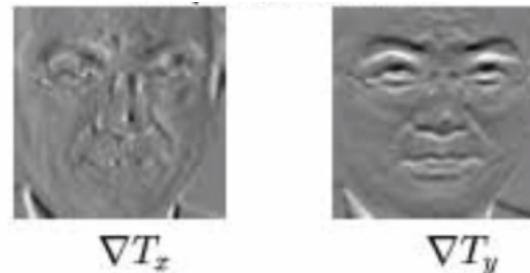
$$E_{\text{LK-SSD}}(\mathbf{u} + \Delta\mathbf{u}) = \sum_i [I_1(\mathbf{x}_i + \mathbf{u} + \Delta\mathbf{u}) - I_0(\mathbf{x}_i)]^2 \quad (8.33)$$

$$\approx \sum_i [I_1(\mathbf{x}_i + \mathbf{u}) + \mathbf{J}_1(\mathbf{x}_i + \mathbf{u})\Delta\mathbf{u} - I_0(\mathbf{x}_i)]^2 \quad (8.34)$$

$$= \sum_i [\mathbf{J}_1(\mathbf{x}_i + \mathbf{u})\Delta\mathbf{u} + e_i]^2, \quad (8.35)$$

where \mathbf{J} is the Jacobian, i.e., gradients at $\mathbf{x}_i + \mathbf{u}$:

$$\mathbf{J}_1(\mathbf{x}_i + \mathbf{u}) = \nabla I_1(\mathbf{x}_i + \mathbf{u}) = \left(\frac{\partial I_1}{\partial x}, \frac{\partial I_1}{\partial y} \right)(\mathbf{x}_i + \mathbf{u}) \quad (8.36)$$

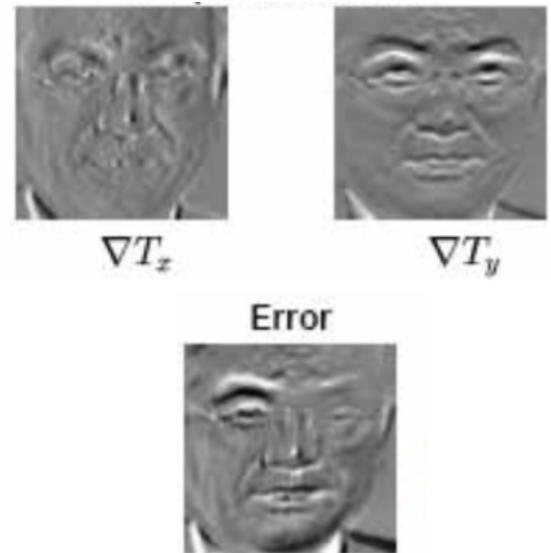


Solve using Normal Equations

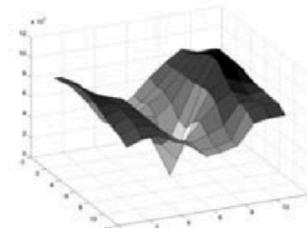
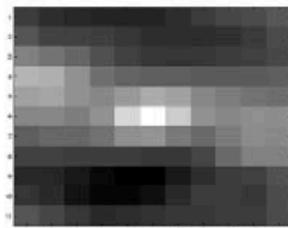
$$\mathbf{A} \Delta \mathbf{u} = \mathbf{b}$$

$$\mathbf{A} = \sum_i \mathbf{J}_1^T(\mathbf{x}_i + \mathbf{u}) \mathbf{J}_1(\mathbf{x}_i + \mathbf{u}) \quad \mathbf{b} = - \sum_i e_i \mathbf{J}_1^T(\mathbf{x}_i + \mathbf{u})$$

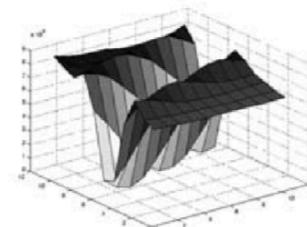
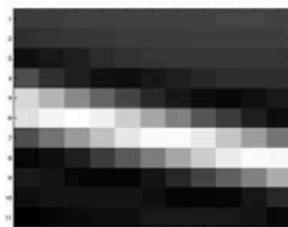
- A is Hessian or "information matrix", same as Harris uses!
- RHS b is just dot product of gradient images with error ->
- Remember: feature-based translation: just mean of flow vectors !



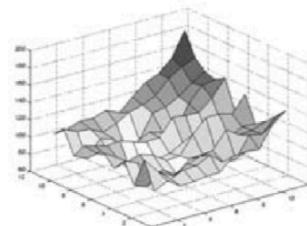
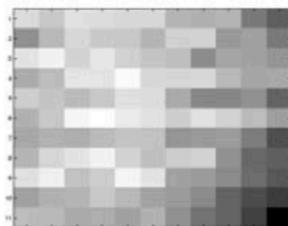
Aperture Problems and Harris



(a)



(b)



(c)

Revisiting Video Stabilization



Motion Models: Translation



- * Translation in x and y
- * 2 DOF
- * Still very shaky



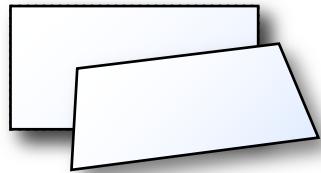
Motion Models: Similarity



- * Translation in x and y
- * Uniform scale and rotation
- * 4 DOF
- * Not shaky, but wobbly



Motion Models: Homography

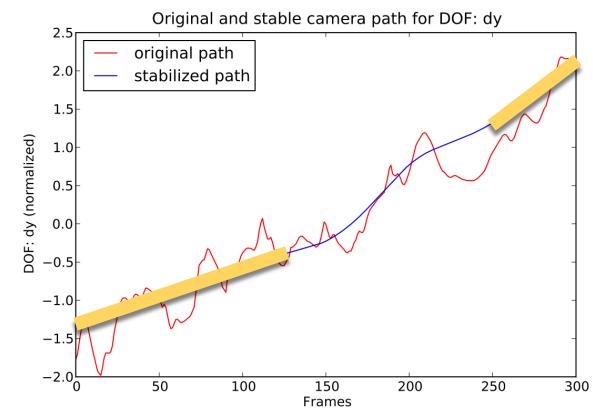
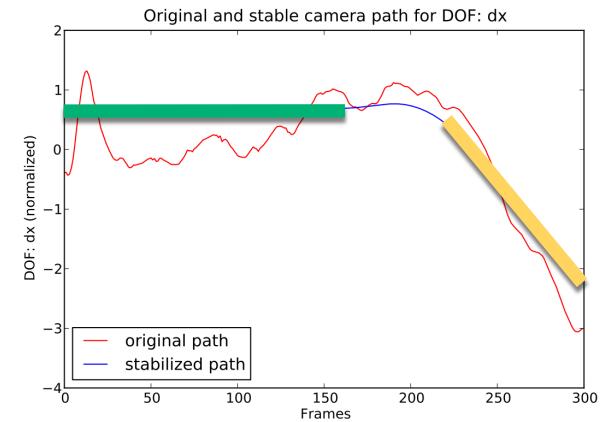


- * Translation in x and y, scale and rotation
- * Skew and perspective
- * 8 DOF
- * Stable



Path Smoothing

- * Goal: Approximate original path with stable one
- * Cinematography inspired:
Properties of a stable path?
- * Tripod → Constant segment
- * Dolly or pan → Linear segment
- * Ease in and out transitions
→ Parabolic segment



Parametric Motion

Template T



Image I

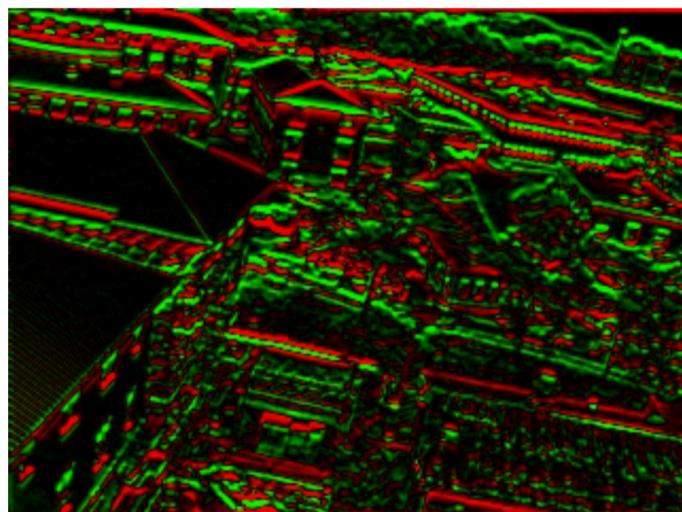
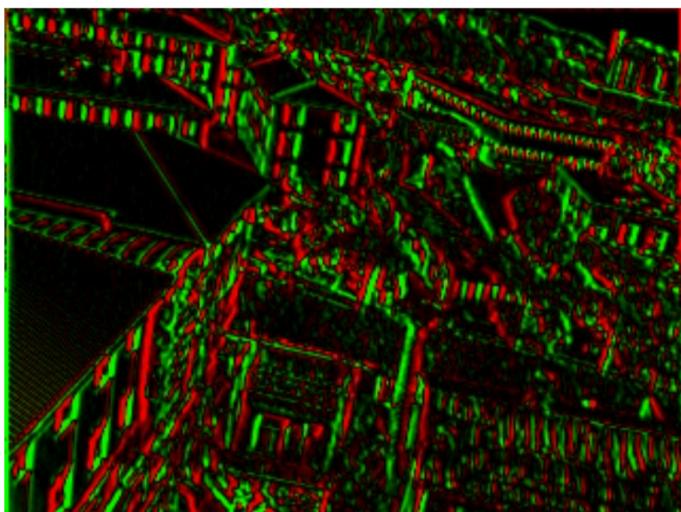
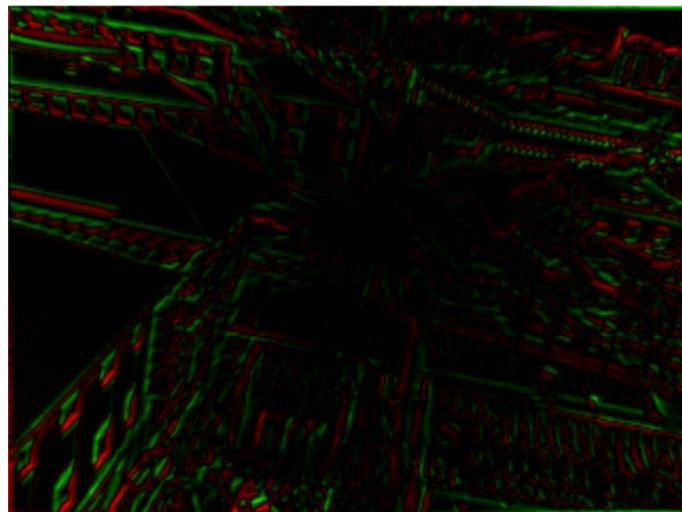


$$\mathbf{x}'(\mathbf{x}; \mathbf{p})$$

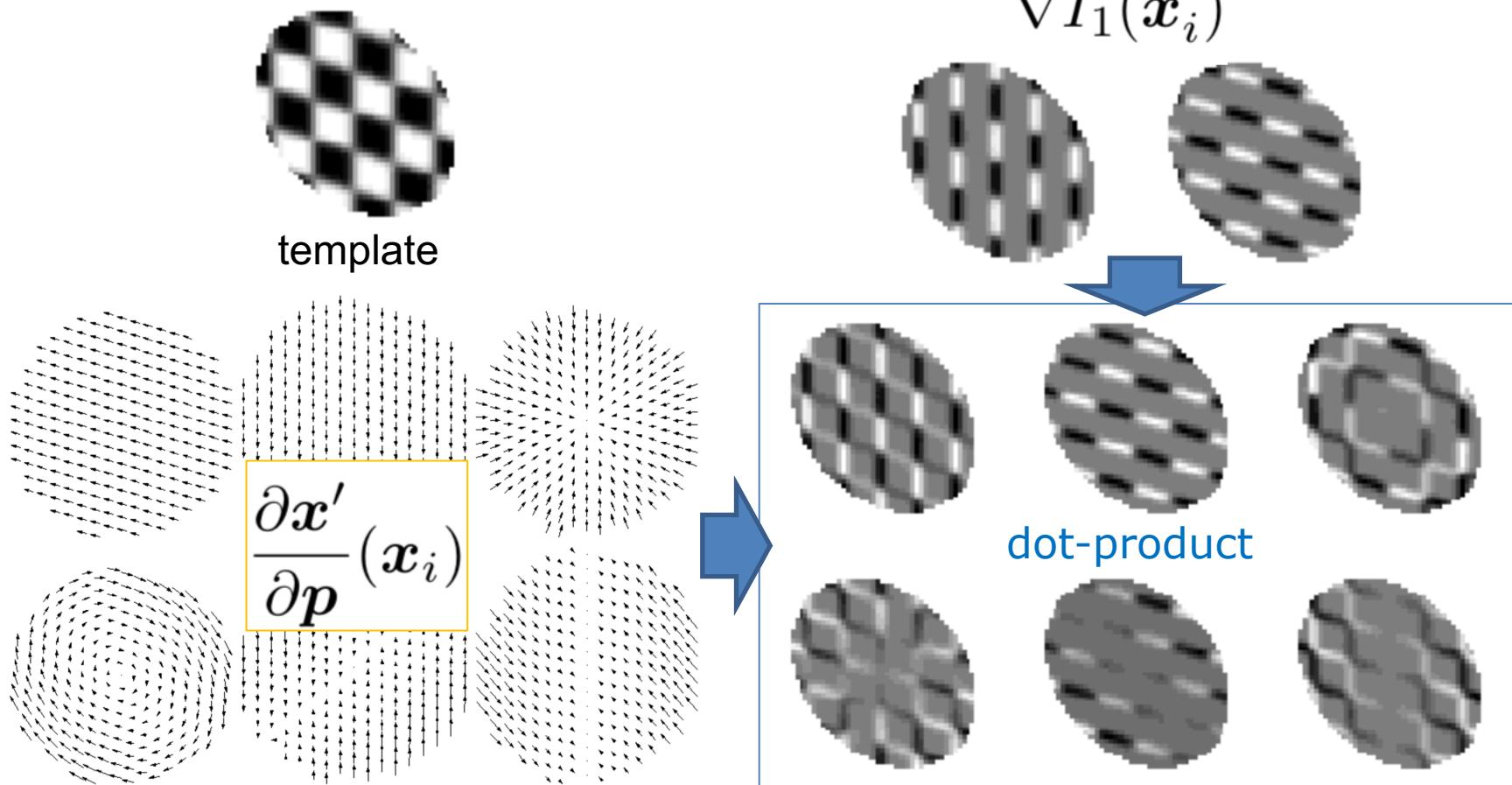
- E.g., image-based homography estimation

$$\begin{aligned} E_{\text{LK-PM}}(\mathbf{p} + \Delta\mathbf{p}) &= \sum_i [I_1(\mathbf{x}'(\mathbf{x}_i; \mathbf{p} + \Delta\mathbf{p})) - I_0(\mathbf{x}_i)]^2 \\ &\approx \sum_i [I_1(\mathbf{x}'_i) + \mathbf{J}_1(\mathbf{x}'_i)\Delta\mathbf{p} - I_0(\mathbf{x}_i)]^2 \end{aligned}$$

"Jacobian Images"



Computing Jacobian Images



$$\mathbf{J}_1(\mathbf{x}'_i) = \frac{\partial I_1}{\partial \mathbf{p}} = \nabla I_1(\mathbf{x}'_i) \frac{\partial \mathbf{x}'}{\partial \mathbf{p}}(\mathbf{x}_i), \quad (8.52)$$

Compositional and Inverse Compositional

- Compare three variants:

- Original:
$$\sum_i [I_1(\mathbf{x}'(\mathbf{x}_i; \mathbf{p} + \Delta\mathbf{p})) - I_0(\mathbf{x}_i)]^2 \quad (8.49)$$

- Compositional:
$$\sum_i [\tilde{I}_1(\tilde{\mathbf{x}}(\mathbf{x}_i; \Delta\mathbf{p})) - I_0(\mathbf{x}_i)]^2 \quad (8.60)$$

- Inverse Comp:
$$\sum_i [\tilde{I}_1(\mathbf{x}_i) - I_0(\tilde{\mathbf{x}}(\mathbf{x}_i; \Delta\mathbf{p}))]^2 \quad (8.64)$$

- In compositional approach we *warp* the image I_1 and solve for an incremental update.
- Inverse compositional: search for incremental update to template instead
 - Jacobians and Hessian can now be *precomputed*

The Inverse Compositional Algorithm

[S. Baker and I. Matthews, 04]

$$\mathbf{r}_k(\mathbf{0}) = \mathbf{I}(\boldsymbol{\xi}_k) - \mathbf{T}(\mathbf{0})$$



$$\Delta\boldsymbol{\xi} = (\mathbf{J}^T \mathbf{W} \mathbf{J} + \lambda \text{diag}(\mathbf{J}^T \mathbf{W} \mathbf{J}))^{-1} \mathbf{J}^T \mathbf{W} \mathbf{r}_k(\mathbf{0})$$

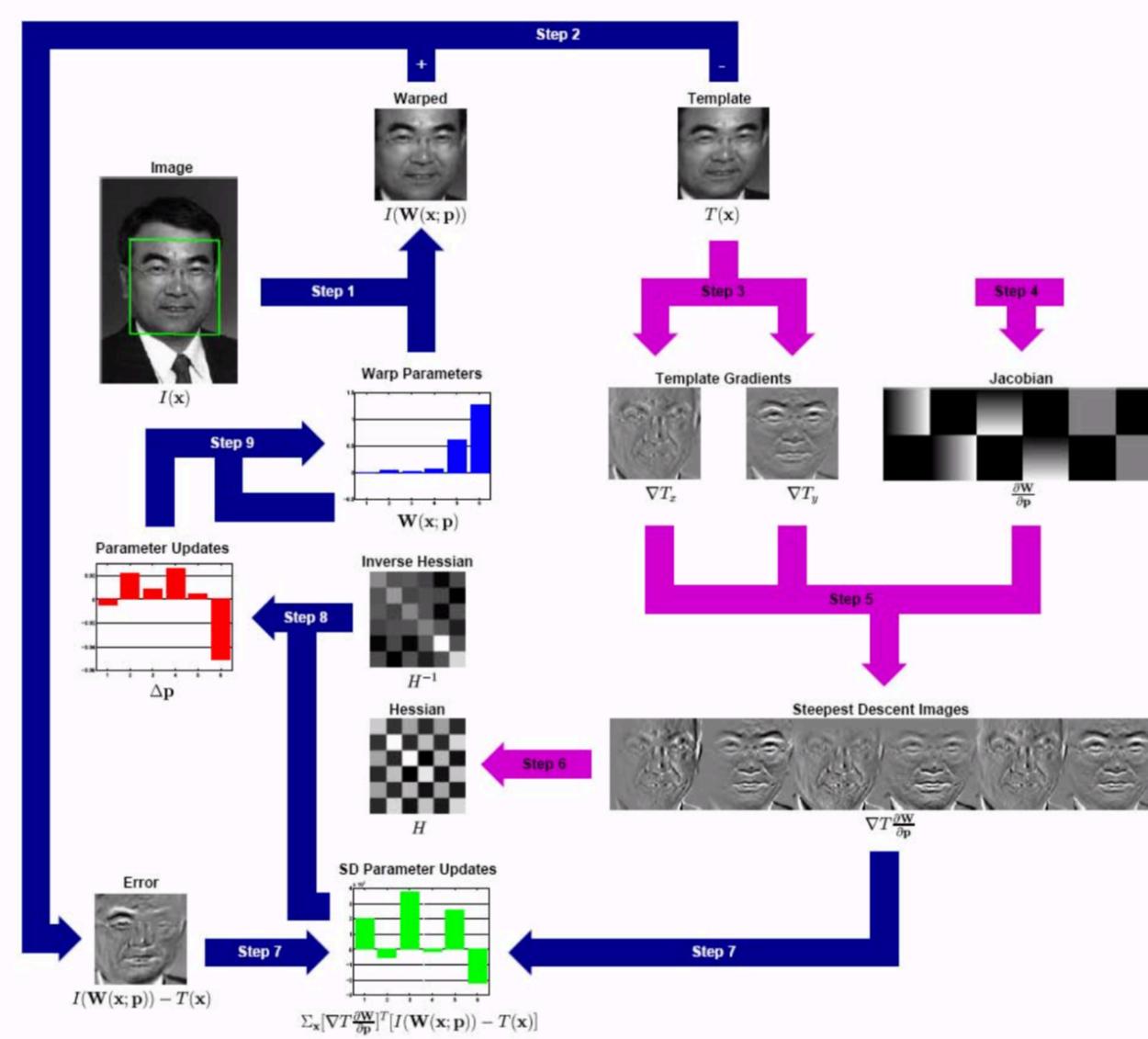


$$\boldsymbol{\xi}_{k+1} = \boldsymbol{\xi}_k \circ (\Delta\boldsymbol{\xi})^{-1}$$

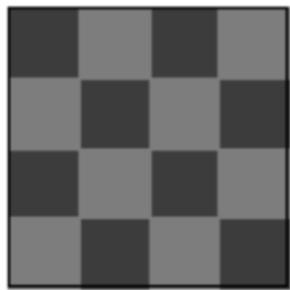
\mathbf{W} Weight matrix

$\lambda \text{diag}(\mathbf{J}^T \mathbf{W} \mathbf{J})$ Damping: very frequently used in non-linear optimization to make sure gradients are valid;
“Levenberg-Marquardt”

Inverse Compositional Approach



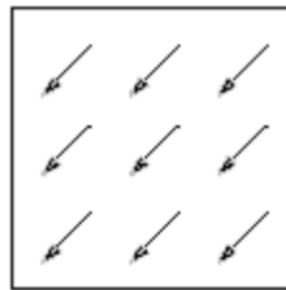
Layered Motion



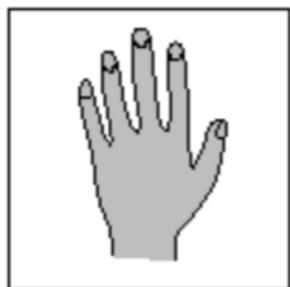
Intensity map



Alpha map



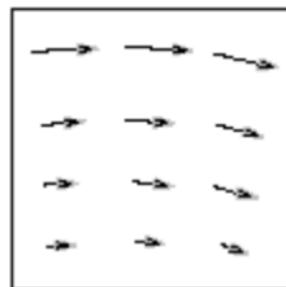
Velocity map



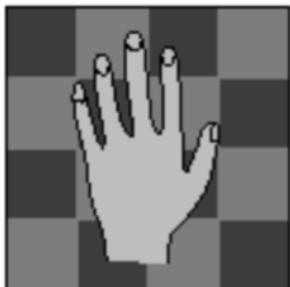
Intensity map



Alpha map



Velocity map



Frame 1



Frame 2



Frame 3

- One type of assumption to “regularize” optical flow
- Estimate FG and BG layers

Layered Motion Results



(a)



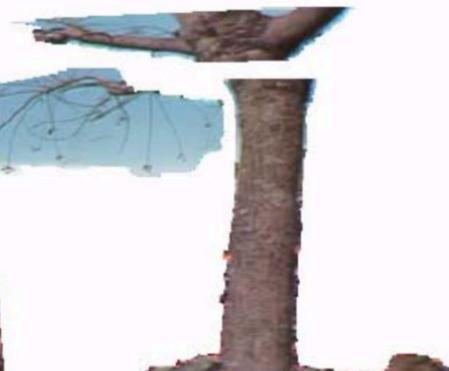
(b)



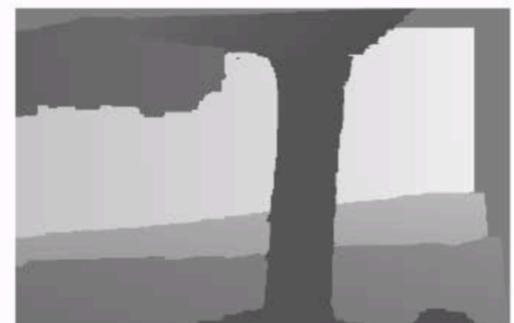
(c)



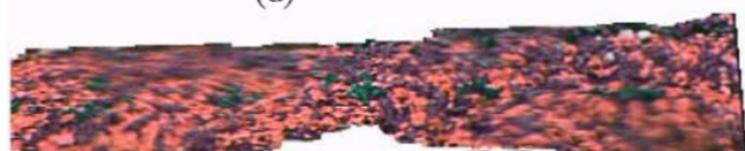
(d)



(e)



(f)



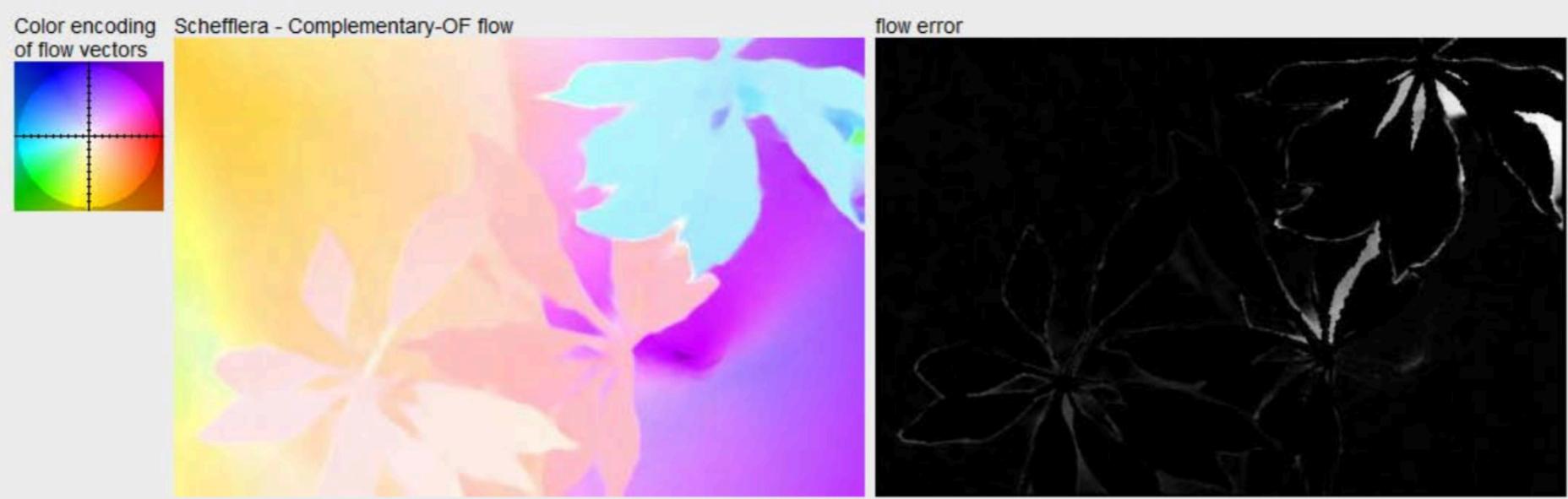
(g)



(h)

Baker, Szeliski, and Anandan 1998

Optical Flow: fully non-parametric



- Fully non-parametric model of motion
- N pixels $\rightarrow N$ flow vectors $\rightarrow 2N$ parameters
- Need some smoothness assumptions!
- Hard to deal with occlusion

Taking a Deeper Look at the Inverse Compositional Algorithm

Zhaoyang Lv¹, Frank Dellaert¹, James M.
Rehg¹, Andreas Geiger²

¹Georgia Institute of Technology

²Autonomous Vision Group, MPI-IS and
University of Tübingen



The Inverse Compositional Algorithm

[S. Baker and I. Matthews, 04]

$$\rightarrow \mathbf{r}_k(\mathbf{0}) = \mathbf{I}(\boldsymbol{\xi}_k) - \mathbf{T}(\mathbf{0})$$



$$\Delta \boldsymbol{\xi} = (\mathbf{J}^T \mathbf{W} \mathbf{J} + \lambda \text{diag}(\mathbf{J}^T \mathbf{W} \mathbf{J}))^{-1} \mathbf{J}^T \mathbf{W} \mathbf{r}_k(\mathbf{0})$$

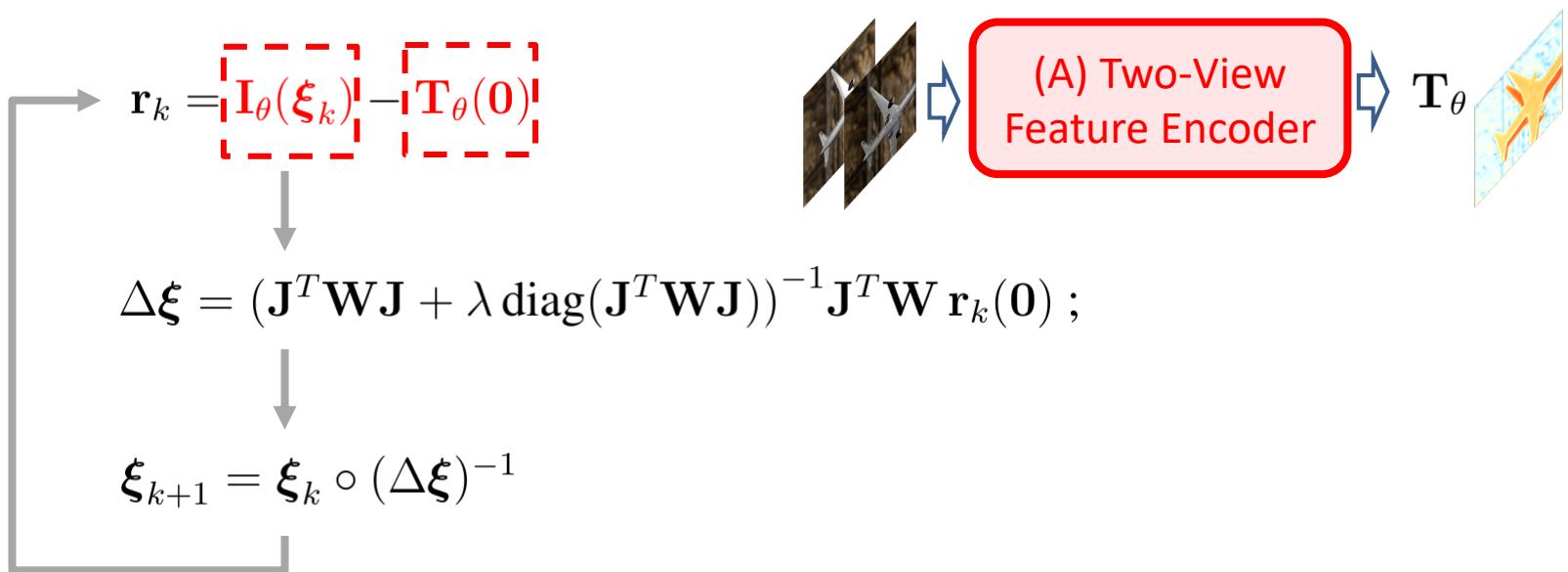


$$\boldsymbol{\xi}_{k+1} = \boldsymbol{\xi}_k \circ (\Delta \boldsymbol{\xi})^{-1}$$

We propose to take a **deeper** look at
the Inverse Compositional algorithm
from a learning perspective.

Take a Deeper Look at the Inverse Compositional algorithm

Contribution (A): Two-view Feature Encoder



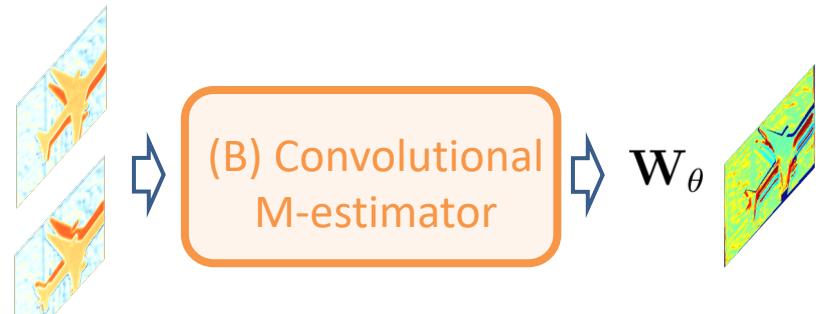
Take a Deeper Look at the Inverse Compositional algorithm

Contribution (B): Convolutional M-estimator

$$\rightarrow \mathbf{r}_k = \mathbf{I}_\theta(\boldsymbol{\xi}_k) - \mathbf{T}_\theta(\mathbf{0})$$

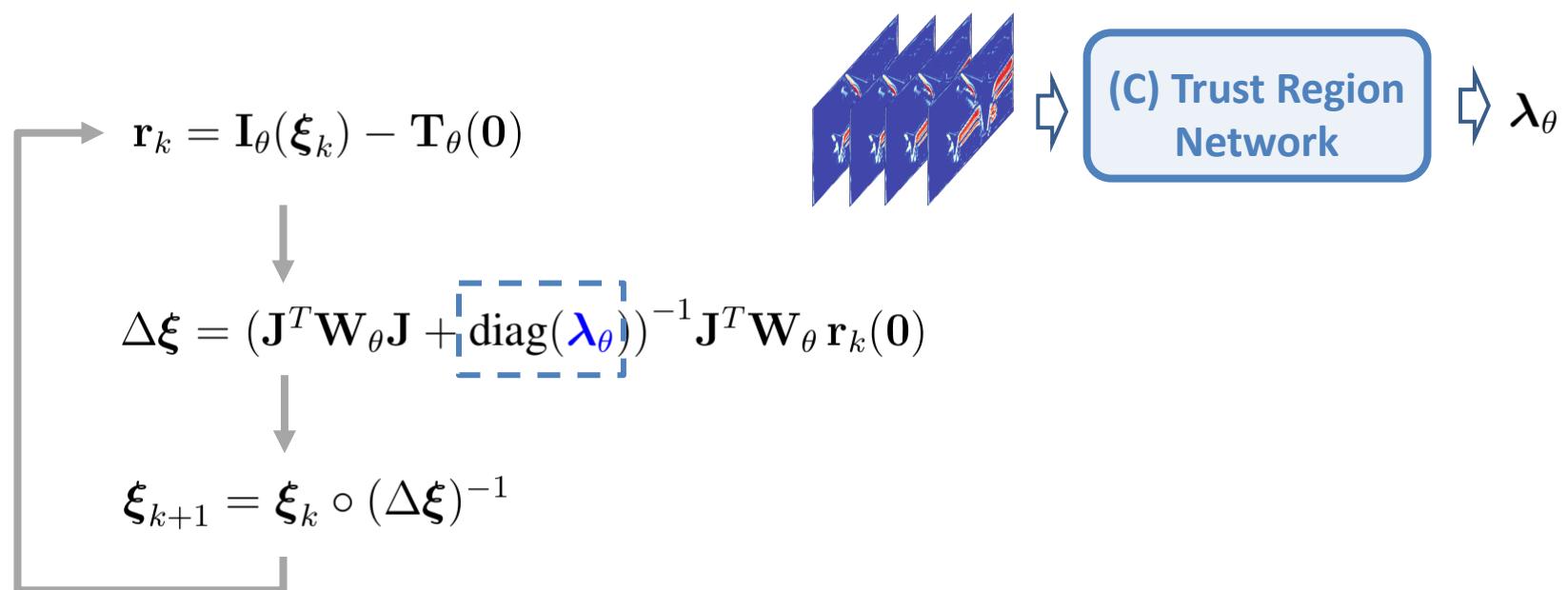
$$\Delta \boldsymbol{\xi} = (\mathbf{J}^T \boxed{\mathbf{W}_\theta} \mathbf{J} + \text{diag}(\mathbf{J}^T \boxed{\mathbf{W}_\theta} \mathbf{J})^{-1} \mathbf{J}^T \boxed{\mathbf{W}_\theta} \mathbf{r}_k(\mathbf{0}))$$

$$\boldsymbol{\xi}_{k+1} = \boldsymbol{\xi}_k \circ (\Delta \boldsymbol{\xi})^{-1}$$

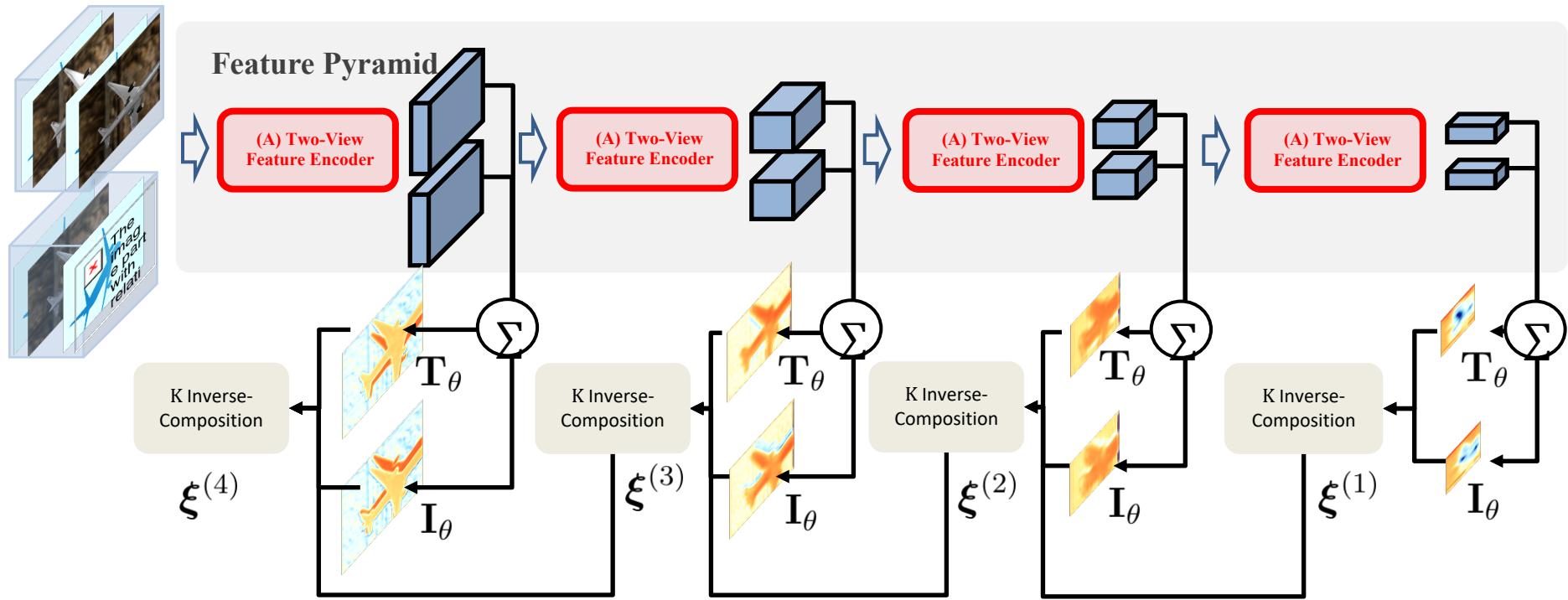


Take a Deeper Look at the Inverse Compositional algorithm

Contribution (C): Trust Region Network



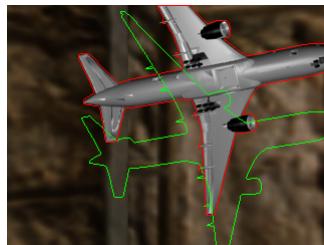
Coarse-to-Fine Inverse Compositional Algorithm



Visualization of Iterative 3D Rigid Motion Alignment



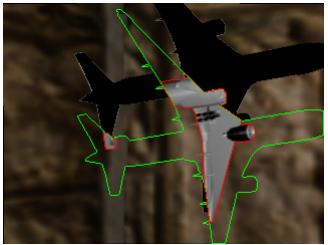
T



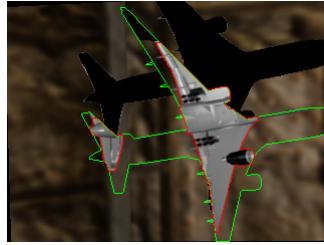
I



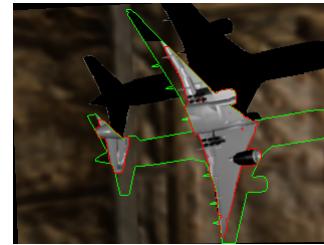
$I(\xi^{GT})$



DeepLK
[Wang et al. ICRA, 2018]



Ours (A)



Ours (A)+(B)



Ours (A)+(B)+(C)

Conclusion

We have taken a deeper look at the inverse compositional algorithm by reformulating it with

- (A) Two-view Feature Encoder
- (B) Convolutional M-estimator
- (C) Trust Region Network

The proposed solution is **learnable, accurate, small, and fast** in inference.