Video Stabilization

CS448V — Computational Video Manipulation

April 2019

Fundamental problem that became even more relevant in recent years


SONY
CONTRAST
BRIGHT
ZEBRA
TALLY
SONY
Power HAD FX

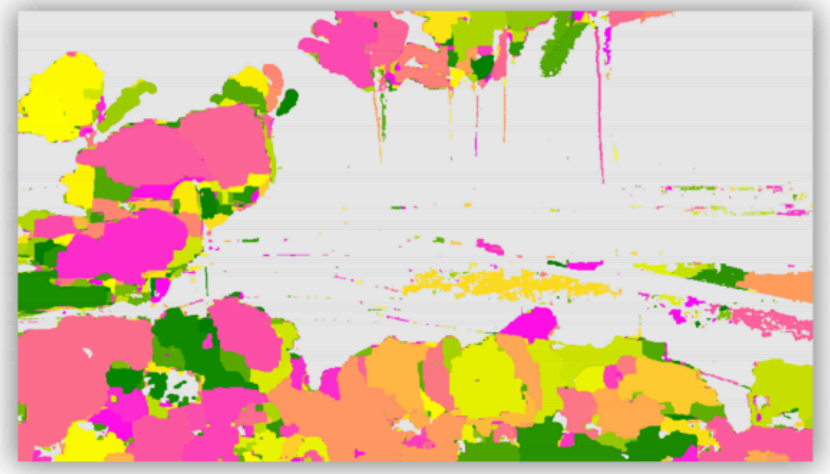

VÍDEO
FOTO
QUADRADO

Important for producing high quality video
and as a first step of many algorithms

Important for producing high quality video and as a first step of many algorithms

"In forming a video loop, we assume that the input video has already been stabilized."

Input video



Loop regions

[Liao et al. '15]

Many ways to stabilize

Many ways to stabilize

Both at capture time and in post

Many ways to stabilize

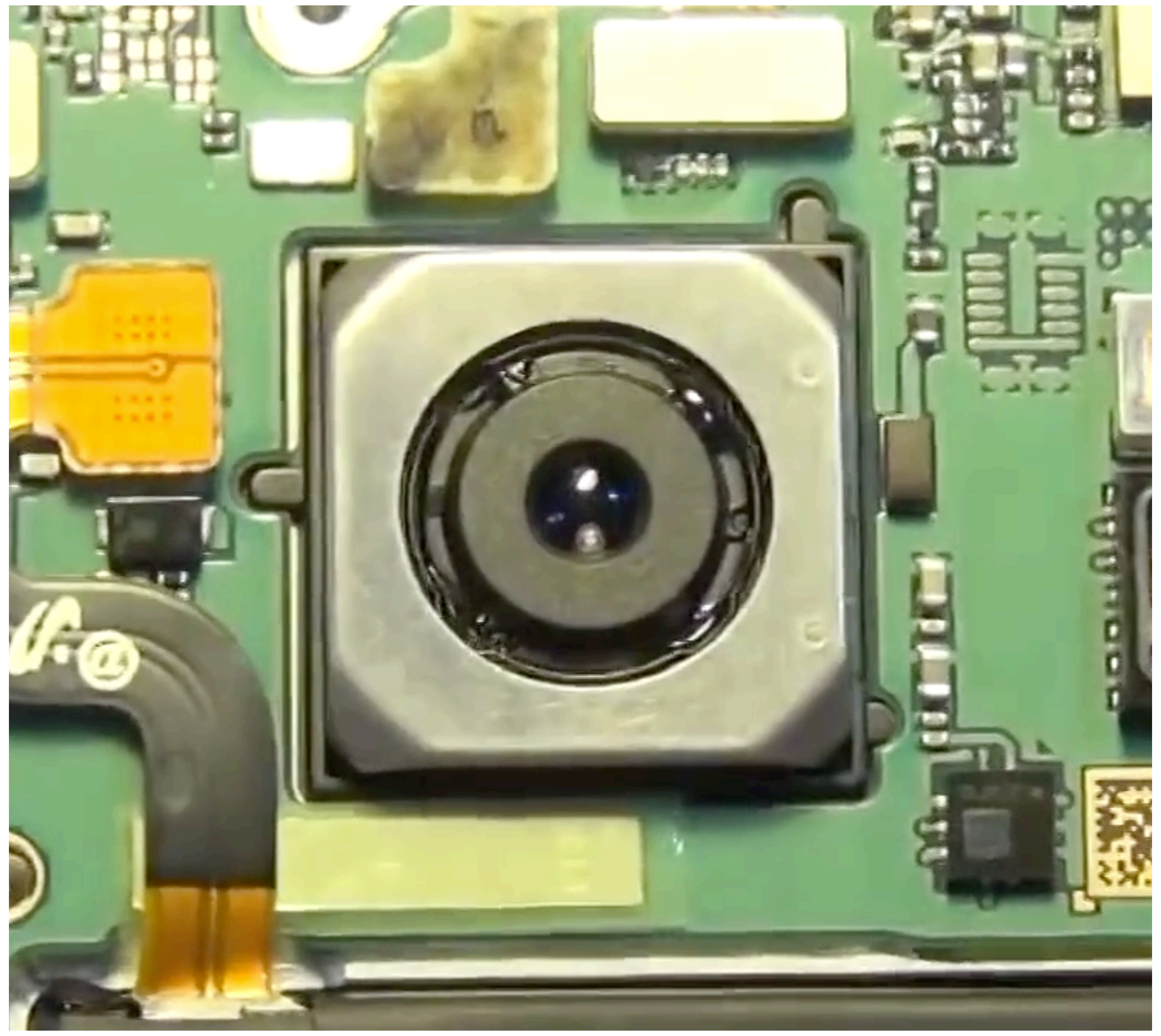
Both at capture time and in post

Tripod

Many ways to stabilize

Both at capture time and in post

Tripod

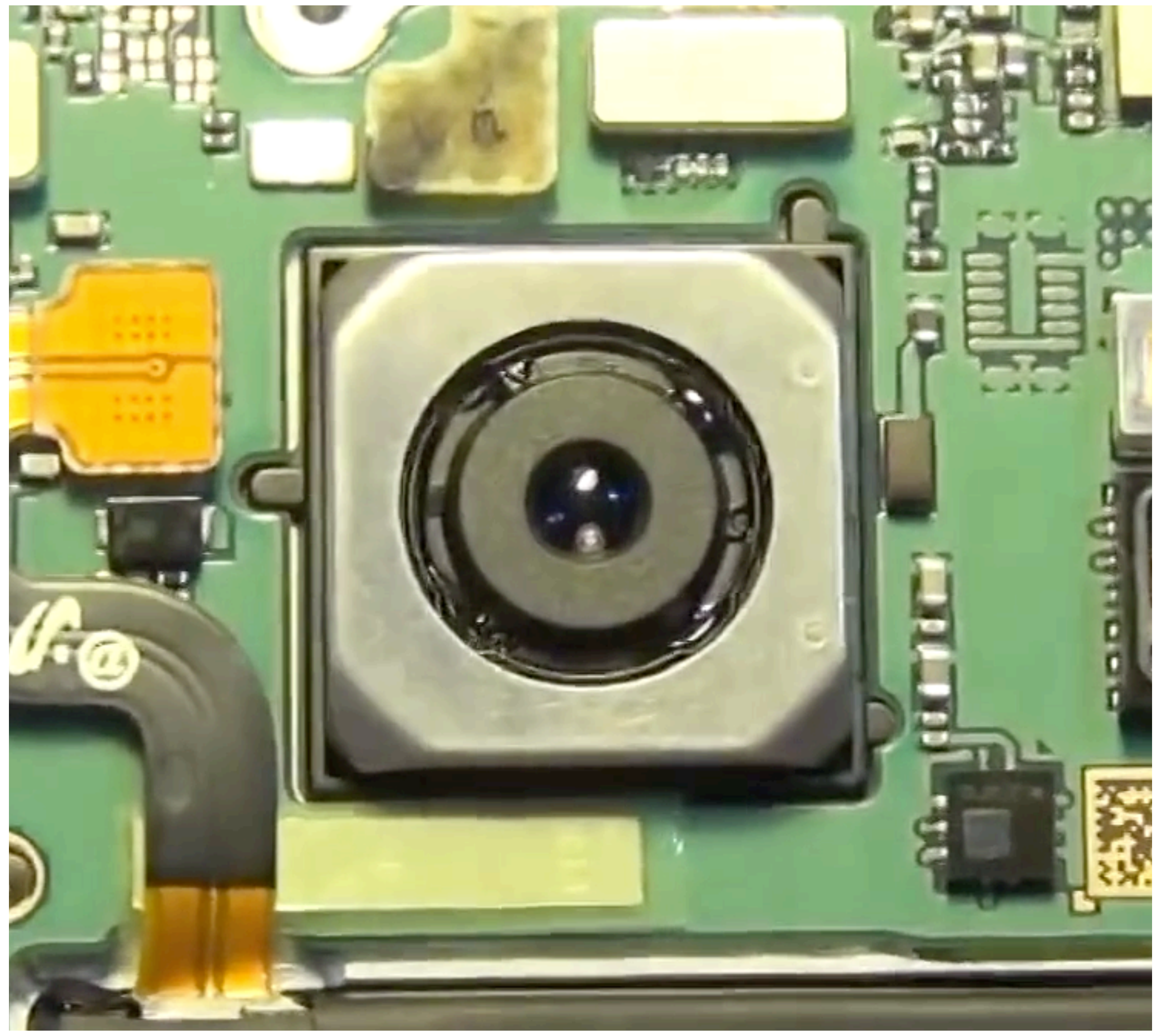


OIS

Many ways to stabilize

Both at capture time and in post

Tripod



OIS

Gimbal

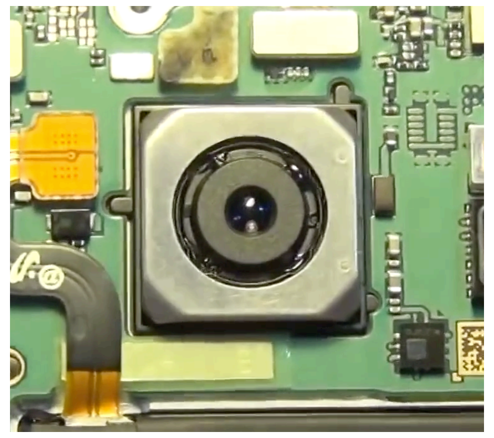
Many ways to stabilize

Both at capture time and in post

capture time

Many ways to stabilize

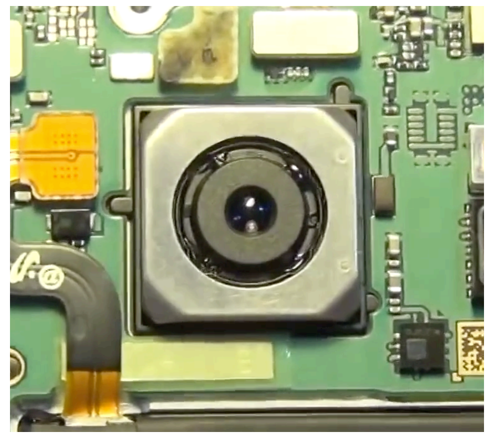
Both at capture time and in post



capture time

Many ways to stabilize

Both at capture time and in post

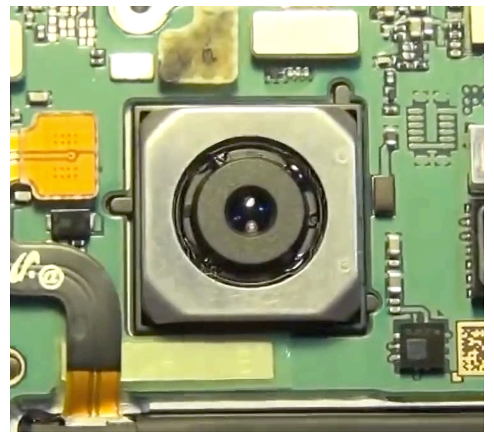




capture time

post production

Many ways to stabilize

Both at capture time and in post



capture time

manual automatic

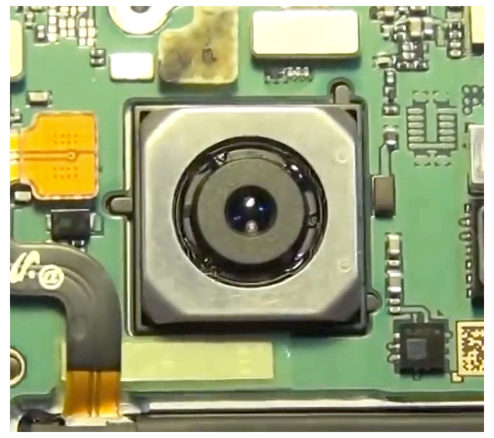
2D

3D

post production

Many ways to stabilize

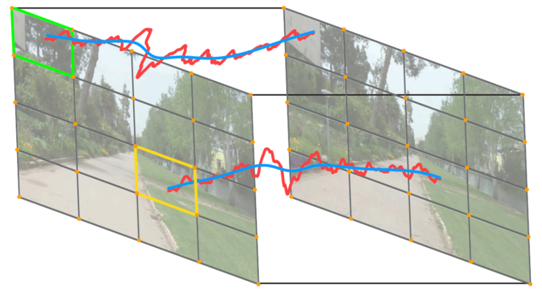
Both at capture time and in post



capture time

manual

automatic



[Liu et al. ’13]

2D

3D

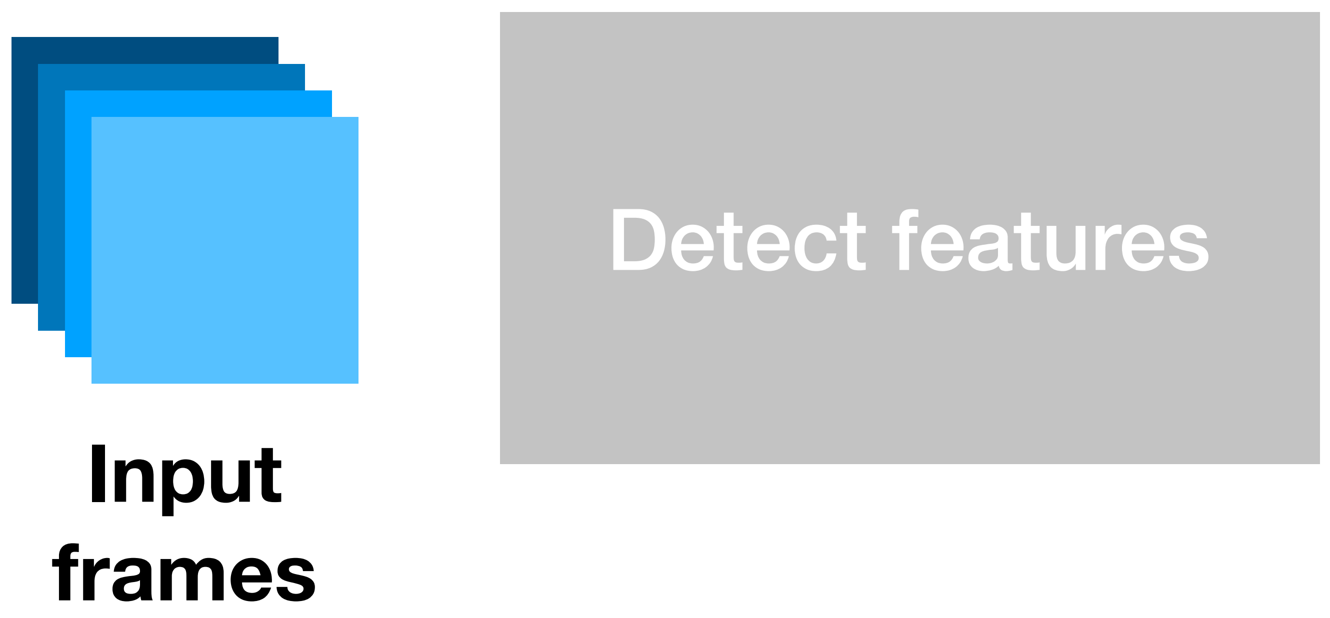
post production

Recipe for video stabilization

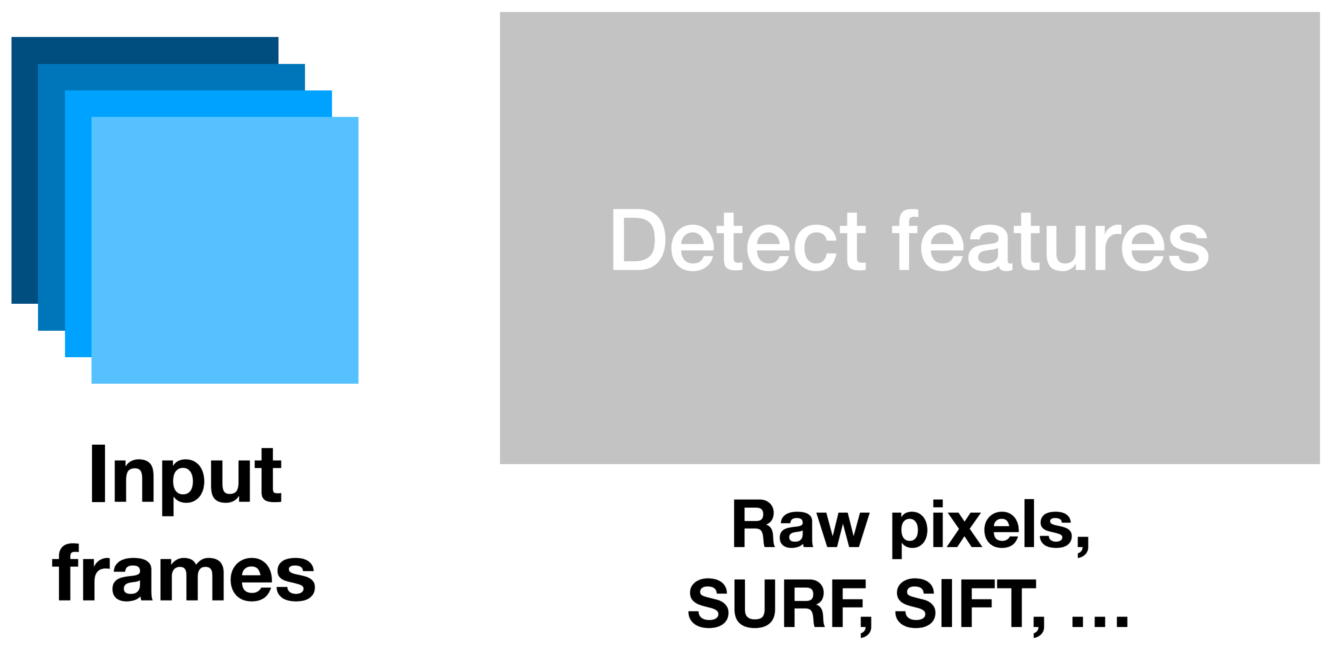
Recipe for video stabilization

Input frames

Recipe for video stabilization



Recipe for video stabilization



Recipe for video stabilization

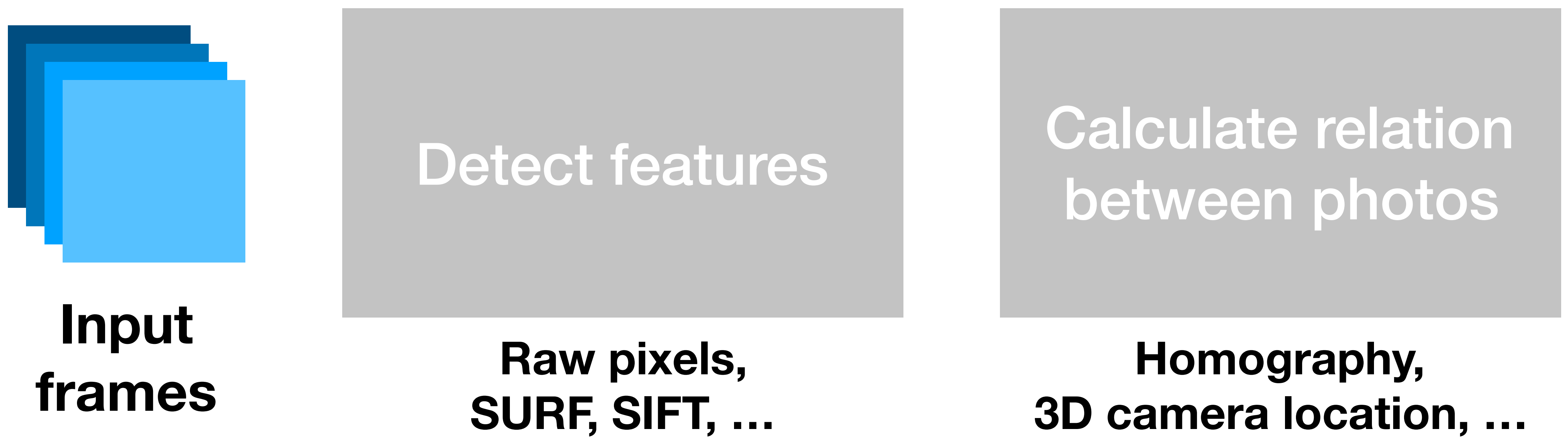
Input frames

Detect features

Raw pixels, SURF, SIFT, …

Calculate relation between photos

Recipe for video stabilization



Recipe for video stabilization

Input frames

Detect features

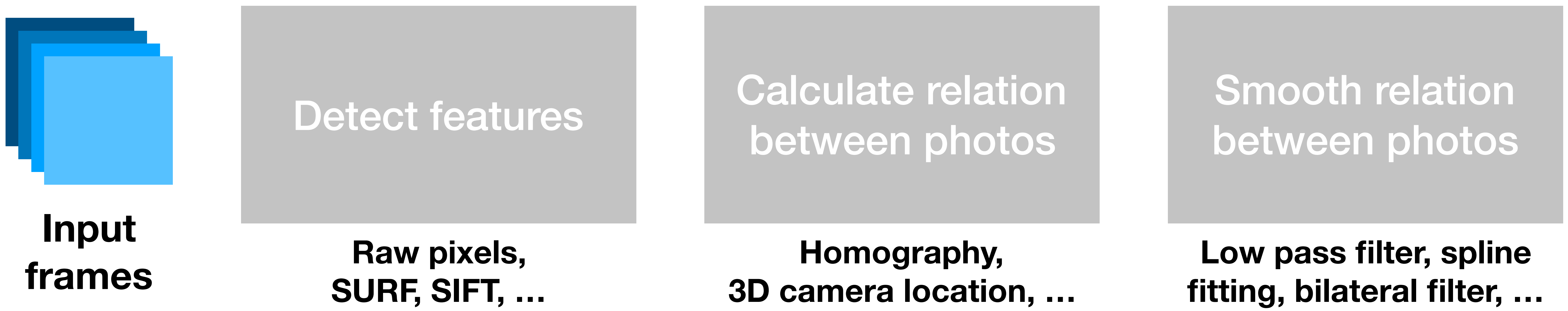
Raw pixels,
SURF, SIFT, …

Calculate relation between photos

Homography,
3D camera location, …

Smooth relation between photos

Recipe for video stabilization



Recipe for video stabilization

Input frames

Detect features

Raw pixels, SURF, SIFT, …

Calculate relation between photos

Homography, 3D camera location, …

Smooth relation between photos

Low pass filter, spline fitting, bilateral filter, …

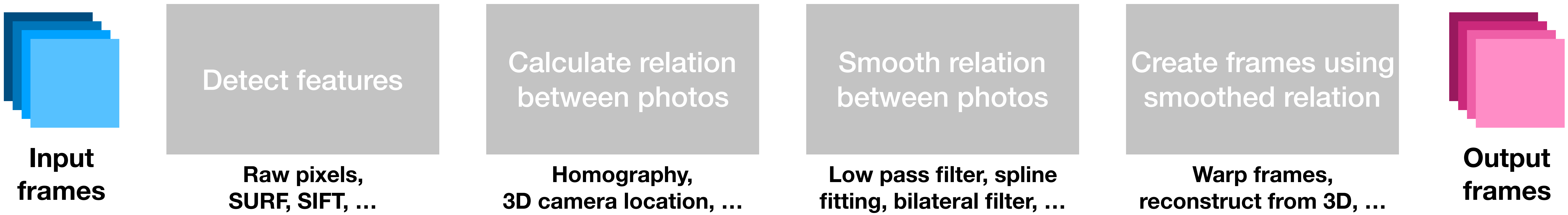
Create frames using smoothed relation

Recipe for video stabilization

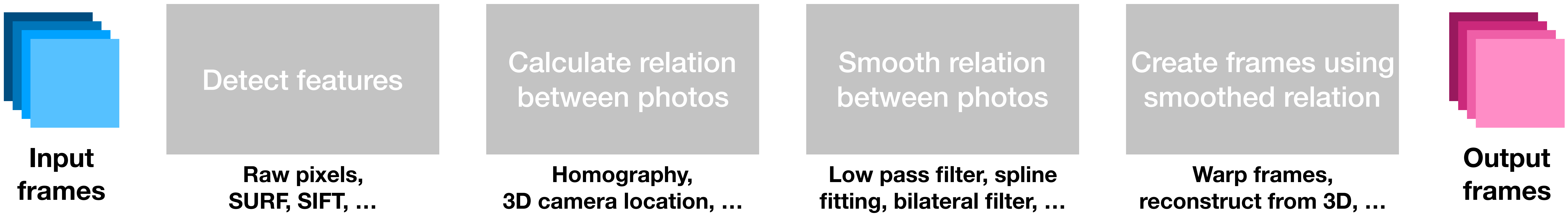
Input frames

Detect features	Calculate relation between photos	Smooth relation between photos	Create frames using smoothed relation
Raw pixels, SURF, SIFT, …	**Homography, 3D camera location, …**	**Low pass filter, spline fitting, bilateral filter, …**	**Warp frames, reconstruct from 3D, …**

Recipe for video stabilization



Recipe for video stabilization



Toy example:

Recipe for video stabilization

Input frames	Detect features	Calculate relation between photos	Smooth relation between photos	Create frames using smoothed relation	Output frames
	Raw pixels, SURF, SIFT, …	Homography, 3D camera location, …	Low pass filter, spline fitting, bilateral filter, …	Warp frames, reconstruct from 3D, …	

Toy example:

SIFT

Recipe for video stabilization

Input frames	Detect features	Calculate relation between photos	Smooth relation between photos	Create frames using smoothed relation	Output frames
	Raw pixels, SURF, SIFT, …	Homography, 3D camera location, …	Low pass filter, spline fitting, bilateral filter, …	Warp frames, reconstruct from 3D, …	

Toy example:

SIFT	2D translation

Recipe for video stabilization

Input frames

Detect features
Raw pixels, SURF, SIFT, …

Calculate relation between photos
Homography, 3D camera location, …

Smooth relation between photos
Low pass filter, spline fitting, bilateral filter, …

Create frames using smoothed relation
Warp frames, reconstruct from 3D, …

Output frames

Toy example:

SIFT

2D translation

Gaussian

Recipe for video stabilization

Input frames

Detect features

Raw pixels, SURF, SIFT, …

Calculate relation between photos

Homography, 3D camera location, …

Smooth relation between photos

Low pass filter, spline fitting, bilateral filter, …

Create frames using smoothed relation

Warp frames, reconstruct from 3D, …

Output frames

Toy example:

SIFT

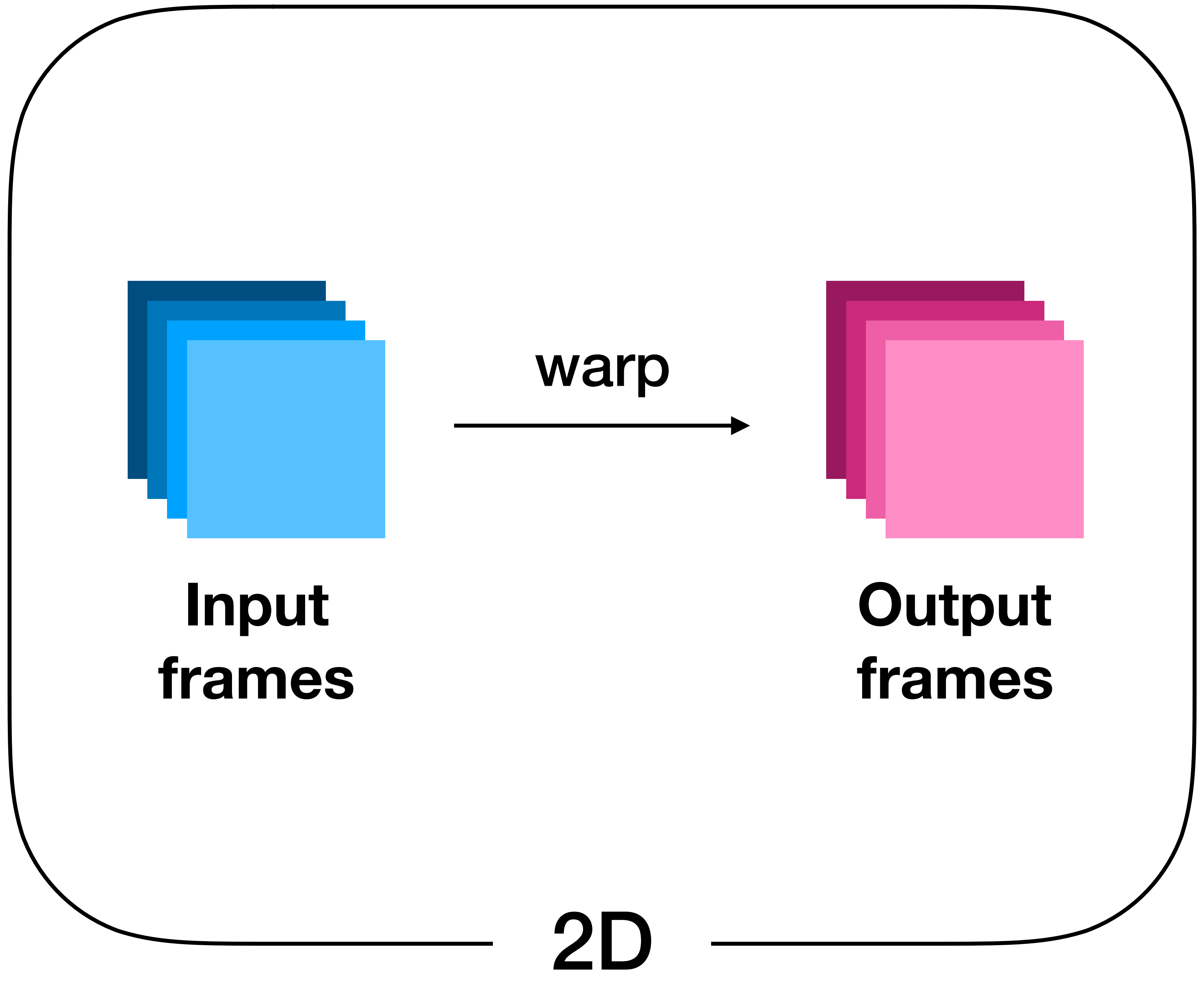
2D translation

Gaussian

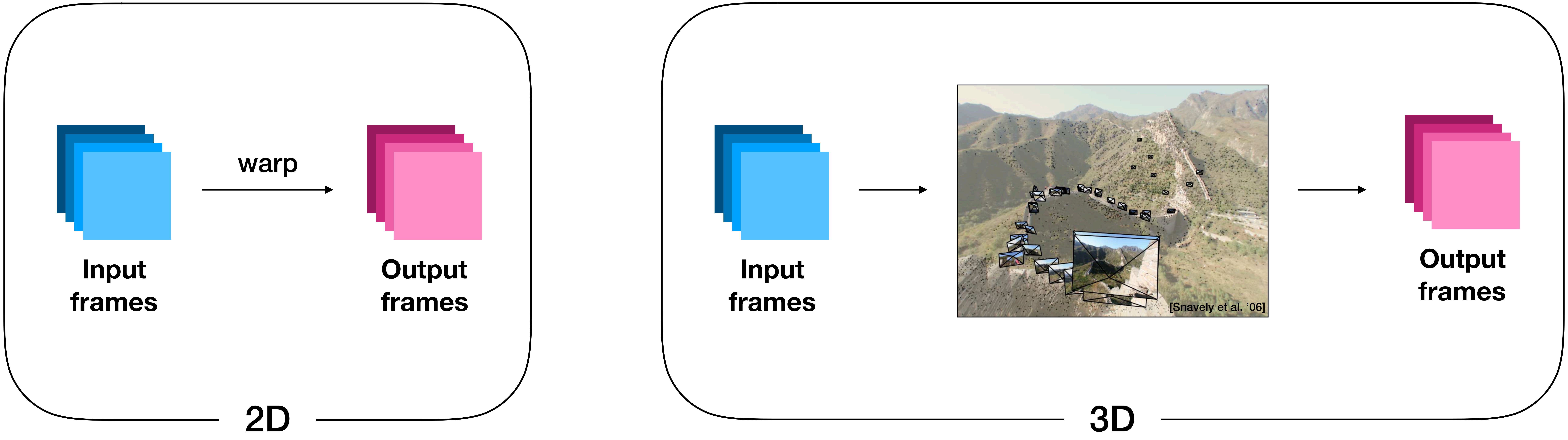
Warp

2D vs. 3D

2D vs. 3D

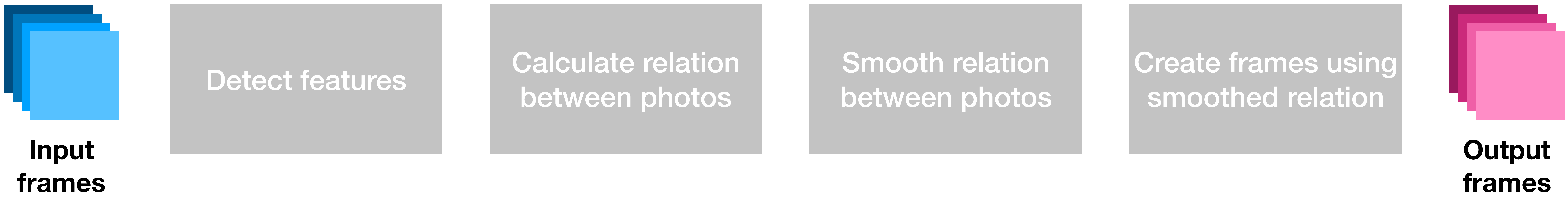


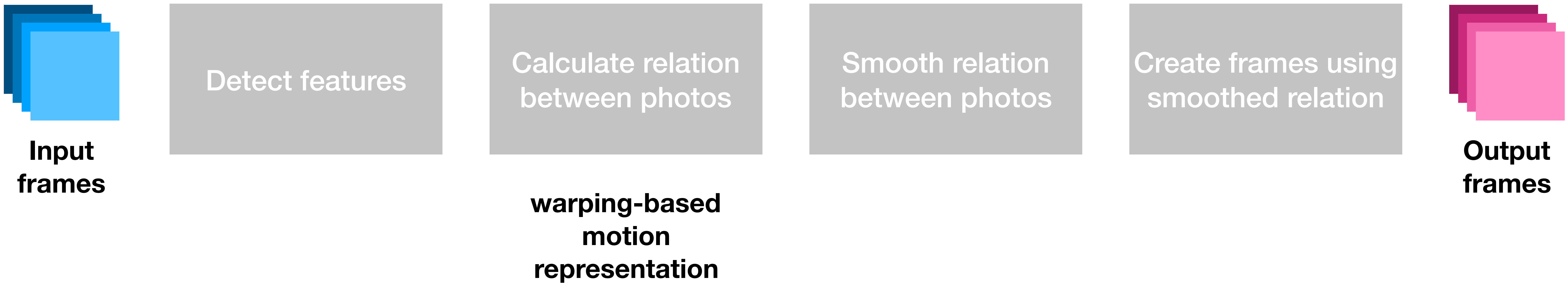
2D vs. 3D

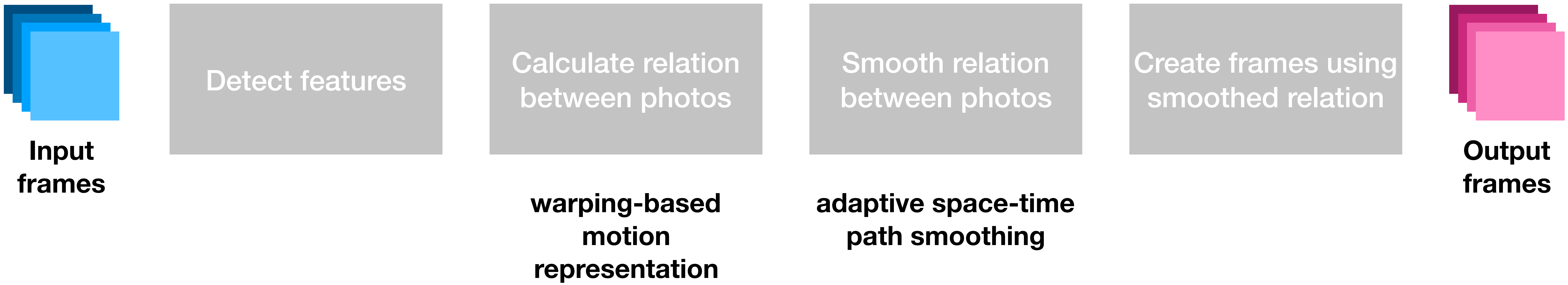

Input
frames
warp
Output
frames
2D
Input
frames
[Snavely et al. '06]
Output
frames
3D

Bundled Camera Paths for Video Stabilization

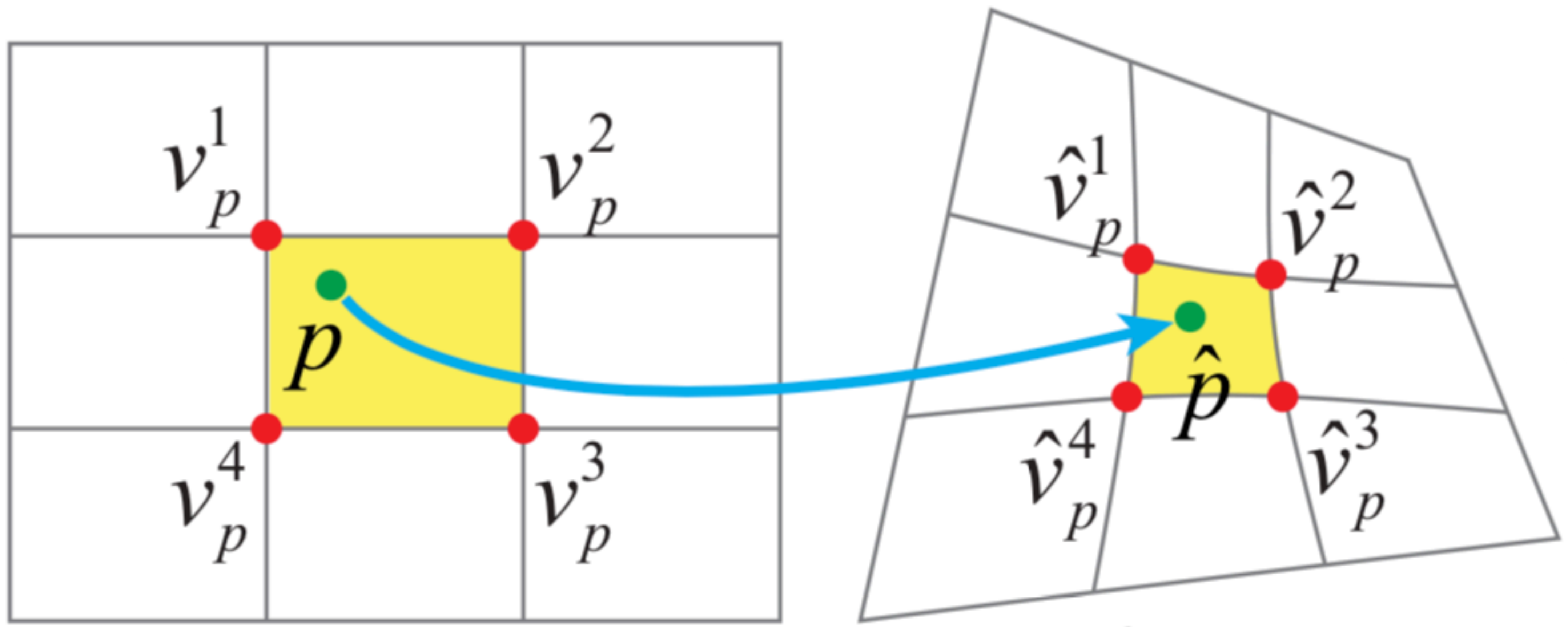
Liu et al. SIGGRAPH 2013


Input frames
Detect features
Calculate relation between photos
Smooth relation between photos
Create frames using smoothed relation
Output frames


Input frames
Detect features
Calculate relation between photos
warping-based motion representation
Smooth relation between photos
Create frames using smoothed relation
Output frames

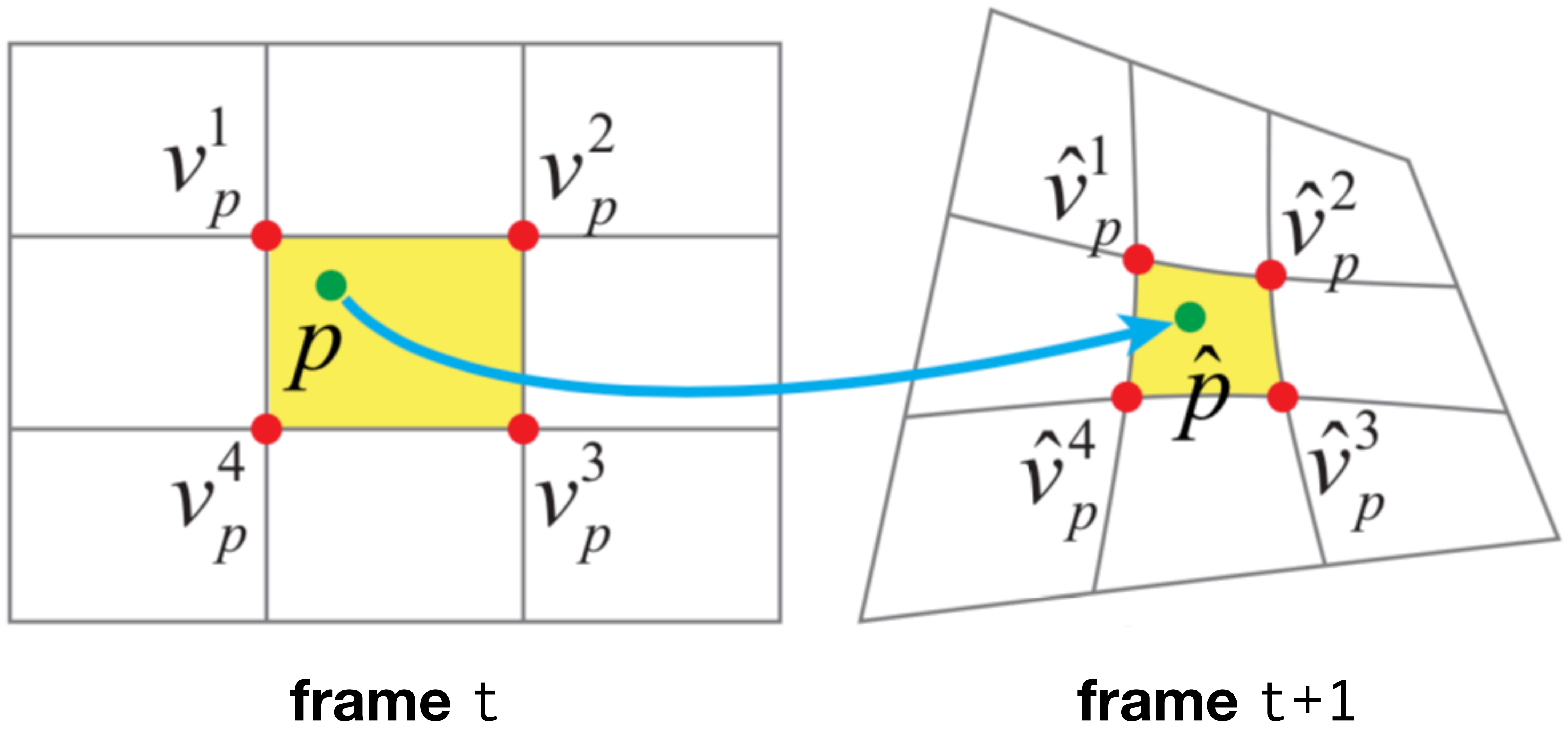

Input frames
Detect features
Calculate relation between photos
warping-based motion representation
Smooth relation between photos
adaptive space-time path smoothing
Create frames using smoothed relation
Output frames

Warping-based motion representation



frame t   **frame** t+1

Warping-based motion representation

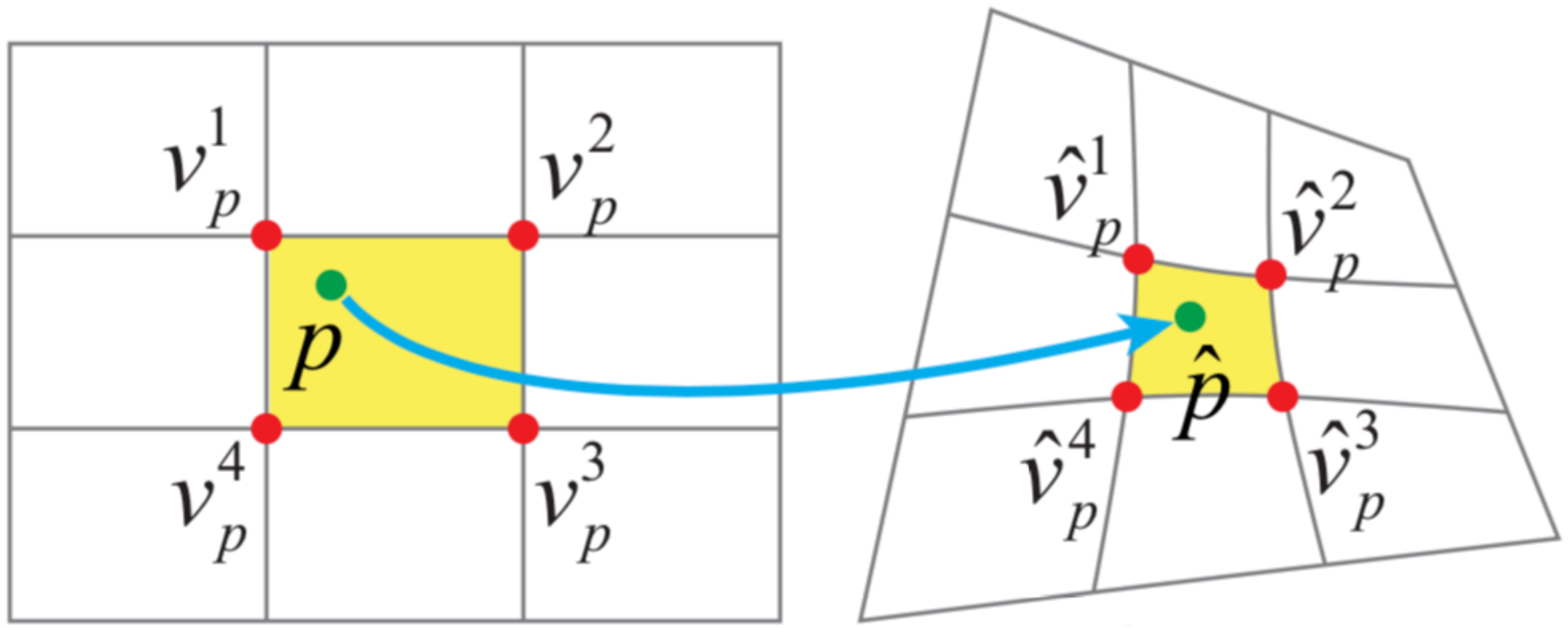


Given that:

$$p = \begin{bmatrix} v_p^1 & v_p^2 & v_p^3 & v_p^4 \end{bmatrix} \begin{bmatrix} w_p^1 \\ w_p^2 \\ w_p^3 \\ w_p^4 \end{bmatrix}$$

$$\sum_{i=1}^{4} w_p^i = 1$$

Warping-based motion representation



frame `t` **frame** `t+1`

Given that:

$$p = \begin{bmatrix} v_p^1 & v_p^2 & v_p^3 & v_p^4 \end{bmatrix} \begin{bmatrix} w_p^1 \\ w_p^2 \\ w_p^3 \\ w_p^4 \end{bmatrix}$$

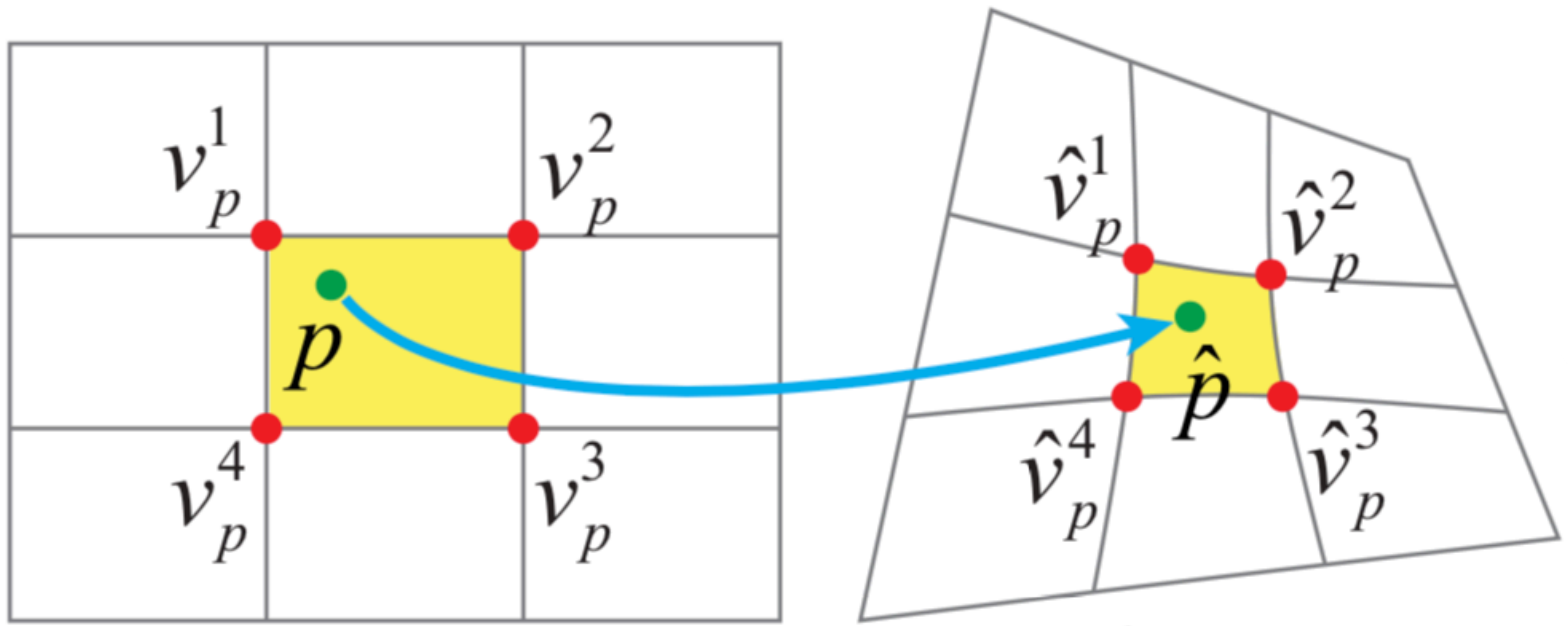
$$\sum_{i=1}^{4} w_p^i = 1$$

We would like:

$$\hat{p} = \begin{bmatrix} \hat{v}_p^1 & \hat{v}_p^2 & \hat{v}_p^3 & \hat{v}_p^4 \end{bmatrix} \begin{bmatrix} w_p^1 \\ w_p^2 \\ w_p^3 \\ w_p^4 \end{bmatrix}$$

$$\hat{p} = \hat{V}_p w_p$$

Warping-based motion representation



Given that:

$$p = \begin{bmatrix} v_p^1 & v_p^2 & v_p^3 & v_p^4 \end{bmatrix} \begin{bmatrix} w_p^1 \\ w_p^2 \\ w_p^3 \\ w_p^4 \end{bmatrix}$$

$$\sum_{i=1}^{4} w_p^i = 1$$

We would like:

$$\hat{p} = \begin{bmatrix} \hat{v}_p^1 & \hat{v}_p^2 & \hat{v}_p^3 & \hat{v}_p^4 \end{bmatrix} \begin{bmatrix} w_p^1 \\ w_p^2 \\ w_p^3 \\ w_p^4 \end{bmatrix}$$

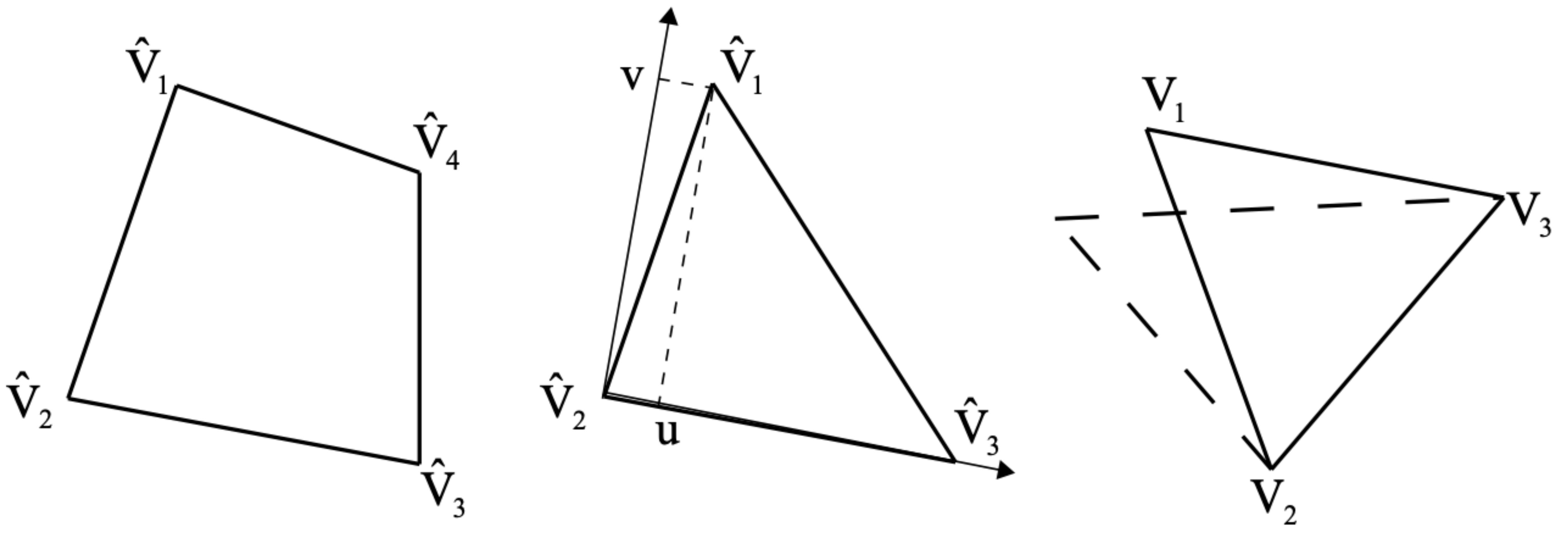
$$\hat{p} = \hat{V}_p w_p$$

Data term:

$$\sum_p \| \hat{V}_p w_p - \hat{p} \|^2$$

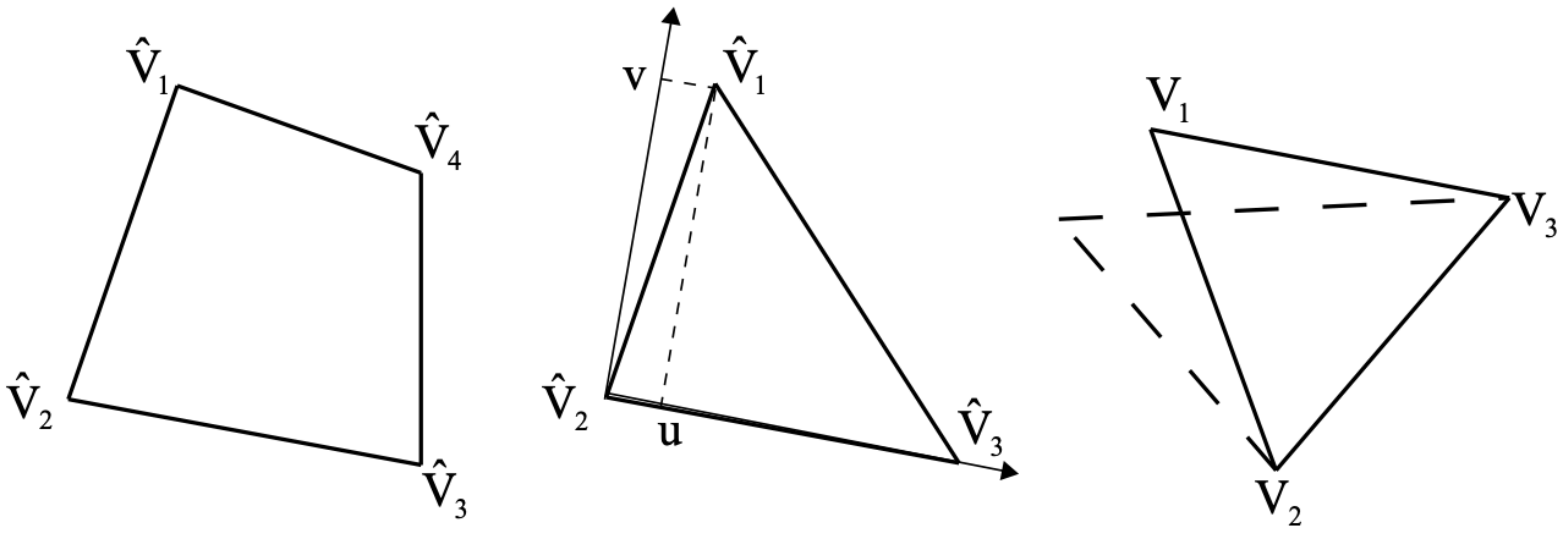
Warping-based motion representation

Shape-preserving term:
Distance from similarity transform

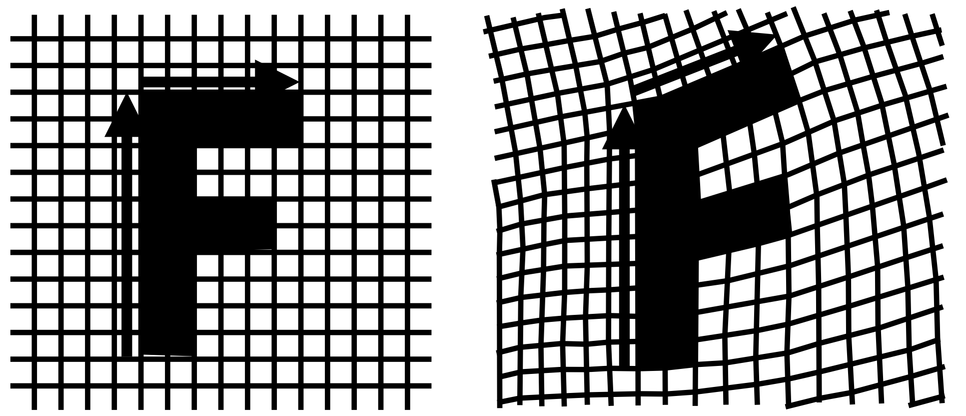


Warping-based motion representation

Shape-preserving term:
Distance from similarity transform



sounds familiar?...



Warping-based motion representation

$$E(\hat{V}) = E_d(\hat{V}) + \alpha E_s(\hat{V})$$

data

shape-preserving

Warping-based motion representation

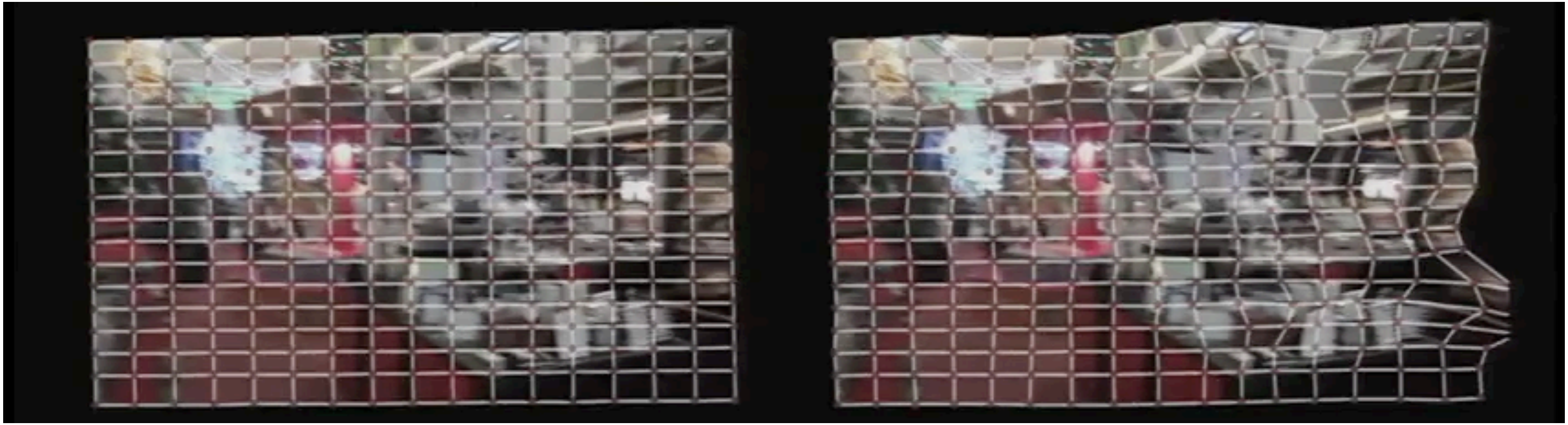
Shape-preserving term

with

without

Warping-based motion representation

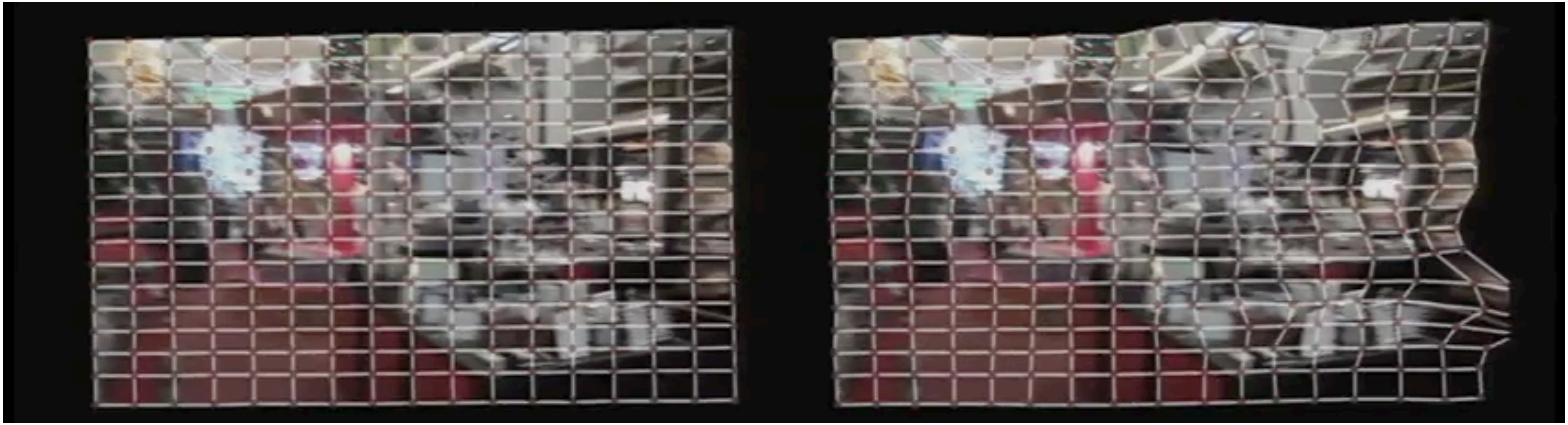
Shape-preserving term



with **without**

Warping-based motion representation

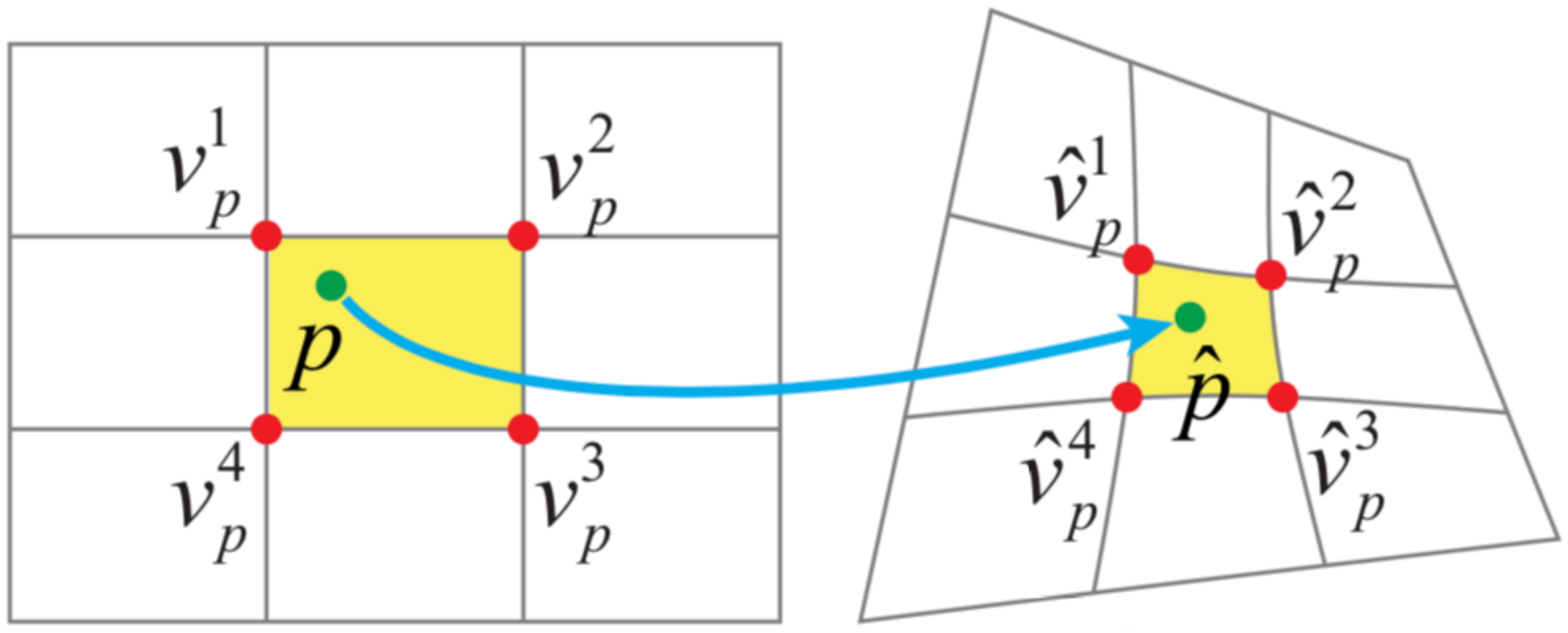
Shape-preserving term



with **without**

Warping-based motion representation

We now have a local homography $F_i(t)$ for each cell i of frame t



frame t

frame t+1

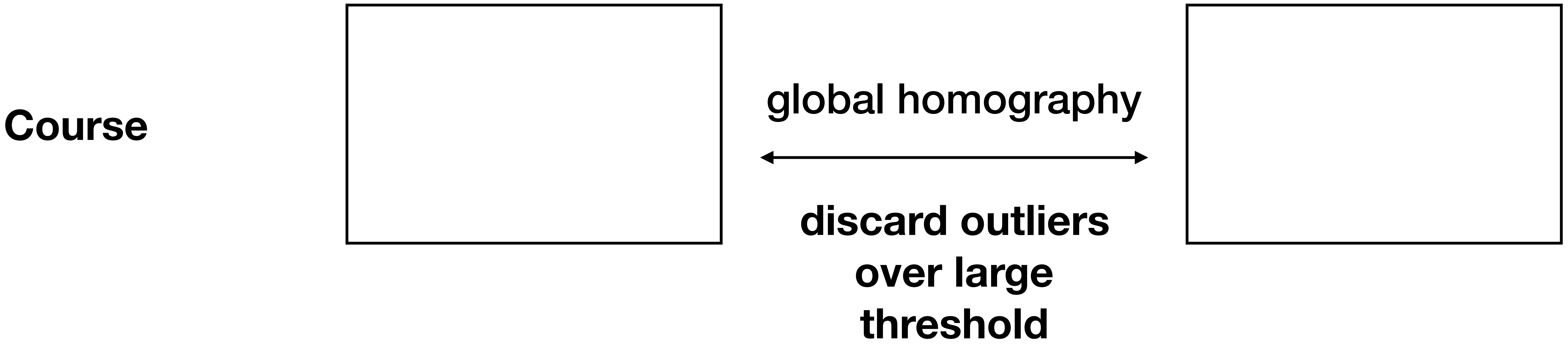
Extensions for robust estimation

Extensions for robust estimation

Outlier rejection: dual-scale RANSAC

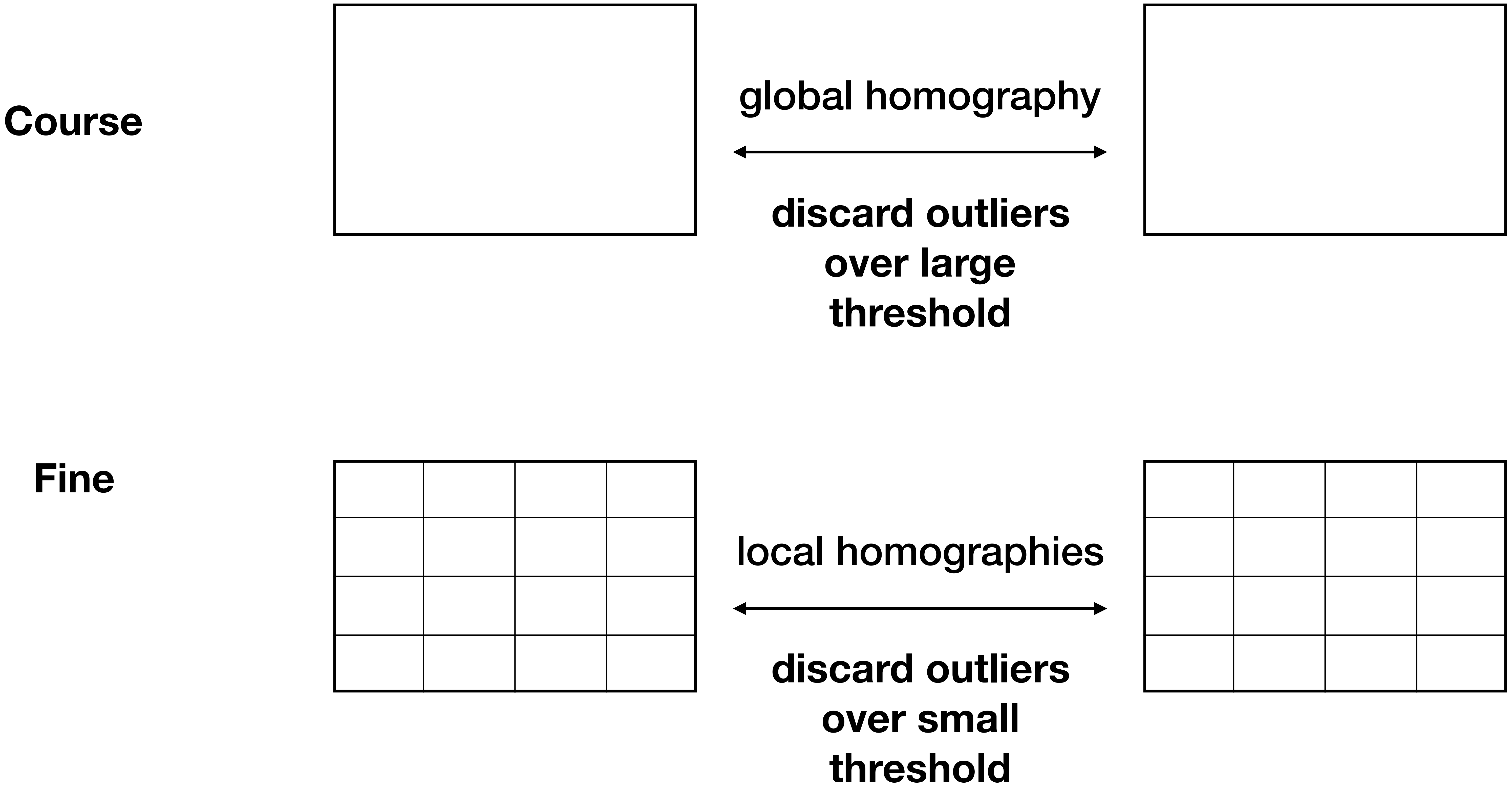
Extensions for robust estimation

Outlier rejection: dual-scale RANSAC



Extensions for robust estimation

Outlier rejection: dual-scale RANSAC

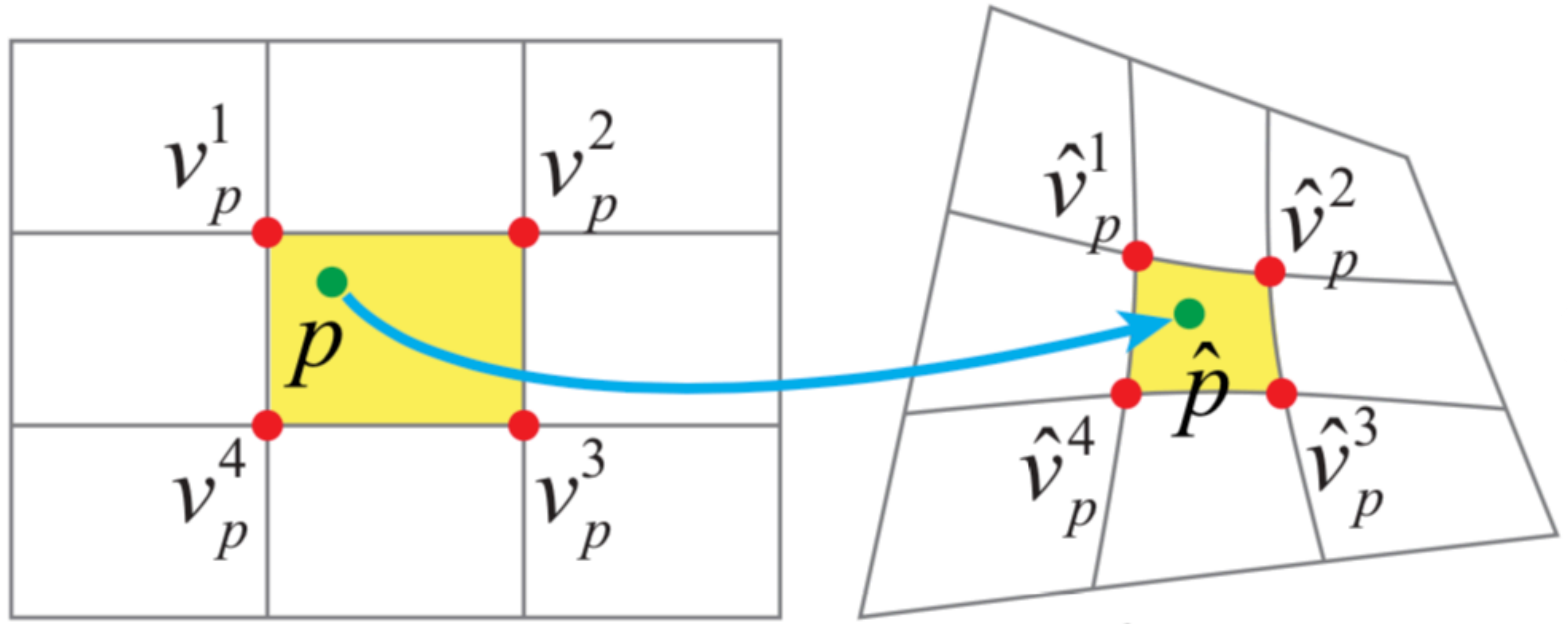


Extensions for robust estimation

Adaptive regularization

$$E(\hat{V}) = E_d(\hat{V}) + \alpha E_s(\hat{V})$$

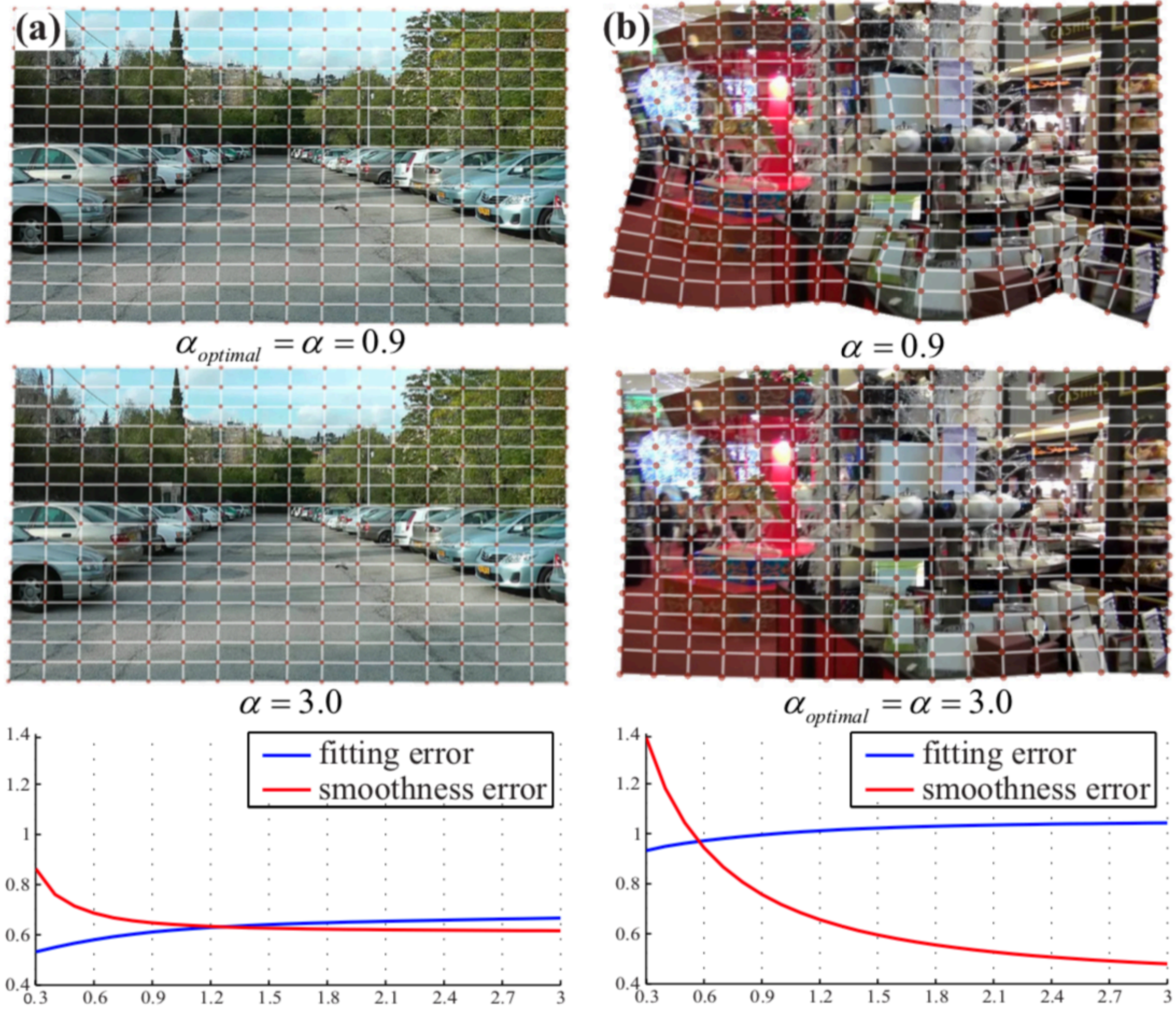
Calculate *α* per frame



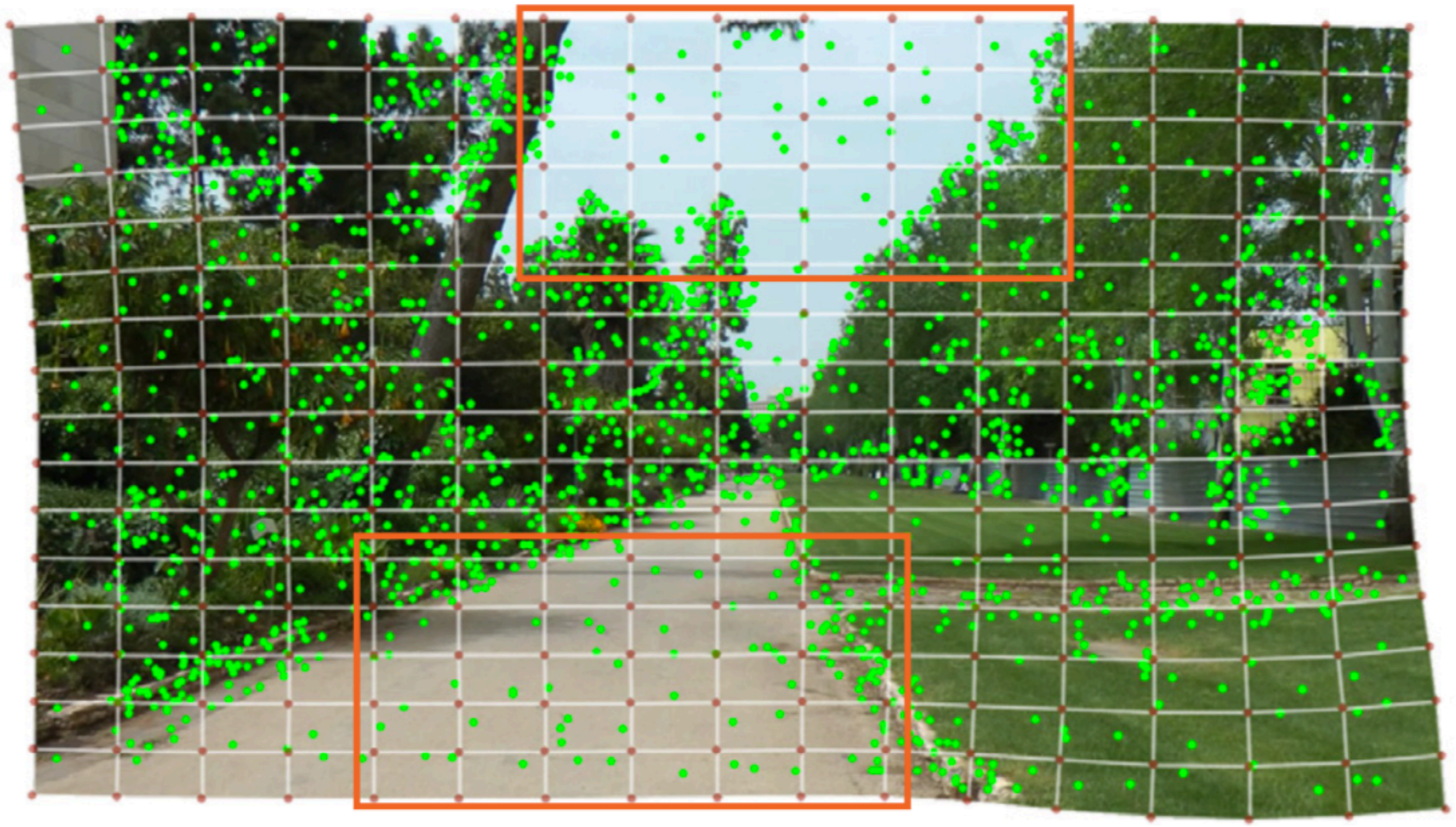
Fitting error:
average residual of feature matching

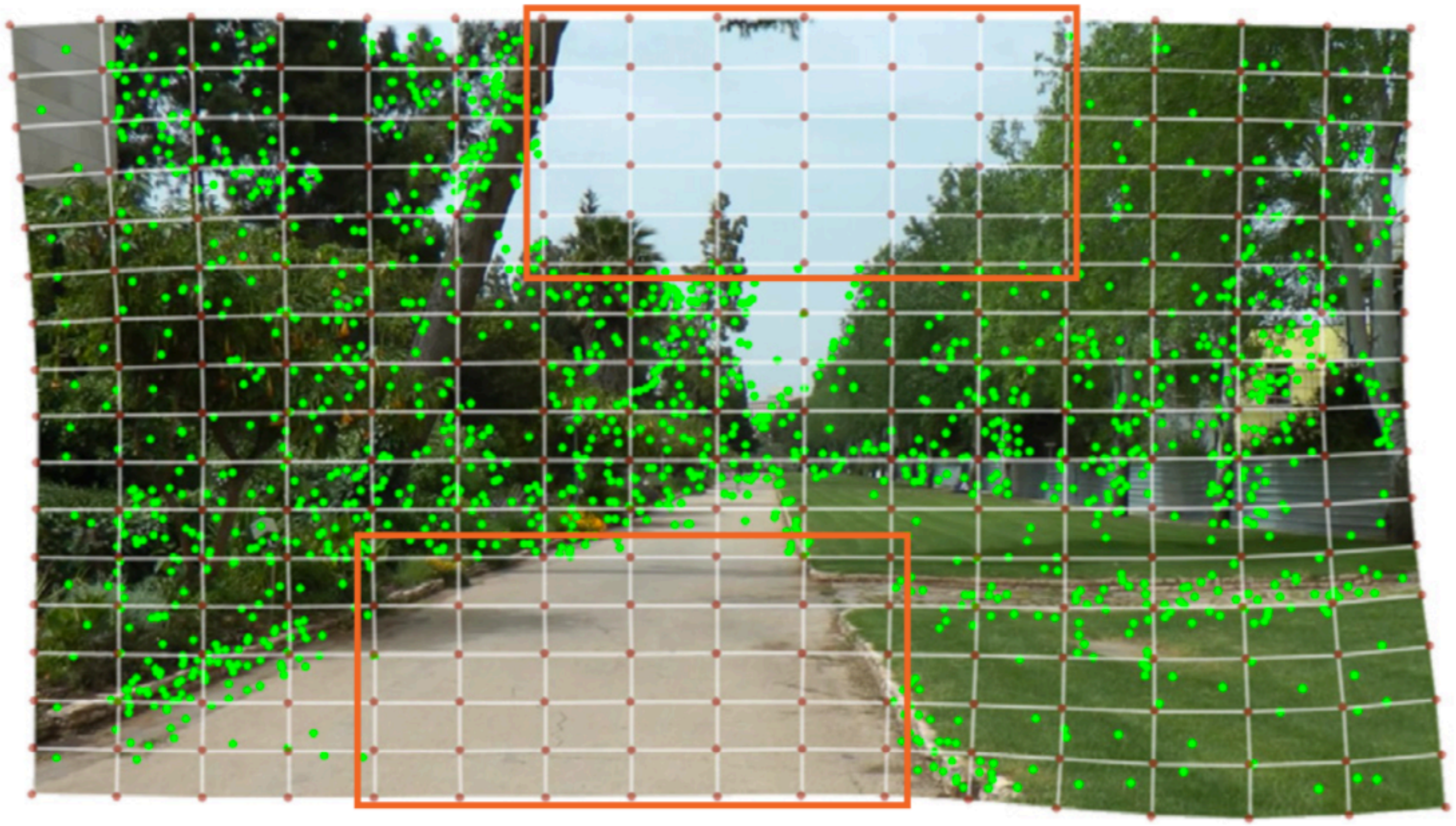
Smoothness error:
L2 distance between neighboring homographies

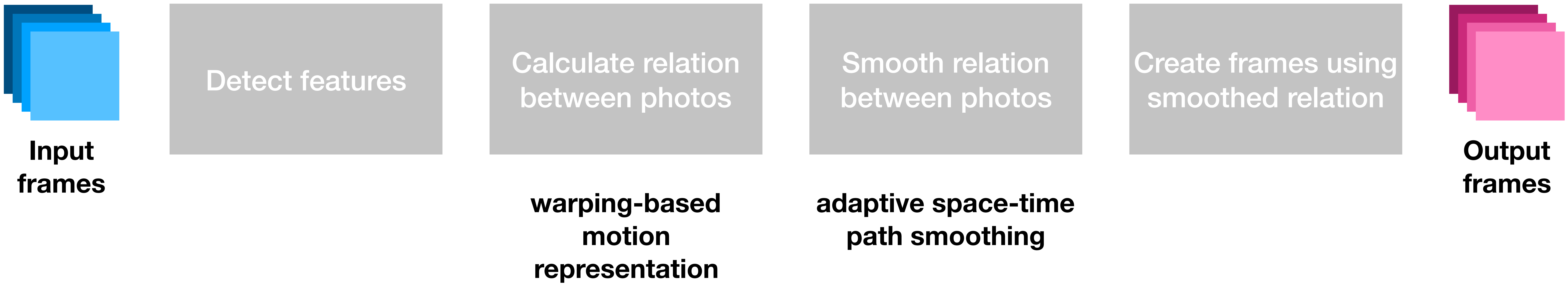
Estimate for different α and pick minimal error

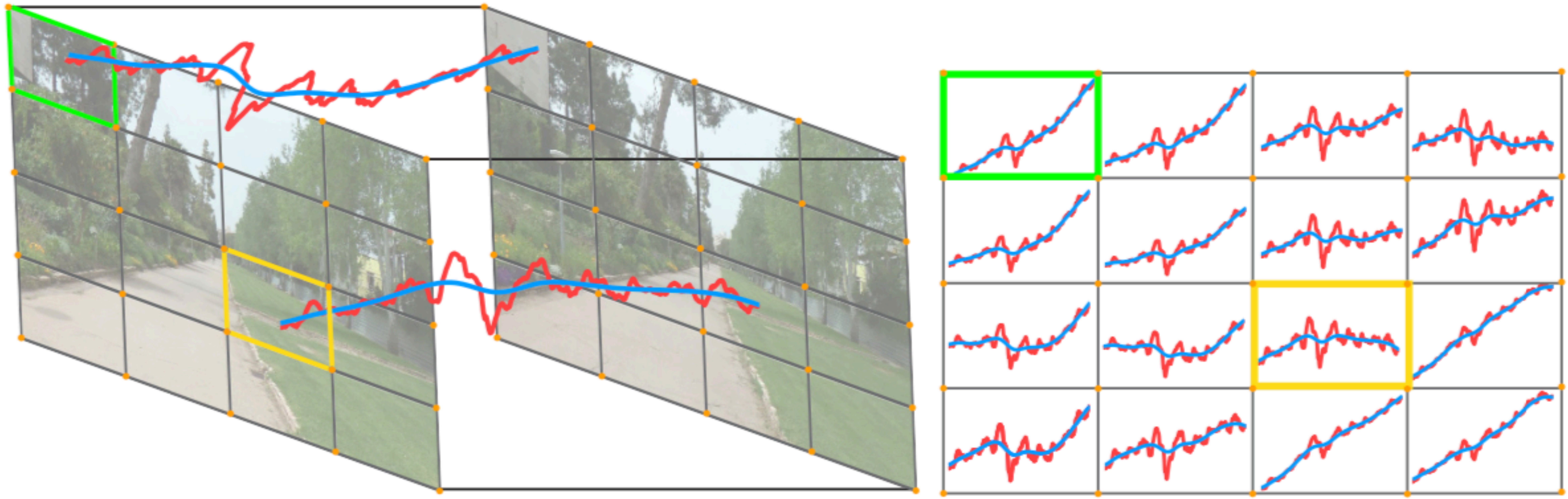


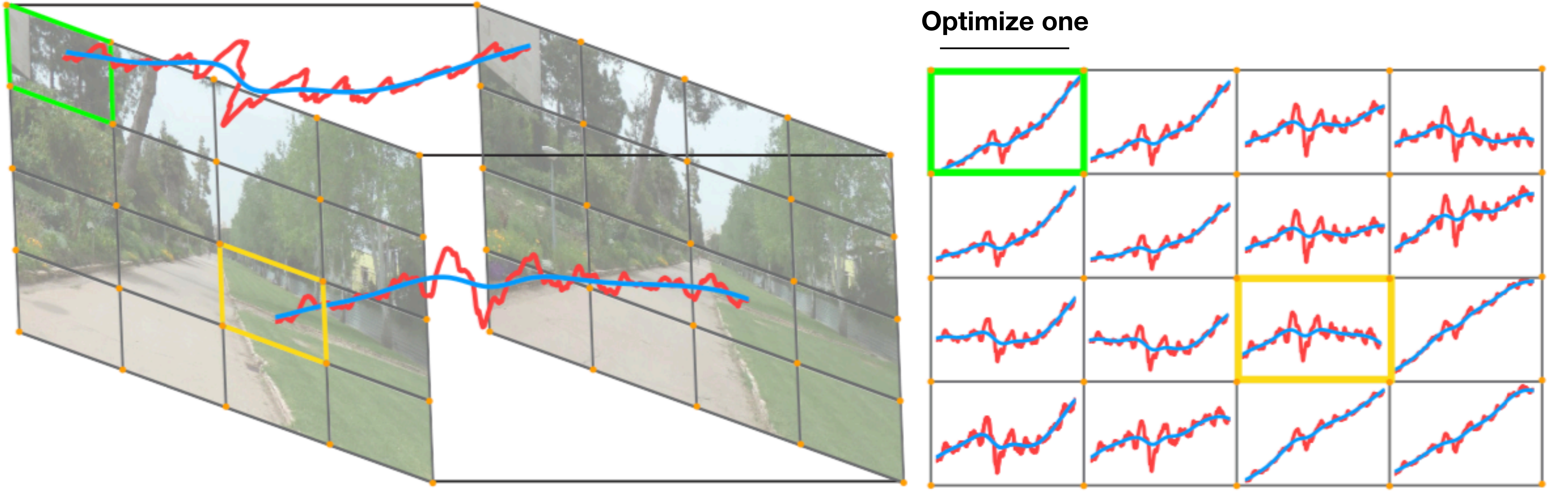
Benefits of regularization

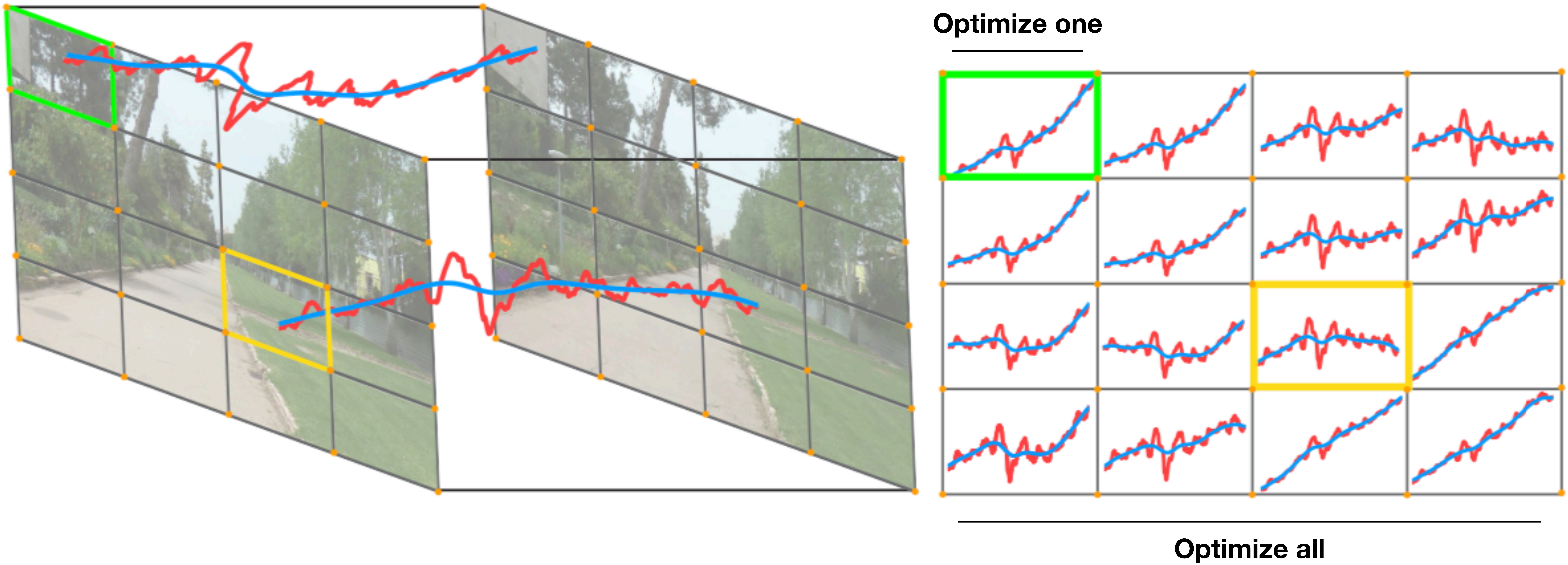




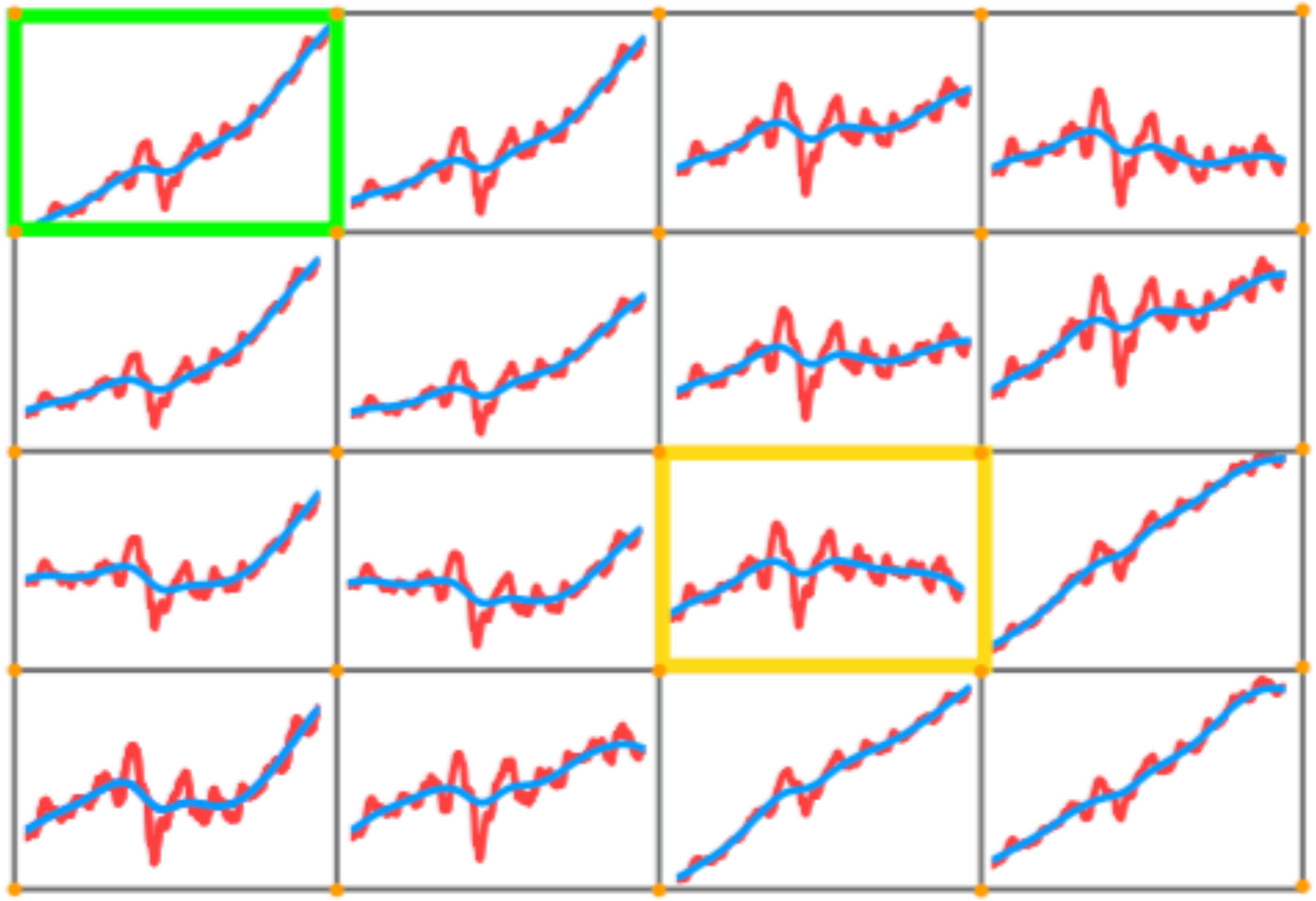

Input frames
Detect features
Calculate relation between photos
warping-based motion representation
Smooth relation between photos
adaptive space-time path smoothing
Create frames using smoothed relation
Output frames



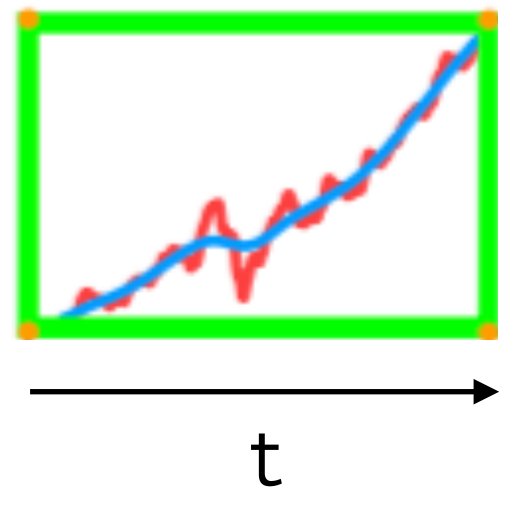

Optimize one


Optimize one
Optimize all

Optimizing a single path

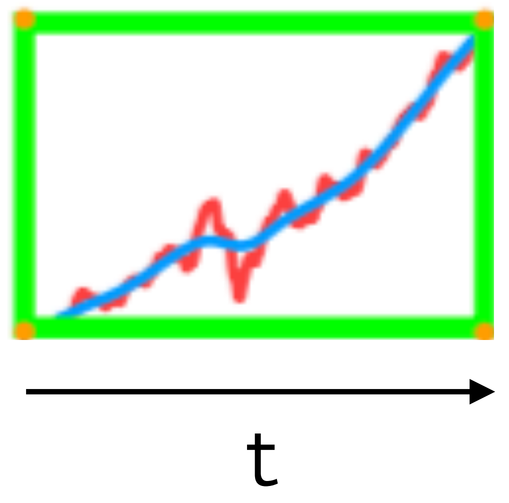


Optimizing a single path



Optimizing a single path

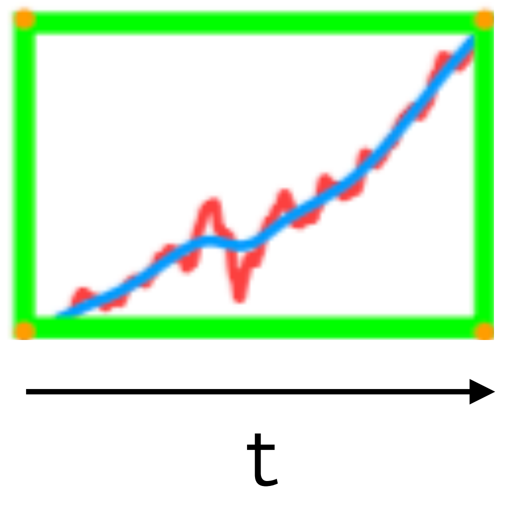
Data term:
blue should match red



Optimizing a single path

Data term:
blue should match red

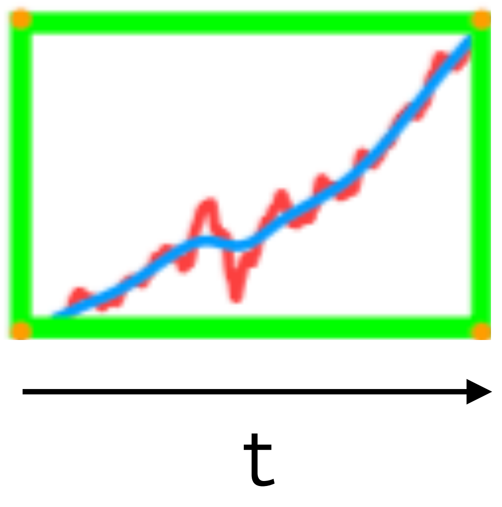
Smoothness term:
blue at time `t` should match the (60) frames around `t`



Optimizing a single path

Data term:
blue should match red

Smoothness term:
blue at time `t` should match the (60) frames around `t`



$$\min \sum_t \left(\|P(t) - C(t)\|^2 + \lambda_t \sum_{r \in \Omega_t} \omega_{t,r}(C) \cdot \|P(t) - P(r)\|^2 \right)$$

data term   smoothness term

Detour: bilateral filter

Slides adapted from Sylvain Paris

Objective of bilateral filtering

- Smooth texture
- Preserve edges

Illustration in 1D

Illustration in 1D

1D image = line of pixels

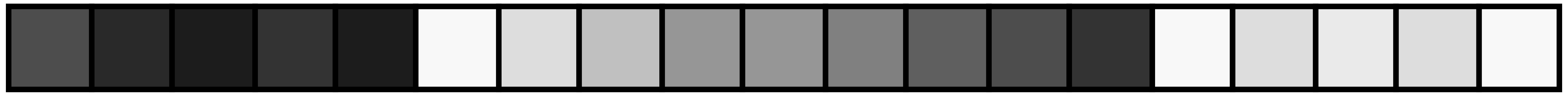
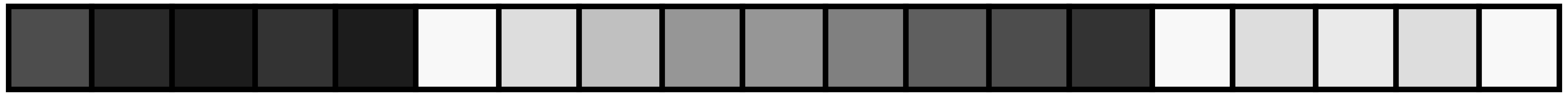
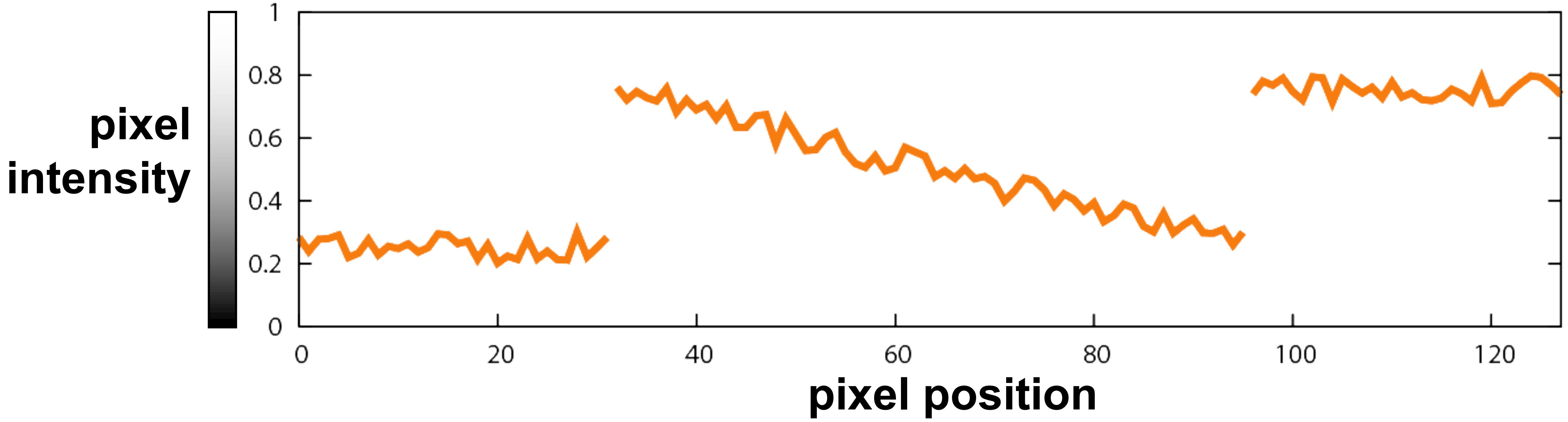


Illustration in 1D

1D image = line of pixels



Better visualized as a plot



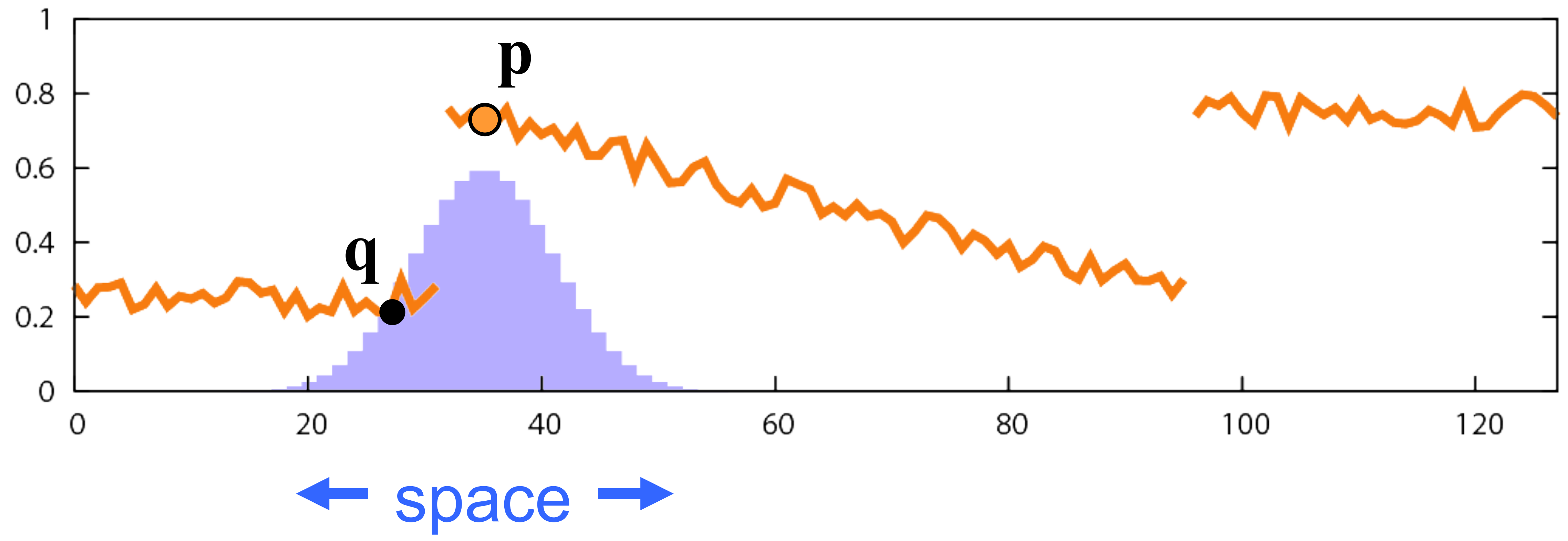
Definition

Definition

Gaussian blur

$$I_p = \sum_q G_{\sigma_s}(\|p - q\|) I_q$$

only spatial distance, intensity ignored

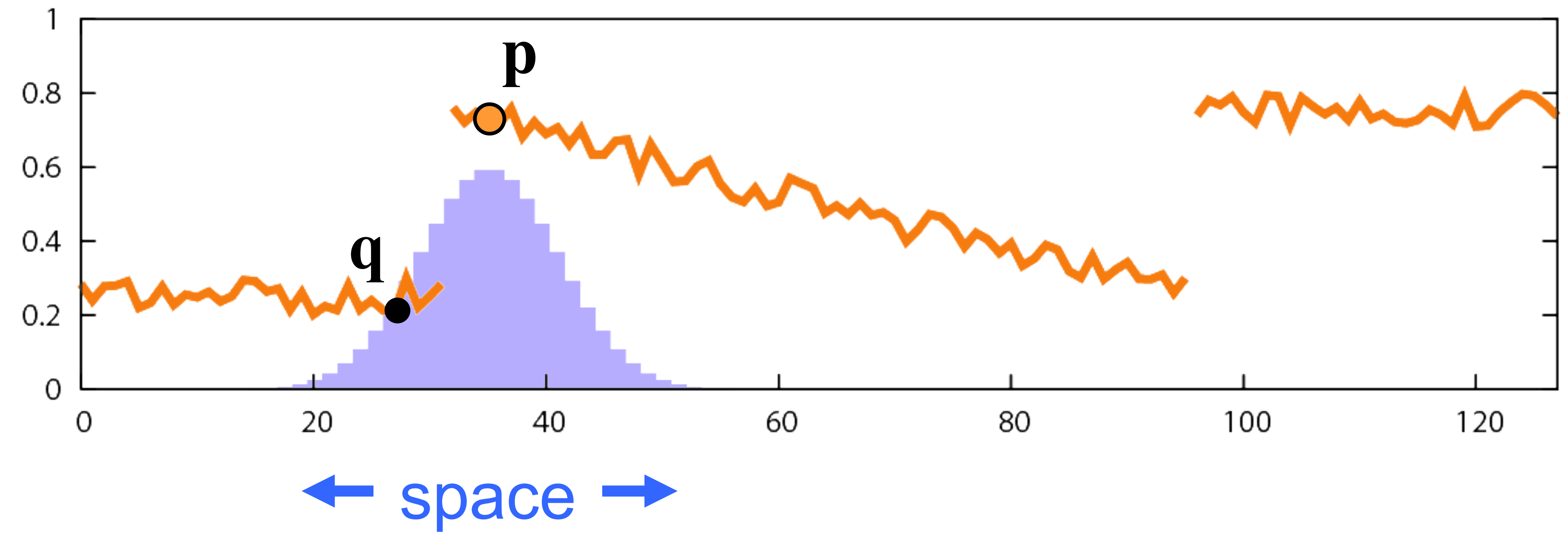


Definition

Gaussian blur

$$I_p = \sum_q G_{\sigma_s}(\|p - q\|) I_q$$

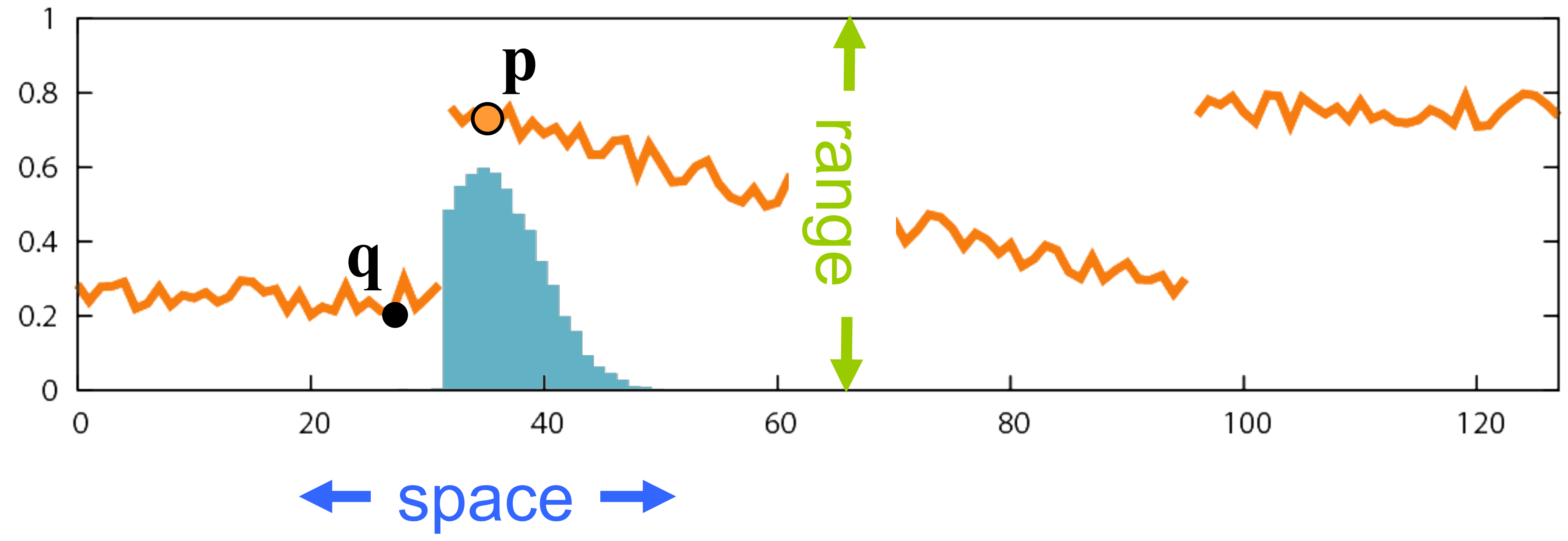
only spatial distance, intensity ignored



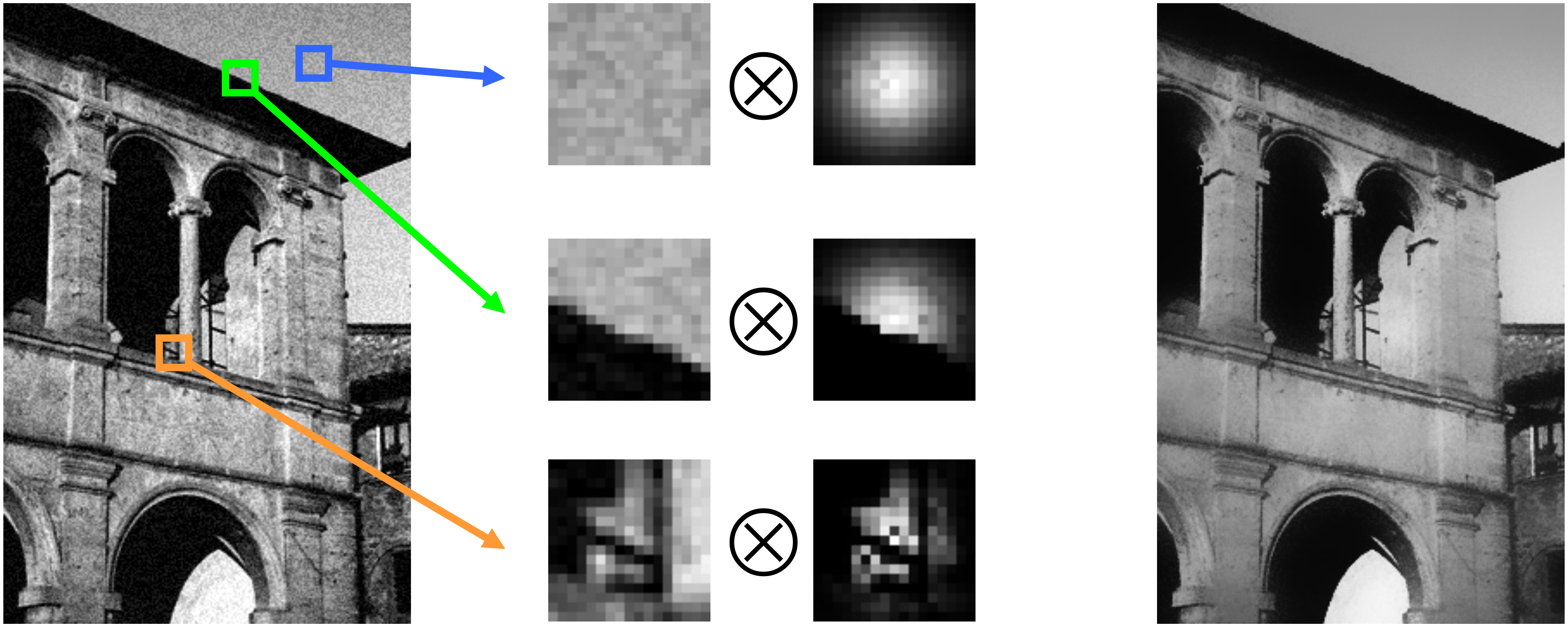
Bilateral filter [Aurich 95, Smith 97, Tomasi 98]

$$I_p = \frac{1}{W_p} \sum_q G_{\sigma_s}(\|p - q\|) G_{\sigma_r}(|I_p - I_q|) I_q$$

spatial and range distances



Example on a real image



Bilateral filter is not just for pixel values!

Bilateral filter is not just for pixel values!

Back to stabilization…

Optimizing a single path

$$\min \sum_t \left(\|P(t) - C(t)\|^2 + \lambda_t \sum_{r \in \Omega_t} \omega_{t,r}(C) \cdot \|P(t) - P(r)\|^2 \right)$$

data term

smoothness term

Optimizing a single path

$$\min \sum_t \left(\|P(t) - C(t)\|^2 + \lambda_t \sum_{r \in \Omega_t} \omega_{t,r}(C) \cdot \|P(t) - P(r)\|^2 \right)$$

data term **smoothness term**

$$\omega_{t,r} = G_t(\|r - t\|) \cdot G_m(\|C(r) - C(t)\|)$$

Optimizing a single path

$$\min \sum_t \left(\|P(t) - C(t)\|^2 + \lambda_t \sum_{r \in \Omega_t} \omega_{t,r}(C) \cdot \|P(t) - P(r)\|^2 \right)$$

data term

smoothness term

$$\omega_{t,r} = G_t(\|r - t\|) \cdot G_m(\|C(r) - C(t)\|)$$

distance between frames

Optimizing a single path

$$\min \sum_t \left(\|P(t) - C(t)\|^2 + \lambda_t \sum_{r \in \Omega_t} \omega_{t,r}(C) \cdot \|P(t) - P(r)\|^2 \right)$$

data term

smoothness term

distance between camera poses

$$\omega_{t,r} = G_t(\|r - t\|) \cdot G_m(\|C(r) - C(t)\|)$$

distance between frames

Optimizing a single path

$$\min \sum_t \left(\|P(t) - C(t)\|^2 + \lambda_t \sum_{r \in \Omega_t} \omega_{t,r}(C) \cdot \|P(t) - P(r)\|^2 \right)$$

data term　　smoothness term

setting the weights λ_t

Optimizing a single path

$$\min \sum_t \left(\|P(t) - C(t)\|^2 + \lambda_t \sum_{r \in \Omega_t} \omega_{t,r}(C) \cdot \|P(t) - P(r)\|^2 \right)$$

data term

smoothness term

setting the weights λ_t

Run optimization with global weight

For each frame

While too much cropping or distortion

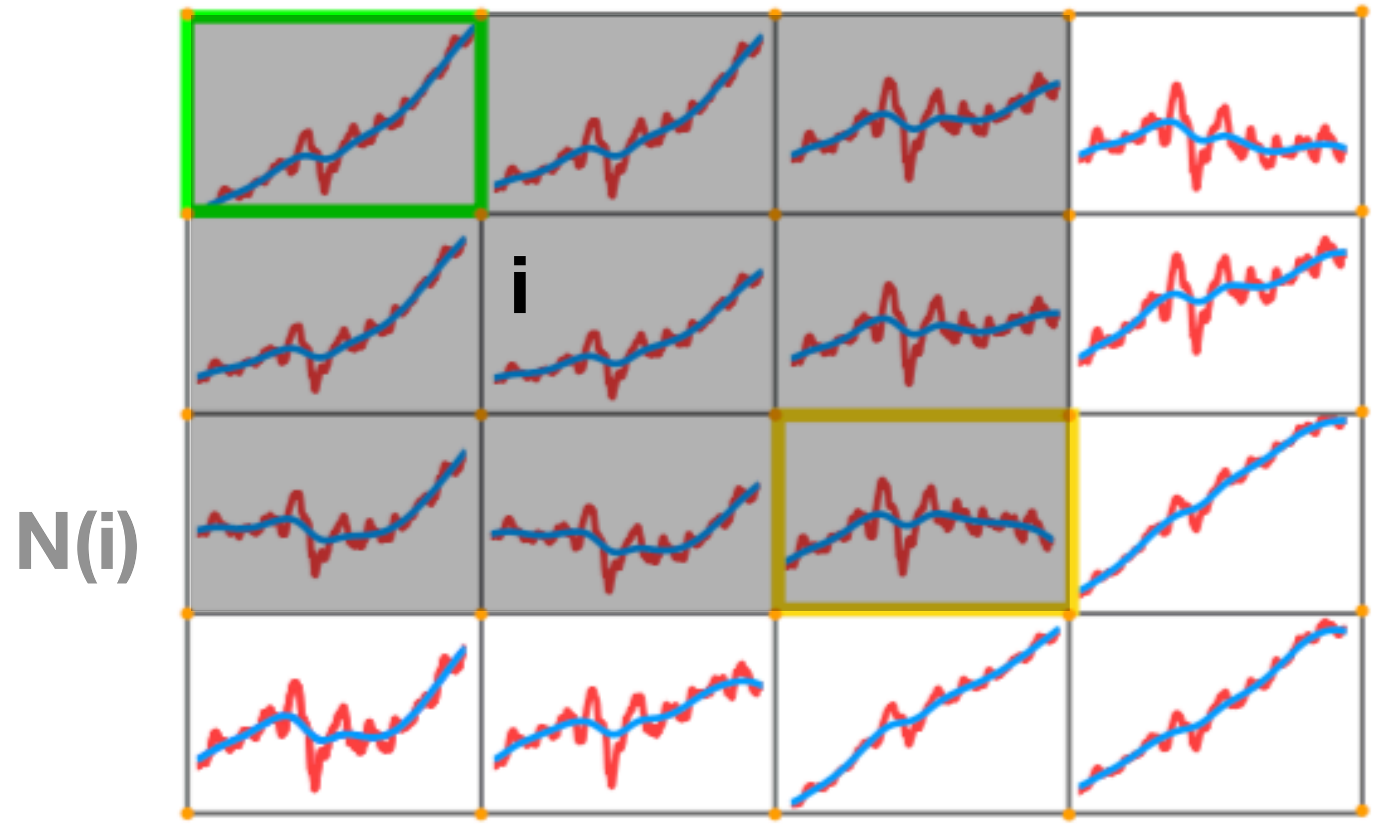
Decrease weight and re-run

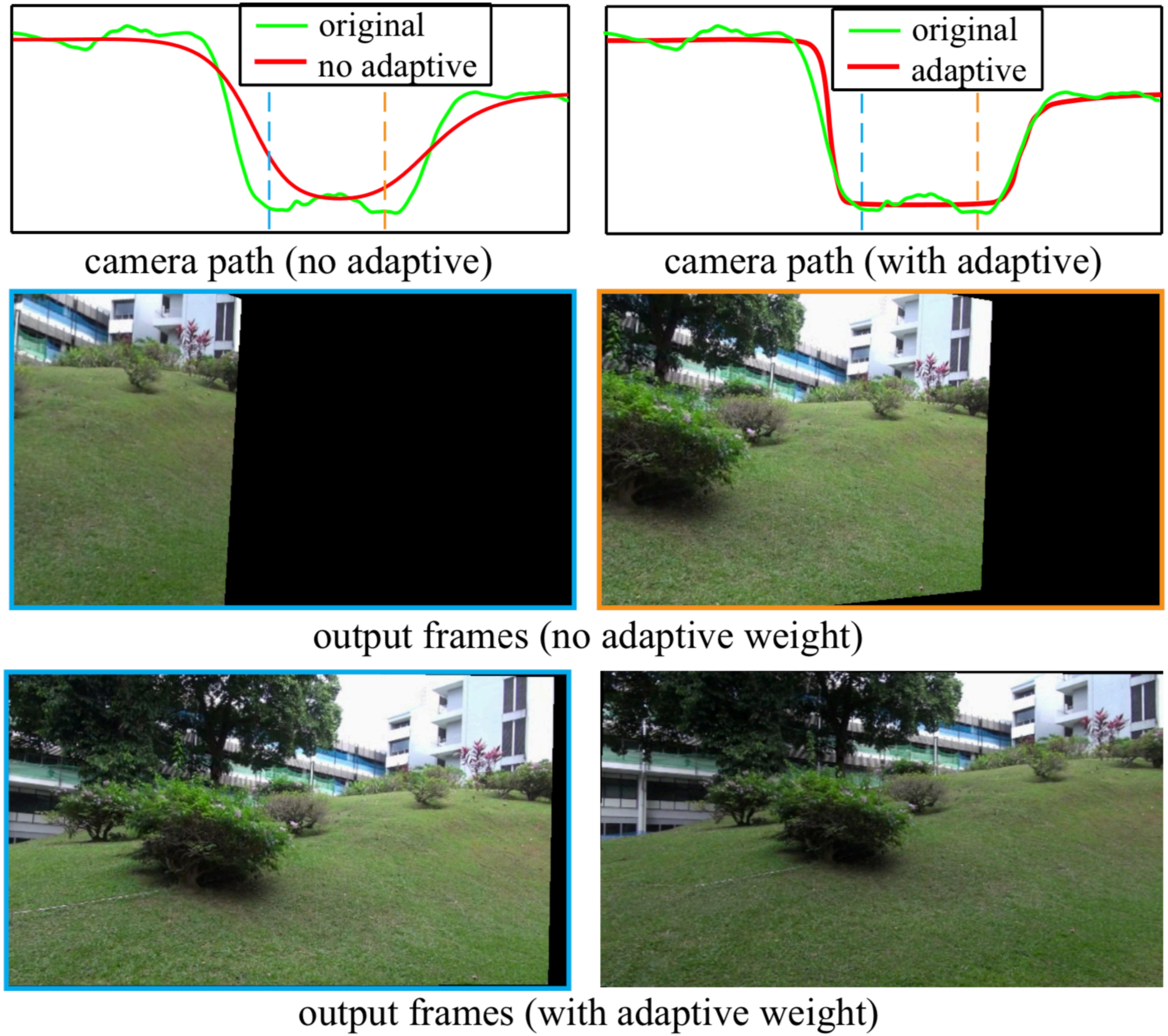
Optimizing bundled paths

Optimizing bundled paths

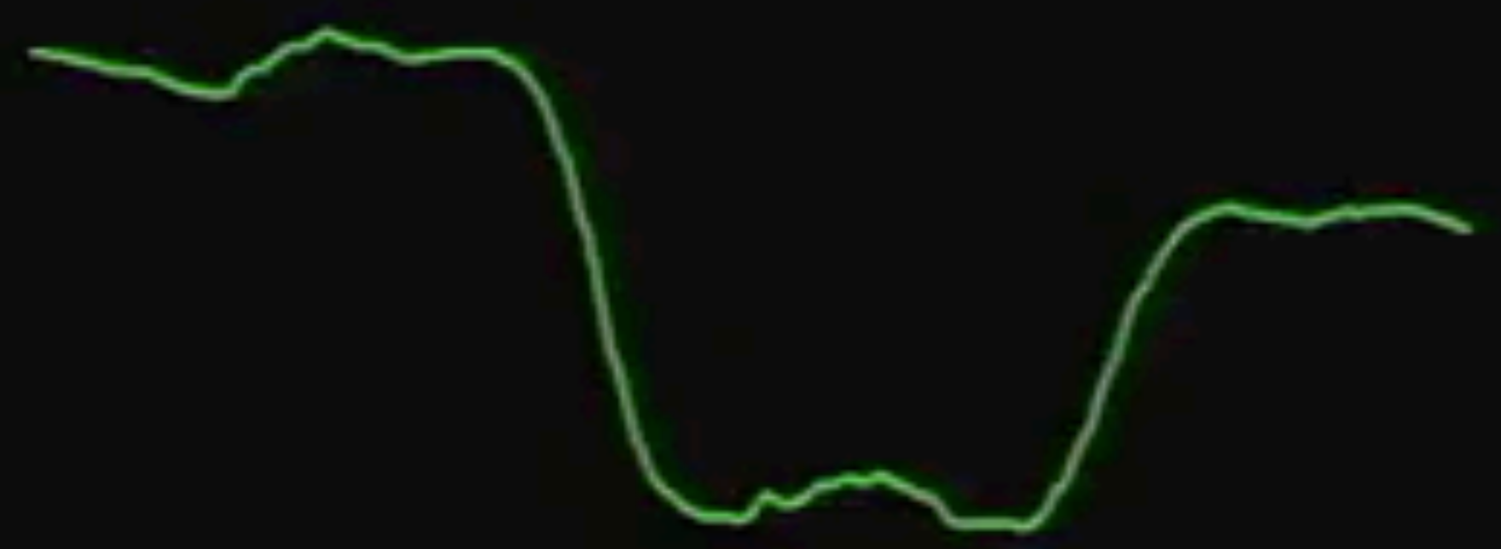
single path — smoothness between neighboring paths

$$\min \sum_i O(\{P_i(t)\}) + \sum_t \sum_{j \in N(i)} \|P_i(t) - P_j(t)\|^2$$



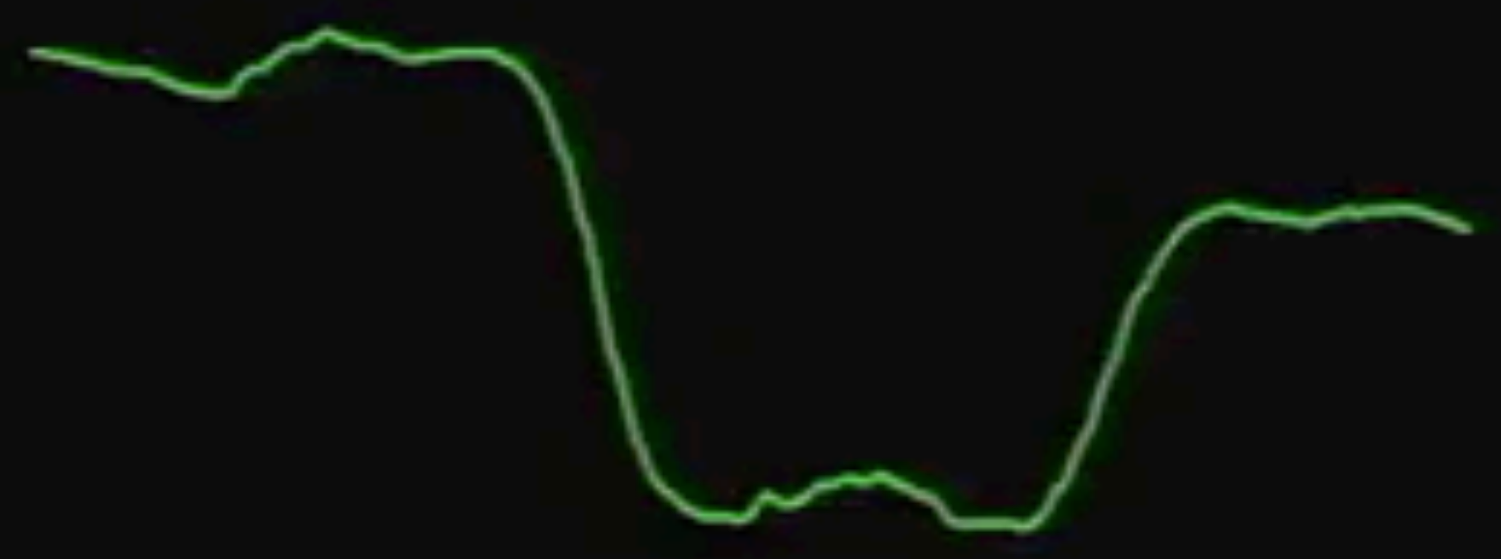

original
no adaptive
camera path (no adaptive)
original
adaptive
camera path (with adaptive)
output frames (no adaptive weight)
output frames (with adaptive weight)

camera path



original

camera path



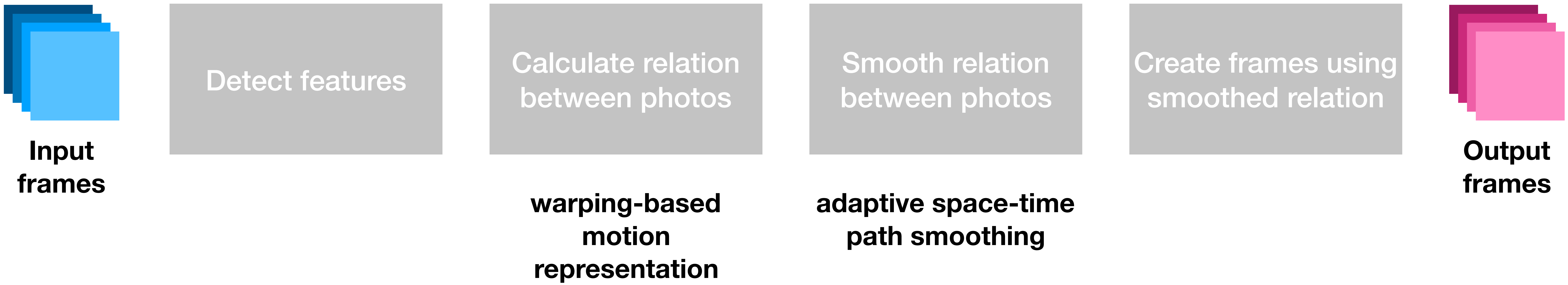
original

without spatial constraint

with spatial constraint

without spatial constraint

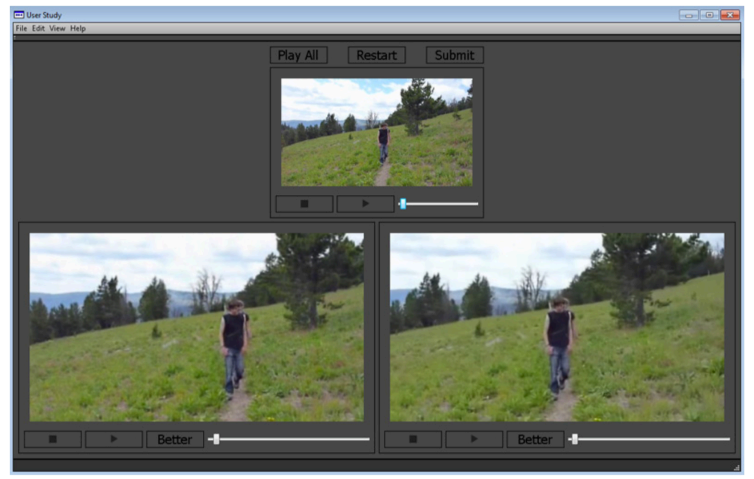
with spatial constraint


Input frames
Detect features
Calculate relation between photos
Smooth relation between photos
Create frames using smoothed relation
Output frames
warping-based motion representation
adaptive space-time path smoothing

Evaluation & Results

Comparison to previous methods

Comparison to commercial products



User study


(I) simple
(II) quick rotation
(III) zooming
(IV) large parallax
(V) driving
(VI) crowd
(VII) running
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
Google YouTube
Adobe After Effect CS6 ('Warp Stabilizer')
Ours

input

single path

bundled paths

input

single path

bundled paths

input:

Subspace [Liu et. al. 2011]

our result

input:

Subspace [Liu et. al. 2011]

our result

input:

3D Warp [Liu et. al. 2009]

our result

input:

3D Warp [Liu et. al. 2009]

our result

input:

Epipolar [Goldstein and Fattal 2012]

our result

input:

Epipolar [Goldstein and Fattal 2012]

our result

input:

L1 path [Grundmann, et. al. 2011]

our result

input:

L1 path [Grundmann, et. al. 2011]

our result

input:
(IV-4)

YouTube result

our result

input:
(IV-4)

YouTube result

our result

input:
(IV-2)

After Effect CS6 result

our result

input:
(IV-2)

After Effect CS6 result

our result

input videos

Homography mixture
[*Grundmann et. al. 2012*]

our result

input videos

Homography mixture
[*Grundmann et. al. 2012*]

our result

input videos | Homography mixture [*Grundmann et. al. 2012*] | our result

input videos | Homography mixture [*Grundmann et. al. 2012*] | our result

always check supplemental…

always check supplemental…

always check supplemental…

always check supplemental…

Recap

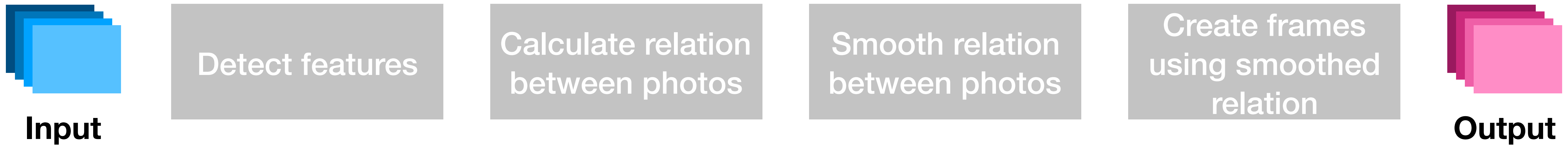
Recap

- Video stabilization is important!

Recap

- Video stabilization is important!

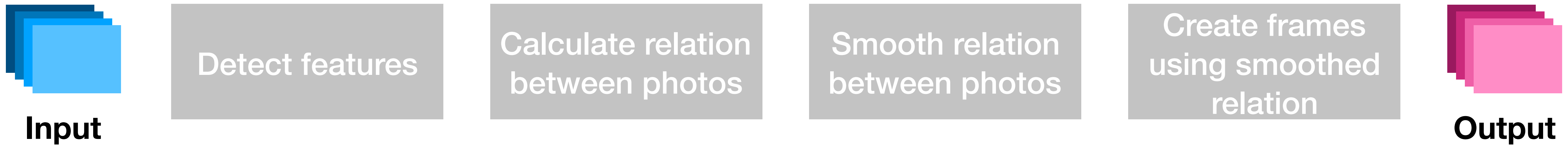
- General recipe for stabilization



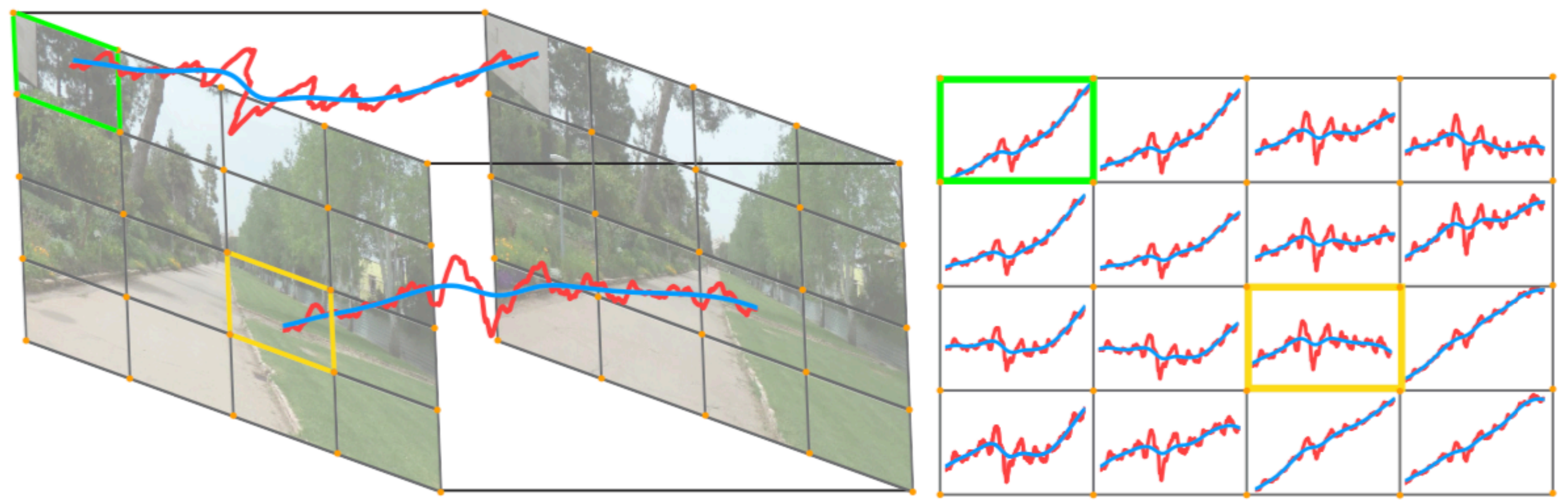
Recap

- Video stabilization is important!

- General recipe for stabilization



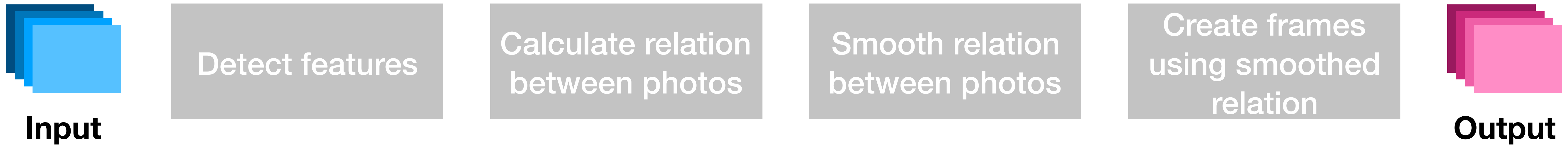
- [Liu et al. ’13]



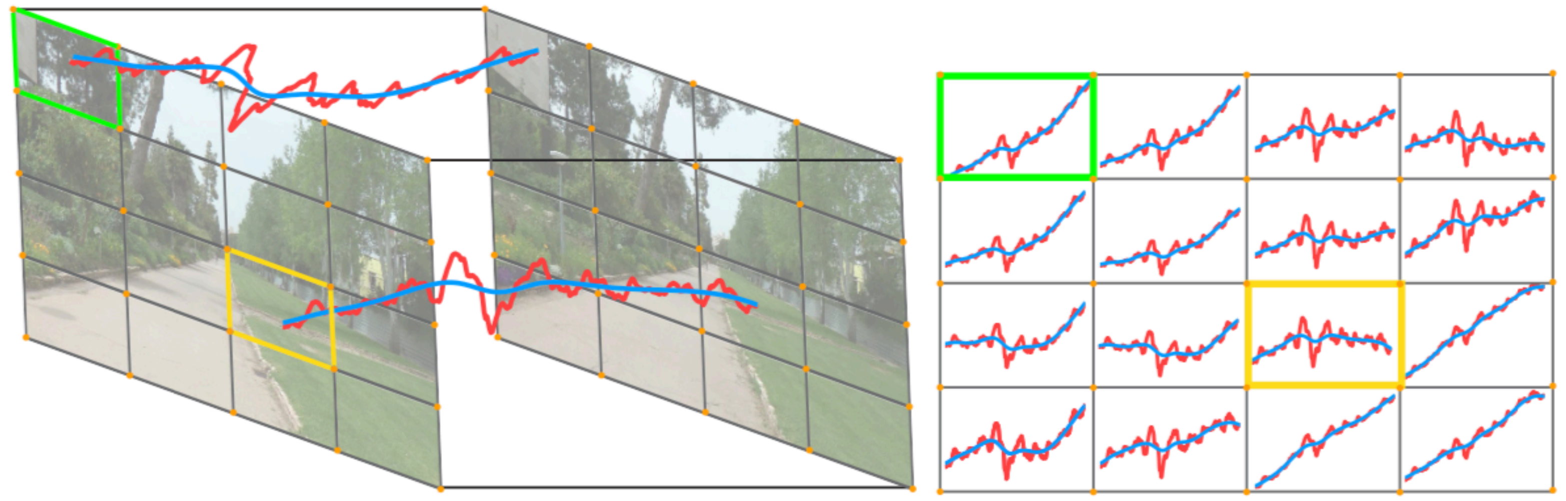
Recap

- Video stabilization is important!

- General recipe for stabilization



- [Liu et al. ’13]



- Bilateral filter