Image Stitching and Homography

Gundimeda Venugopal

VIT-AP UNIVERSITY

Computer Vision

(Course Code: 4047)

Module-5:Lecture-1: Image Stitching and Homography

Gundimeda Venugopal, Professor of Practice, SCOPE

What is Image Stitching?

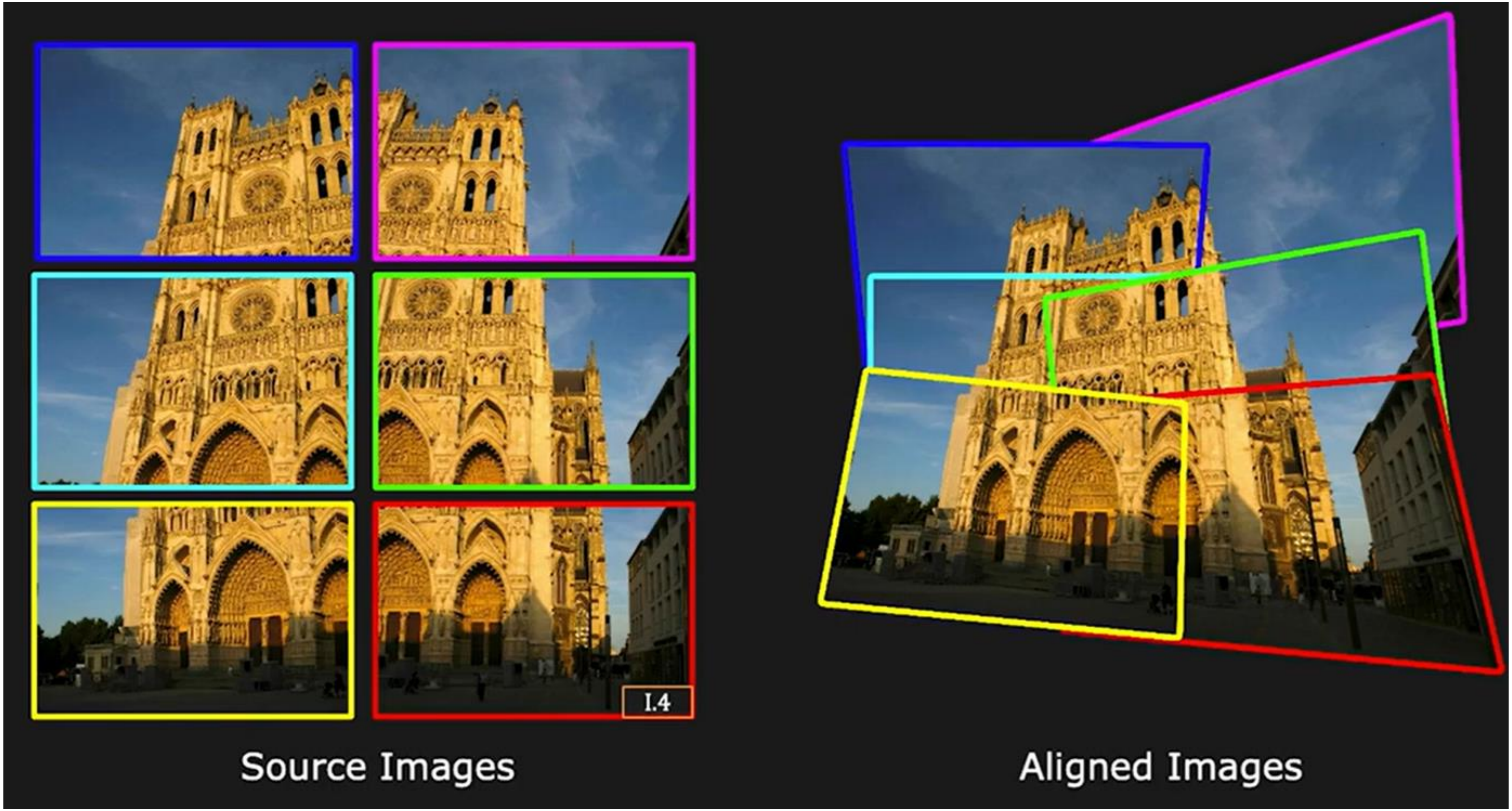


Image Stitching

Image Stitching

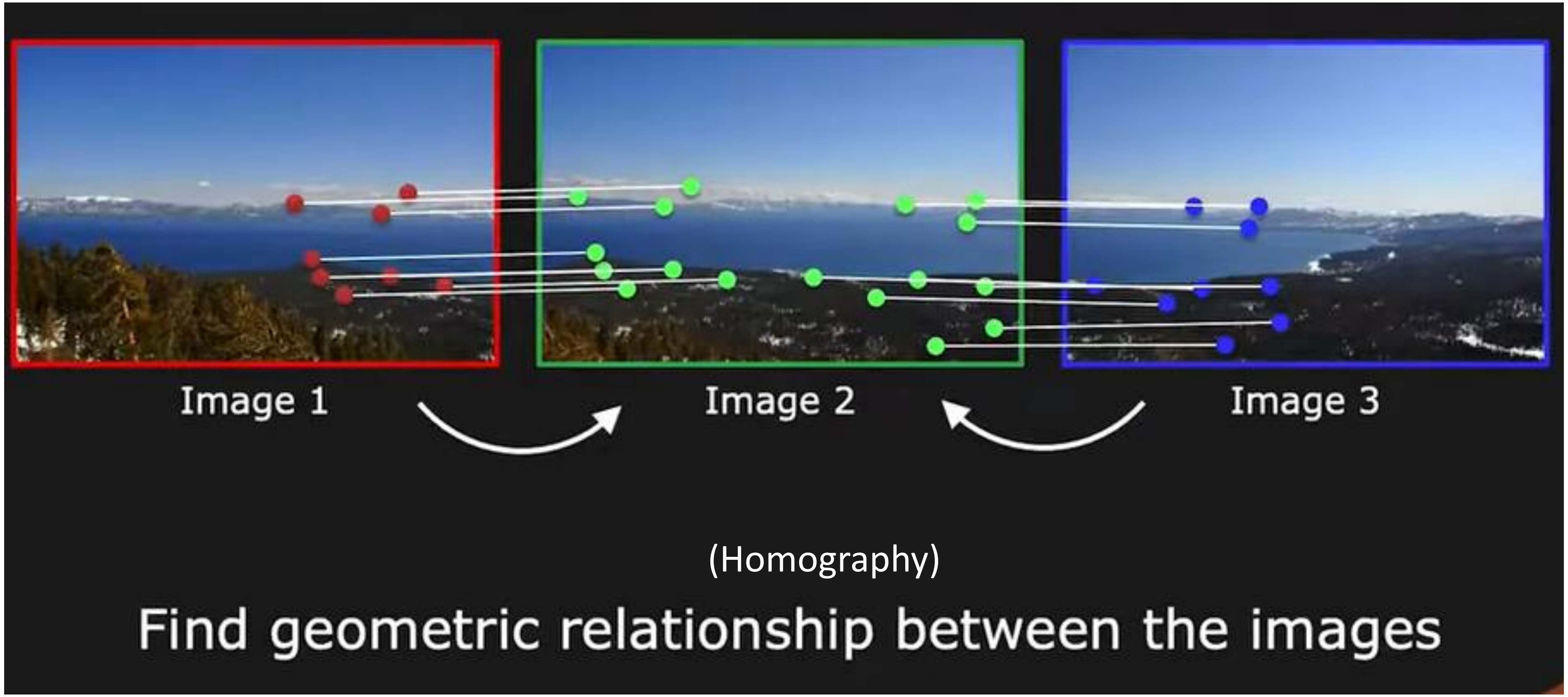


Image Stitching: Steps involved in Image Stitching

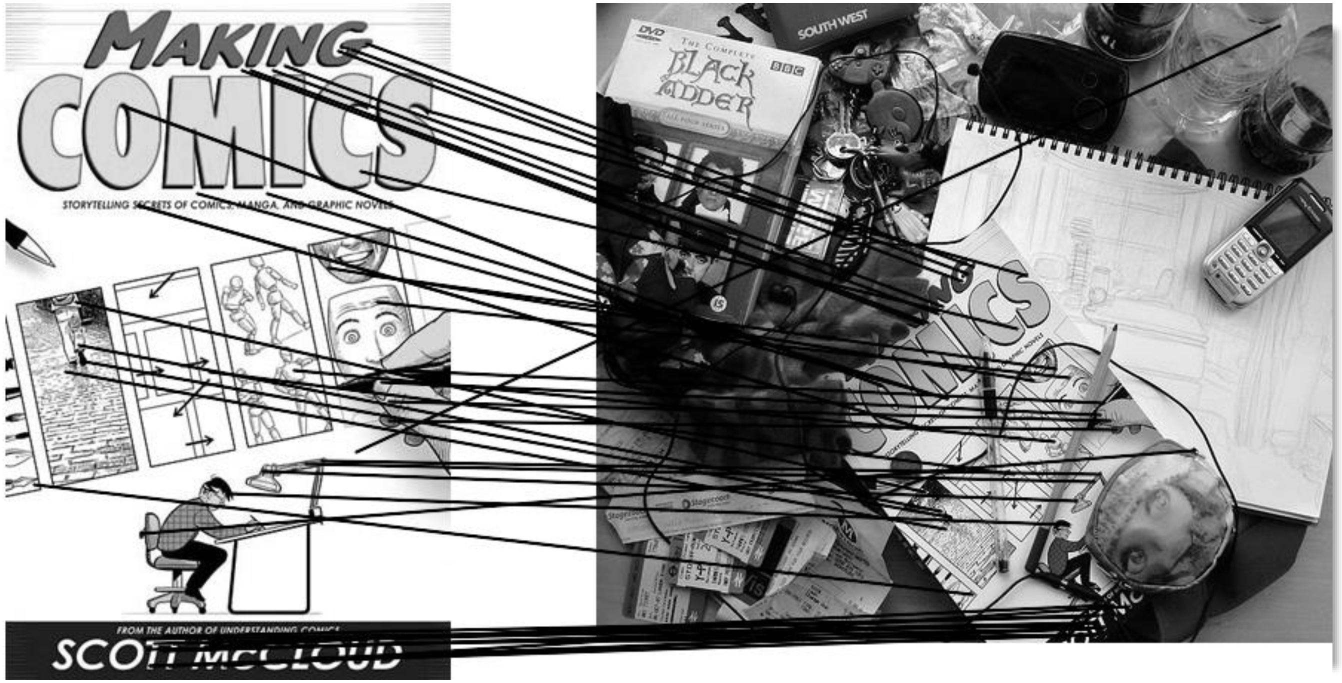
- Image stitching is a process in computer vision and image processing where multiple images, typically of overlapping scenes, are combined to produce a single panoramic image. This technique involves aligning and blending the images to create a seamless and high-resolution composite.
- Here are the key steps involved in image stitching:

1. **Image Acquisition**: Capture multiple images of the scene with overlapping areas. These images are usually taken with a consistent orientation and similar exposure settings.
2. **Feature Detection**: Identify distinctive features (like corners, edges, or specific patterns) in each image. Common algorithms for this task include SIFT (Scale-Invariant Feature Transform), SURF (Speeded-Up Robust Features).
3. **Feature Matching**: Corresponding features between overlapping images are matched. This step aligns the images by finding pairs of similar features.
4. **Homography Estimation**: Compute a transformation matrix (homography) that aligns one image with the next. This matrix describes how to warp one image to match the perspective of another.
5. **Image Warping and Alignment**: Apply the homography matrix to warp images into a common coordinate frame so that they overlap correctly.
6. **Blending**: Seamlessly blend the overlapping areas to reduce visible seams and ensure a smooth transition between images. Techniques like feathering, multi-band blending, and exposure compensation are often used.
7. **Rendering**: Combine the aligned and blended images into a single panoramic image. This may involve cropping to remove unwanted edges and adjusting the final image's exposure and color balance.

- Image stitching is widely used in various applications, such as creating panoramic photos in consumer cameras, generating large-scale maps from satellite images, and constructing virtual environments in video games and simulations.

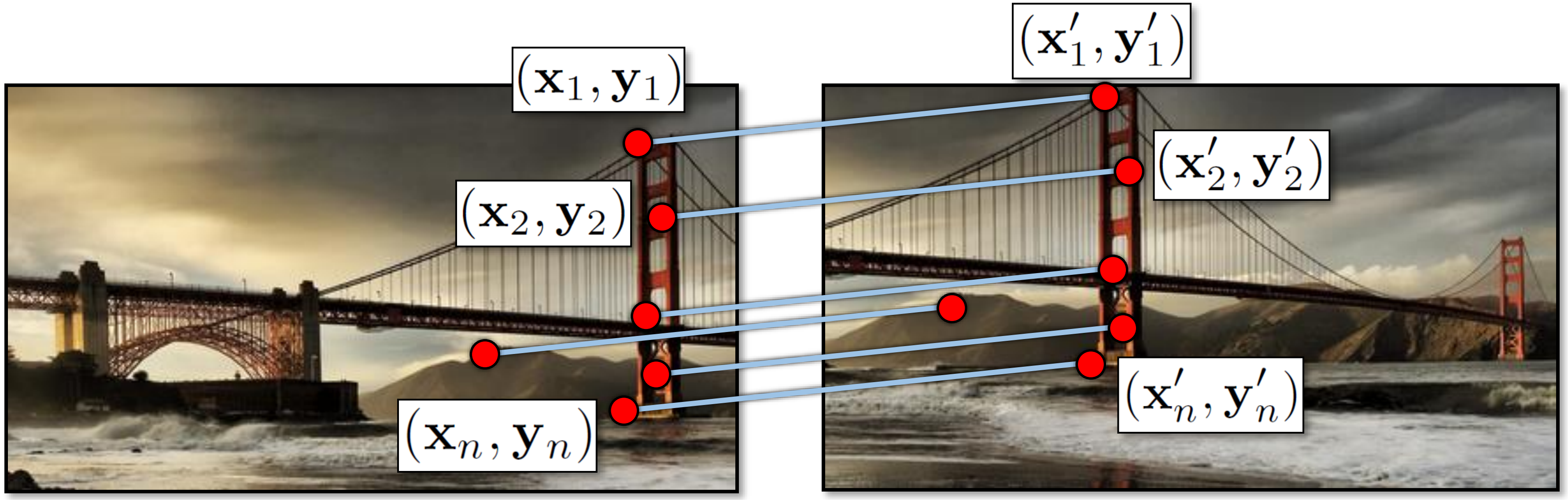
Computing transformations

- Image alignment problem
- Given a set of matches between images A and B
  - How can we compute the transform T from A to B?



  - Find transform T that best “agrees” with the matches

Simple case: translations

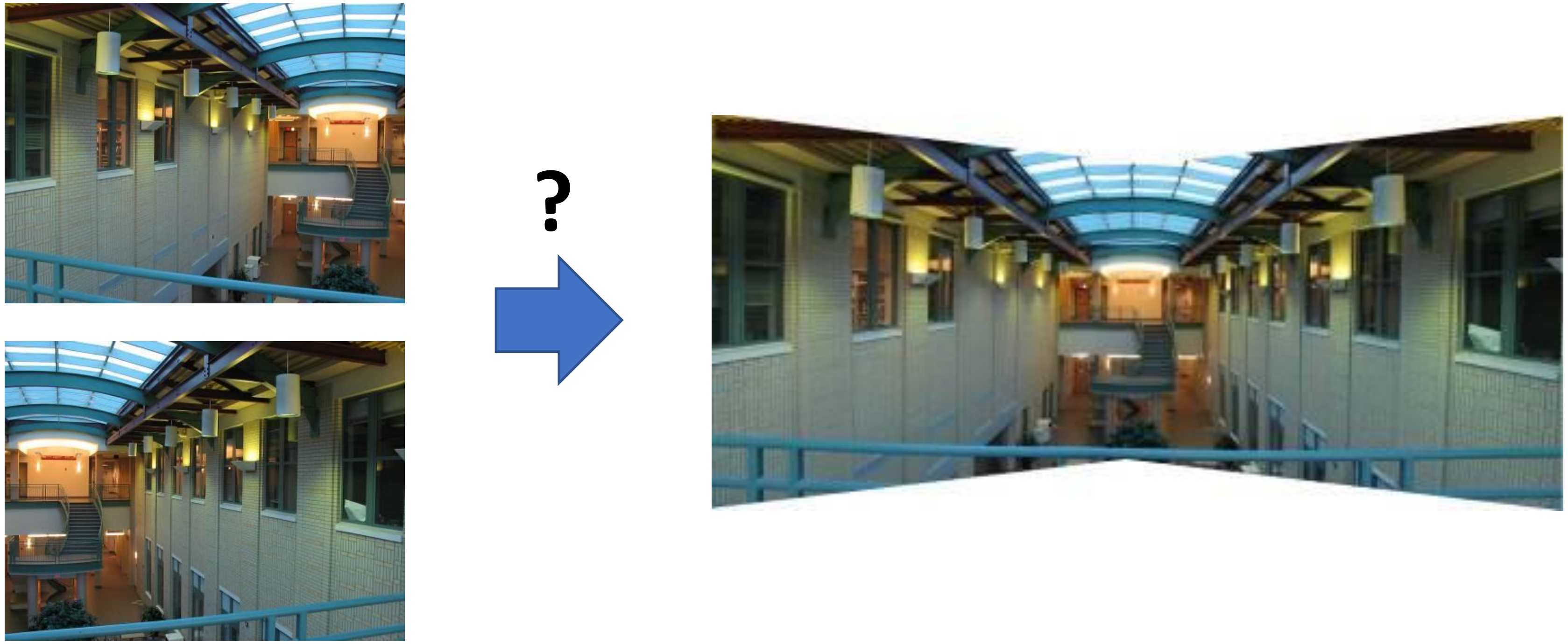


Displacement of match *i* = $(\mathbf{x}'_i - \mathbf{x}_i, \mathbf{y}'_i - \mathbf{y}_i)$

$$(\mathbf{x}_t, \mathbf{y}_t) = \left(\frac{1}{n} \sum_{i=1}^{n} \mathbf{x}'_i - \mathbf{x}_i, \frac{1}{n} \sum_{i=1}^{n} \mathbf{y}'_i - \mathbf{y}_i \right)$$

Computing transformations

Image Stitching (Perspective Transformation)

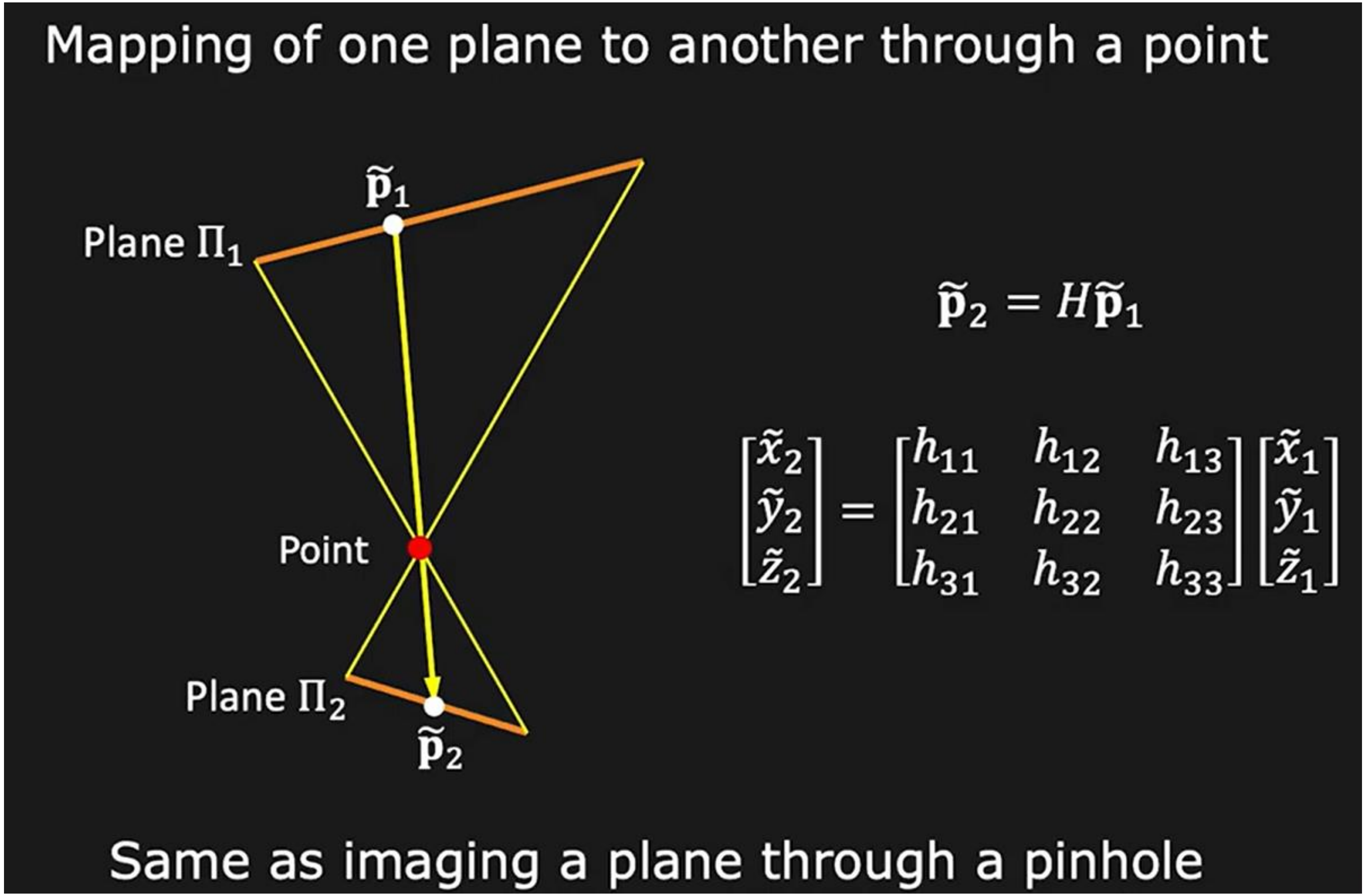


Projective Transformation

Any transformation of the form:

$$\begin{bmatrix}\tilde{x}_2\\ \tilde{y}_2\\ \tilde{z}_2\end{bmatrix} = \begin{bmatrix}h_{11} & h_{12} & h_{13}\\ h_{21} & h_{22} & h_{23}\\ h_{31} & h_{32} & h_{33}\end{bmatrix}\begin{bmatrix}\tilde{x}_1\\ \tilde{y}_1\\ \tilde{z}_1\end{bmatrix} \qquad \tilde{\mathbf{p}}_2 = H\tilde{\mathbf{p}}_1$$

Projective Transformation



Projective Transformation

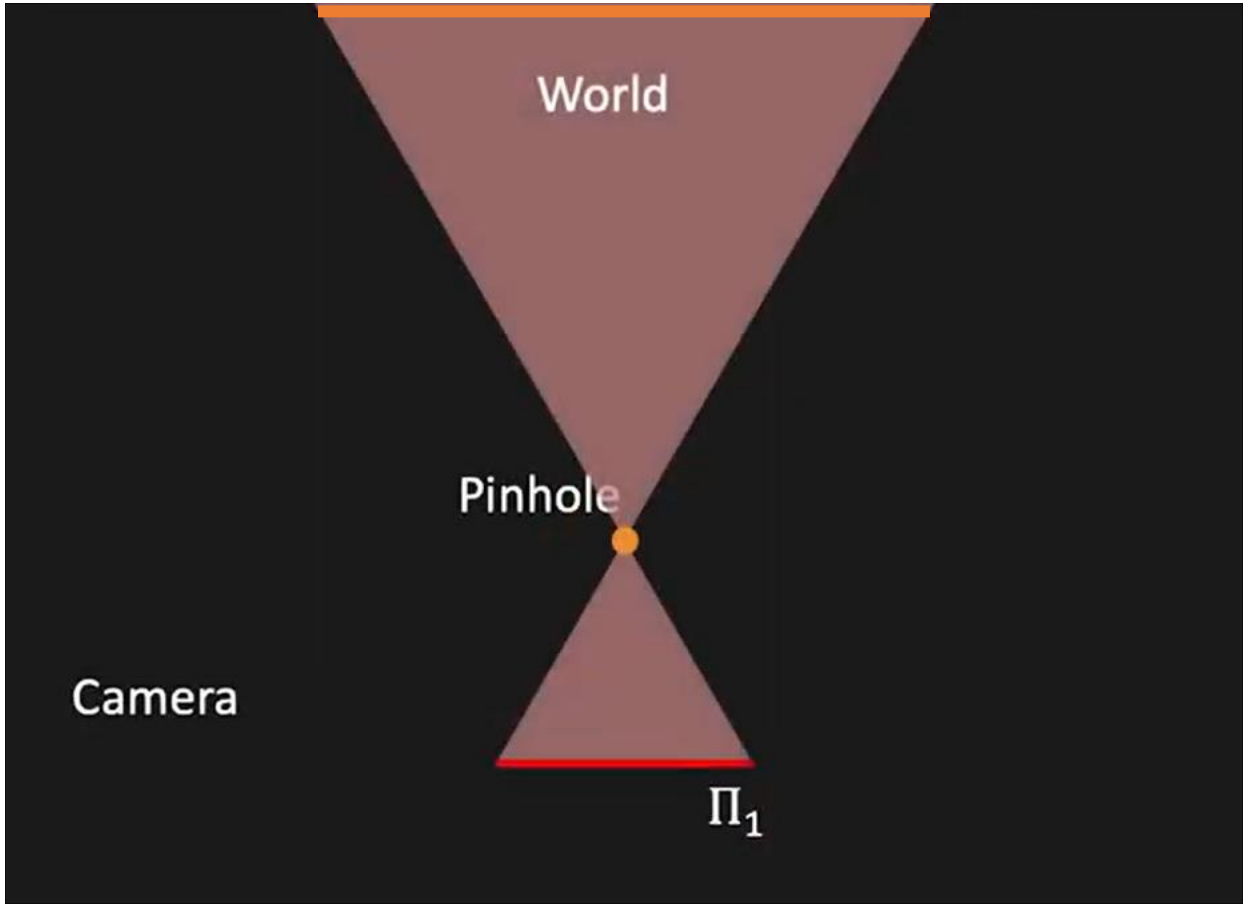
Homography can only be defined up to a scale.

$$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} \tilde{x}_1 \\ \tilde{y}_1 \\ \tilde{z}_1 \end{bmatrix} \equiv \begin{bmatrix} \tilde{x}_2 \\ \tilde{y}_2 \\ \tilde{z}_2 \end{bmatrix} \equiv k \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} \tilde{x}_1 \\ \tilde{y}_1 \\ \tilde{z}_1 \end{bmatrix}$$

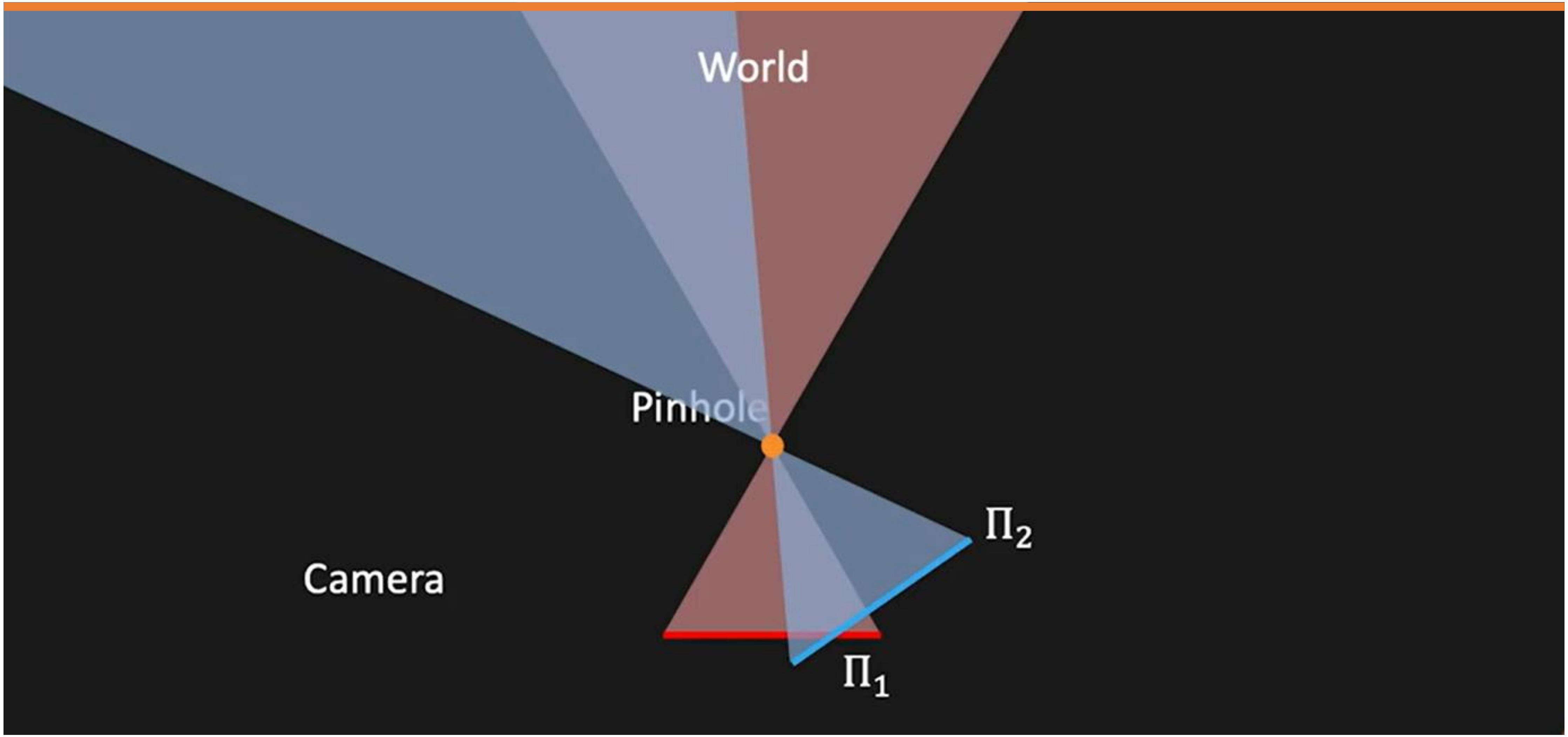
If we fix scale such that $\sqrt{\Sigma(h_{ij})^2} = 1$ then 8 free parameters

- Origin does not necessarily map to the origin
- Lines map to lines
- Parallel lines do not necessarily remain parallel
- Closed under composition

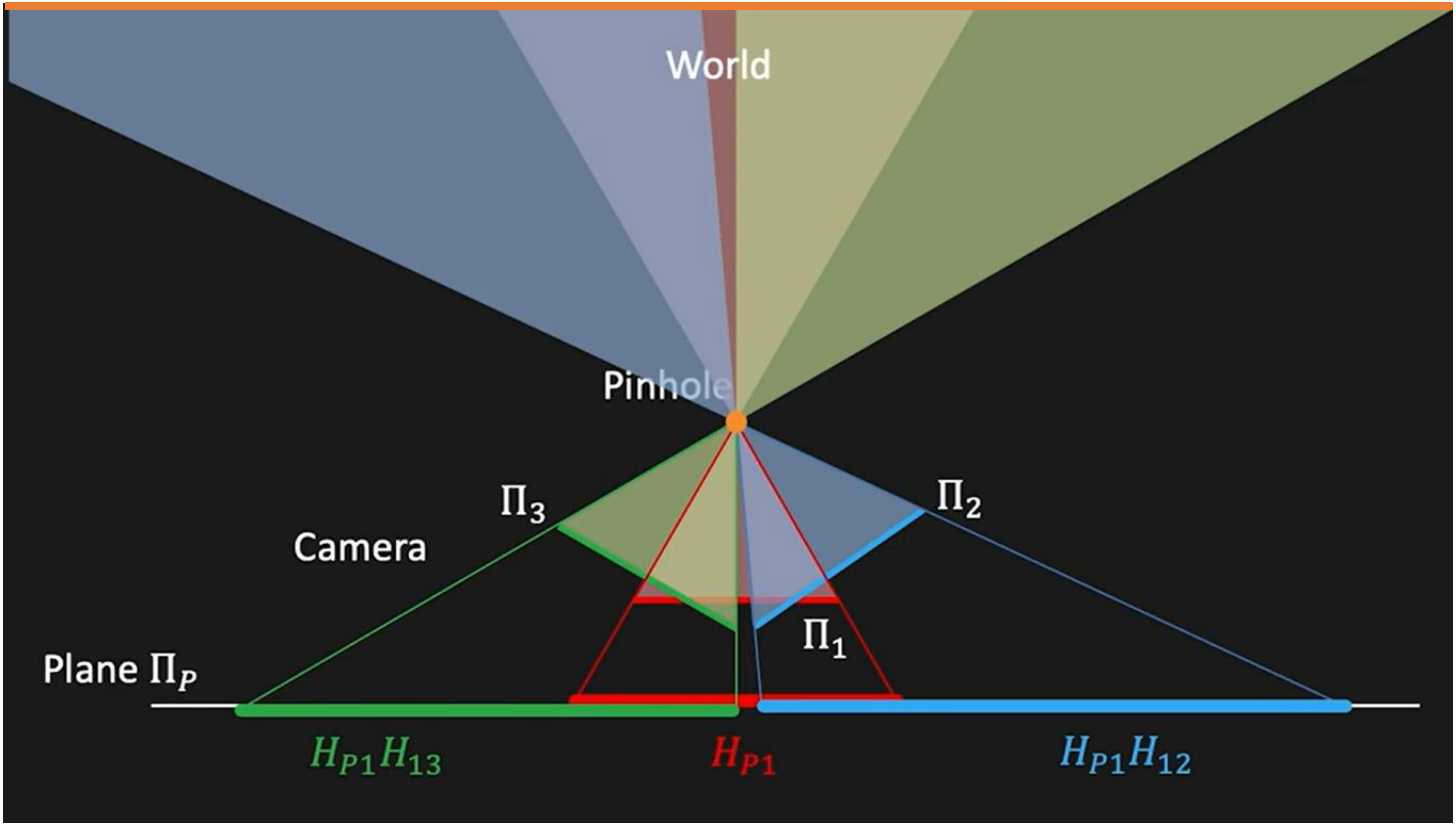
Homography Composition



Homography Composition



Homography Composition



Homography Composition

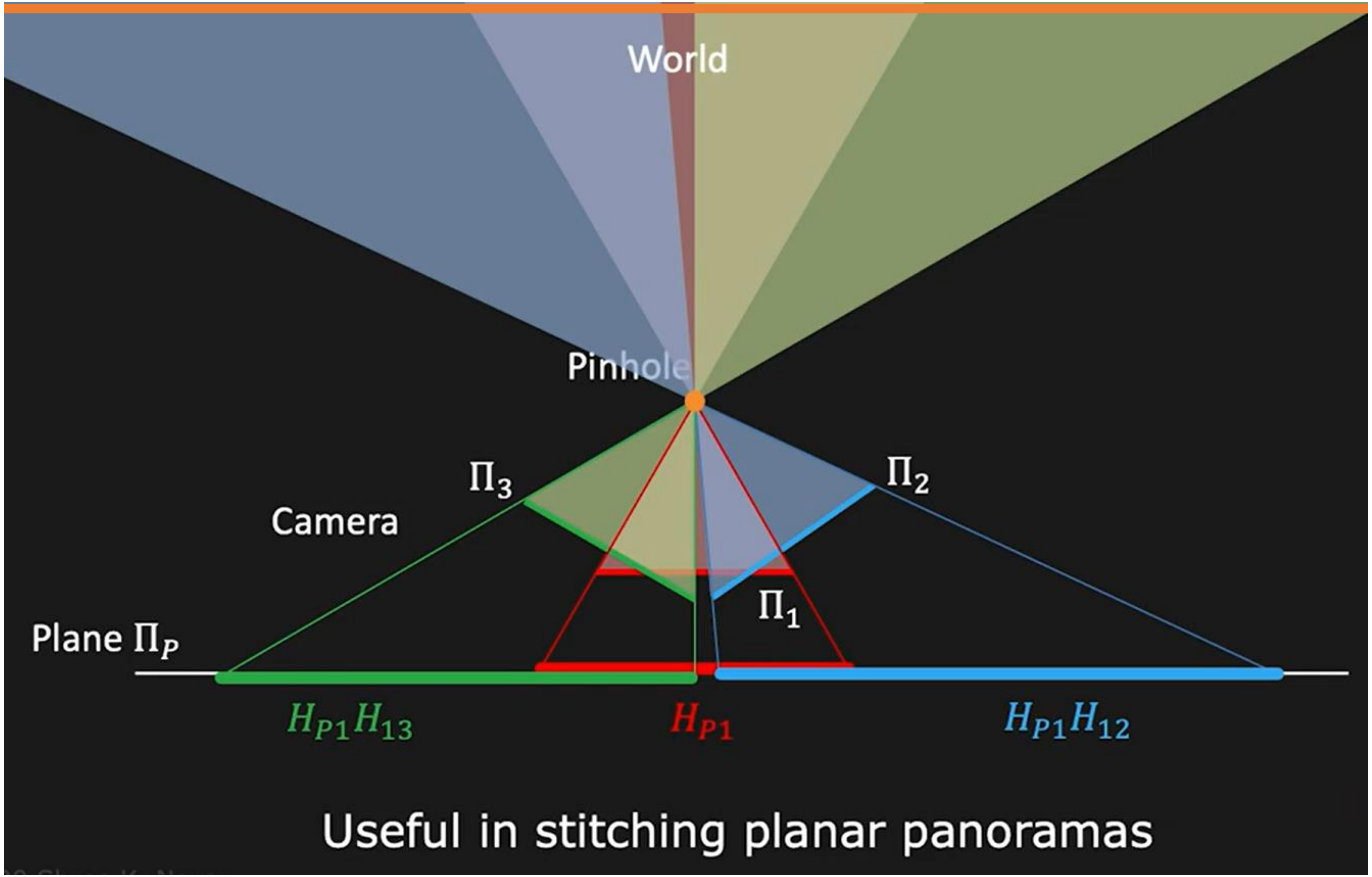
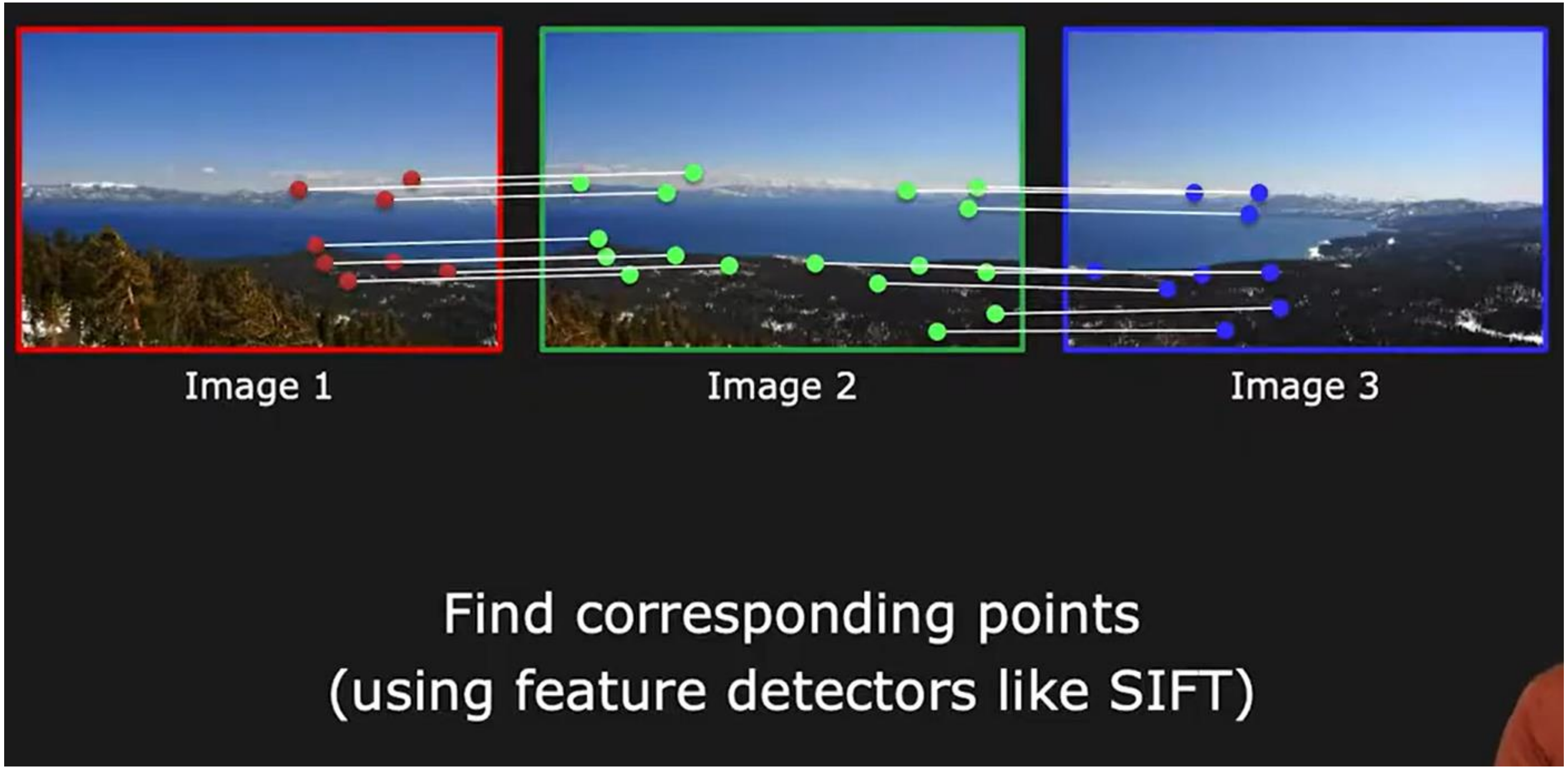
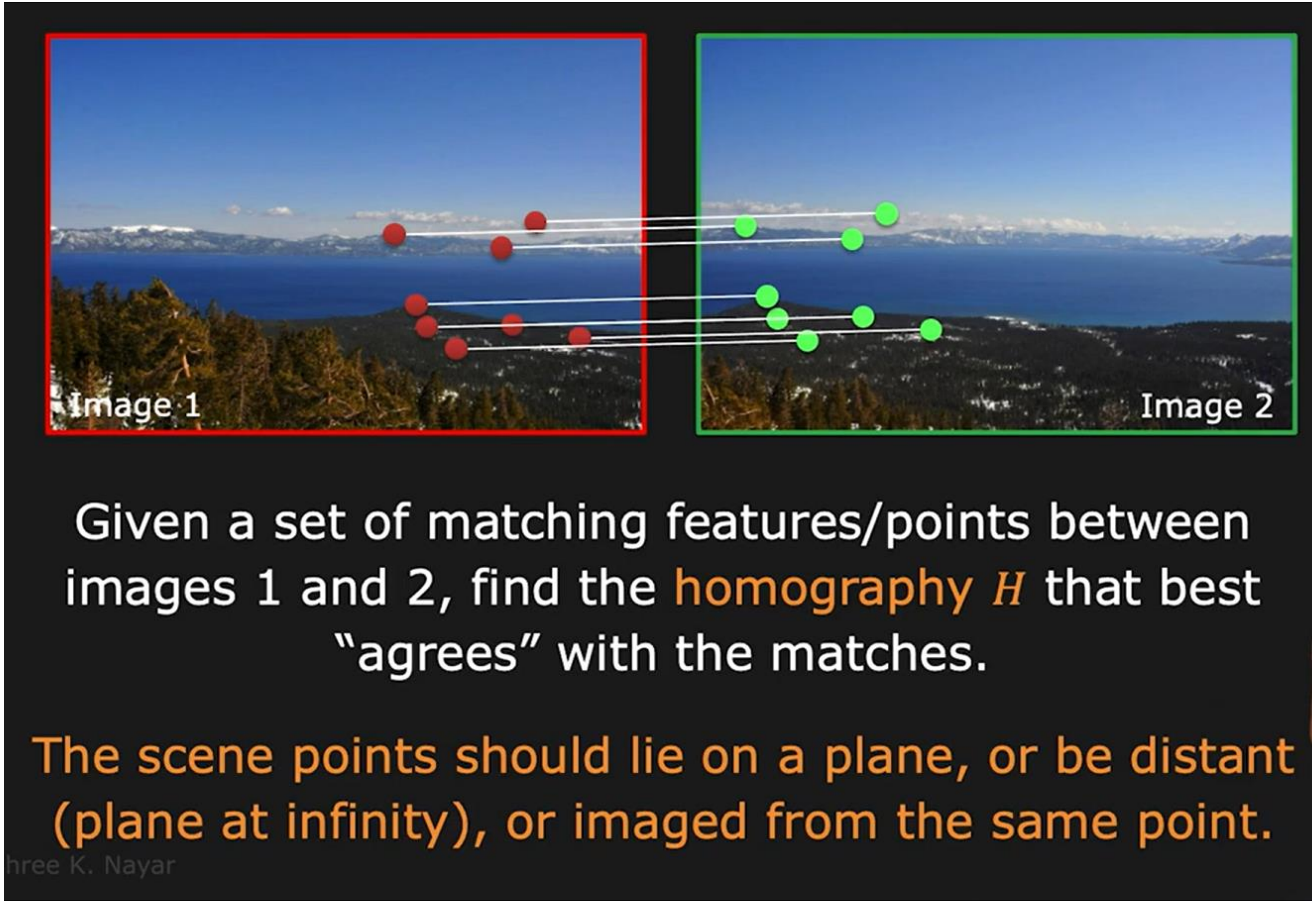


Image Stitching

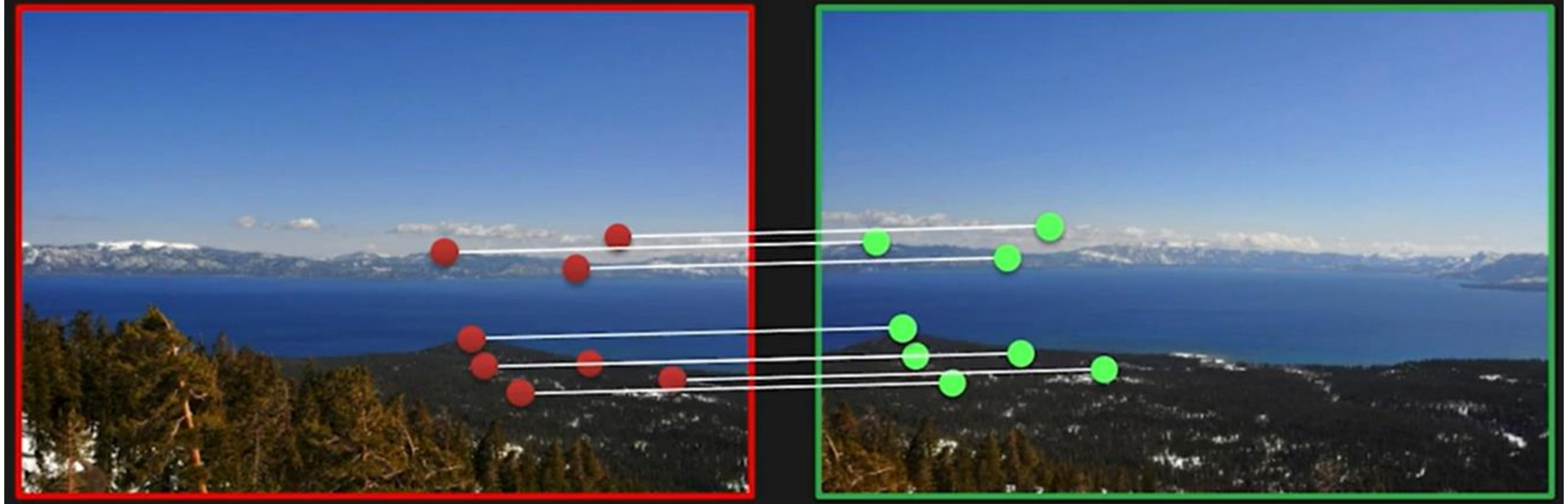
1. Capture multiple images of the scene with overlapping areas. These images are usually taken with a consistent orientation and similar exposure settings.
2. Extract features using SIFT or SURF



Computing Homography



Computing Homography



Source Image

Destination Image

Homogeneous
Coordinate system

$$\begin{bmatrix} x_d \\ y_d \\ 1 \end{bmatrix} \equiv \begin{bmatrix} \tilde{x}_d \\ \tilde{y}_d \\ \tilde{z}_d \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_s \\ y_s \\ 1 \end{bmatrix}$$

How many unknowns? 9 ...But 8 degrees of freedom

How many minimum pairs of matching points? 4

Computing Homography

For a given pair i of corresponding points:

$$x_d{}^{(i)} = \frac{\tilde{x}_d{}^{(i)}}{\tilde{z}_d{}^{(i)}} = \frac{h_{11}x_s^{(i)} + h_{12}y_s^{(i)} + h_{13}}{h_{31}x_s^{(i)} + h_{32}y_s^{(i)} + h_{33}}$$

$$y_d{}^{(i)} = \frac{\tilde{y}_d{}^{(i)}}{\tilde{z}_d{}^{(i)}} = \frac{h_{21}x_s^{(i)} + h_{22}y_s^{(i)} + h_{23}}{h_{31}x_s^{(i)} + h_{32}y_s^{(i)} + h_{33}}$$

Rearranging the terms:

$$x_d{}^{(i)}\left(h_{31}x_s^{(i)} + h_{32}y_s^{(i)} + h_{33}\right) = h_{11}x_s^{(i)} + h_{12}y_s^{(i)} + h_{13}$$

$$y_d{}^{(i)}\left(h_{31}x_s^{(i)} + h_{32}y_s^{(i)} + h_{33}\right) = h_{21}x_s^{(i)} + h_{22}y_s^{(i)} + h_{23}$$

Computing Homography

$$x_d^{(i)}\left(h_{31}x_s^{(i)} + h_{32}y_s^{(i)} + h_{33}\right) = h_{11}x_s^{(i)} + h_{12}y_s^{(i)} + h_{13}$$

$$y_d^{(i)}\left(h_{31}x_s^{(i)} + h_{32}y_s^{(i)} + h_{33}\right) = h_{21}x_s^{(i)} + h_{22}y_s^{(i)} + h_{23}$$

Rearranging the terms and writing as linear equation:

$$\underbrace{\boxed{\begin{bmatrix} x_s^{(i)} & y_s^{(i)} & 1 & 0 & 0 & 0 & -x_d^{(i)}x_s^{(i)} & -x_d^{(i)}y_s^{(i)} & -x_d^{(i)} \\ 0 & 0 & 0 & x_s^{(i)} & y_s^{(i)} & 1 & -y_d^{(i)}x_s^{(i)} & -y_d^{(i)}y_s^{(i)} & -y_d^{(i)} \end{bmatrix}}}_{\text{(Known)}} \underbrace{\boxed{\begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix}}}_{\substack{\mathbf{h} \\ \text{(Unknown)}}} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Compute Homography

Combining the equations for all corresponding points:

$$
\underbrace{\begin{bmatrix}
x_s^{(1)} & y_s^{(1)} & 1 & 0 & 0 & 0 & -x_d^{(1)}x_s^{(1)} & -x_d^{(1)}y_s^{(1)} & -x_d^{(1)} \\
0 & 0 & 0 & x_s^{(1)} & y_s^{(1)} & 1 & -y_d^{(1)}x_s^{(1)} & -y_d^{(1)}y_s^{(1)} & -y_d^{(1)} \\
 & & & & \vdots & & & & \\
x_s^{(i)} & y_s^{(i)} & 1 & 0 & 0 & 0 & -x_d^{(i)}x_s^{(i)} & -x_d^{(i)}y_s^{(i)} & -x_d^{(i)} \\
0 & 0 & 0 & x_s^{(i)} & y_s^{(i)} & 1 & -y_d^{(i)}x_s^{(i)} & -y_d^{(i)}y_s^{(i)} & -y_d^{(i)} \\
 & & & & \vdots & & & & \\
x_s^{(n)} & y_s^{(n)} & 1 & 0 & 0 & 0 & -x_d^{(n)}x_s^{(n)} & -x_d^{(n)}y_s^{(n)} & -x_d^{(n)} \\
0 & 0 & 0 & x_s^{(n)} & y_s^{(n)} & 1 & -y_d^{(n)}x_s^{(n)} & -y_d^{(n)}y_s^{(n)} & -y_d^{(n)}
\end{bmatrix}}_{\substack{A \\ \text{(Known)}}}
\underbrace{\begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix}}_{\substack{\mathbf{h} \\ \text{(Unknown)}}}
=
\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}
$$

Solve for $\mathbf{h}$: $\boxed{A\,\mathbf{h} = \mathbf{0}}$ such that $\|\mathbf{h}\|^2 = 1$

Constrained Least Squares

Define least squares problem:

$$\min_{\mathbf{h}} \|A\mathbf{h}\|^2 \quad \text{such that } \|\mathbf{h}\|^2 = 1$$

We know that:

$$\|A\mathbf{h}\|^2 = (A\mathbf{h})^T(A\mathbf{h}) = \mathbf{h}^T A^T A\mathbf{h} \quad \text{and} \quad \|\mathbf{h}\|^2 = \mathbf{h}^T\mathbf{h} = 1$$

$$\boxed{\min_{\mathbf{h}}(\mathbf{h}^T A^T A\mathbf{h}) \quad \text{such that } \mathbf{h}^T\mathbf{h} = 1}$$

Constrained Least Squares

$$\min_{\mathbf{h}}(\mathbf{h}^T A^T A\mathbf{h}) \text{ such that } \mathbf{h}^T\mathbf{h} = 1$$

Define Loss function $L(\mathbf{h},\lambda)$:

$$L(\mathbf{h},\lambda) = \mathbf{h}^T A^T A\mathbf{h} - \lambda(\mathbf{h}^T\mathbf{h} - 1)$$

Taking derivatives of $L(\mathbf{h},\lambda)$ w.r.t $\mathbf{h}$: $2A^T A\mathbf{h} - 2\lambda\mathbf{h} = \mathbf{0}$

$$\boxed{A^T A\mathbf{h} = \lambda\mathbf{h}}$$ Eigenvalue Problem

Eigenvector $\mathbf{h}$ with smallest eigenvalue λ of matrix $A^T A$ minimizes the loss function $L(\mathbf{h})$.

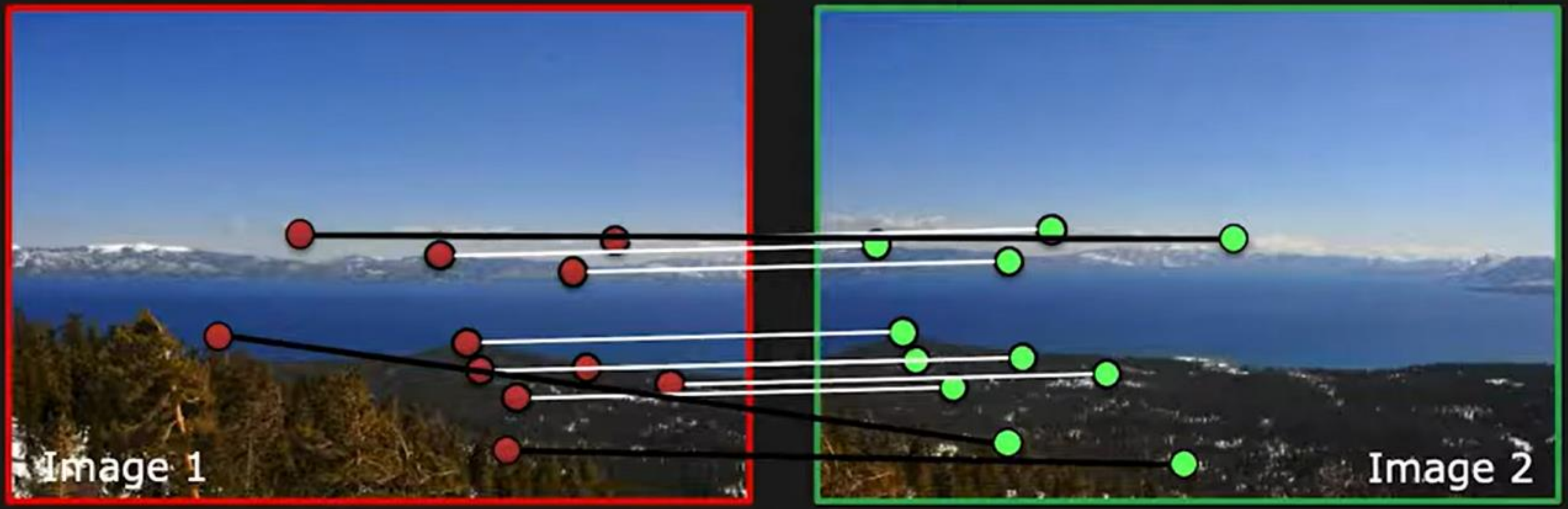
Matlab: `eig(A'*A)` returns eigenvalues and vectors of $A^T A$

Homography matrix

- Take the eigen vector **h.** and convert it to a 3X3 matrix. This is the Homography matrix.

$$\begin{bmatrix} \tilde{x}_2 \\ \tilde{y}_2 \\ \tilde{z}_2 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} \tilde{x}_1 \\ \tilde{y}_1 \\ \tilde{z}_1 \end{bmatrix}$$

What could go wrong (say w.r.t SIFT features matching)?



Outliers!

We need to robustly compute transformation in the presence of wrong matches.

If number of outliers < 50%, then RANSAC to the rescue!

RANdom SAmple Consensus (RANSAC)

General RANSAC Algorithm:

1. Randomly choose s samples. Typically s is the minimum samples to fit a model.
2. Fit the model to the randomly chosen samples.
3. Count the number M of data points (inliers) that fit the model within a measure of error ε.
4. Repeat Steps 1-3 N times
5. Choose the model that has the largest number M of inliers.

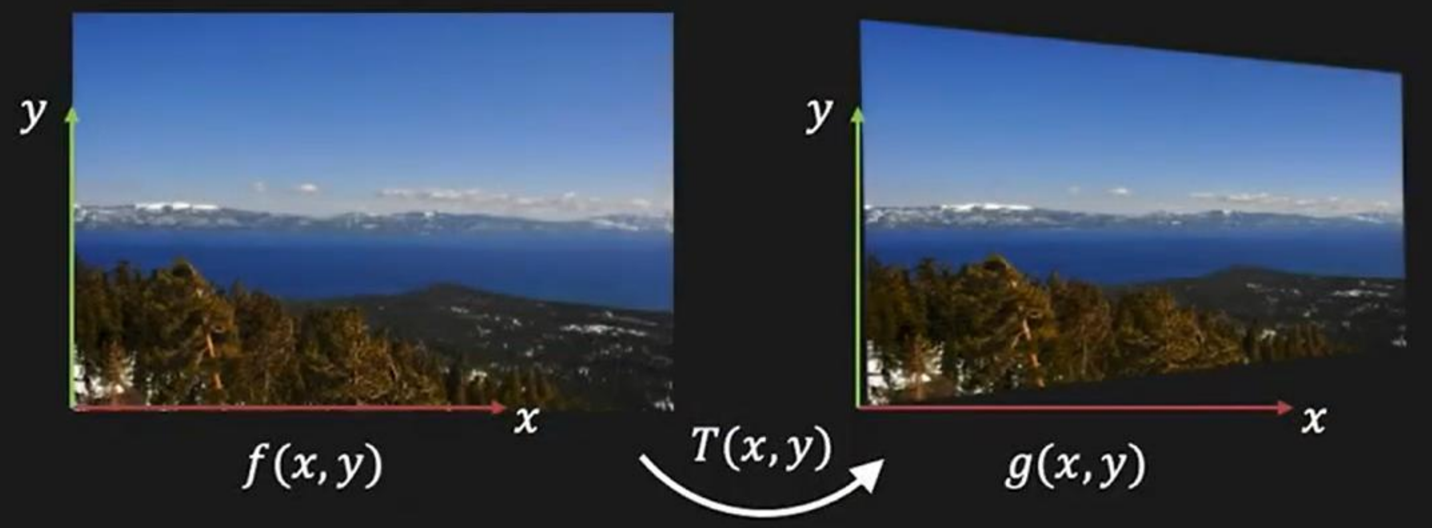
For homography:

$s = 4$ points. ε is acceptable alignment error in pixels.

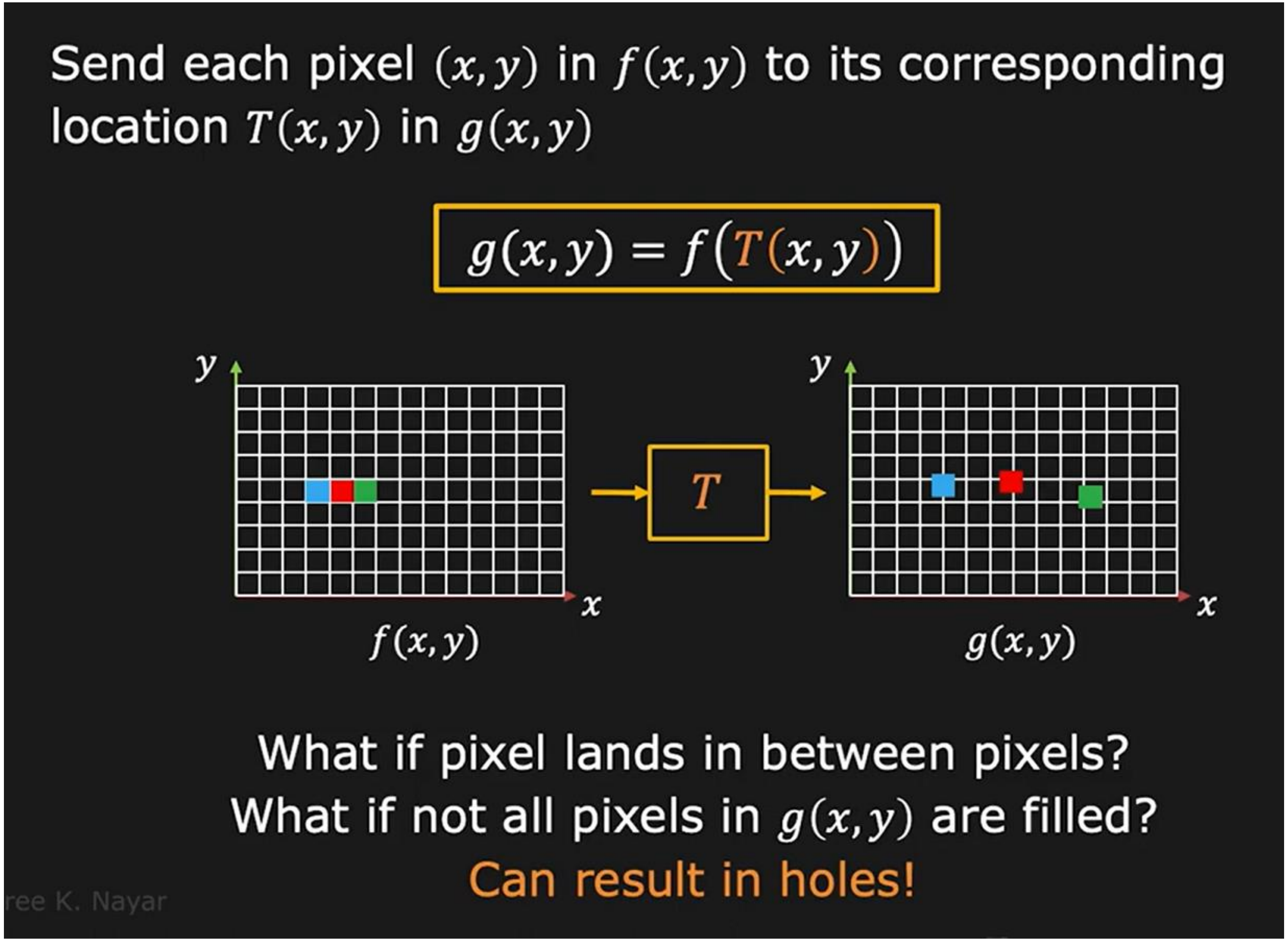
Warping Images

Given a transformation T and a image $f(x, y)$, compute the transformed image $g(x, y)$

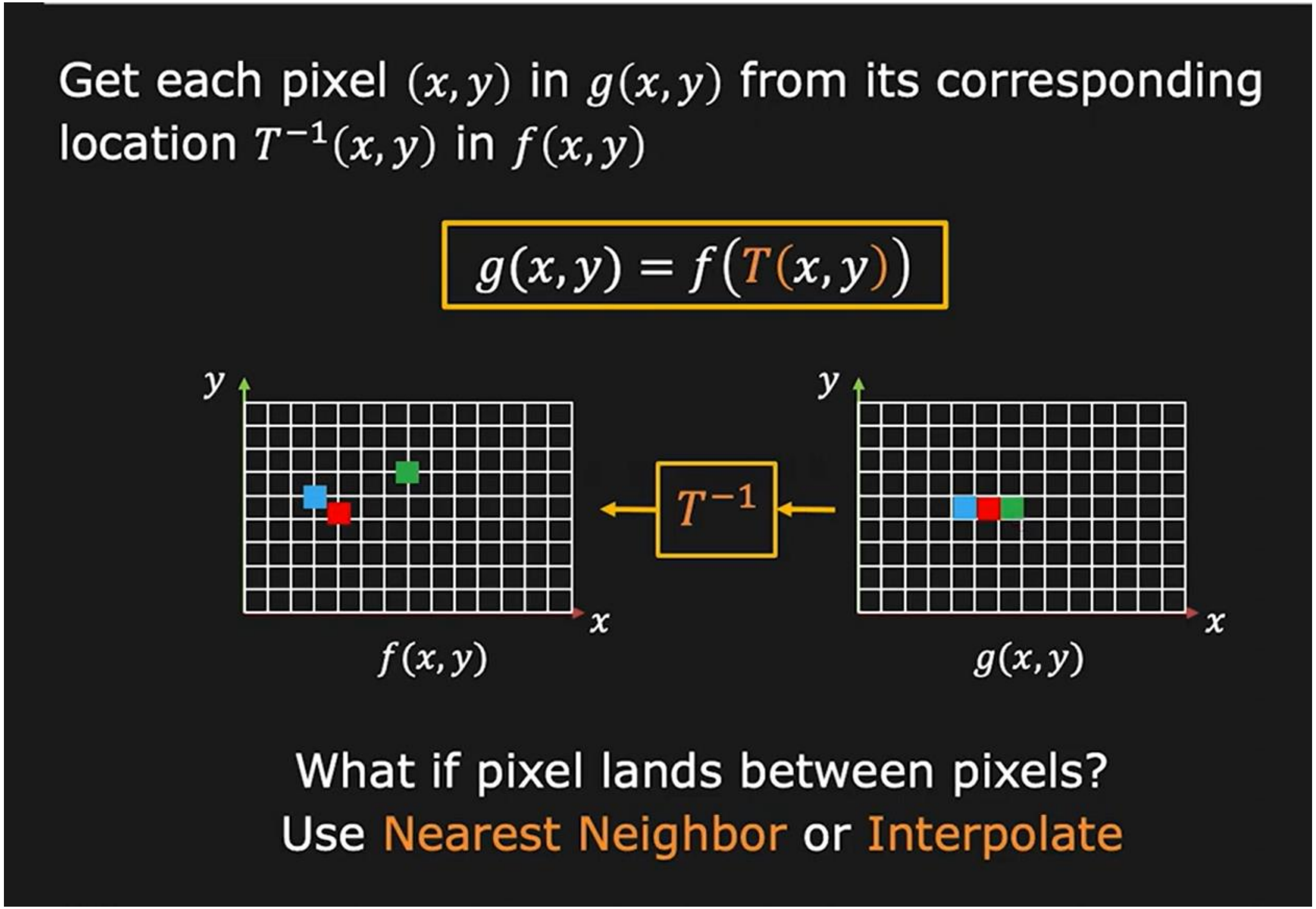
$$\boxed{g(x,y) = f\big(T(x,y)\big)}$$



Forward Warping



Backward Warping



Take the image ***f*** and its four corners and apply forward warp to the four corners of ***g*** (bounding box).

The goal is to fill all the cells with in the bounding box of ***g***.

Take few pixels (e.g., say 3) and apply inverse of the warp and you land up with the pixels in ***f.***

Corresponding pixel values are take from f onto g.

If a pixel lands in the middle of pixels, interpolation of brightness values w.r.t neighbouring pixels is done. The interpolated value is set to the ***g*** pixel.

Do it for all pixels in ***g*** and fill these ***g*** pixel values with respective values from ***f*** using the inverse transform.

Solved issues:

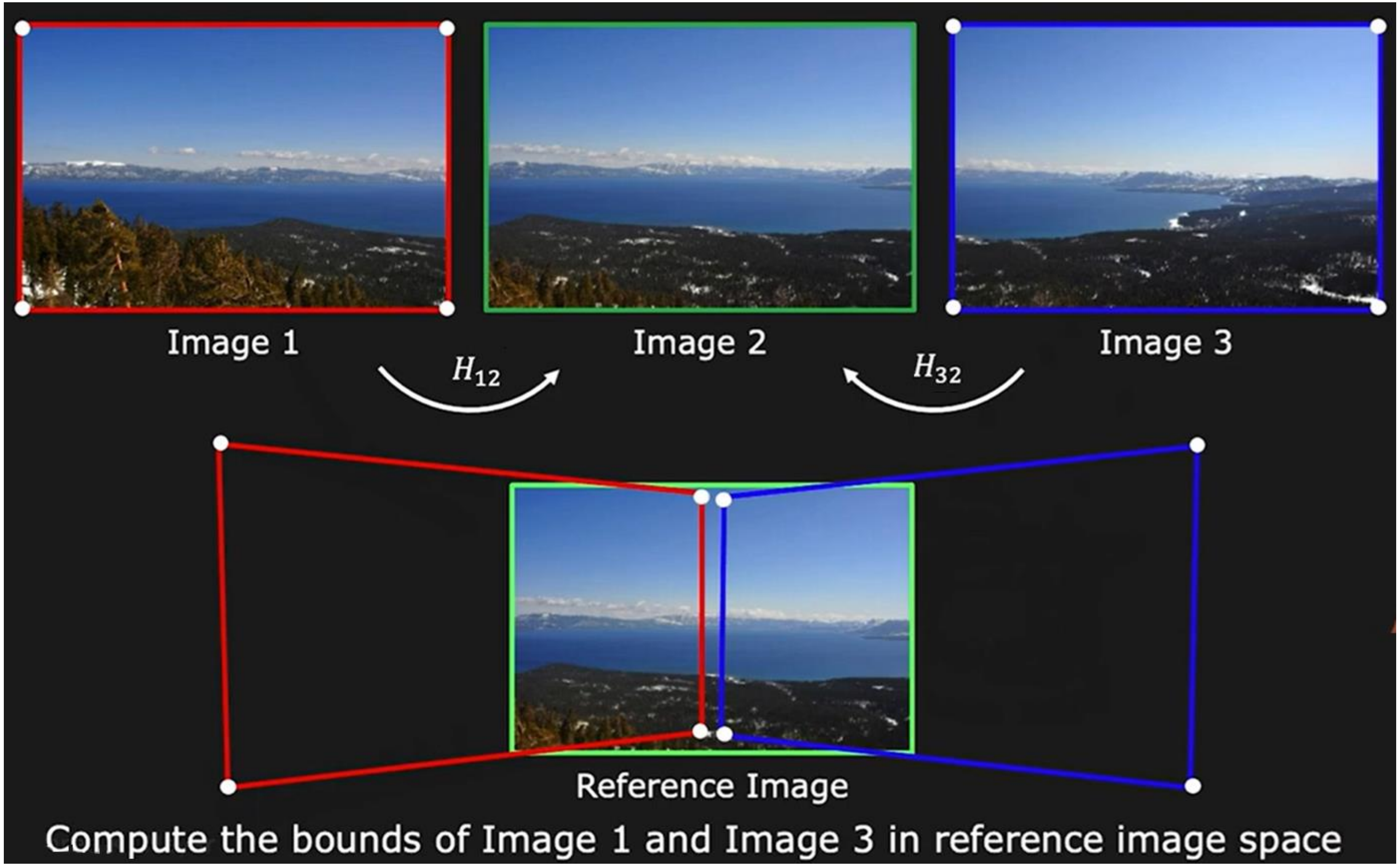
a) Use Interpolation or Nearest Neighbours to get best estimates in ***g***

b) No holes in ***g***

Image Stitching: Image Alignment Process (1)

Image Stitching: Image Alignment Process (2)

Forward warping to align image boundaries after transformation



Take 4 corners of Image 1 and image 2 and find the homography that takes you from Image 1 to image 2 (H12).

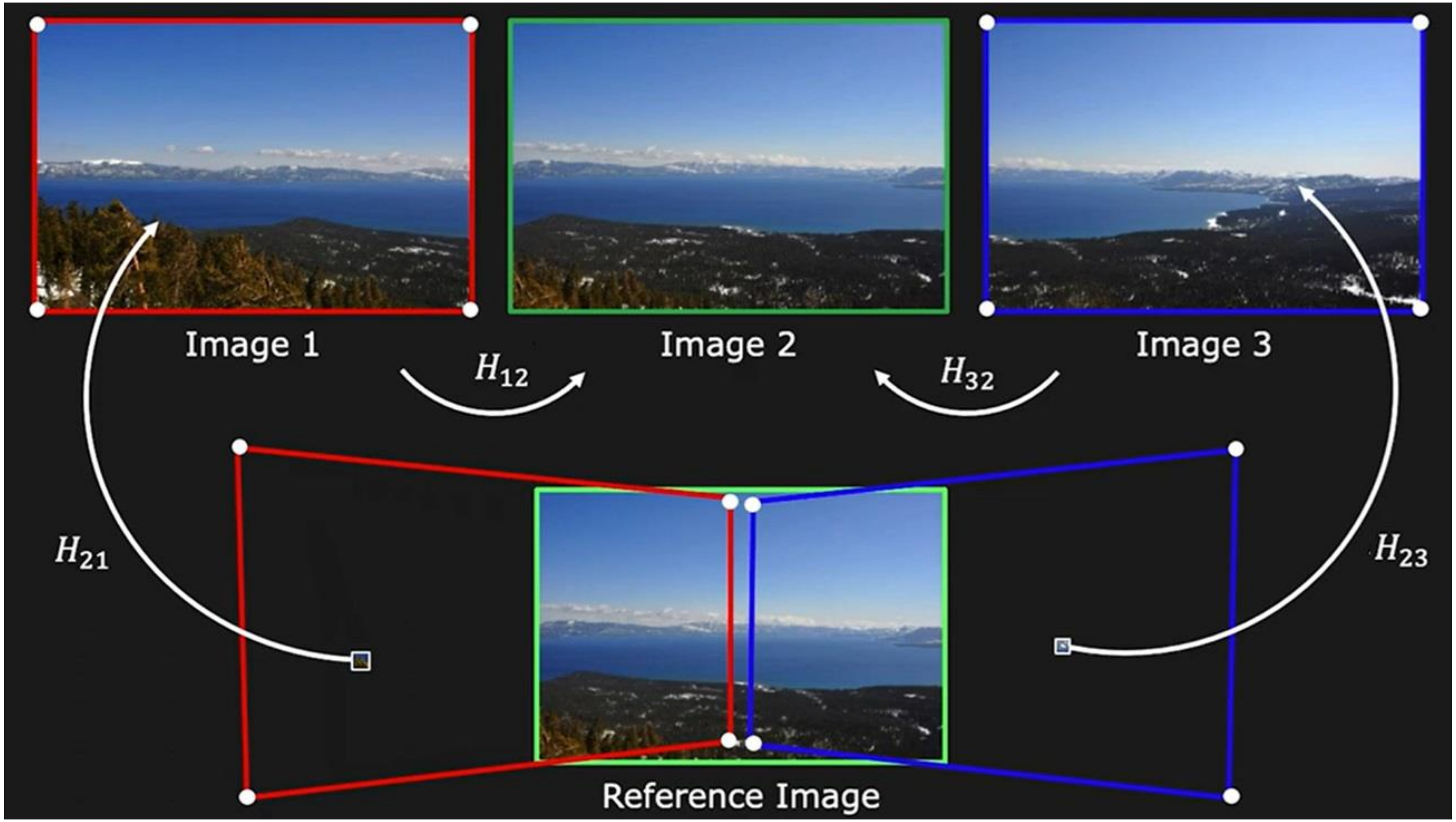
Take the 4 corners of Image 1 and forward map them (using transformation matrix H12) to get the bounding box for Image 1 in the reference frame of Image 2.

Do the same for Image 3.

We need to fill all the points with in these new bounding boxes (including the overlapped regions)

Image Stitching: Image Alignment Process (3)

Backward warping to align all points with in the images (1 and 3) transformation



We need to fill all the points with in these new bounding boxes (including the overlapped regions)

We need to take each pixel within the bounding box and apply backward map H21 (the backward warp) to go in and see where you land in Image 1.

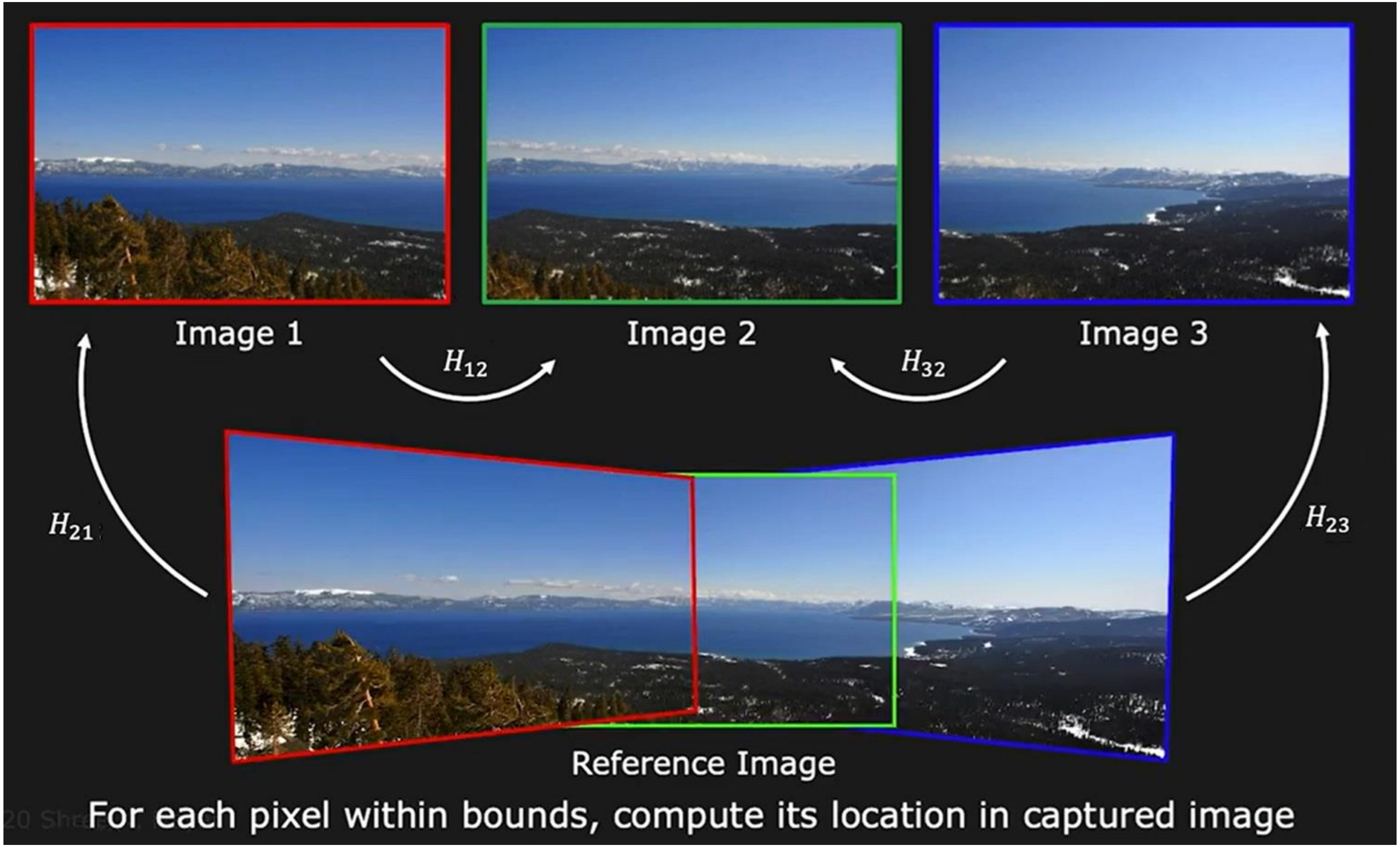
Pick the brightness value from there either using interpolation or nearest neighbors and come back and write it at this location.

Repeat it for all the pixels within the bounding box.

Now you get the image 1 warped or transformed into the reference frame of Image 2

Same applies to Image 3.

Image Stitching: Image Alignment Process (4)



Now we have aligned images that are stacked on each other (with overlapping)

Image Stitching: Blending Images (5)

Image Stitching: Blending Images (Averaging) (6)

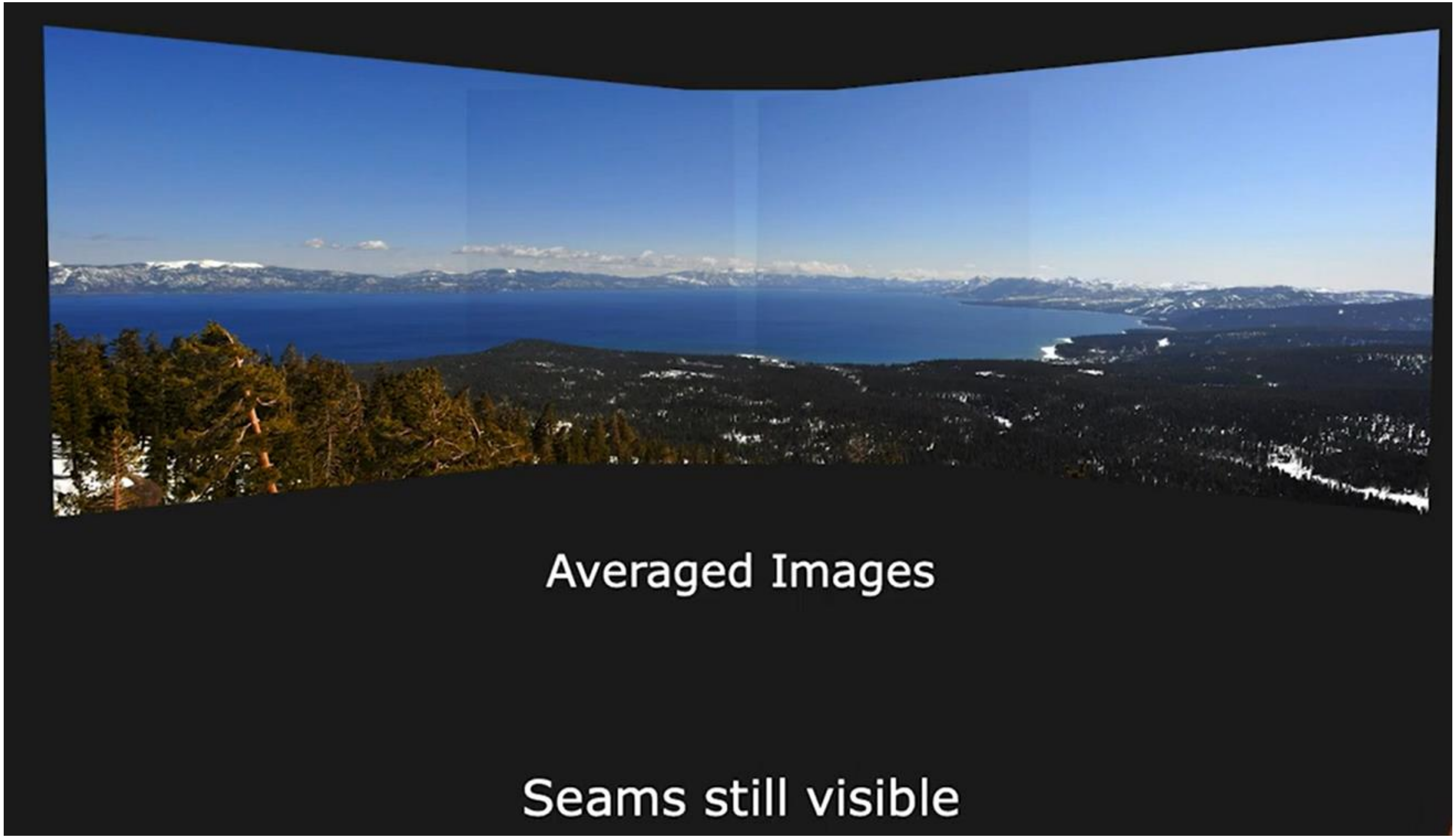
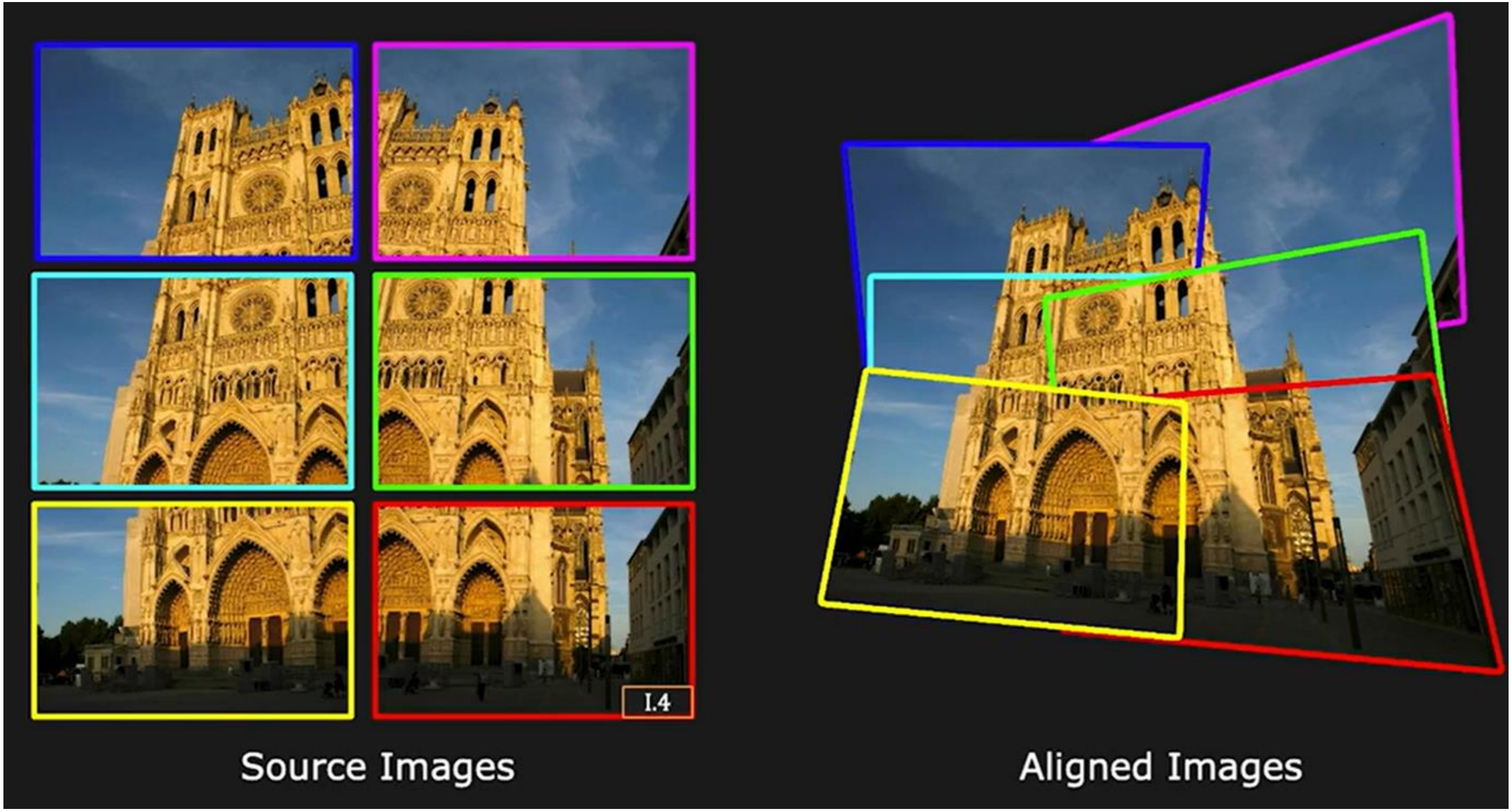


Image Stitching: Blending Images (Weighted Blending)

Image Stitching: Computing Weighting Functions (7)

Image Stitching: Blended Image (Output) (8)

Image Stitching Example



Compositing

Image Compositing

Compositing Procedure

1. Extract Sprites (e.g using *Intelligent Scissors* in Photoshop)




2. Blend them into the composite (in the right order)

Composite by David Dewey

Different Ways to Use Compositing

Whether in video or photos, photo editors use compositing in diverse ways.

- **Realism**
  - When you are compositing different photos and videos, anything is possible as you get creative with it. With compositing techniques, one can create photorealistic and believable imagery. Realistic composite components can place different people together in a single frame that was not present.
- **Fantasy**
  - One can use compositing techniques to create different fantasy images and make them look realistic. The editor can place photo and video subjects in absurd or magical environments via composite imagery.
- **Motion Effect**
  - This is another one of the main effects of compositing; it allows images to be grouped to create a motion-like effect for a single subject. In sports photography, there are different techniques of composite photography that can break down the movements of a jumper or an athlete to analyze them. It would not be possible to achieve such effects with traditional techniques.
- **Storytelling**
  - Composite images are used in storytelling. The photo editor can establish control over different elements of a photograph to narrate a story to the viewers. For instance, composite images come in handy, when it comes to the creation of movie posters. They bring together the different elements of the film into one frame.

How Can You Create a Composite Image?

- **Select Different Image Elements**
  - Firstly, you need to select the different elements from various other photographs. Then, overlay these elements on the composite photo that you want to create. Now, you must open every image in a separate file. Make sure you choose elements with similar pixel count or lighting to produce the best effects.
- **Create The Layers**
  - Label a new layer for every single element so that each layer is easy to manipulate later. Choose a proper background for the image and make it the first layer. Use precise selection tools to choose the specific elements that you want for your photo.
- **Use Masks for Hiding or Blending Elements**
  - There will be a mask button in your editing software. Use it to create a layer mask on different layers. This will allow you to fade, hide, or reveal parts of the photo, according to your liking.
- **Adjust the Colors of the Image**
  - It is the ultimate step in the process, and now that you have the image elements placed into one single frame, you need to adjust the image colors to make it look more realistic. There are different color grading tools and buttons that can help you produce the desired effect.

Alpha Channel

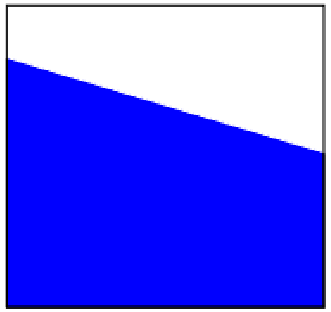
Add one more channel:

- Image(R,G,B,alpha)

Encodes transparency (or pixel coverage):

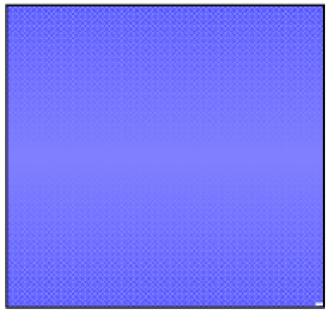
- Alpha = 1: opaque object (complete coverage)
- Alpha = 0: transparent object (no coverage)
- 0<Alpha<1: semi-transparent (partial coverage)

Example: alpha = 0.3



Partial coverage

or

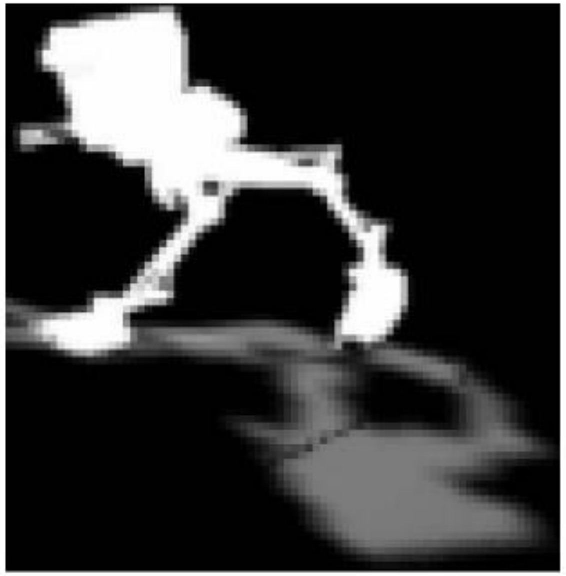


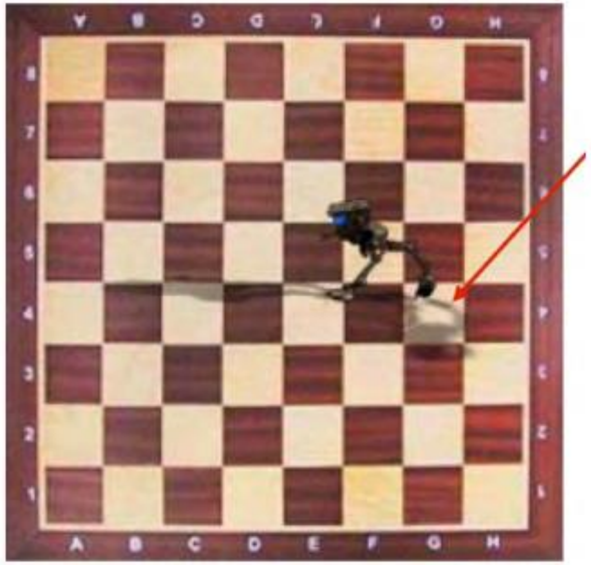
semi-transparency

Alpha Blending

$$I_{comp} = \alpha I_{fg} + (1-\alpha) I_{bg}$$

alpha
mask





shadow

Multiple Alpha Blending

So far we assumed that one image (background) is opaque.

If blending semi-transparent sprites (the "A over B" operation):

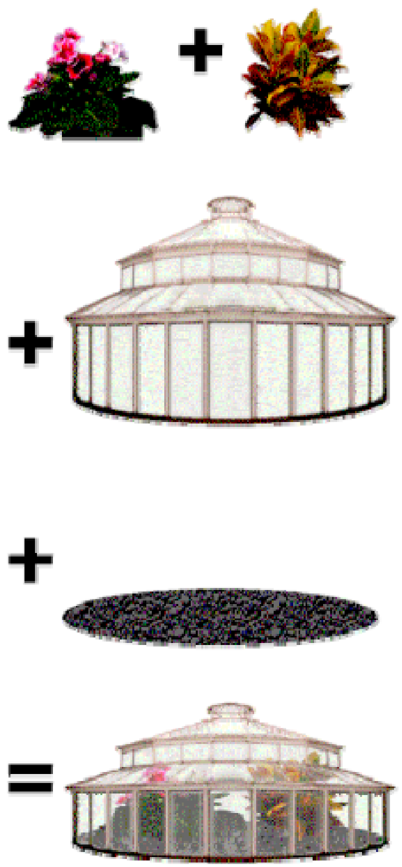
$$I_{comp} = \alpha_a I_a + (1-\alpha_a)\alpha_b I_b$$

$$\alpha_{comp} = \alpha_a + (1-\alpha_a)\alpha_b$$

Note: sometimes alpha is premultiplied: im(αR,αG,αB,α):

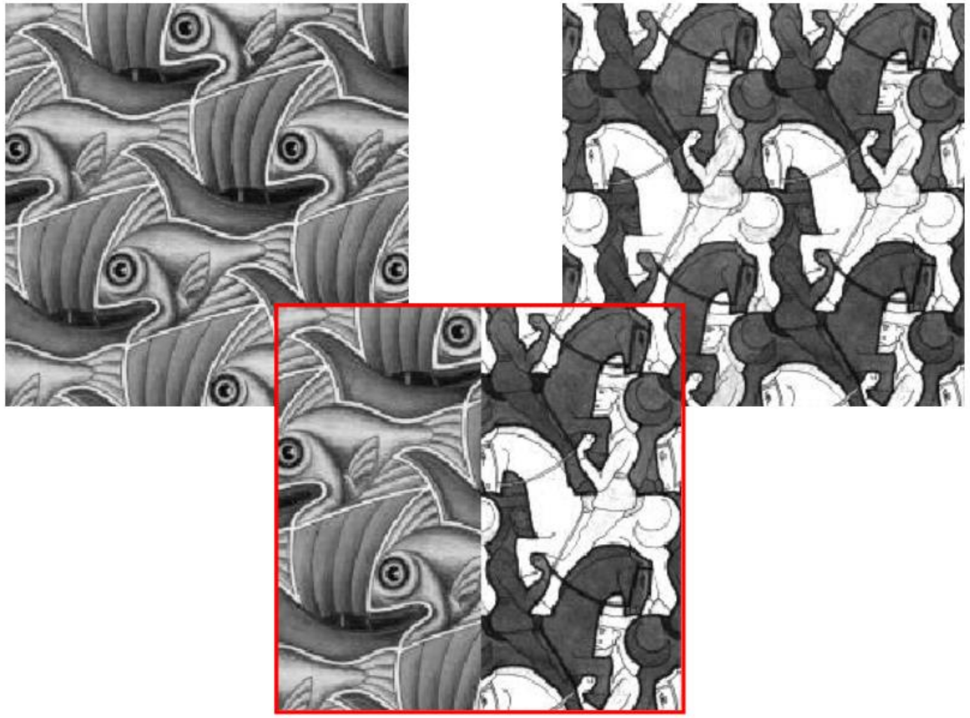
$$I_{comp} = I_a + (1-\alpha_a) I_b$$

(same for alpha!)



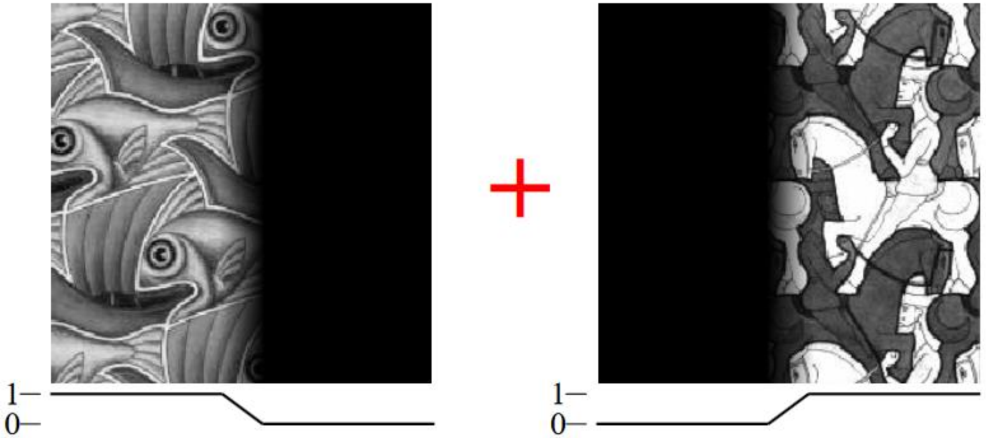
Simple Blending Approaches

Alpha Hacking



No physical interpretation, but it smoothes the seams

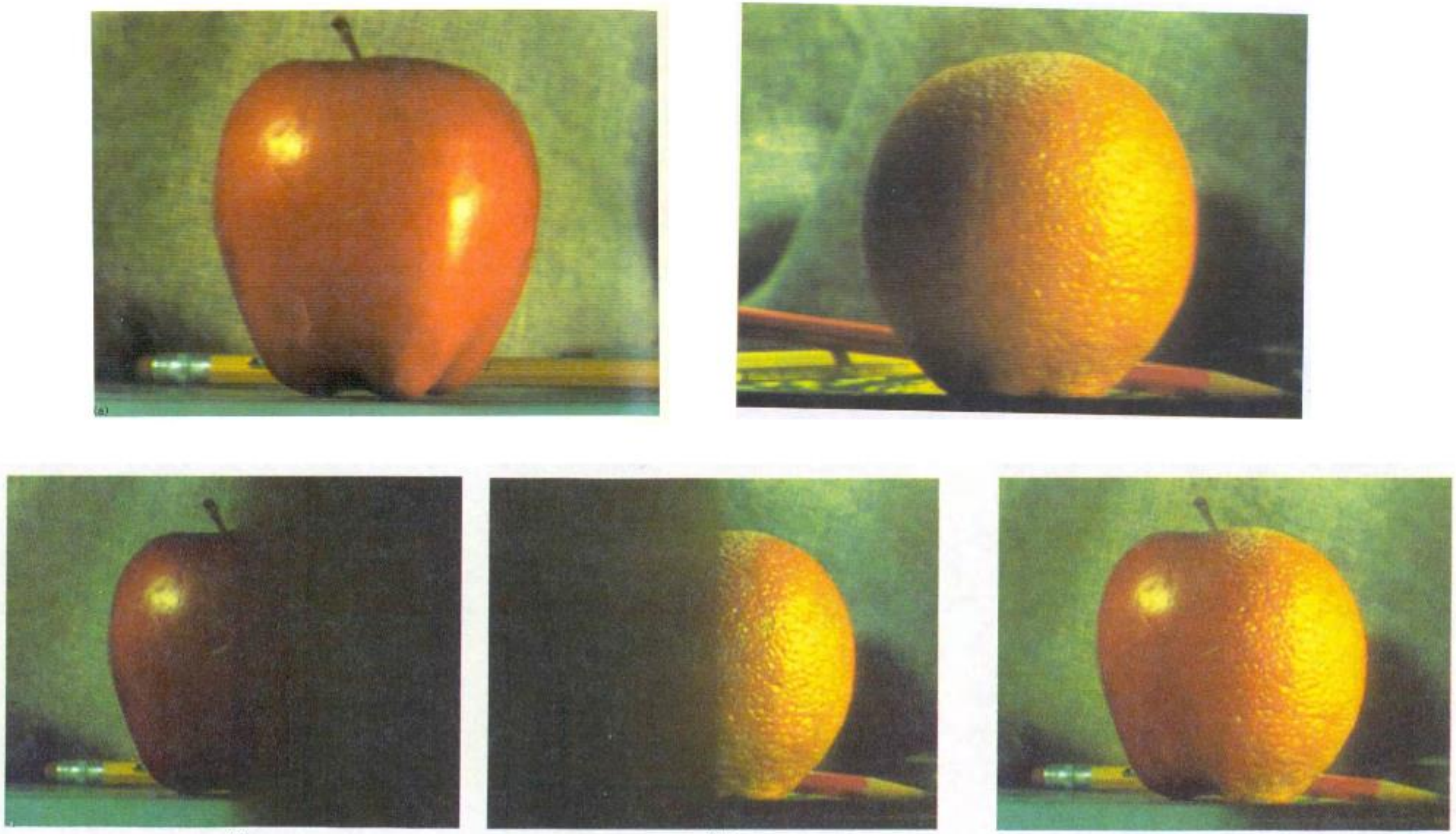
Feathering



Encoding as transparency

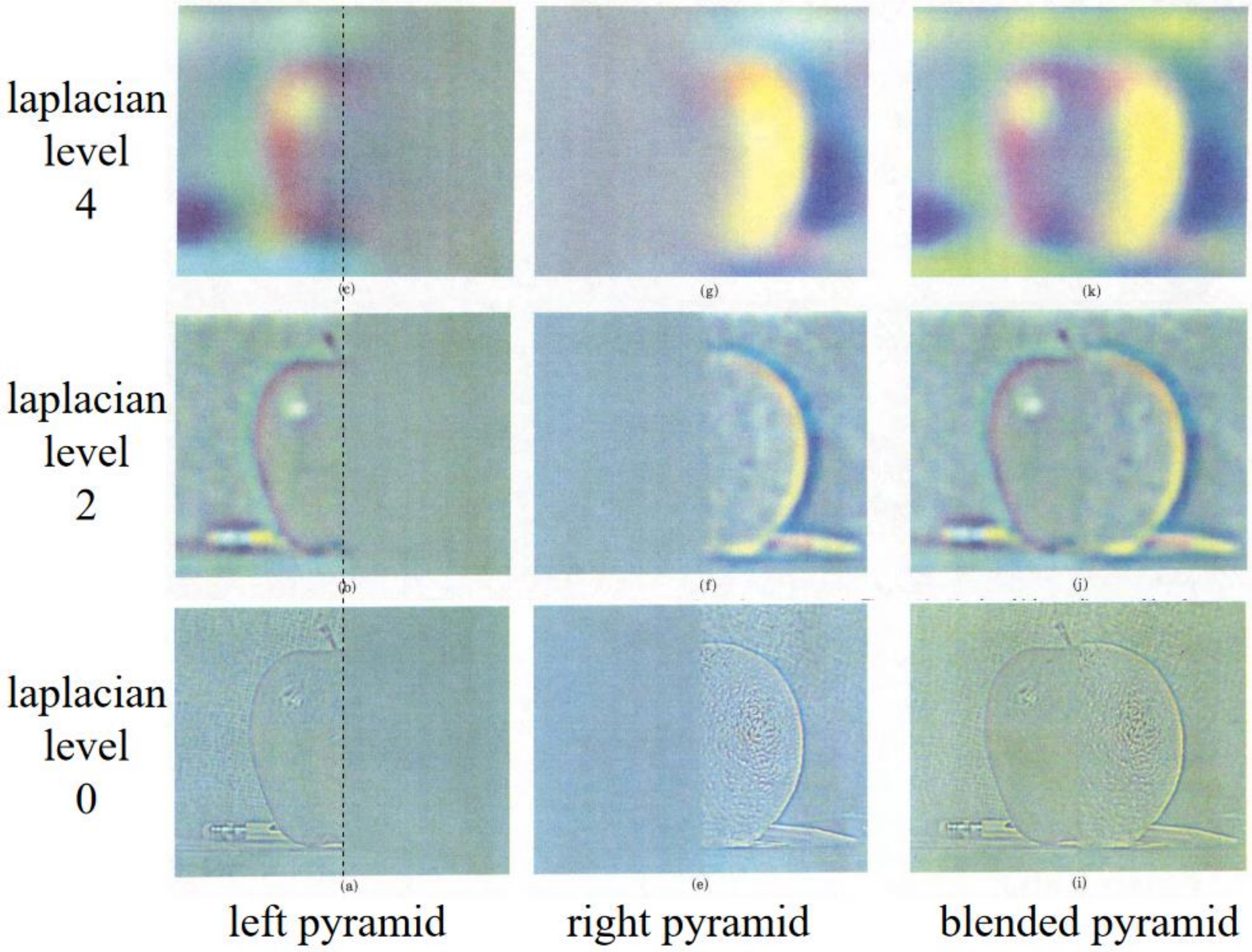
$I_{blend} = \alpha I_{left} + (1-\alpha) I_{right}$

Pyramid blending (already covered in pyramids)



(d) (h) (l)

Pyramid blending (already covered in pyramids)



Laplacian Pyramid: Blending (already covered in Pyramids)

General Approach:

1. Build Laplacian pyramids *LA* and *LB* from images *A* and *B*
2. Build a Gaussian pyramid *GR* from selected region *R*
3. Form a combined pyramid *LS* from *LA* and *LB* using nodes of *GR* as weights:
   - $LS(i,j) = GR(I,j,)*LA(I,j) + (1-GR(I,j))*LB(I,j)$
4. Collapse the *LS* pyramid to get the final blended image

References

- Image Stitching | face Detection play list by Shree K Nayar (Columbia university): https://www.youtube.com/playlist?list=PL2zRqk16wsdp8KbDfHKvPYNGF2L-zQASc

- https://www.geeksforgeeks.org/image-stitching-with-opencv/

- https://www.cs.cmu.edu/~16385/lectures/lecture9.pdf
- http://graphics.cs.cmu.edu/courses/15-463/2007_fall/Lectures/blending.pdf (Good one)

- https://www.flatworldsolutions.com/digital-photography/articles/what-is-image-compositing-how-can-you-make-composite-images.php