**EECS C106B / 206B Robotic Manipulation and Interaction** **Spring 2022**

# Lecture 6B: (Feature Detection -Chpt 4 MaSKS)

*Scribes: Federico Palacios, Adam Curtis*

## 6.1 Detection & Correspondence

"Early days of AI, people were actually pretty cocky. They think vision is part of intelligent... I can have one PhD summer project and solve the vision problem - that's a real comment. Here, 60 years after, I would say we've barely scratched the surface of it." -Yi Ma

### 6.1.1 Concepts

What is computer vision? Computer vision is all about where is what? -Projections of 3D information onto 2D space requires us to solve the image recognition problem. The "what" is about appearance and it is essentially a recognition problem. The "where" is a geometry problem. But which of these comes first? It's a chicken and egg scenario. Recognition helps reconstruction and vice versa.

### 6.1.2 Matching

We want to find matches between images so we start with some assumptions:

Lambertian assumption: $I_1(\mathrm{x}_1) = R(p) = I_2(\mathrm{x}_2)$

This is essentially saying that the intensity of a pixel should be the same no matter where you look at it from.

Rigid body motion: $\mathrm{x}_2 = h(\mathrm{x}_1) = \frac{1}{\lambda_2(\mathbf{X})}(R\lambda_1(\mathbf{X})\mathrm{x}_1 + T)$

In the rigid body motion equation $\lambda_i i = 1, 2$ is a scaling factor of the 3-D coordinates of $\mathbf{X}$.

Correspondence: $I_1(\mathrm{x}_1) = I_2(h(\mathrm{x}_1))$

Which says that a point in the image has the same brightness as a transformed point in another image, where $h$ describes the "image motion."

### 6.1.3 Local Deformation Models

In looking at the transformation of an image, we prefer to examine local changes because finding a global transformation can require us to know the entire 3-D structure of the environment.

Translation model: $h(\mathrm{x}) = \mathrm{x} + d$ $\quad I_1(\mathrm{x}_1) = I_2(h(x_1))$

Affine model: $h(\mathrm{x}) = A\mathrm{x} + d \qquad I_1(\mathrm{x}_1) = I_2(h(x_1))$

Transformation of the intensity values and occlusions:

$I_1(\mathrm{x}_1) = f_o(\mathbf{X}, g) I_2(h(\mathrm{x}_1)) + n(h(\mathrm{x}_1))$

### 6.1.4 Region Based Similarities

Instead of just looking at single pixels, it is more helpful to examine areas of an image. One tool we can use is the sum of squared differences in a region around x.

$$SSD(h) = \Sigma_{\tilde{\mathrm{x}} \in W(\mathrm{x})} \|I_1(\tilde{\mathrm{x}}) - I_2(h(\tilde{\mathrm{x}}))\|^2$$

Where W is a window of interest. Unfortunately SSD is effected by image scaling and changes in intensities, so it is not a robust method to track features. Normalize Cross Correlation is a better feature-tracking algorithm for template tracking. $I_1(\tilde{\mathrm{x}}$ and $I_2(h(\tilde{\mathrm{x}}))$ are feature filled regions. We define the NCC as

$$NCC(h) = \frac{\Sigma_{W(\mathrm{x})}(I_1(\tilde{\mathrm{X}}) - \bar{I}_1)(I_2(h(\tilde{\mathrm{X}}) - \bar{I}_2))}{\sqrt{\Sigma_{W(\mathrm{x})}(I_1(\tilde{\mathrm{X}}) - \bar{I}_1)^2 \Sigma_{W(\mathrm{x})}(I_2(h(\tilde{\mathrm{X}}) - \bar{I}_2))^2}}$$

$\bar{I}_1$ and $\bar{I}_2$ are the mean intensities of the regions in question. $NCC$ is always between $-1$ and $+1$. When $NCC = +1$ the features match exactly.

The sum of absolute differences is similar to sum of square differences.

$$SAD(h) = \Sigma_{\tilde{\mathrm{x}} \in W(\mathrm{x})} |I_1(\tilde{\mathrm{x}}) - I_2(h(\tilde{\mathrm{x}}))|$$

## 6.2 Tracking & Optical Flow

”Life for robots is miserable”
-Yi Ma

Let us consider a simple model.

### 6.2.1 Models and Brightness Constancy Constraints

If you move your camera over time $(t)$, you can describe the motion of a single point on the image as $x(t)$. Therefore the intensity $(I)$ of a given spot is a function of the motion trajectory $x(t)$ and time $(t)$. Translational Model:

$$I_1(x_1) = I_2(x_1 + \Delta x)$$

Small Baseline:

$$I(x(t), t) = I(x(t) + udt, t + dt)$$

RHS approximation by first two terms of Taylor series:

$$\boxed{\nabla I(x(t),t)^T u + I_t(x(t),t) = 0}$$

This is the Brightness Constancy Constraint

### 6.2.2 Optical Flow: Connecting 2D and 3D Motions

Let's first describe 3-Dimensional motion. Where $\dot{X}, \dot{Y}$, and $\dot{Z}$ represent instantaneous change in position. Equation 3.2:

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{bmatrix} = - \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} - \begin{bmatrix} \omega_y z - \omega_z y \\ \omega_z x - \omega_x z \\ \omega_x y - \omega_y x \end{bmatrix}$$

Assume the image plane lies at f=1, then $x = \frac{X}{Z}$ and $y = \frac{Y}{Z}$. Taking the derivative, we get our change in instantaneous position over time on the 2D plane: Equation 3.3:

$$\dot{x} = \frac{\dot{X}Z - \dot{Z}X}{Z^2}, \dot{y} = \frac{\dot{Y}Z - \dot{Z}Y}{Z^2}$$

We thus define our $u$ and $v$ variables similarly. Equation 3.4:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \frac{1}{Z} \begin{bmatrix} -1 & 0 & x \\ 0 & -1 & y \end{bmatrix} \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} + \begin{bmatrix} xy & -(1+x^2) & y \\ 1+y^2 & -xy & -x \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$$

This description is important. If we look closely at our definition of $u$:

$$\bar{u} = \frac{t_x}{Z} \begin{bmatrix} x \\ y \end{bmatrix}$$

It shows us how the velocity $t_x$ over the distance $Z$ results in $\bar{u}$ being a function of time. This is how we are able to determine things like time of impact or depth from an image.

### 6.2.3 Aperture Problem

The above formulas assume that we are able to physically track these points, but the reality is not so simple. Sometimes we may only have limited information. In the image below, we see how the limited view of an aperture might fool us into thinking that movement is only happening in a single direction.

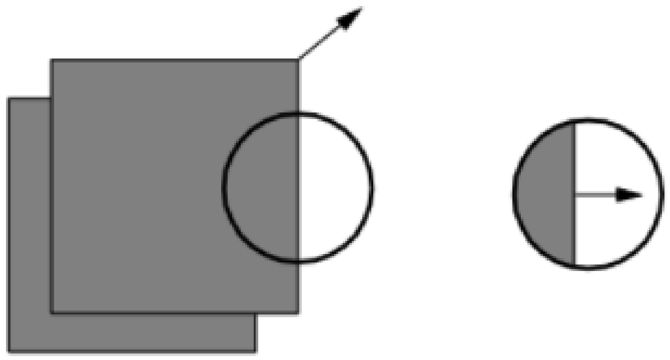

Fig:6.2.3.1 (From lecture slides) Aperture Depiction

Normal Flow:

$$\mathbf{u}_n \doteq \frac{\nabla I^T \mathbf{u}}{\|\nabla I\|} \cdot \frac{\nabla I}{\|\nabla I\|} = -\frac{I_t}{\|\nabla I\|} \cdot \frac{\nabla I}{\|\nabla I\|}$$

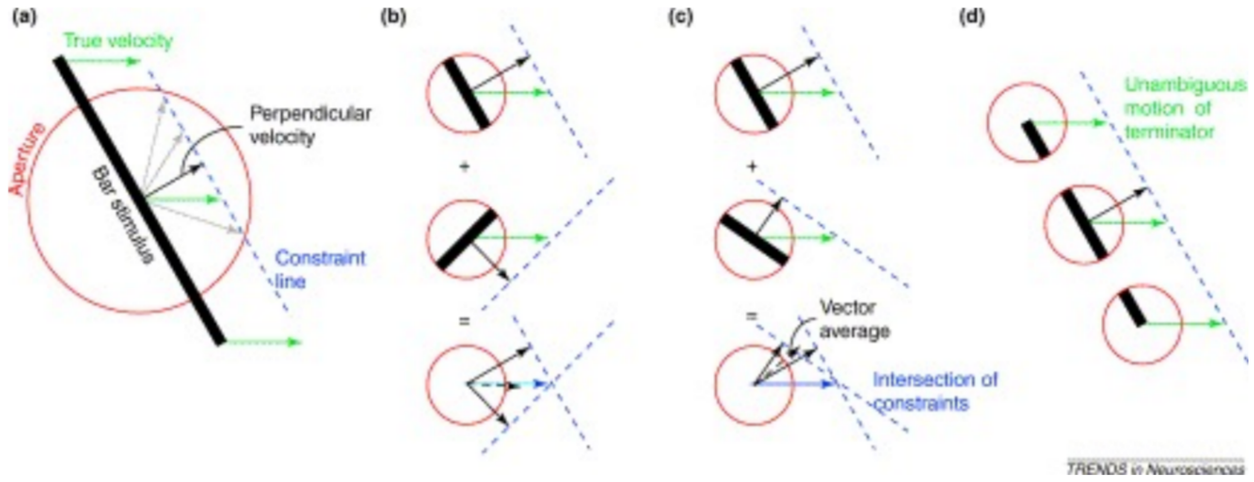

Fig:6.2.3.2 (From lecture slides) Aperture Problem Depictions: (a) An example of the perceived velocity differing from the true velocity due to the aperture and the single observable bar. (b) Even with separate points, we may be able to determine the correct direction of movement, but the magnitude is still incorrect. What we can do though is observe the intersection of perceived constraint lines to determine the True Velocity. (c) We can apply the technique mentioned in (b) here as well, despite the fact that the vector average does not correspond to the True Velocity (d) Finally, if we have the end of an edge or 'terminator', we can use that to help determine true velocity of other locations as the motion of a terminator is unambiguous

### 6.2.4 Optical Flow: Solving Flow

By observing a region or window (W), we can (hopefully)find at least 2 edges with different slopes. We can then apply the method described in Fig 6.2.3.2(b) in order to solve for u. Integrate around over an image window:

$$E_b(\mathbf{u}) = \sum_{W(x,y)} [\nabla I^T(x,y,t)\mathbf{u}(x,y) + I_t(x,y,t)]^2$$

Then Solve:

$$\nabla E_b(\mathbf{u}) = 2 \sum_{W(x,y)} \nabla I(\nabla I^T \mathbf{u} + I_t)$$

$$= 2 \sum_{W(x,y)} \left( \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \mathbf{u} + \begin{bmatrix} I_x I_t \\ I_y I_t \end{bmatrix} \right)$$

$$\begin{bmatrix} \Sigma I_x^2 & \Sigma I_x I_y \\ \Sigma I_x I_y & \Sigma I_y^2 \end{bmatrix} \mathbf{u} + \begin{bmatrix} \Sigma I_x I_t \\ \Sigma I_y I_t \end{bmatrix} = 0$$

$$G\mathbf{u} + b = 0$$

And so we get that:

$$\boxed{\mathbf{u} = -G^{-1}b}$$

Where:

$$G = \begin{bmatrix} \Sigma I_x^2 & \Sigma I_x I_y \\ \Sigma I_x I_y & \Sigma I_y^2 \end{bmatrix}$$

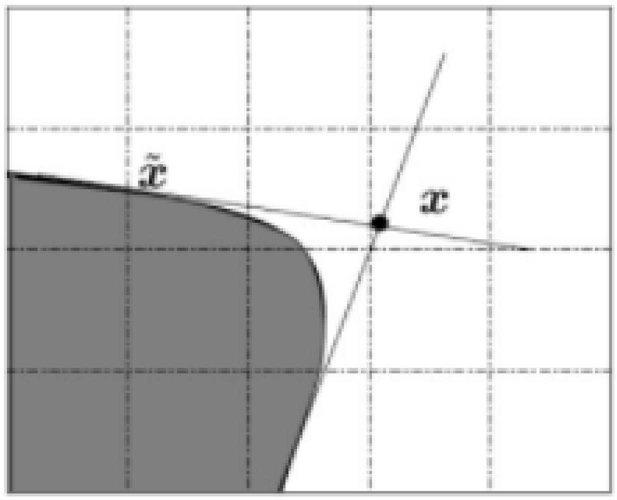

Fig:6.2.4.1 (From lecture slides) Feature detection example

Note that Conceptually:

at rank(G) = 0, there is the blank wall problem
at rank(G) = 1, there is the aperture problem
at rank(G) = 2, enough texture - good feature candidates

In reality: the choice of threshold is more involved.

### 6.2.5 Computing Derivatives

In reality, images are not so clear or easy to work with. We can use convolutions to actually perform these techniques across discretized images.
We need to compute the following derivatives(from Appendix 4.A):

$$I_x(x,y) = \frac{\partial I}{\partial x}(x,y), I_y(x,y) = \frac{\partial I}{\partial y}(x,y)$$

Now consider convolution with difference of Gaussians:

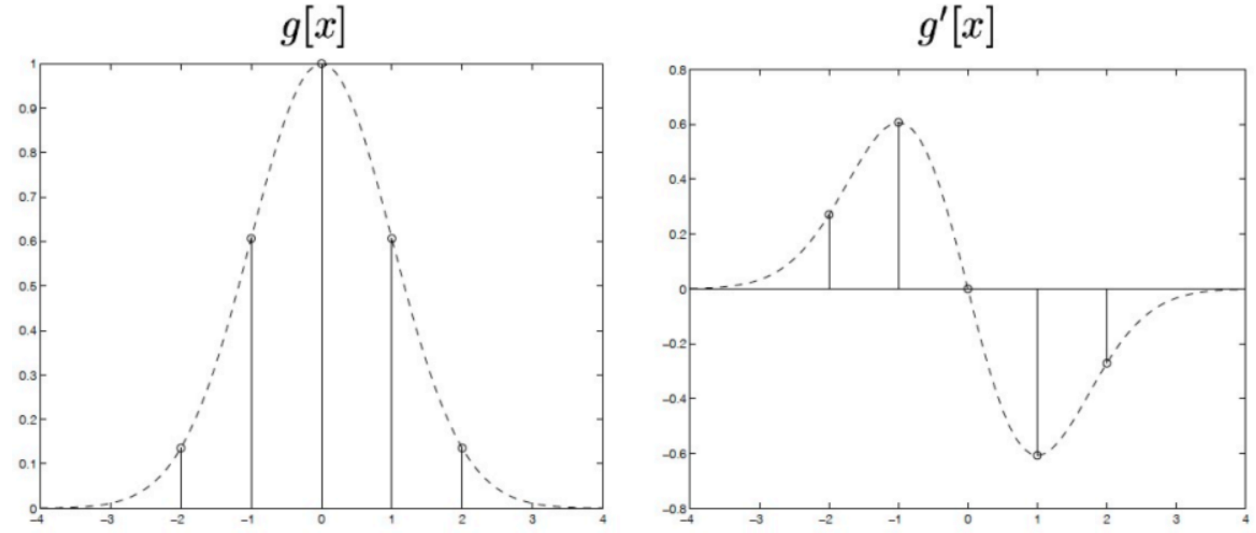

Fig:6.2.5.1 (From lecture slides) Difference of Gaussians example

$$I_x[x,y] = I[x,y] * g'[x] * g[y] = \sum_{k=-\frac{w}{2}}^{\frac{w}{2}} \sum_{l=-\frac{w}{2}}^{\frac{w}{2}} I[k,l]g'[x-k]g[y-l],$$

$$I_y[x,y] = I[x,y] * g[x] * g'[y] = \sum_{k=-\frac{w}{2}}^{\frac{w}{2}} \sum_{l=-\frac{w}{2}}^{\frac{w}{2}} I[k,l]g[x-k]g'[y-l],$$

### 6.2.6 Tracking and Reconstruction

Here is an example of a feature rich image being reconstructed in 3D:

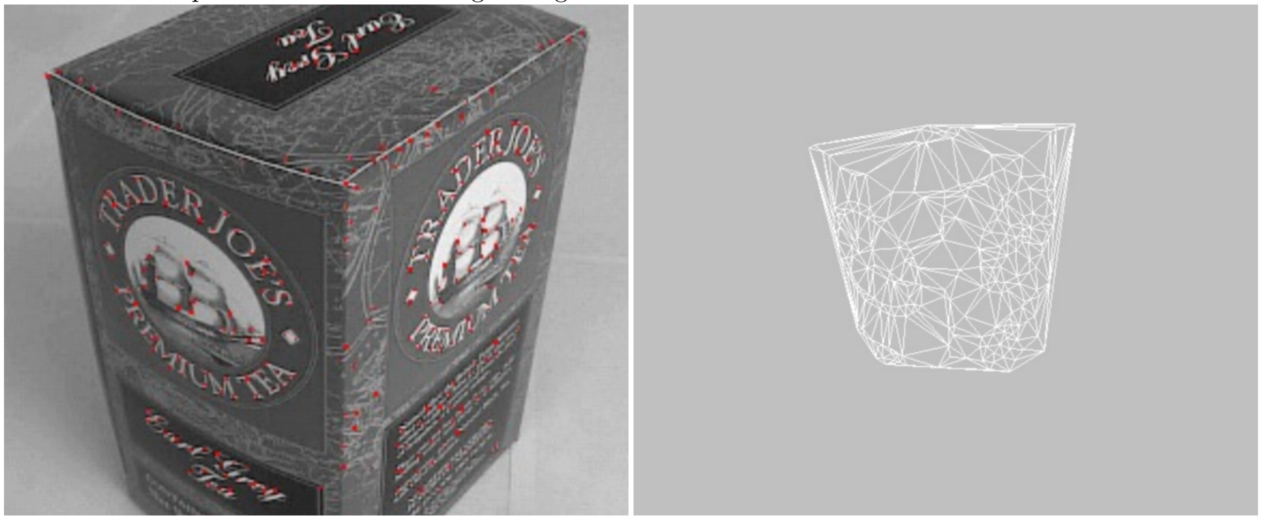

Fig:6.2.6.1 (From lecture slides) Feature Rich Image(Left) 3D Reconstruction Preview(Right)

## 6.3 Methodologies

Computer vision is still very much an open problem with new methods in continuous development. One constant is the need for edges and corners to track - "for in a world devoid of features, we are a ship lost in the fog"
- Made up quote

### 6.3.1 Point Feature Extraction

To extract features lets consider if a given pixel $(x, y)$ is a corner feature:
1) Compute the gradient of the image $(I_x, I_y)$ Refer to Appendix 4.A of the MaSKS textbook for information on filters.
2) Compute $G$ for every pixel in a window around $(x, y)$.

$$G = \begin{bmatrix} \Sigma I_x^2 & \Sigma I_x I_y \\ \Sigma I_x I_y & \Sigma I_y^2 \end{bmatrix}$$

3) Check the smallest singular value $\sigma_2(G)$ is larger than some threshold  to determine if the pixel is indeed a corner. Change  to be more or less selective!

### 6.3.2 Harris Corner Detector

The Harris Corner Detector is a variation of the above algorithm created by Harris and Stephen in 1988.

$$C(G) = det(G) + k \cdot trace^2(G)$$

$k$ is used to change the type of features selected by the Harris Corner Detector. A small value of k will select for $G$ with at least one large eigenvalue, which corresponds to only one dimension of large change in intensity. Setting $k$ larger will select for $G$ only with two large eigenvalues, corresponding to a strong corner feature.

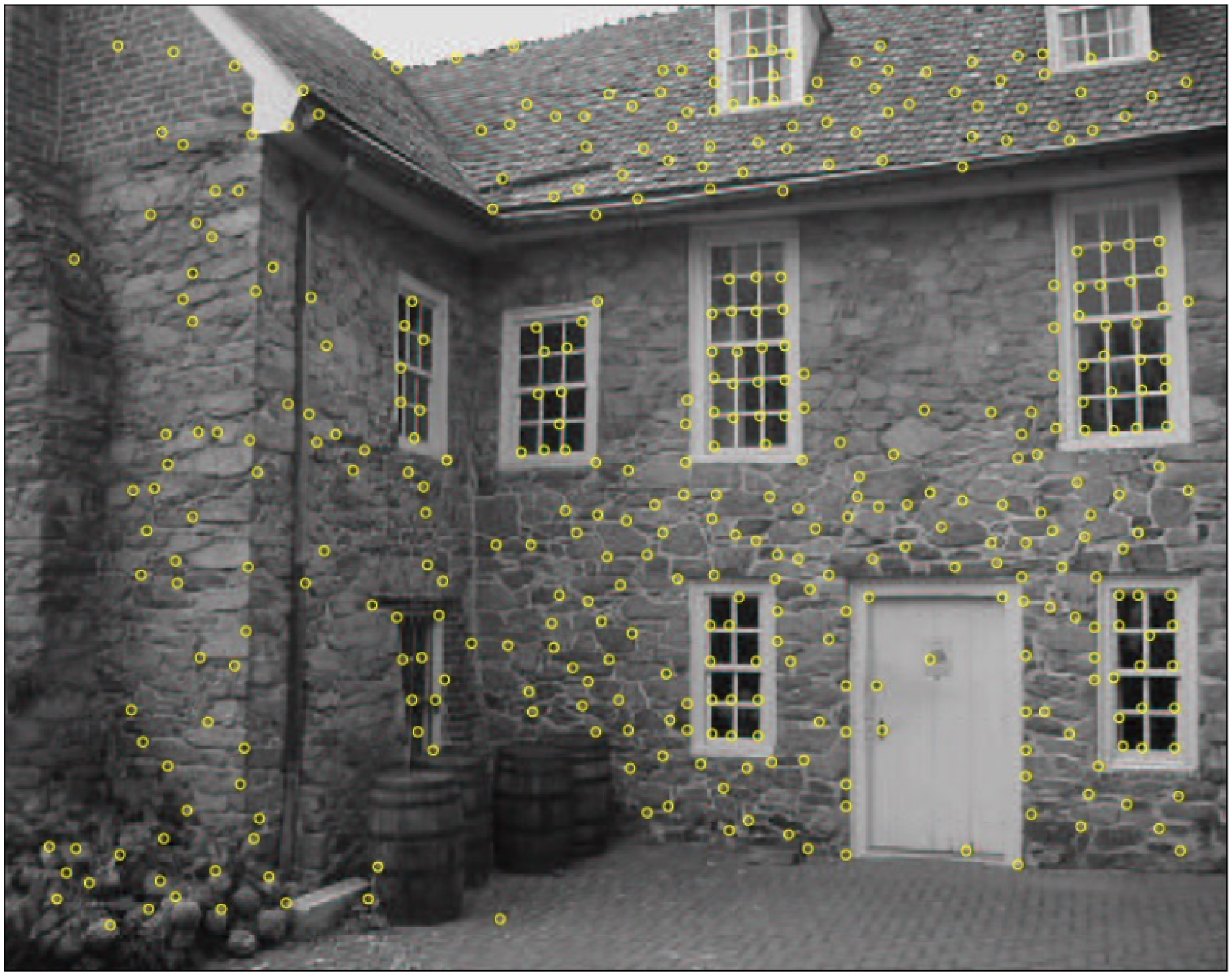
Fig:6.3.2.1 (From lecture slides) Harris corner detection example

### 6.3.3 SIFT

SIFT (Scale-Invariant Feature Transform) is a more advanced approach for matching features in the presence of scale changes in the images.

$$G(x, y, kS) = \frac{1}{2\pi (kS)^2} e^{-(x^2+y^2)/2k^2 S^2}$$

$$k = \sqrt{2}$$

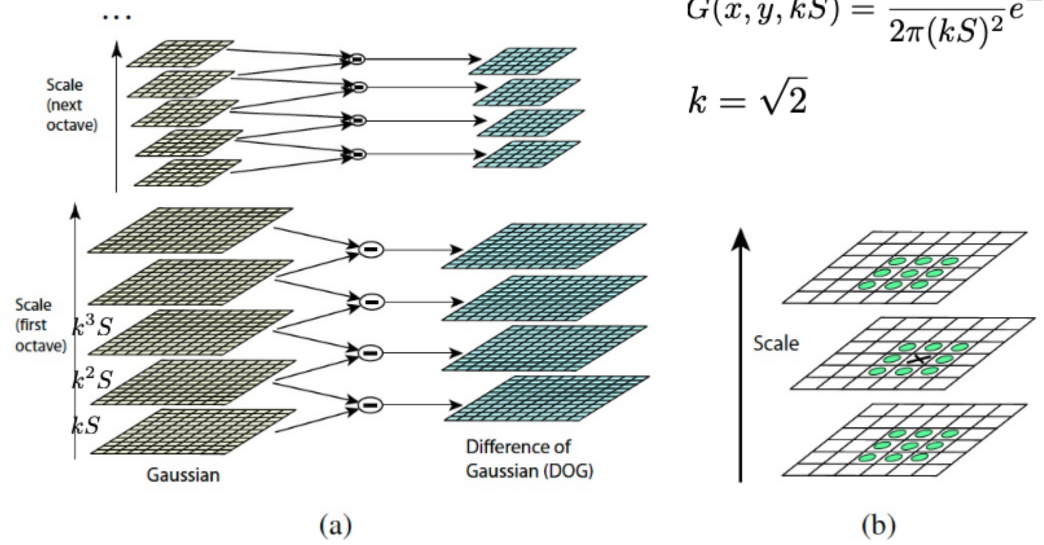

Fig:6.3.4.1 (From lecture slides) Scale-space feature detection using a sub-octave Difference of Gaussian Pyramid (Lowe 2004) © 2004 Springer: (a) adjacent levels of a sub-octave Gaussian pyramid are subtracted to produce Difference of Gaussian images; (b) extrema (maxima and minima) in the resulting 3D volume are detected by comparing a pixel to its 26 neighbors.

SIFT uses a concept called Keys to track these features. These keys are a set of image gradients in some window. The direction of these gradients are then compiled into a histogram, which will be unique and invariant to scale.

### 6.3.4 Affine Invariance

The Affine Invariant tracker is not as fast or efficient as Small-baseline tracking algorithms but can track features more robustly for an extended duration without loosing them to environment, scale, or angle changes in the images. If two images are taken from drastically different view points, we need to use Affine Invariant detectors to match features.

### 6.3.5 Other Examples

Many other examples exist and each has pros and cons. It's important to choose the right feature detection algorithm for a particular use based on resources and requirements.

- Maximally Stable Extremal Regions (**MSER**) detector developed by(Matas, Chum et al. 2004).
- **SURF** (Bay, Ess et al. 2008), which uses integral images for faster convolutions.
- **FAST** and **FASTER** (Rosten, Porter, and Drummond 2010), one of the first learned detectors.
- **BRISK** (Leutenegger, Chli, and Siegwart 2011), which uses a scale-space FAST detector together with a bit-string descriptor.

- **ORB** (Rublee, Rabaud et al. 2011), which adds orientation to FAST.
- **KAZE** (Alcantarilla, Bartoli, and Davison 2012) and **Accelerated-KAZE** (Alcantarilla, Nuevo, and Bartoli 2013), which use non-linear diffusion to select the scale for feature detection.
- **Mother of All Features**(Microsoft), used Learned Features.

## References

[1] Szeliski, Richard. Computer Vision: Algorithms and Applications. Springer International Publishing, 2022. DOI.org (Crossref), https://doi.org/10.1007/978-3-030-34372-9.

[2] Ma, Yi, editor. An Invitation to 3-D Vision: From Images to Geometric Models. Springer, 2004.