上海科技大学
ShanghaiTech University

CS283: **Robotics Spring 2025: Vision II**

Sören Schwertfeger

ShanghaiTech University

IMAGE FEATURES

- Lines
- Points
    - Harris
    - SIFT

Example: Build a Panorama

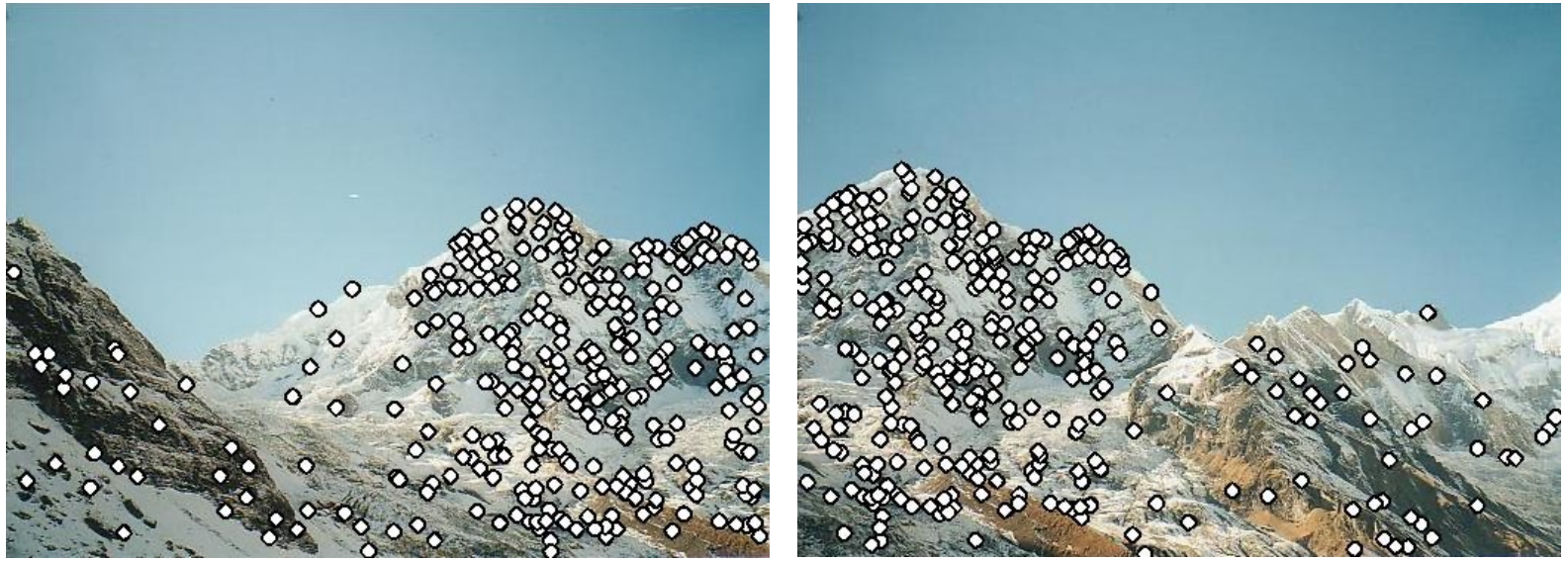
This panorama was generated using **AUTOSTITCH** (freeware), available at
http://www.cs.ubc.ca/~mbrown/autostitch/autostitch.html

How do we build panorama?

- We need to match (align) images

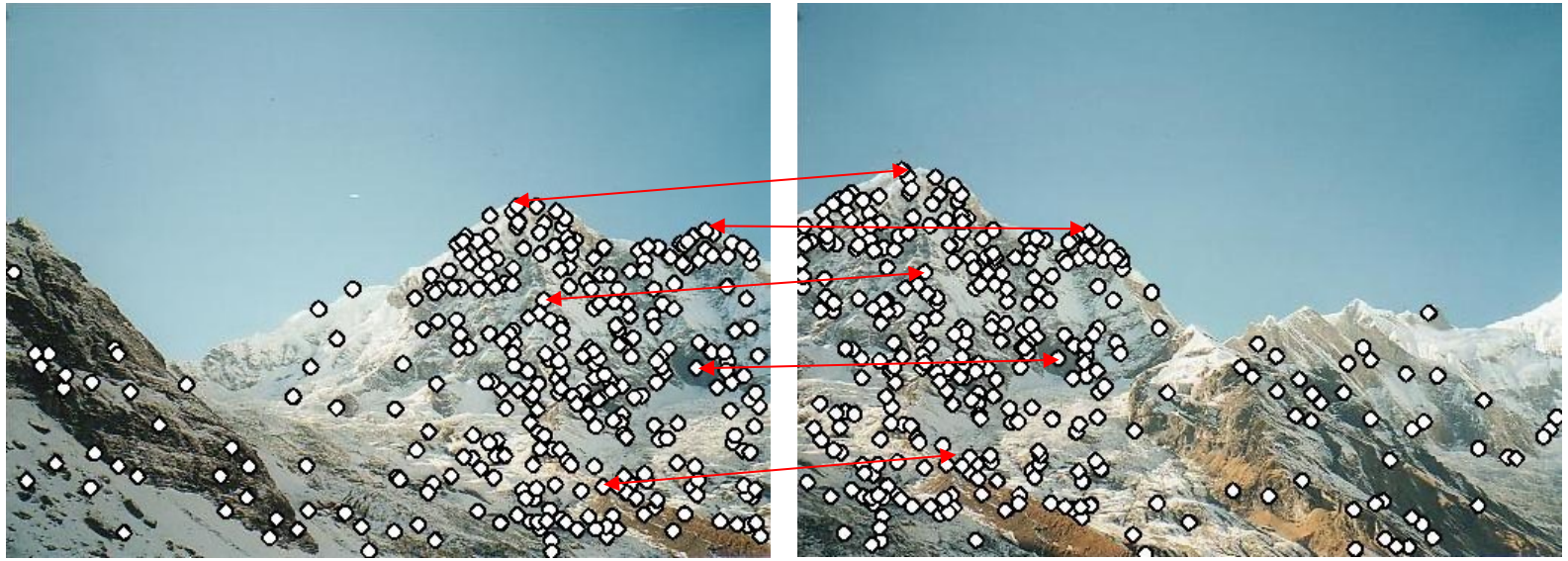
Matching with Features

- Detect feature points in both images



Matching with Features

- Detect feature points in both images
- Find corresponding pairs
  - Use RANSAC to find inliers and outliers

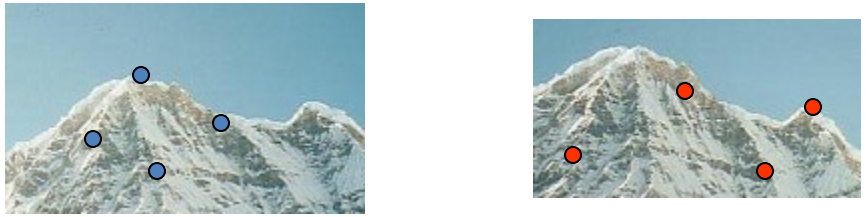


Matching with Features

- Detect feature points in both images
- Find corresponding pairs
- Use these pairs to align images

Matching with Features

- Problem 1:
  - Detect the *same* point *independently* in both images

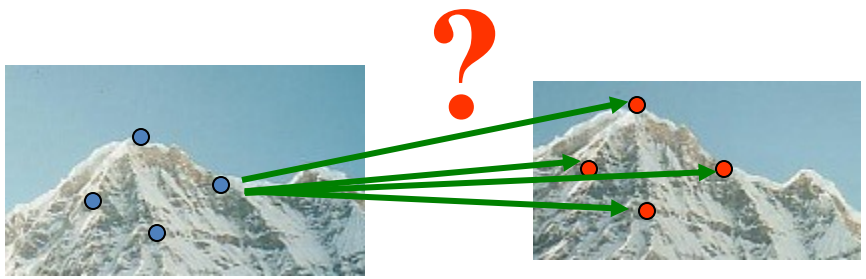


no chance to match!

We need a repeatable detector

Matching with Features

- Problem 2:
  - For each point correctly recognize the corresponding one



We need a reliable and distinctive descriptor

More motivation…

- Feature points are used also for:

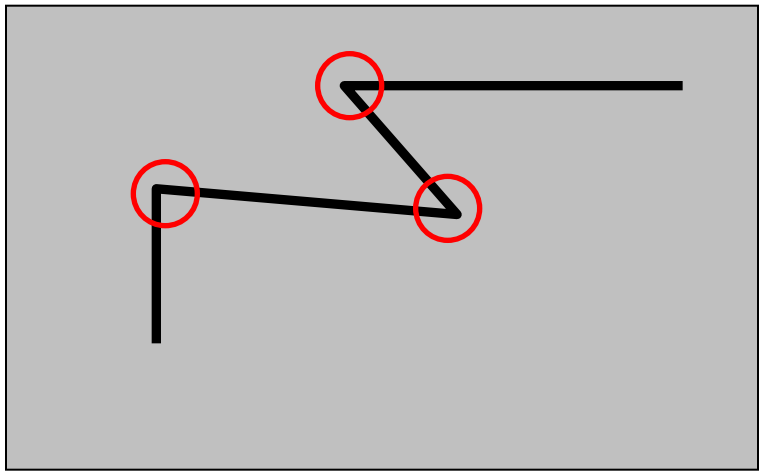
  - Robot navigation
  - Object recognition
  - Image alignment (panoramas)
  - 3D reconstruction
  - Motion tracking
  - Indexing and database retrieval -> Google Images

  … other

Most Famous Feature Detectors

- Overview
  - Harris Detector (1988)
  - SIFT Detector (2004)

HARRIS CORNER DETECTOR



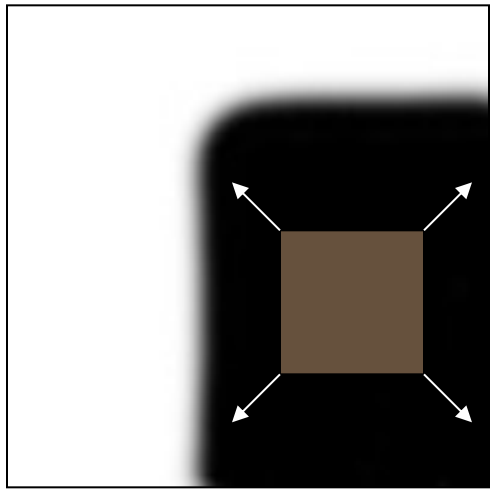
C.Harris, M.Stephens. “A Combined Corner and Edge Detector”. 1988

Finding Corners

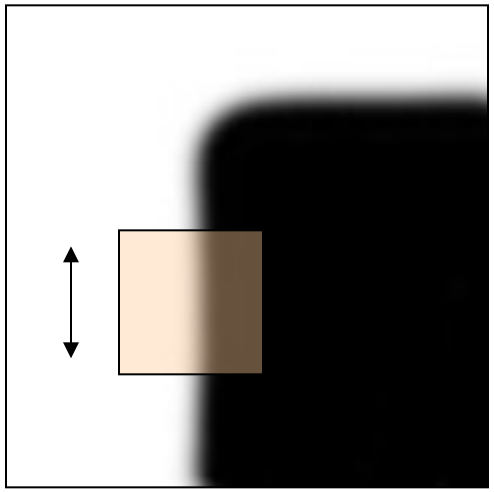
- Key property: in the region around a corner, image gradient has two or more dominant directions
- Corners are repeatable and distinctive

The basic idea

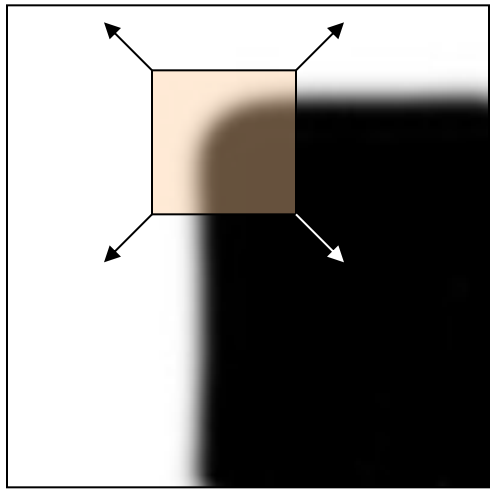
- We should easily recognize the point by looking through a small window
- Shifting a window in *any direction* should give *a large change* in intensity
- => define a corner response function



"flat" region:
no change in all directions



"edge":
no change along the edge direction



"corner":
significant change in all directions

How do we implement this?

- Let *I* be a grayscale image. Consider taking an image patch centered on (u,v) and shifting it by (x,y). The Sum of Squared Differences between these two patches is given by:

$$SSD(x,y) = \sum_u \sum_v ((I(u,v)) - I(u+x, v+y))^2$$

- $I(u+x, v+y)$ can be approximated by a first order Taylor expansion. Let I_x and I_y be the partial derivatives of *I*, such that

$$I(u+x, v+y) \approx I(u,v) + I_x(u,v)x + I_y(u,v)y$$

- This produces the approximation

$$SSD(x,y) \approx \sum_u \sum_v (I_x(u,v)x + I_y(u,v)y)^2$$

- Which can be written in a matrix form as $SSD(x,y) \approx \begin{bmatrix} x & y \end{bmatrix} M \begin{bmatrix} x \\ y \end{bmatrix}$

How do we implement this?

$$SSD(x, y) \approx \begin{bmatrix} x & y \end{bmatrix} M \begin{bmatrix} x \\ y \end{bmatrix}$$

- M is the “second moment matrix”

- Since M is symmetric, we can rewrite M as

$$M = \sum_u \sum_v \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = \begin{bmatrix} \sum\sum I_x^2 & \sum\sum I_x I_y \\ \sum\sum I_x I_y & \sum\sum I_y^2 \end{bmatrix}$$

where λ_1 and λ_2 are the eingenvalues of M

$$M = R^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R$$

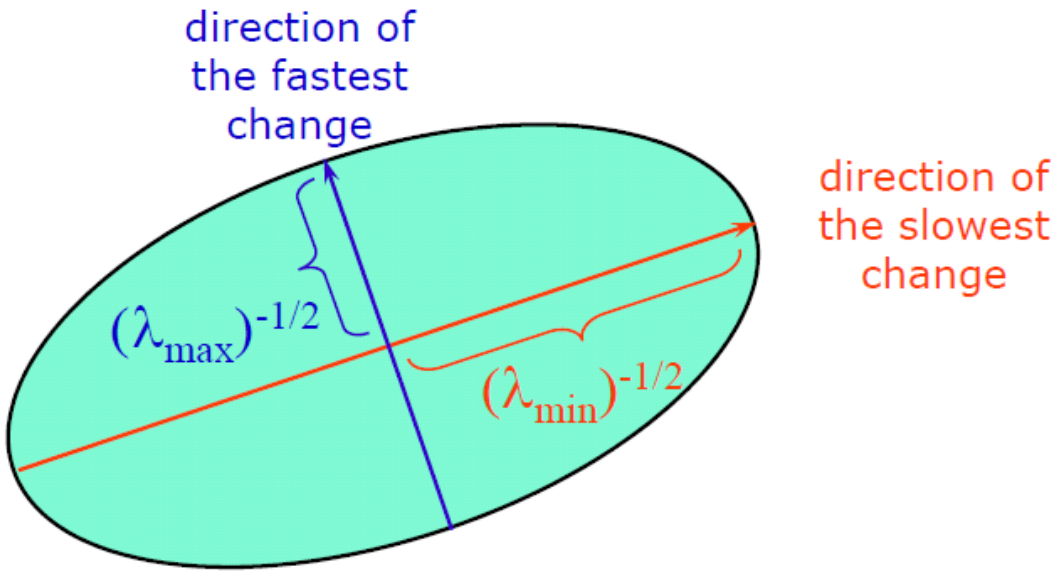
How do we implement this?

- As mentioned before, a corner is characterized by a large variation of in all directions of the vector (x,y). The Harris detector analyses the eigenvalues of M to decide if we are in presence of a corner or not.

- We can visualize *M as an ellipse with axis lengths determined by the* eigenvalues and orientation determined by *R*

Ellipse equation:

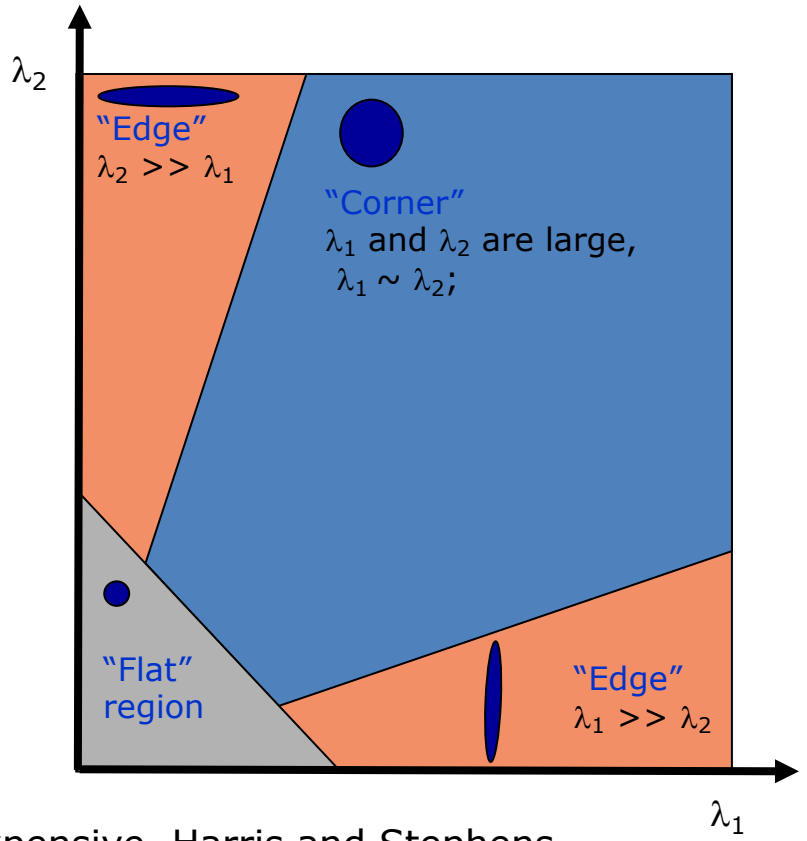
$$[u \;\; v] \; M \begin{bmatrix} u \\ v \end{bmatrix} = \text{const}$$



Corner response function

Based on the magnitudes of the eigenvalues, the following inferences can be made based on this argument:

- If both λ_1 and λ_2 are small, SSD is almost constant in all directions (i.e. we are in presence of a flat region).
- If either $\lambda_1 >> \lambda_2$ or $\lambda_2 >> \lambda_1$, we are in presence of an edge: SSD has a large variation only in one direction, which is the one perpendicular to the edge.
- If both λ_1 and λ_2 are large, SSD has large variations in all directions and then we are in presence of a corner.



Because the calculation of the eigenvalues is computationally expensive, Harris and Stephens suggested the use of the following "**cornerness function**" instead:

$$C = \lambda_1\lambda_2 - \kappa(\lambda_1 + \lambda_2)^2 = det(M) - \kappa \cdot trace^2(M)$$

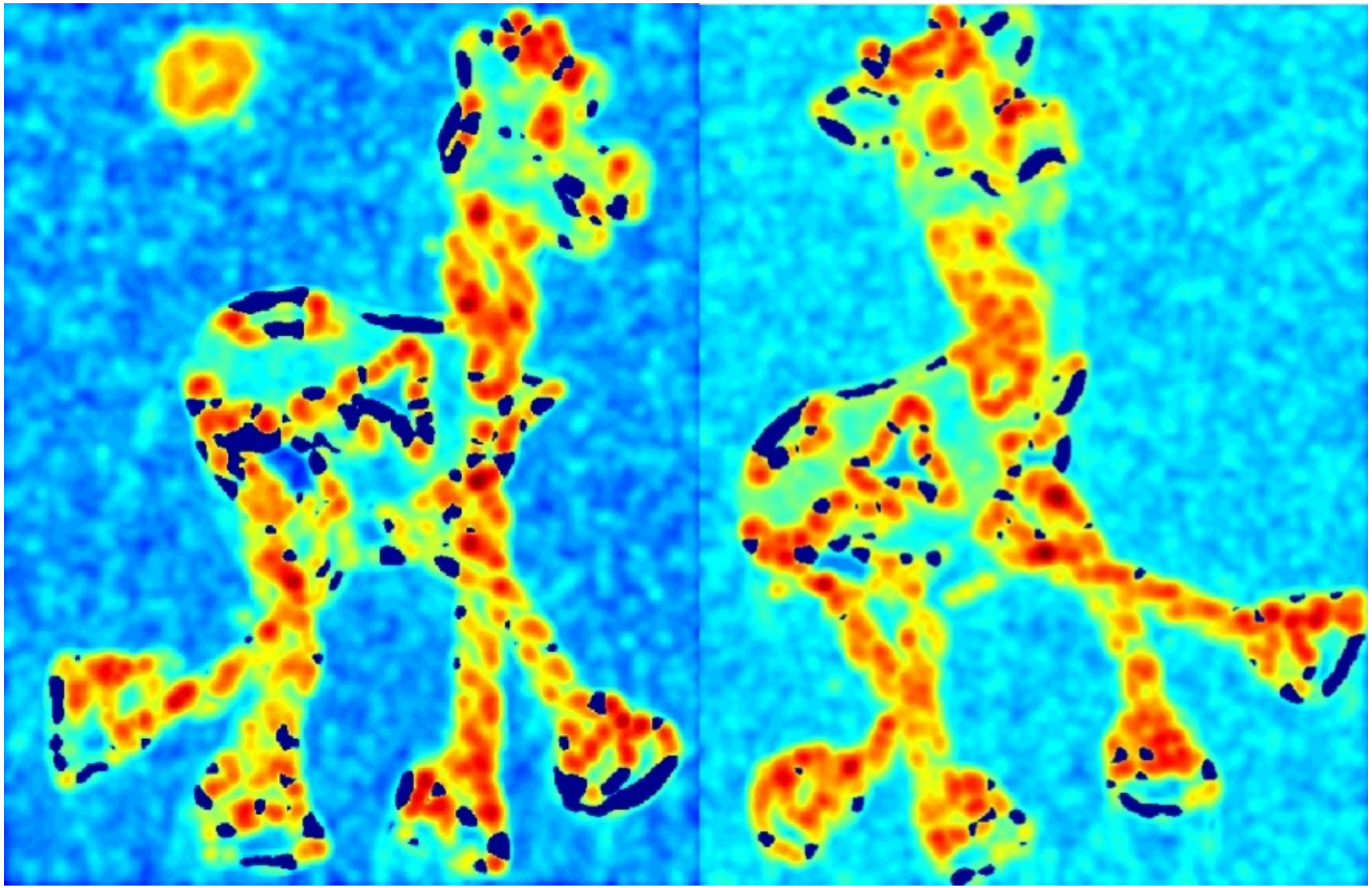
Where k is a between 0.04 and 0.15

- Finally, the last step of the Harris corner detector consists in extracting the local maxima of the cornerness function

Harris Detector: Workflow

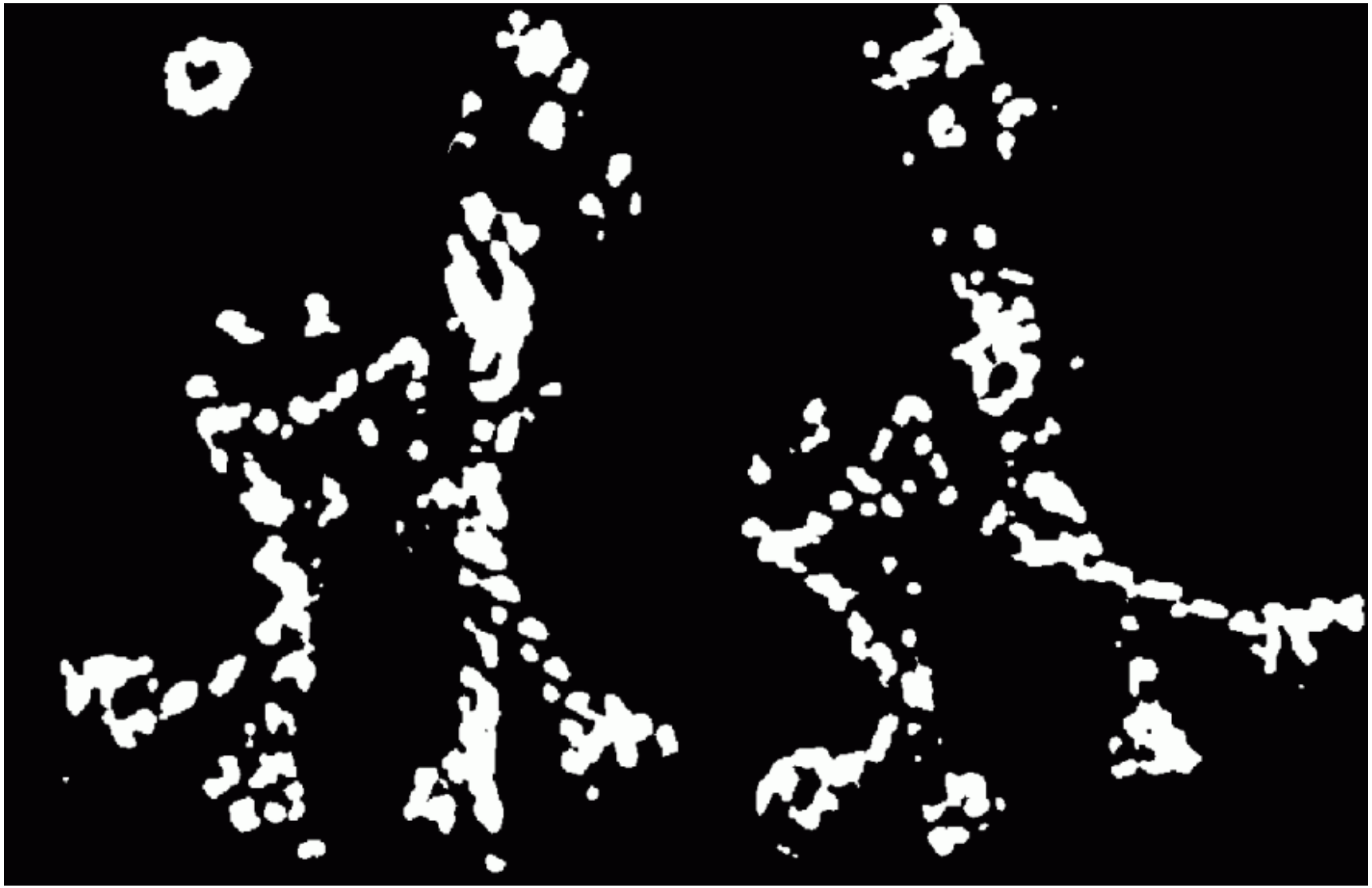
Harris Detector: Workflow

- Compute corner response *R*



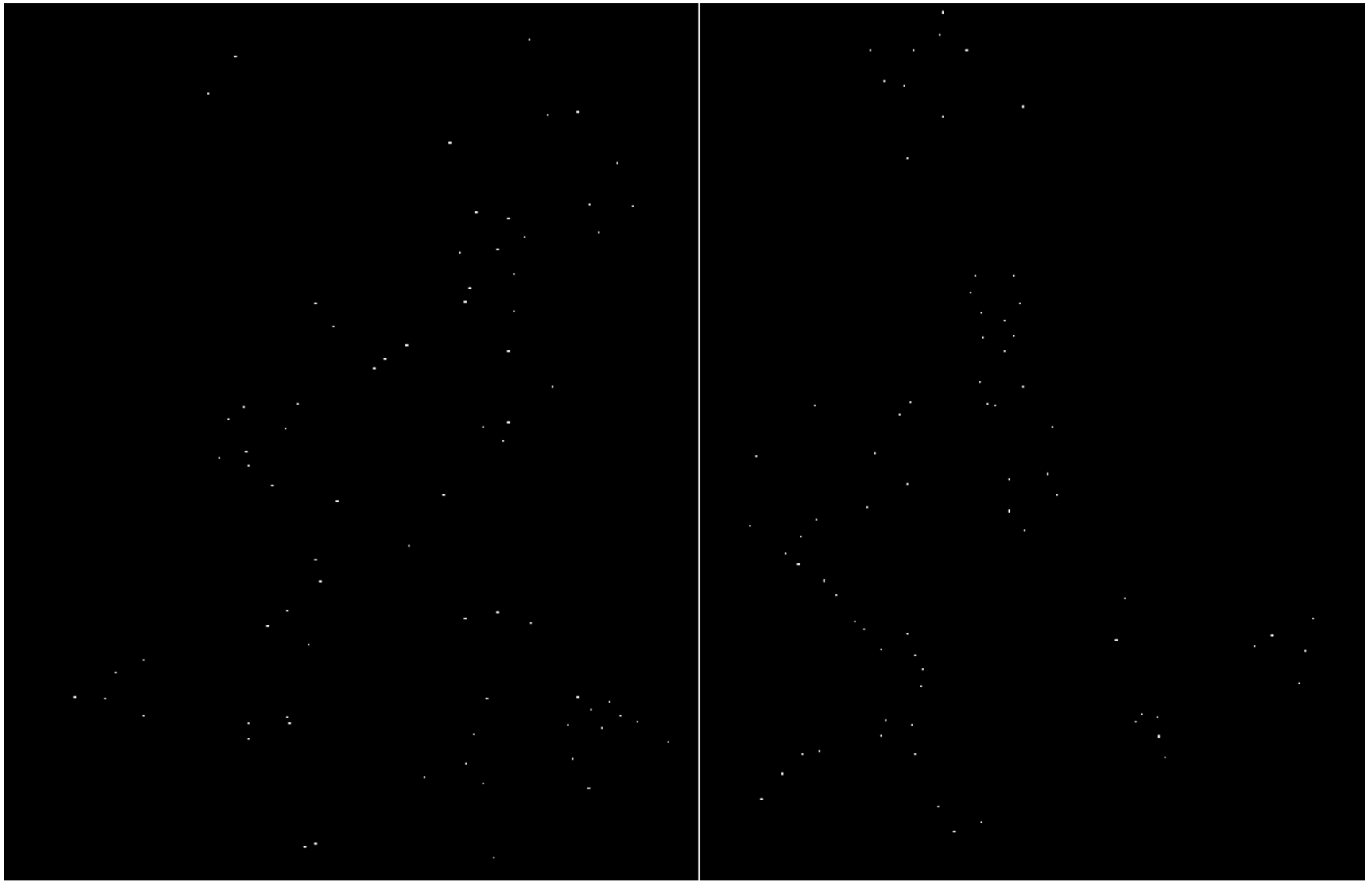
Harris Detector: Workflow

- Find points with large corner response: $R >$ threshold



Harris Detector: Workflow

- Take only the points of local maxima of R



Harris Detector: Workflow

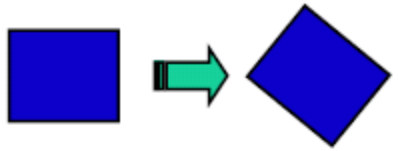
Harris detector: properties

- How does the Harris detector behave to image transformations?
- Will we be able to re-detect the same corners under
  - Rotations
  - View-point changes
  - Zoom changes
  - Illumination changes?

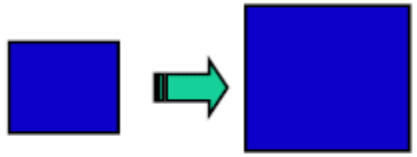
- In order to answer these questions, we need a model of these transformations.
- The detector can then be modified to be invariant to such transformations

Models of Image Change

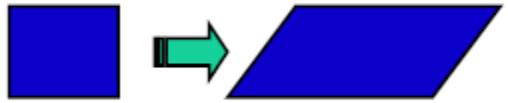
- Geometric
  - **Rotation**



  - **Scale**

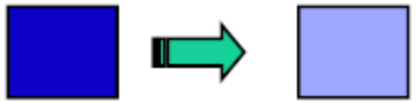


  - **Affine**



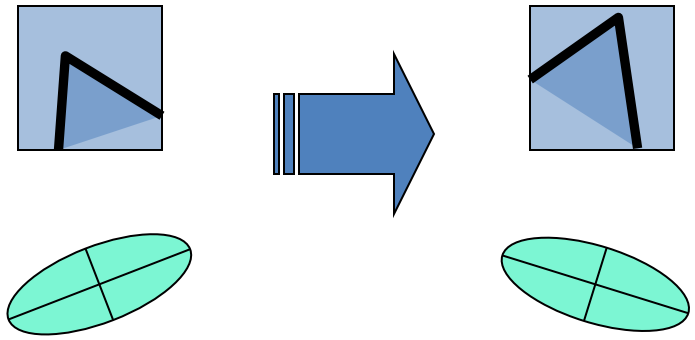
valid for locally planar object

- Photometric
  - **Affine intensity change** ($I \rightarrow a\,I + b$)



Harris Detector: Some Properties

- Rotation invariance

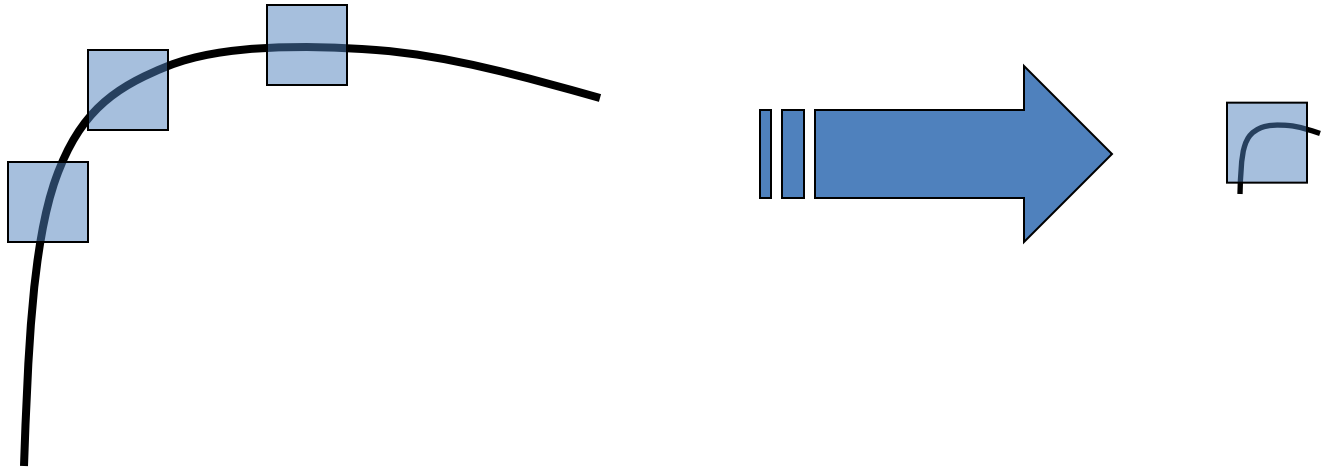


Ellipse rotates but its shape (i.e. eigenvalues) remains the same

Corner response R is invariant to image rotation

Harris Detector: Some Properties

- But: non-invariant to *image scale*!



All points will be classified as edges

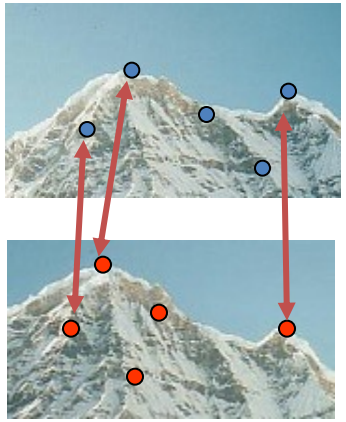
Corner !

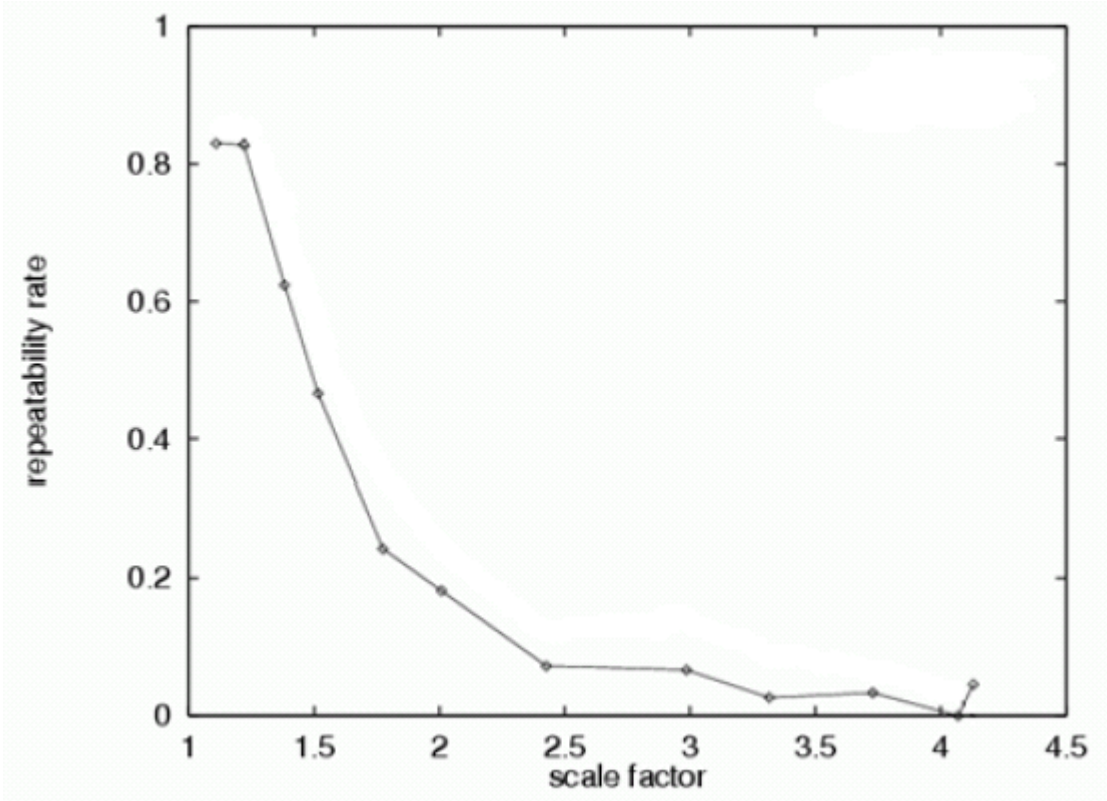
Harris Detector: Some Properties

- Quality of Harris detector for different scale changes

Repeatability rate:

$$\frac{\text{\# correspondences}}{\text{\# possible correspondences}}$$





Summary on Harris properties

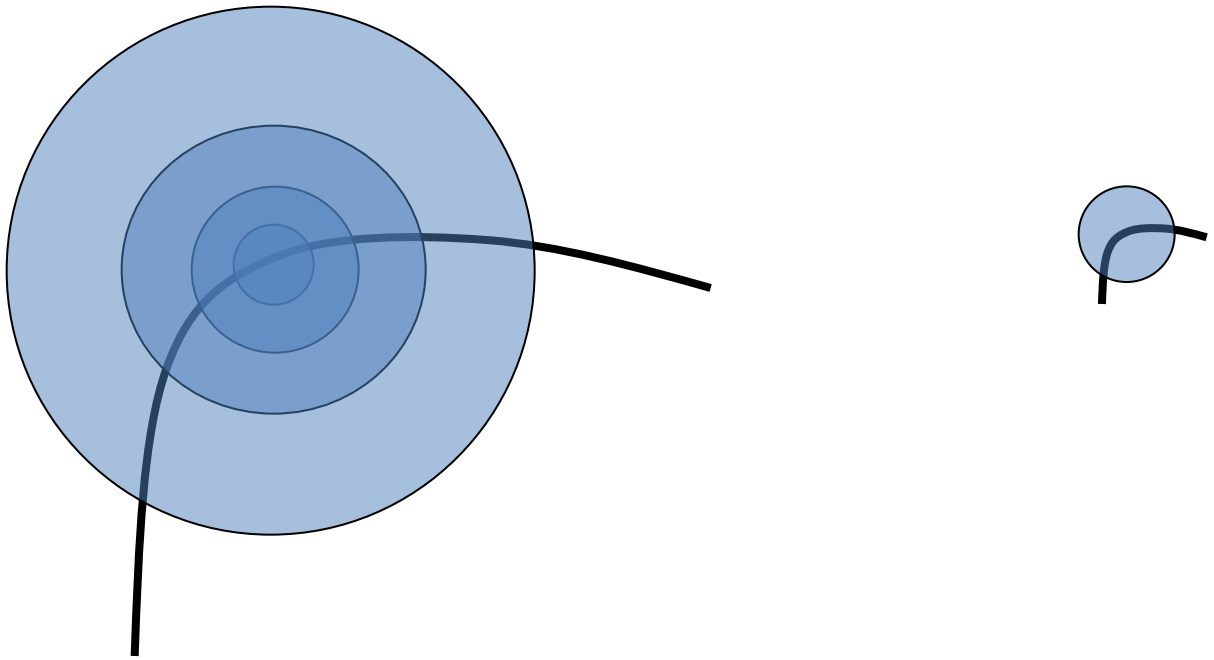
- Harris detector is an approach for detecting and extracting corners (i.e. points with high intensity changes in all directions)

- The detection is Invariant to
  - Rotation
  - Linear intensity changes
  - However, to make the matching invariant to these we need an opprtune matching criterion (for example, SSD in not Rotation nor affine invariant!)

- The detection is NOT invariant to
  - Scale changes
  - Geometric affine changes (Intuitively, an affine transformation distorts the neighborhood of the feature along the x and y directions and, accordingly, a corner can get reduced or increased its curvature)

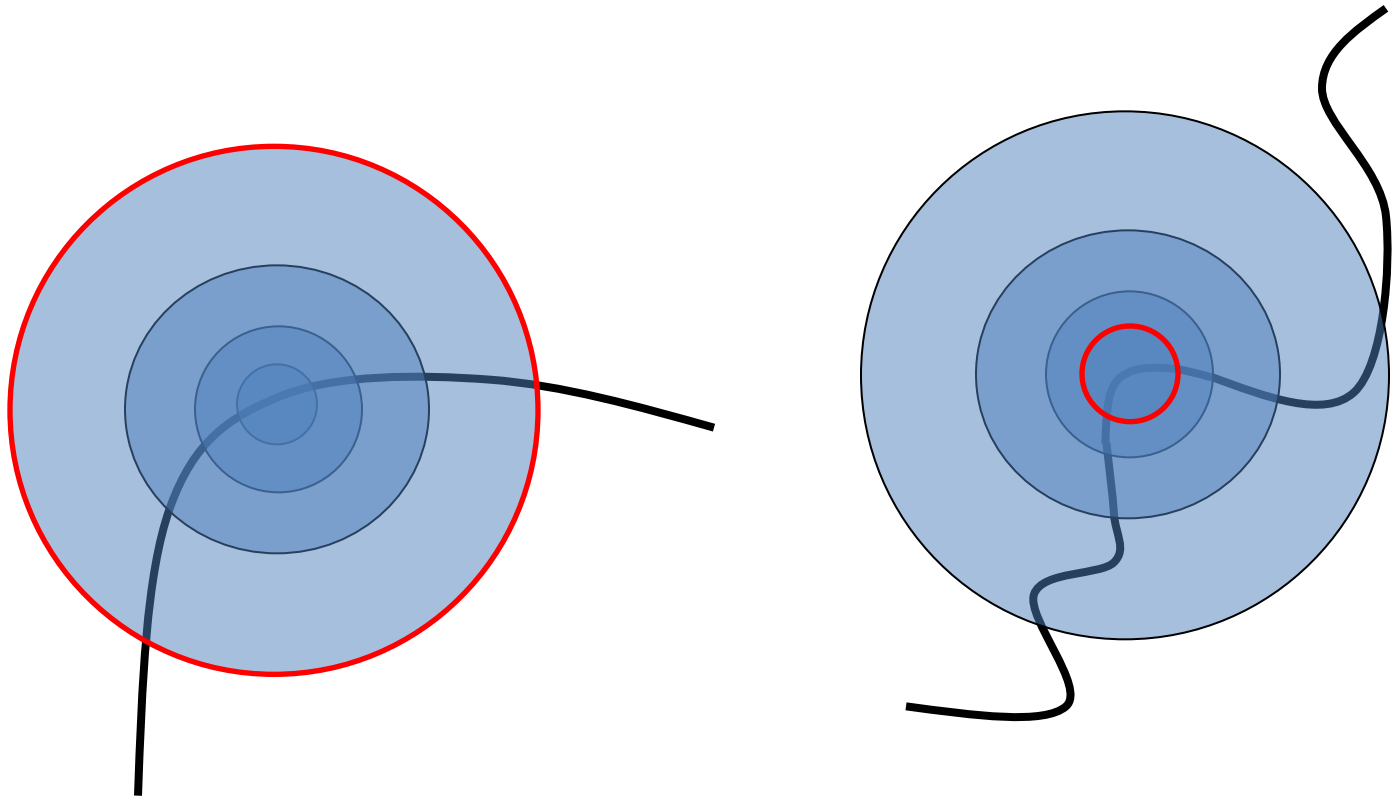
Scale Invariant Detection

- Consider regions (e.g. circles) of different sizes around a point
- Regions of corresponding sizes will look the same in both images



Scale Invariant Detection

- The problem: how do we choose corresponding circles ***independently*** in each image?

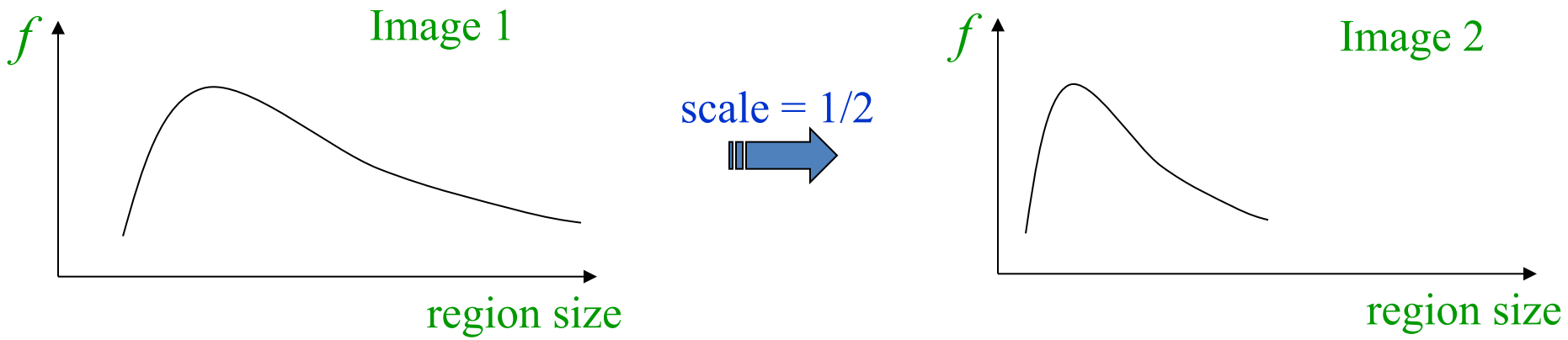


Scale Invariant Detection

- Solution:
  - Design a function on the region (circle), which is “scale invariant” (the same for corresponding regions, even if they are at different scales)

Example: average intensity. For corresponding regions (even of different sizes) it will be the same.

  - For a point in one image, we can consider it as a function of region size (circle radius)



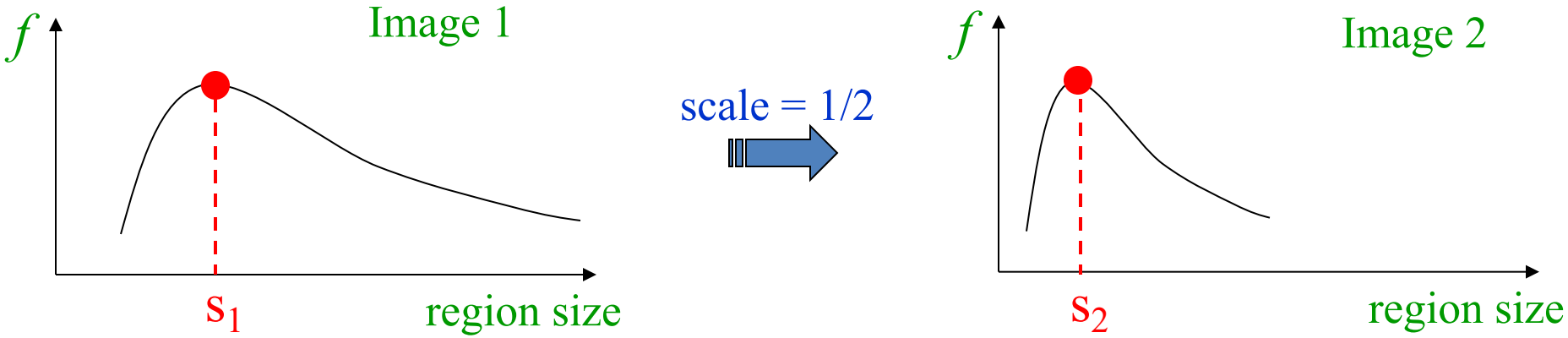
Scale Invariant Detection

- Common approach:

  Take a local maximum of this function

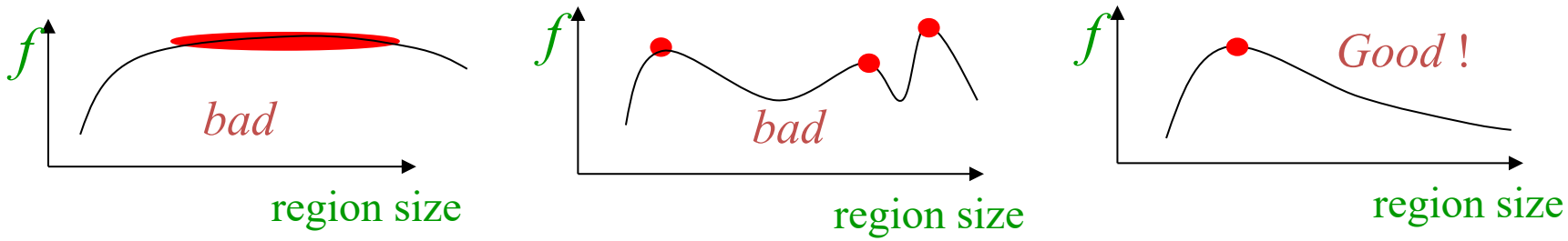
  Observation: region size, for which the maximum is achieved, should be *invariant* to image scale.

Important: this scale invariant region size is found in each image independently!



Scale Invariant Detection

- A “good” function for scale detection:
  has one stable sharp peak



- For usual images: a good function would be a one which responds to contrast (sharp local intensity change)

Scale Invariant Detection

- Functions for determining scale

$$f = \text{Kernel} * \text{Image}$$

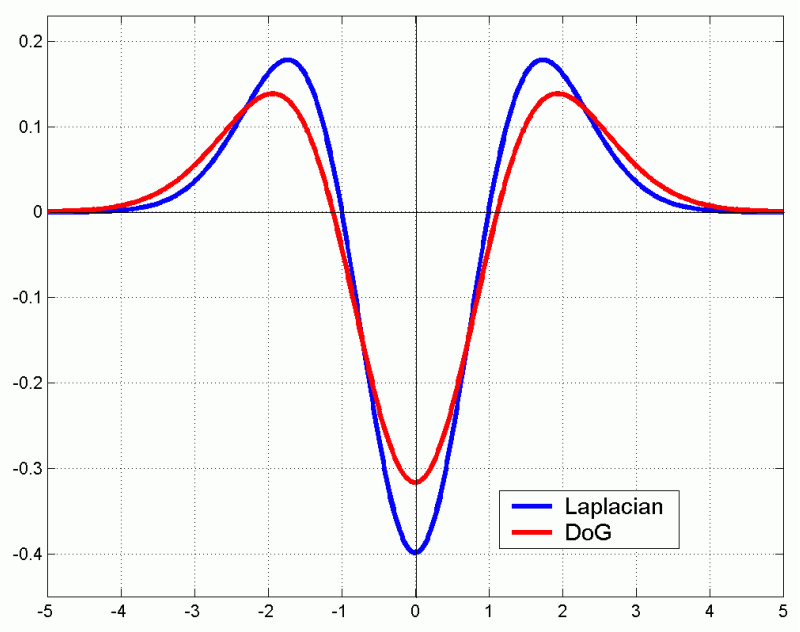
Kernels:

$$DoG = G(x, y, k\sigma) - G(x, y, \sigma)$$

(Difference of Gaussians)

where Gaussian

$$G(x, y, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2+y^2}{2\sigma^2}}$$



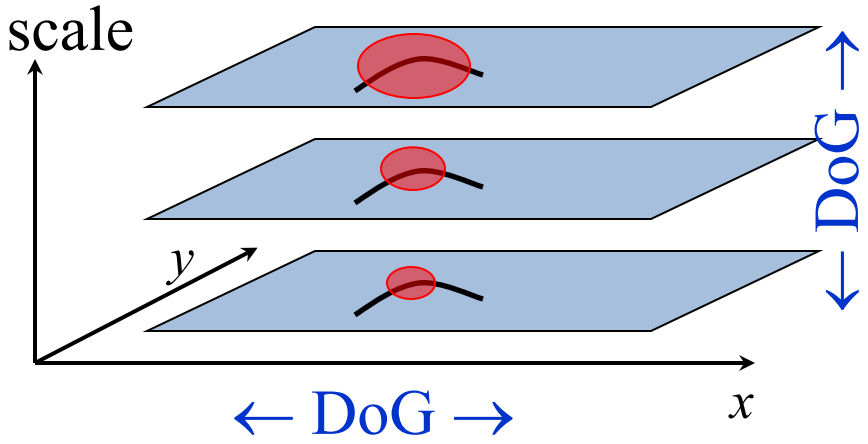
Note: This kernel is invariant to *scale* and *rotation*

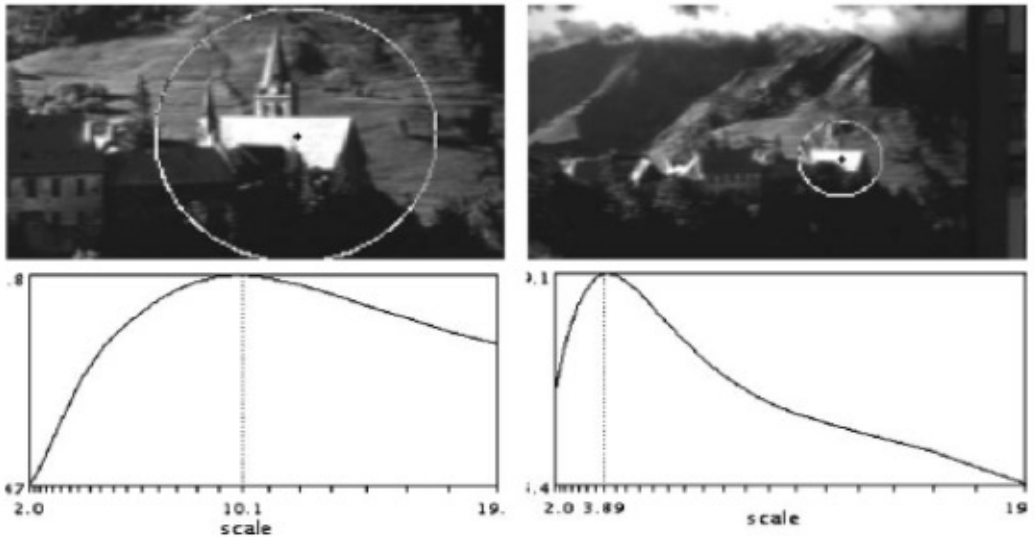
Scale Invariant Detectors

- SIFT

  Find local maximum of:

  – Difference of Gaussians in space and scale



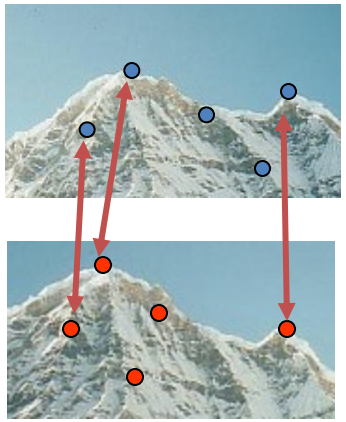


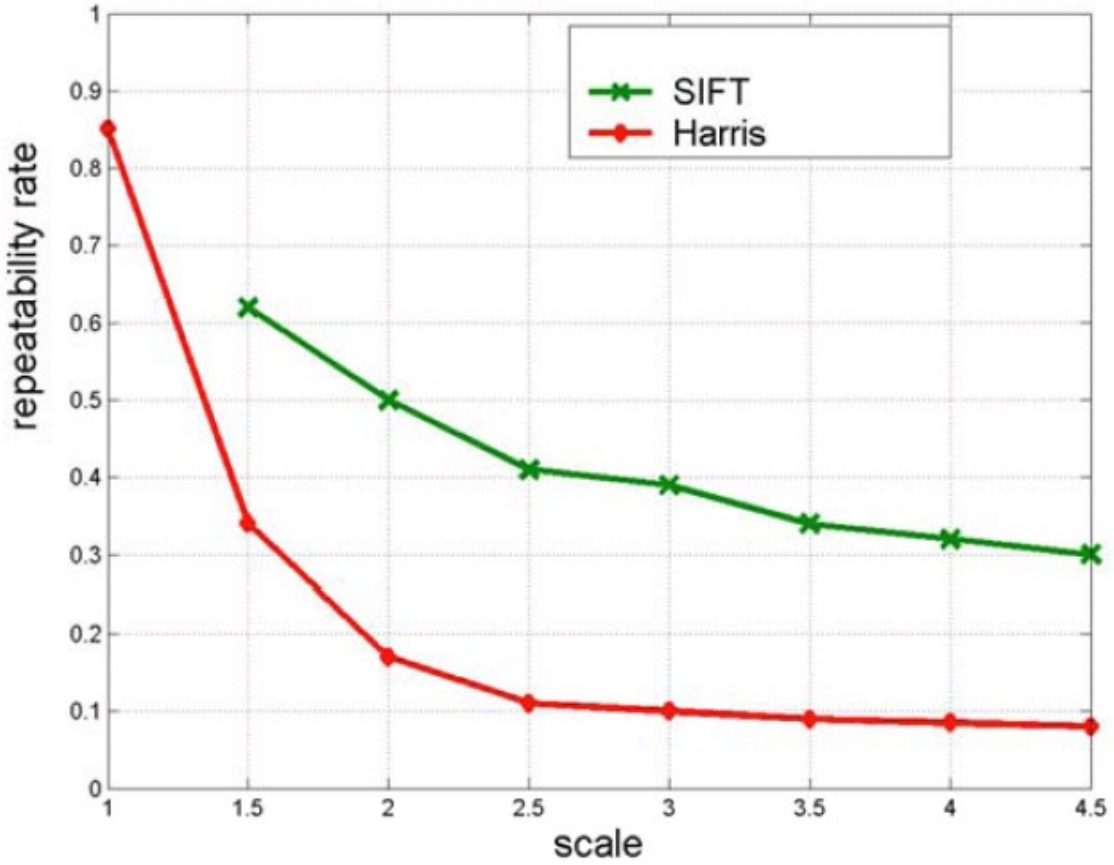
Scale Invariant Detectors

- Experimental evaluation of detectors w.r.t. scale change

Repeatability rate:

$$\frac{\text{\# correspondences}}{\text{\# possible correspondences}}$$



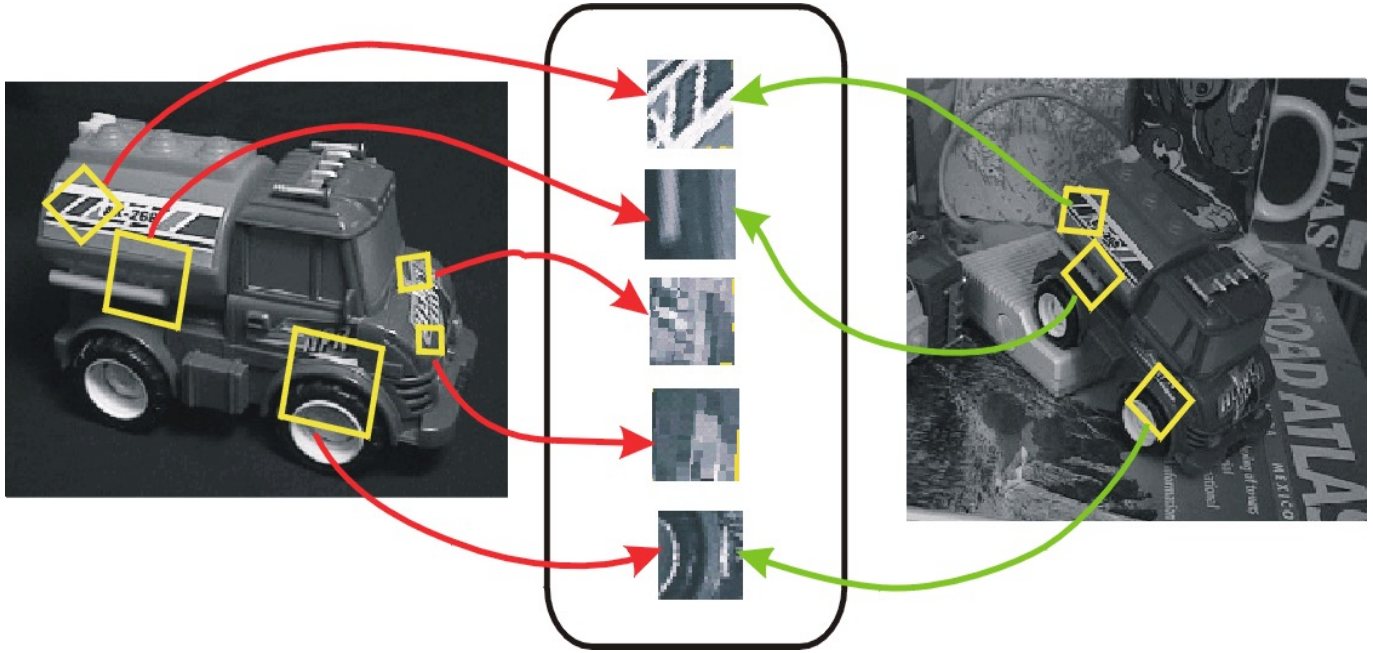


What is SIFT ?

- Scale Invariant Feature Transform (SIFT) is an approach for detecting and extracting local feature descriptors that are reasonably invariant to changes in:
  - rotation
  - scaling
  - small changes in viewpoint
  - illumination

Invariant Local Features

- Image content is transformed into local feature coordinates that are invariant to translation, rotation, scale, and slight view-point and illumination changes



SIFT Descriptor

- SIFT matching features between different images invariant to:
  - Rotation
  - Small image scale
  - Small view point changes

- This is made possible by the use of an opportune feature descriptor (remember, Harris does not use a descriptor for matching)

- What is a descriptor? A descriptor is a “description”, identity card, which allows to recognize a given feature uniquely among many others!

Generation of Keypoint Descriptor

- Keypoint descriptor: computing gradient magnitude and orientation at each image sample point in a region around the keypoint location
- Samples are accumulated into orientation histograms with the length of each bin corresponding to the sum of the gradient magnitudes near that direction within the region.
- The descriptor is therefore a vector containing the values of all the orientations histogram entries
- This vector is normalized to enhance invariance to changes in illumination
- Peaks in the histogram correspond to dominant orientations. If more than one peak is found, a separated feature is assigned to the same point location.

- All the properties of the keypoint are measured relative to the keypoint orientation, this provides invariance to rotation.

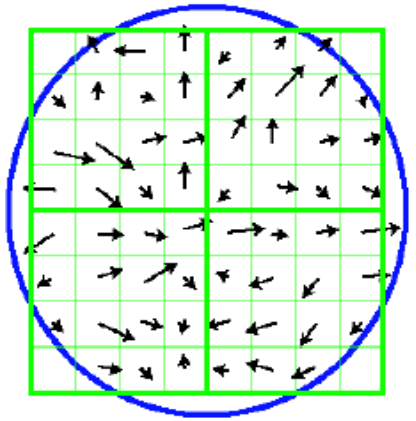
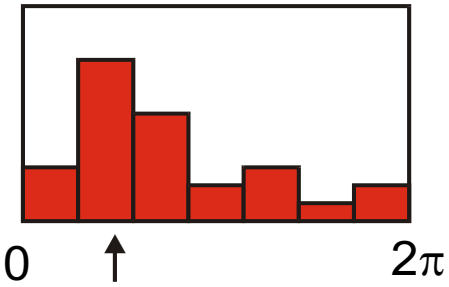
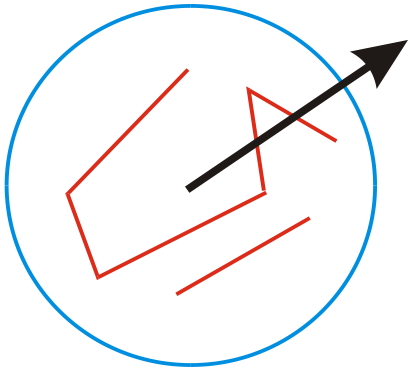


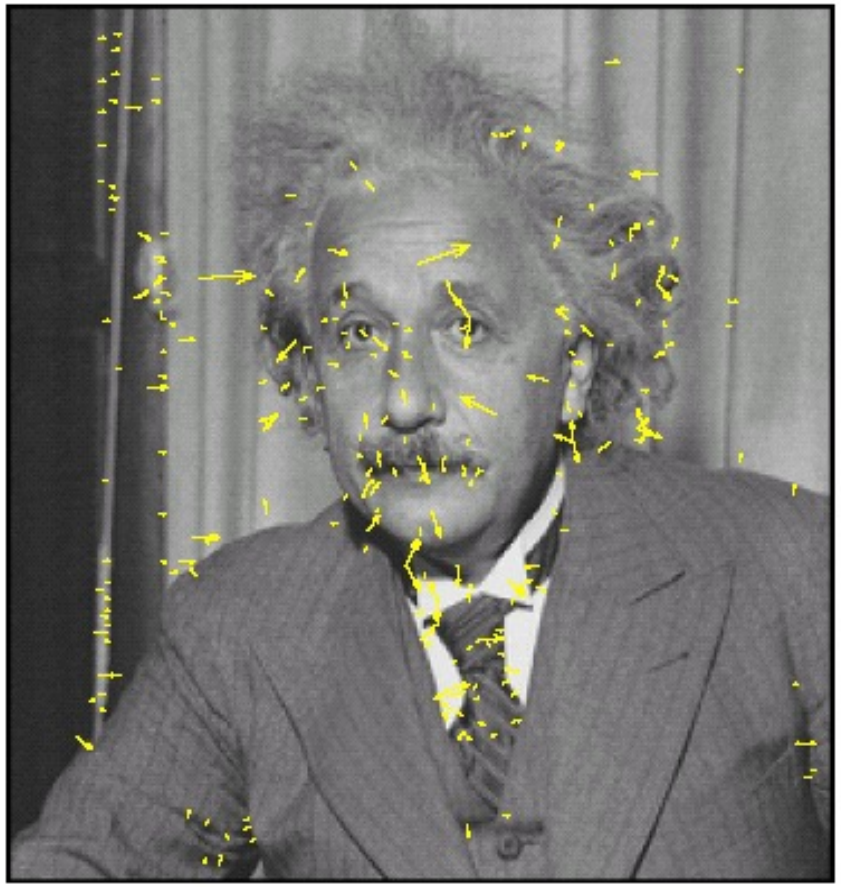
Image gradients



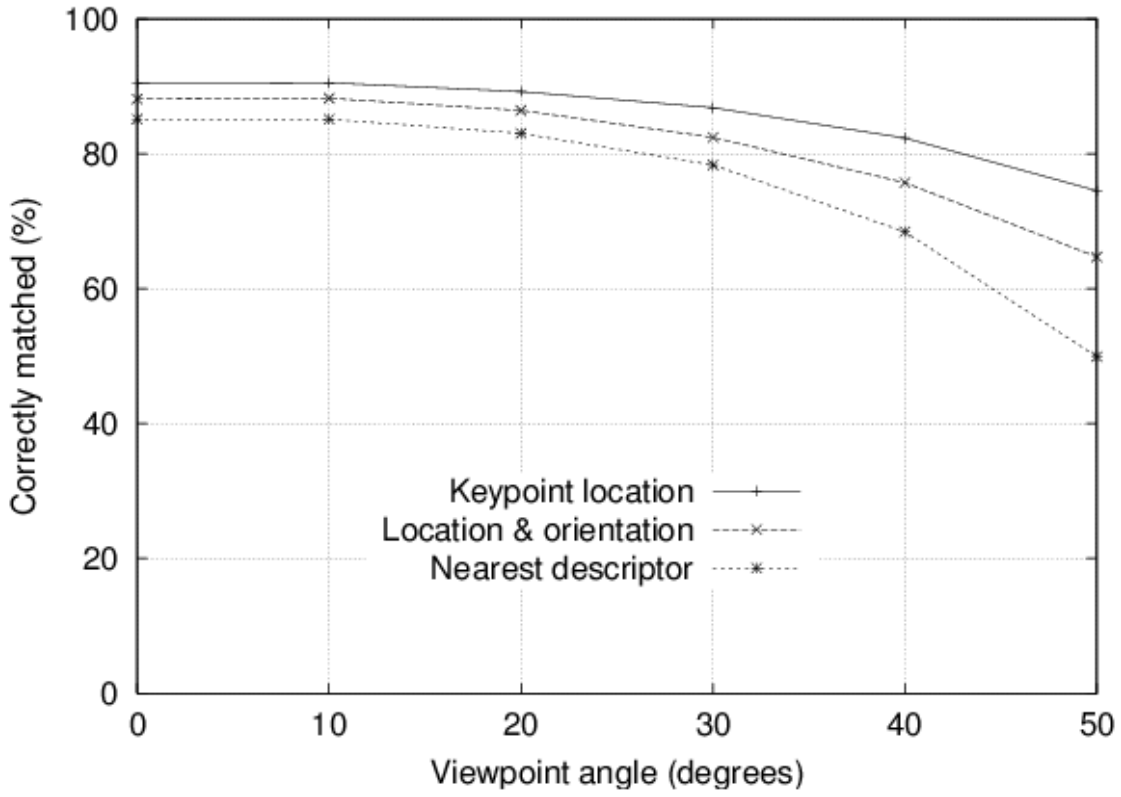


Key point localization

- Final keypoints with selected orientation and scale

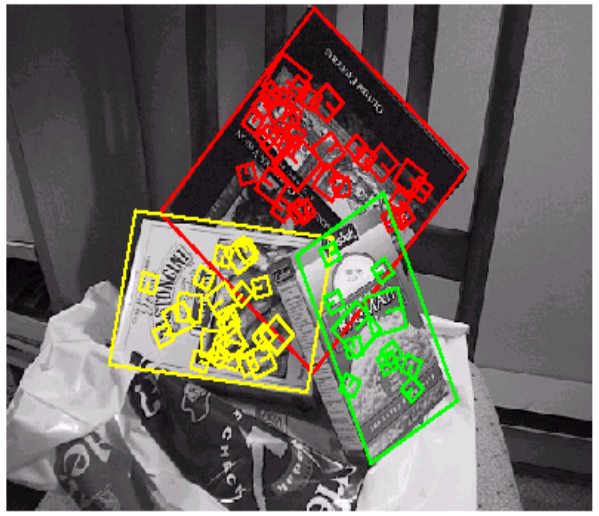


Feature stability to view point change

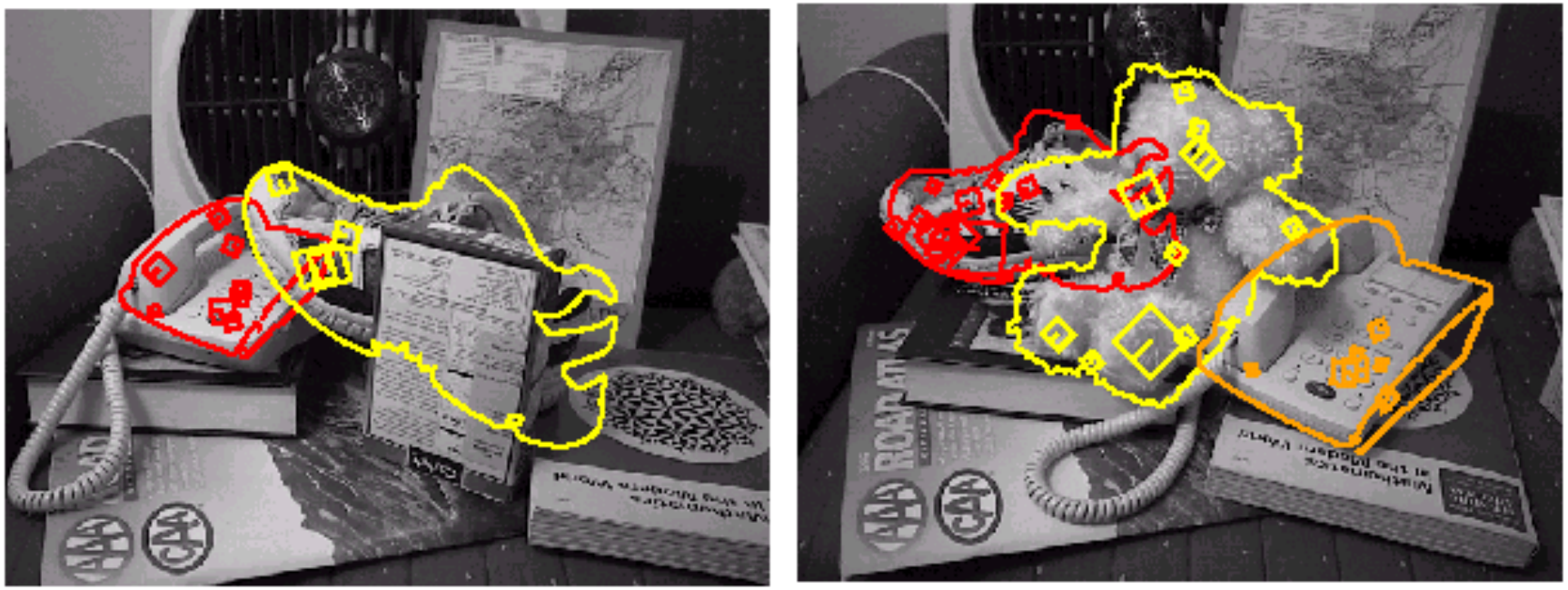


Planar recognition

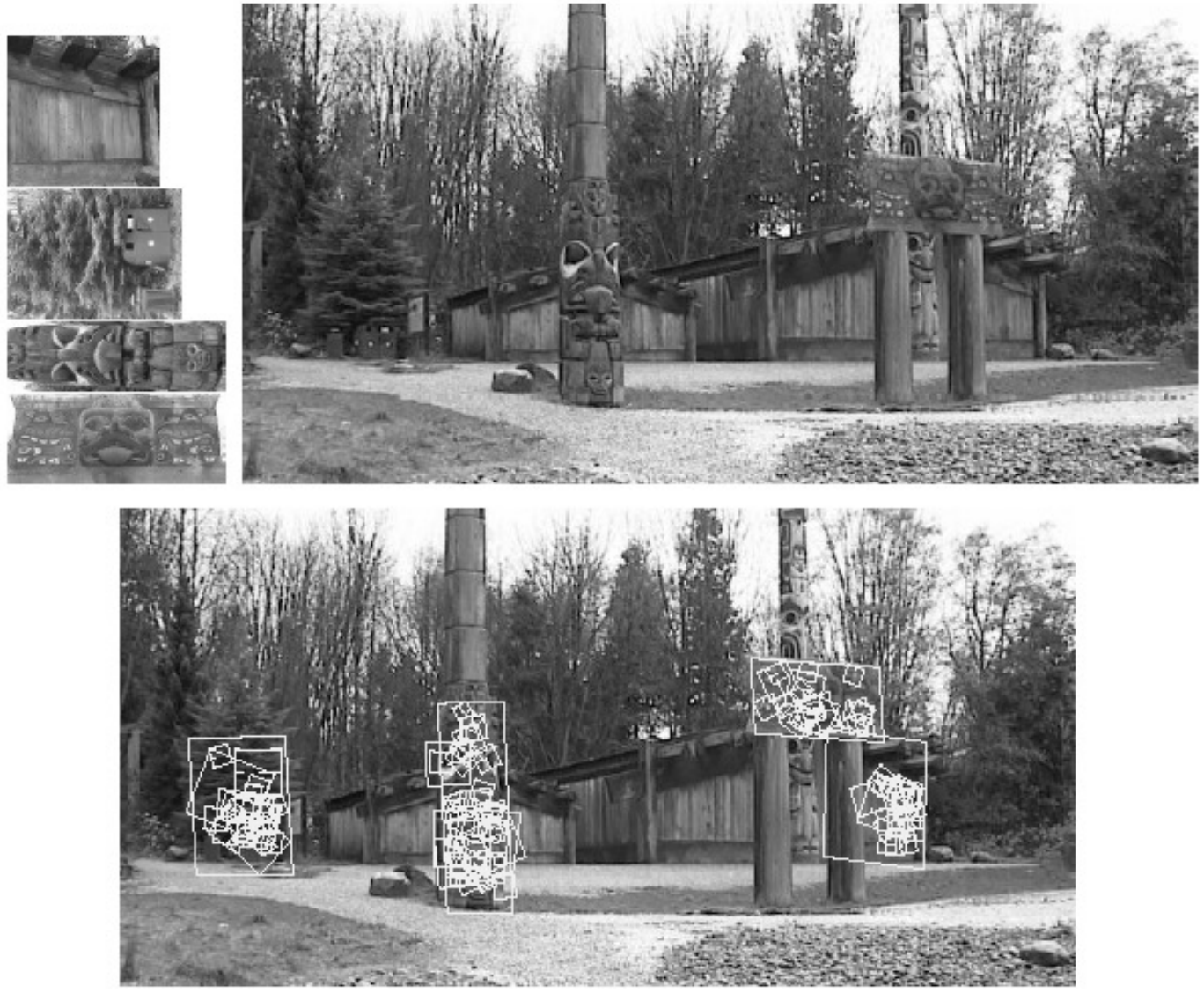
- Planar surfaces can be reliably recognized at a rotation of 60° away from the camera
- Only 3 points are needed for recognition
- But objects need to possess enough texture (i.e. no monochrome objects!)



Recognition under occlusion



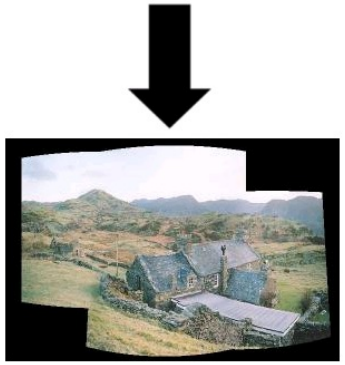
Place recognition



Multiple panoramas from an unordered image set

SIFT is used in current consumer cameras and smart phones to build panoramas from multiple shots!

Input images



Output panorama 1

DEEP LEARNING FOR ROBOTIC VISION

Deep Learning

- Hot trend in Machine Learning / AI
- Traditional approaches: Hand tuned features, e.g.:
  - Feature extraction (Computer Vision);
    Hidden Markov Models & Statistics (Natural Language Processing)
- Deep Learning approach:
  - Learning of parameters for Artificial Neural Network (ANN) based on training data
- Recent success based on:
  - Huge amounts of (training) data
  - Lots of computing power
  - Better DL architectures

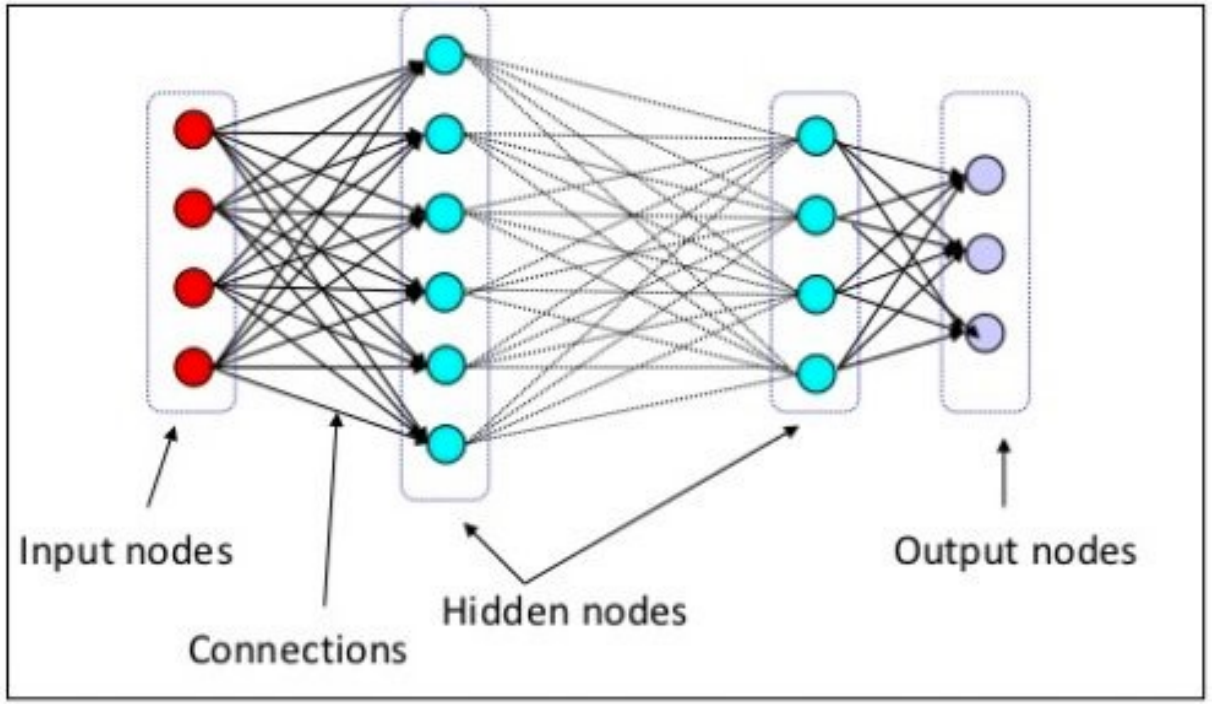


Image Source: Matthew Mayo

Traditional Pattern Recognition: Fixed/Handcrafted Feature Extractor

Feature Extractor → Trainable Classifier

Mainstream Pattern Recognition 9until recently)

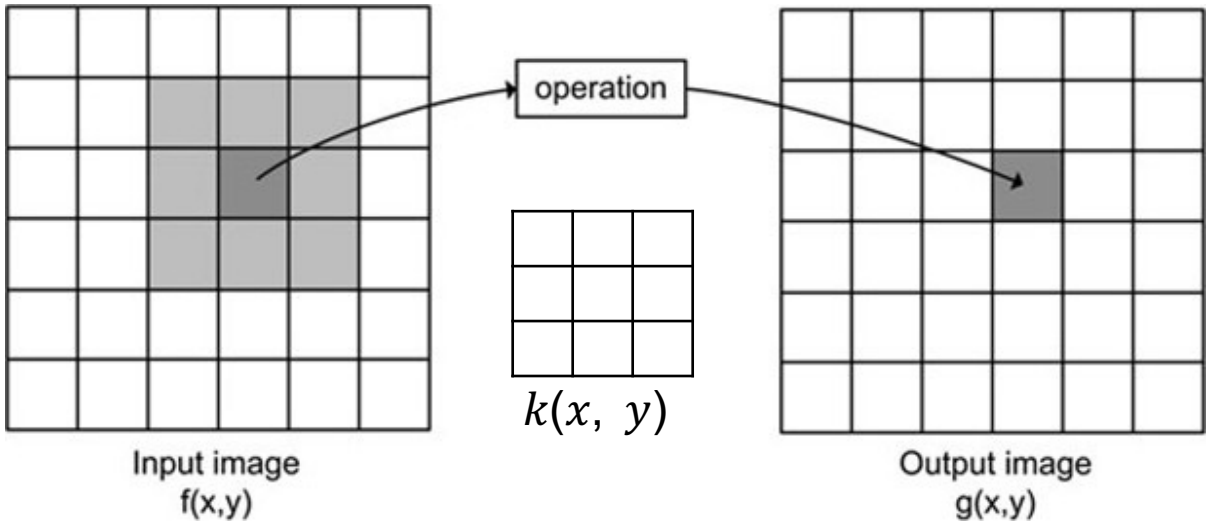
Feature Extractor → Mid-Level Features → Trainable Classifier

Deep Learning: Multiple stages/layers trained end to end

Low-Level Features → Mid-Level Features → High-Level Features → Trainable Classifier

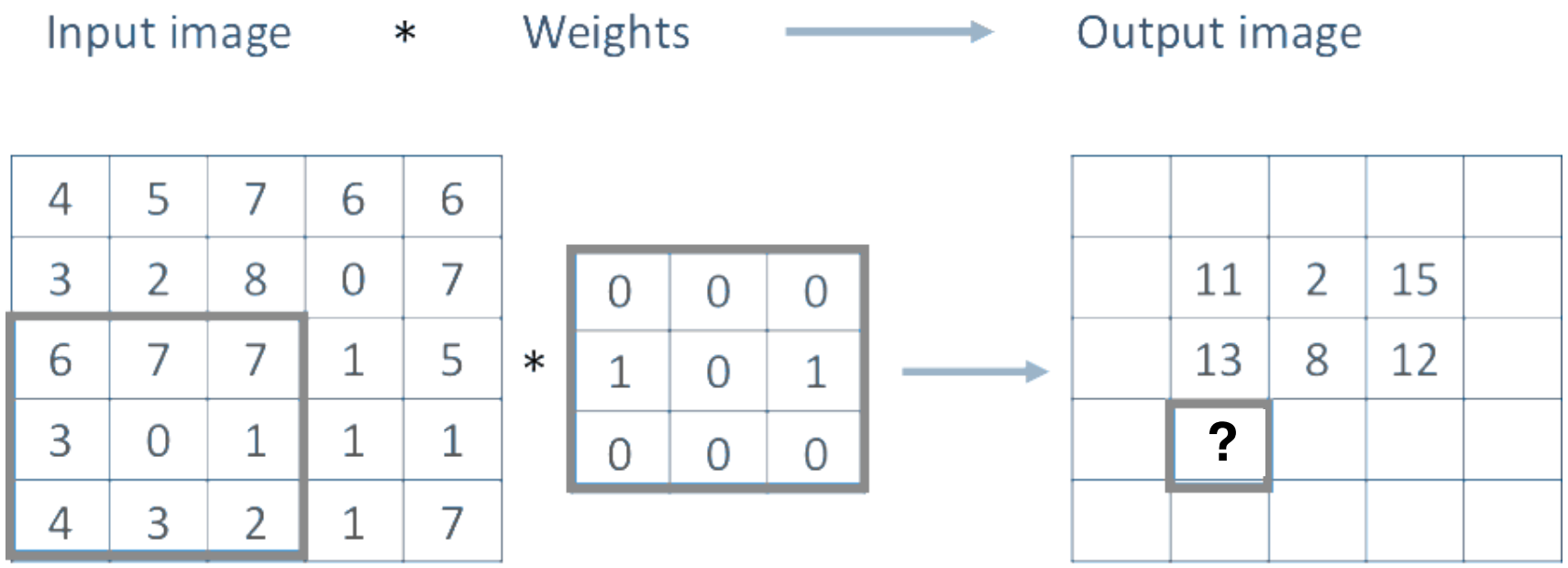
NIPS'2015 DL Tutorial http://www.iro.umontreal.ca/~bengioy/talks/DL-Tutorial-NIPS2015.pdf

Convolution operator

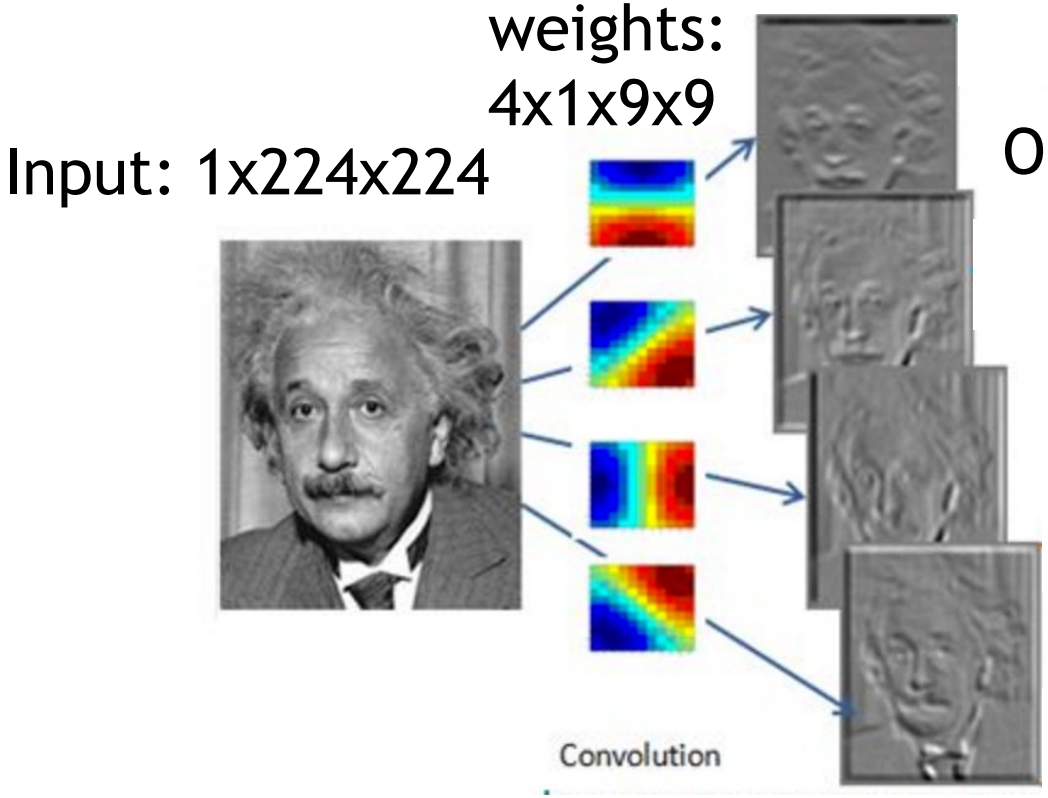


$$g(x,\ y) = \sum_{v} \sum_{u} k(u, v) f(x\ - u,\ y - v)$$

(filter, kernel)

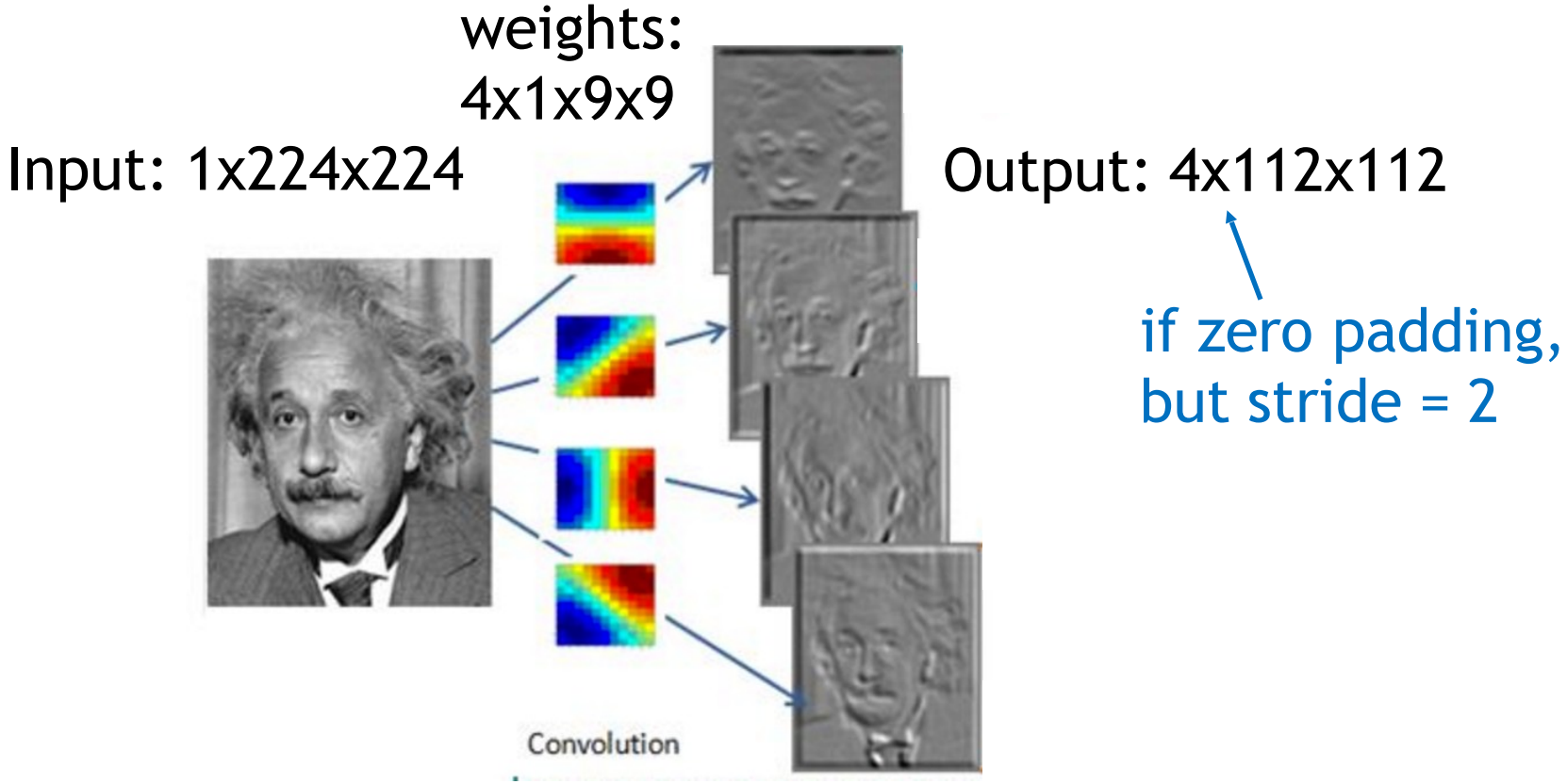


Convolutional Layer (with 4 filters)



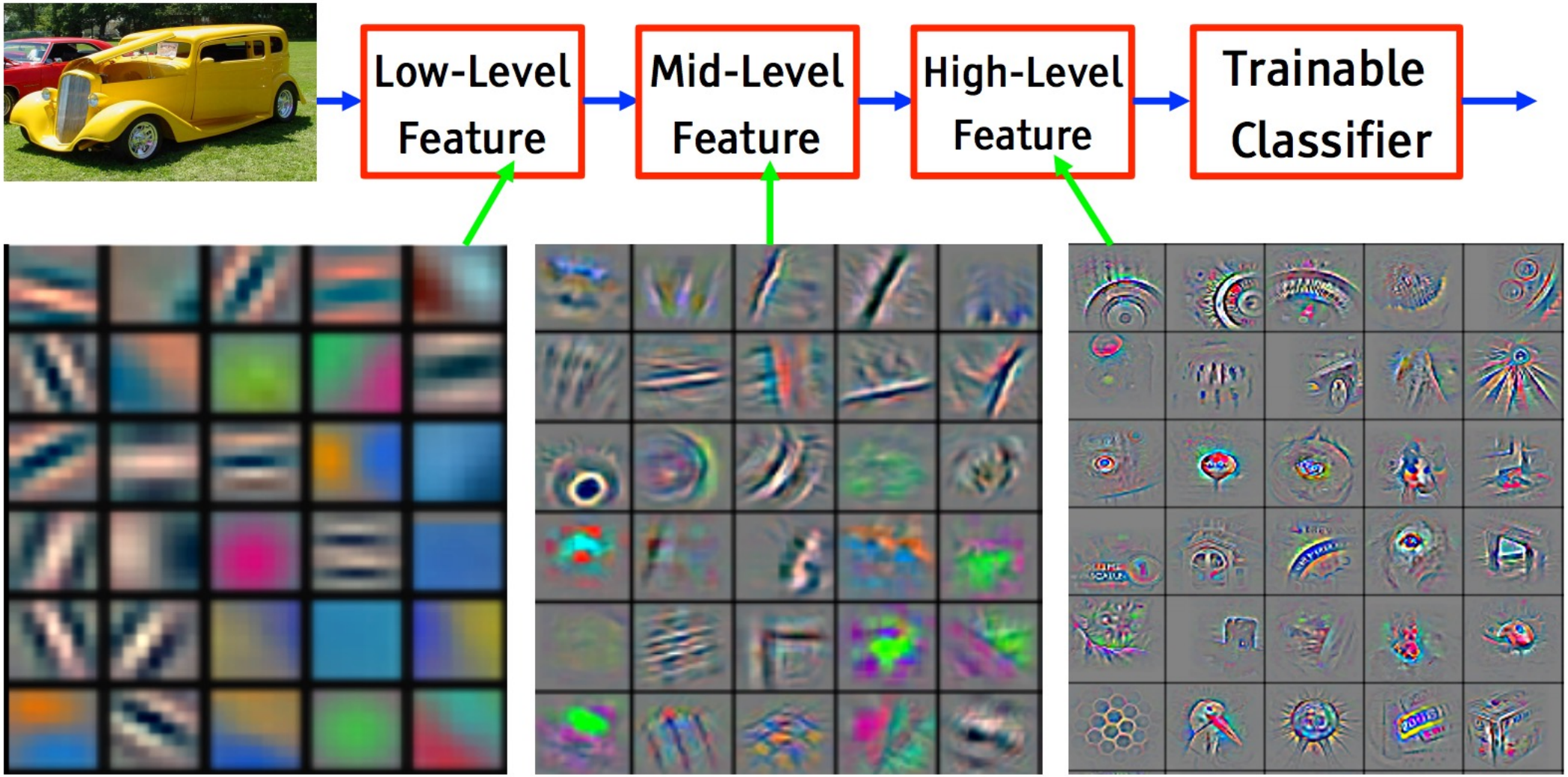
if zero padding,
and stride = 1

Convolutional Layer (with 4 filters)



Multiple Layers: Levels of Abstraction

Source:
NIPS'2015 Tutorial
Geoff Hinton, Yoshua Bengio & Yann LeCun



Alexnet

ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca

the paper that started the deep learning revolution!

Image Classification

Classify an image into 1000 possible classes:

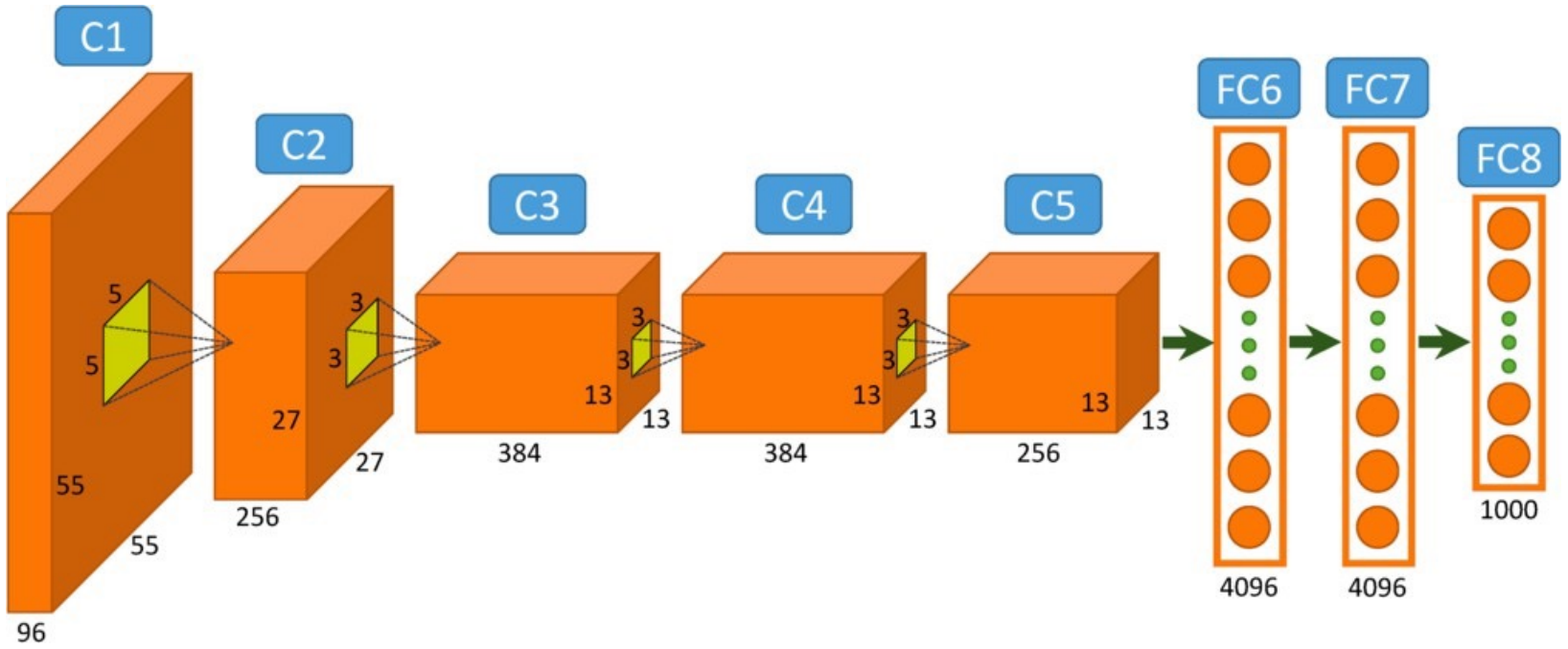
e.g. Abyssinian cat, Bulldog, French Terrier, Cormorant, Chickadee,

red fox, banjo, barbell, hourglass, knot, maze, viaduct, etc.

cat, tabby cat(0.71)
Egyptian cat (0.22)
red fox (0.11)
…..

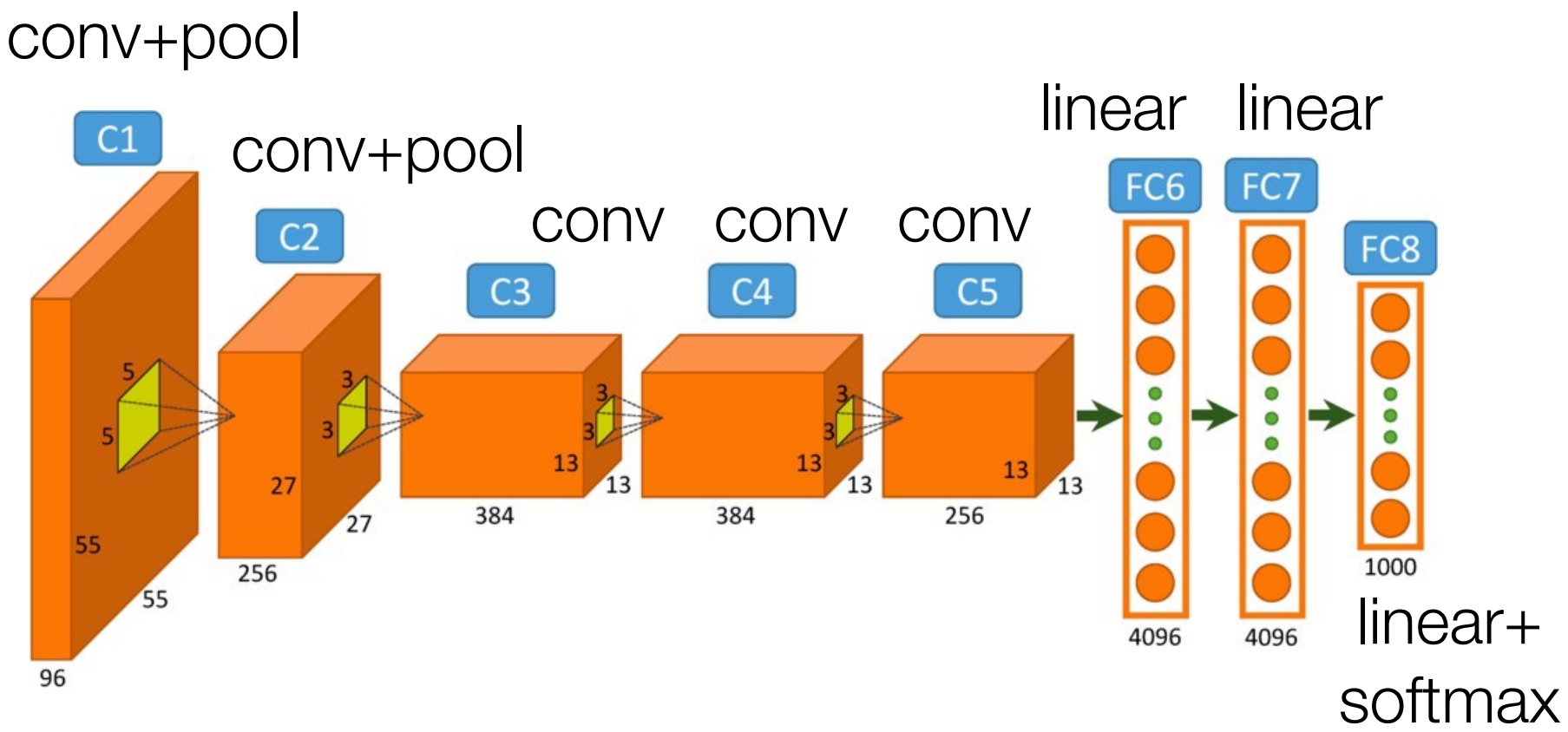
train on the ImageNet challenge dataset, ~1.2 million images

Alexnet



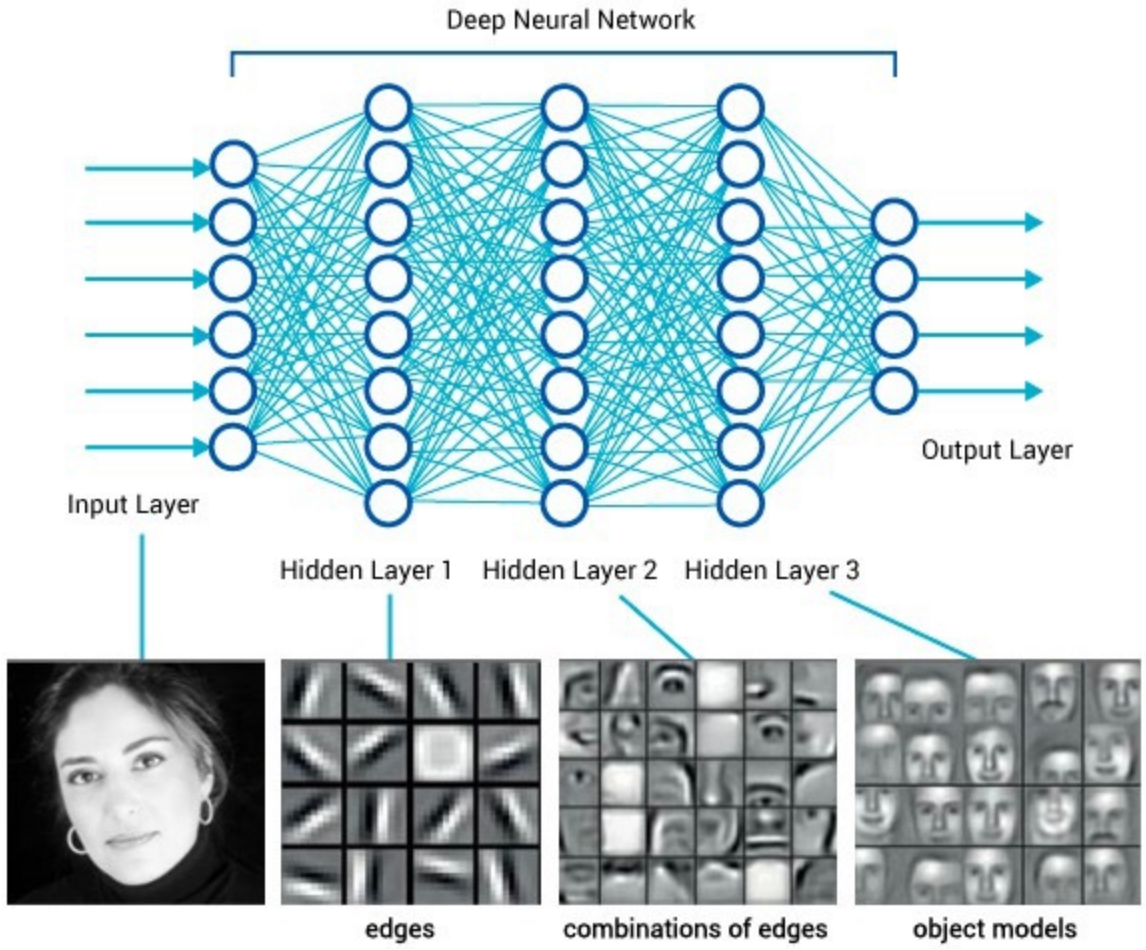
https://www.saagie.com/fr/blog/object- detection-part1

Alexnet



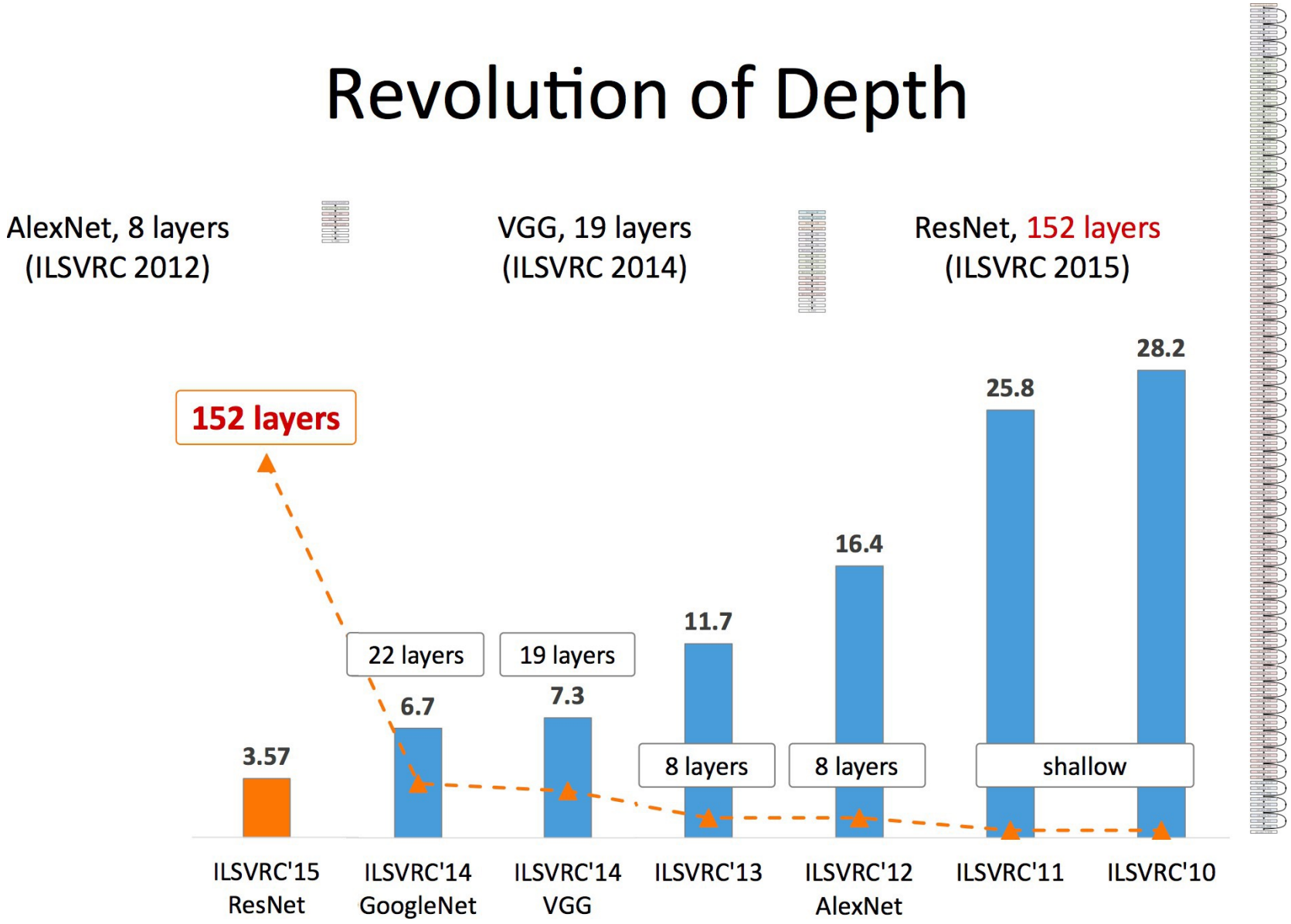
https://www.saagie.com/fr/blog/object- detection-part1

What is happening?



https://www.saagie.com/fr/blog/object- detection-part1

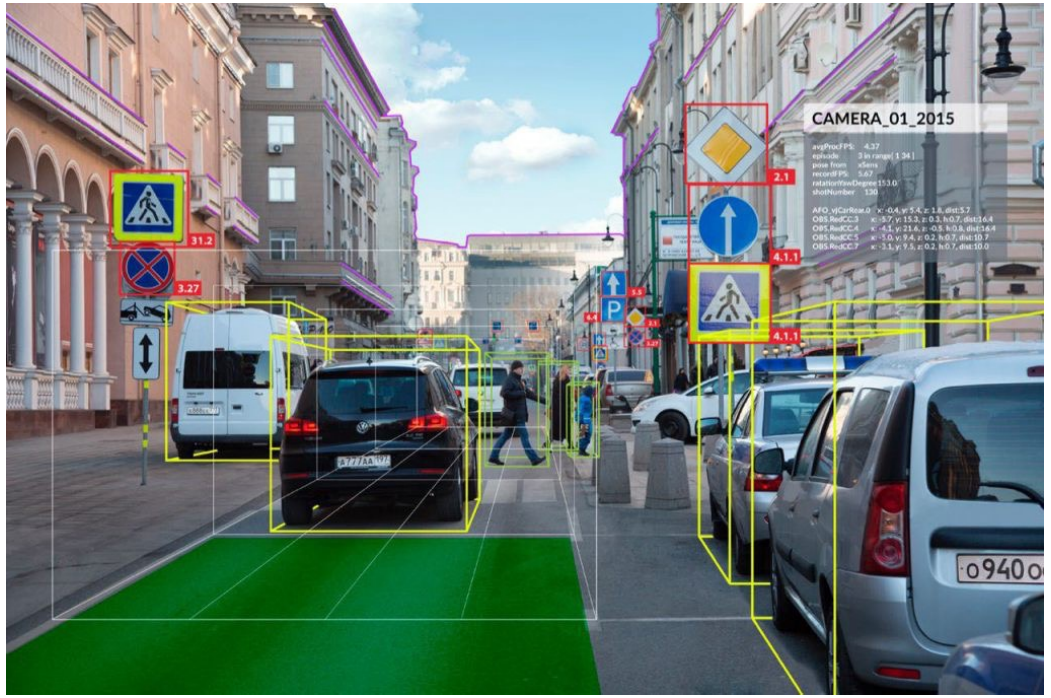
Revolution of Depth



Slide by Mohammad Rastegari

Computer Vision in Autonomous Driving

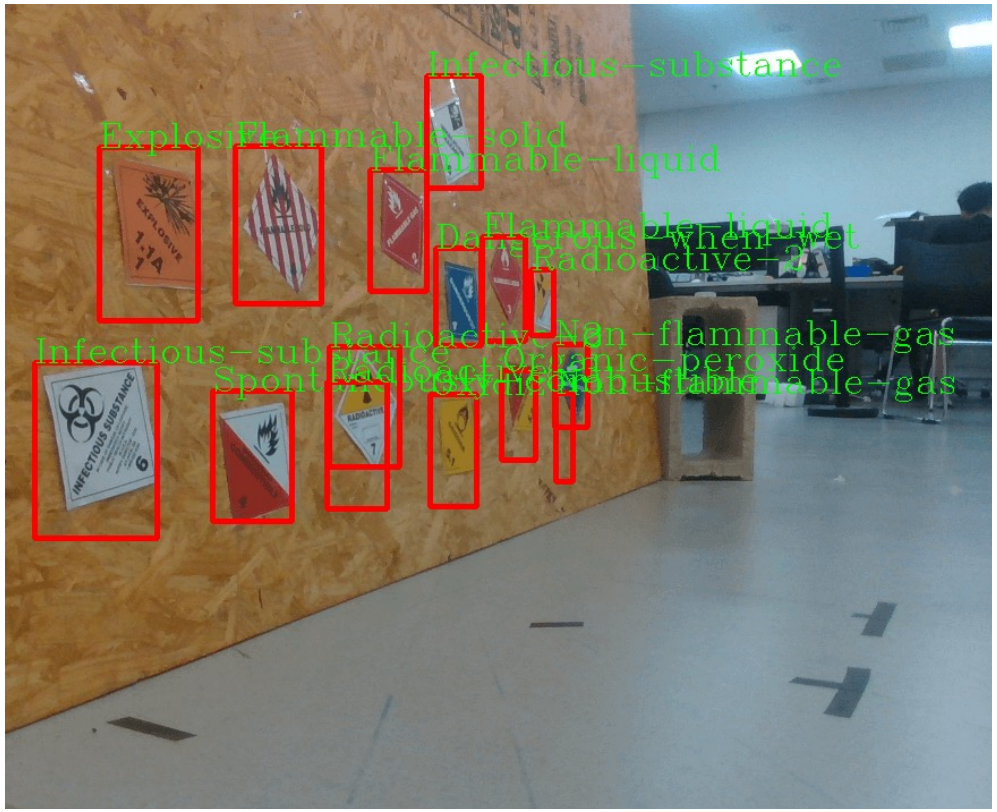
- DL for Computer Vision for detection of:
  - Cars (Truck; Bus; Police car!; …)
  - Pedestrian; Bicycle; …
  - Lane; curb
  - Markings on the road
  - Traffic Signs
  - Traffic Lights
- From Cameras AND Lidars



https://becominghuman.ai/computer-vision-applications-in-self-driving-cars-610561e14118

Computer Vision for Robotics

- DL for Computer Vision for detection of:
  - Doors; Door handle
  - Signs; Read door signs
  - Objects
  - Other Robots



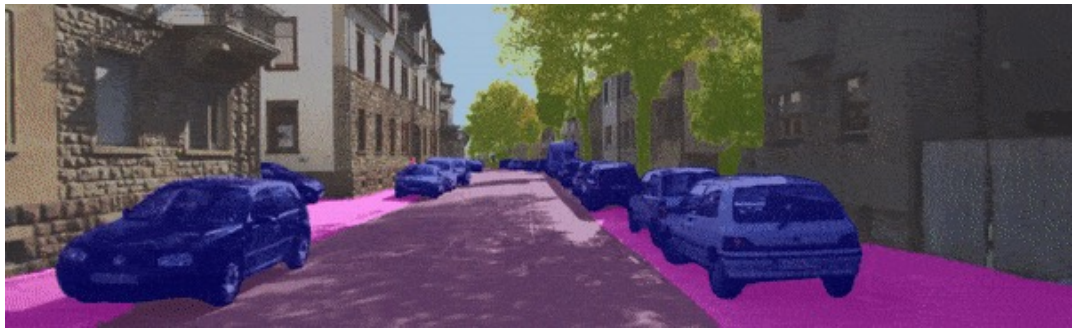
Amazon Picking Challenge

- Robot stow and take (pick) products from shelves
- Challenges in: Computer vision (find object and its pose [position & orientation]) and Manipulation (planning, grasping)
- Since 2015 – now co-located with RoboCup
- Winner 2016: Team Delft
  - Suction gripper (easy!)
  - 100 items per hour (human: 400)
  - Failure rate: 16.7 %
  - Deep learning to find objects:
    - Stereo camera for 3D
  - ROS Industrial

https://www.theverge.com/2016/7/5/12095788/amazon-picking-robot-challenge-2016

Deep Learning for Scene Understanding

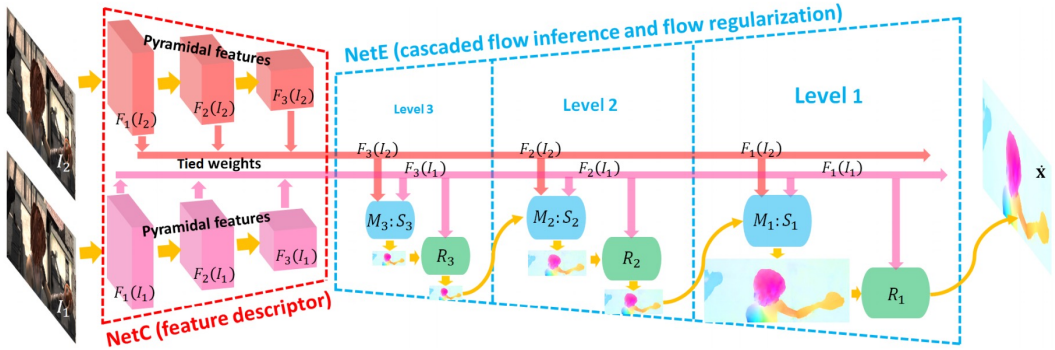
- Semantics Segmentation
  1. single RGB image as input
  2. semantic labeling at pixel-level
  3. divide scene to different regions and objects
     - semantic SLAM
       - help to remove dynamic object (Chen et al. SuMa++)
     - provide cues for loop closing (Chen et al. OverlapNet)



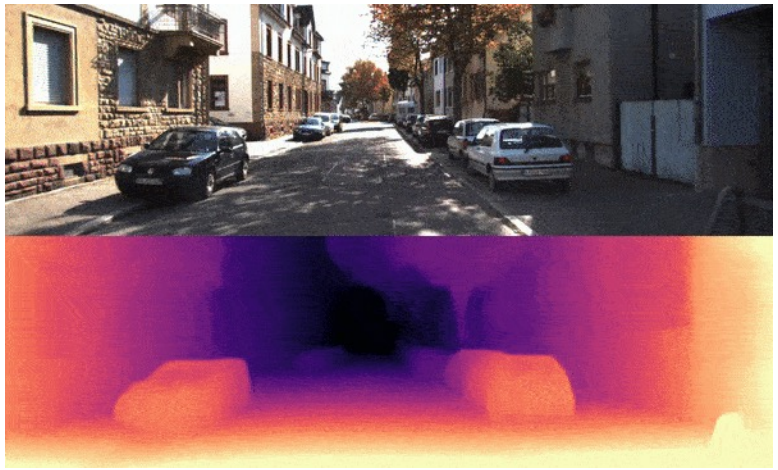
Poudel, Rudra PK, Stephan Liwicki, and Roberto Cipolla. "Fast-scnn: Fast semantic segmentation network." arXiv preprint arXiv:1902.04502 (2019).
Guo, Jian, et al. "GluonCV and GluonNLP: Deep Learning in Computer Vision and Natural Language Processing." Journal of Machine Learning Research 21.23 (2020): 1-7.

Deep Learning for Feature Extraction

- Optical Flow Estimation
  1. two frames as input
  2. output the optical flow
  3. generating the 2D-2D correspondences



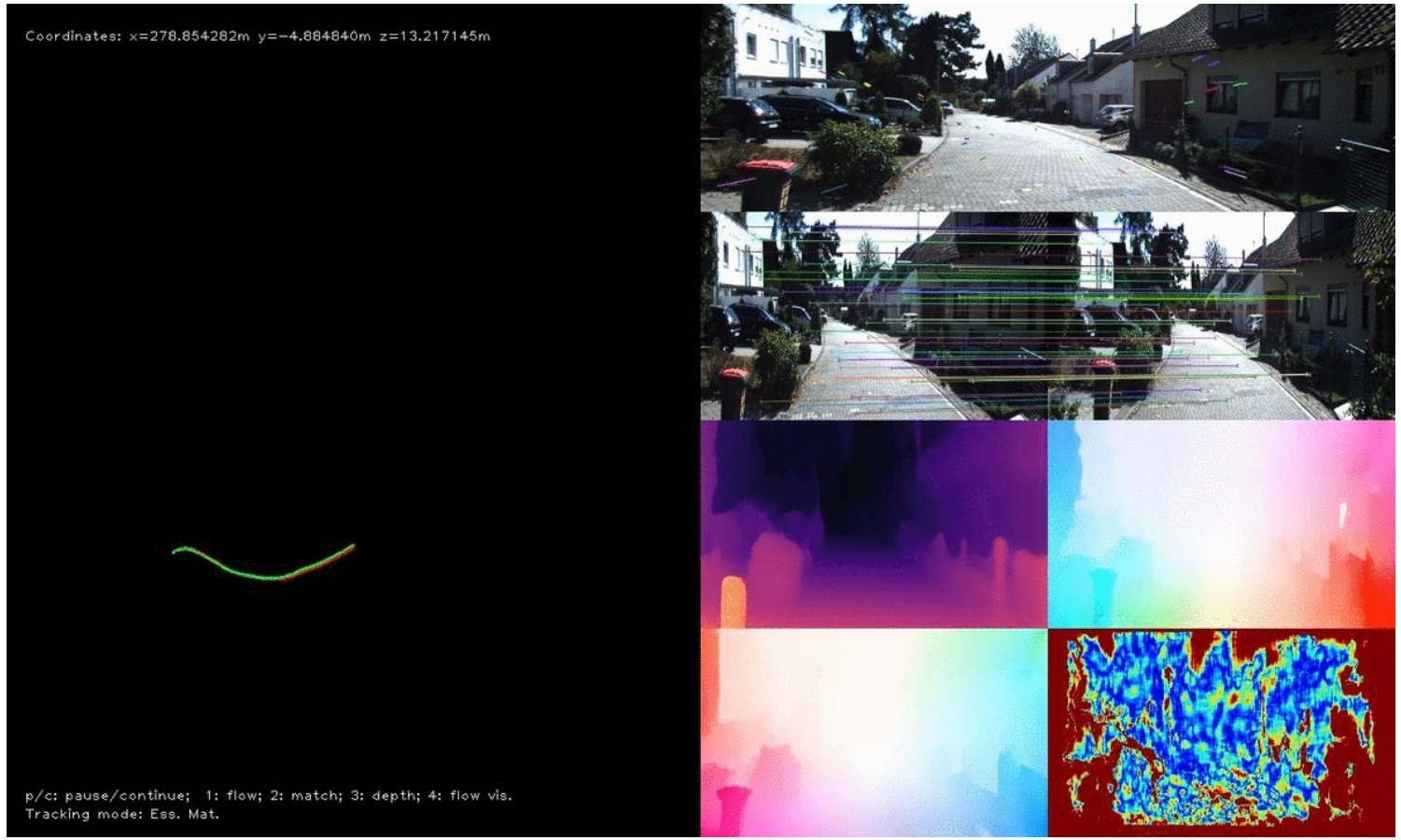
- Monocular Depth Estimation
  1. single RGB image as input
  2. output the depth map
  3. provides 3D information, recover scaling factor



Hui, Tak-Wai, Xiaoou Tang, and Chen Change Loy. "Liteflownet: A lightweight convolutional neural network for optical flow estimation." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.

Godard, Clément, et al. "Digging into self-supervised monocular depth estimation." Proceedings of the IEEE international conference on computer vision. 2019.

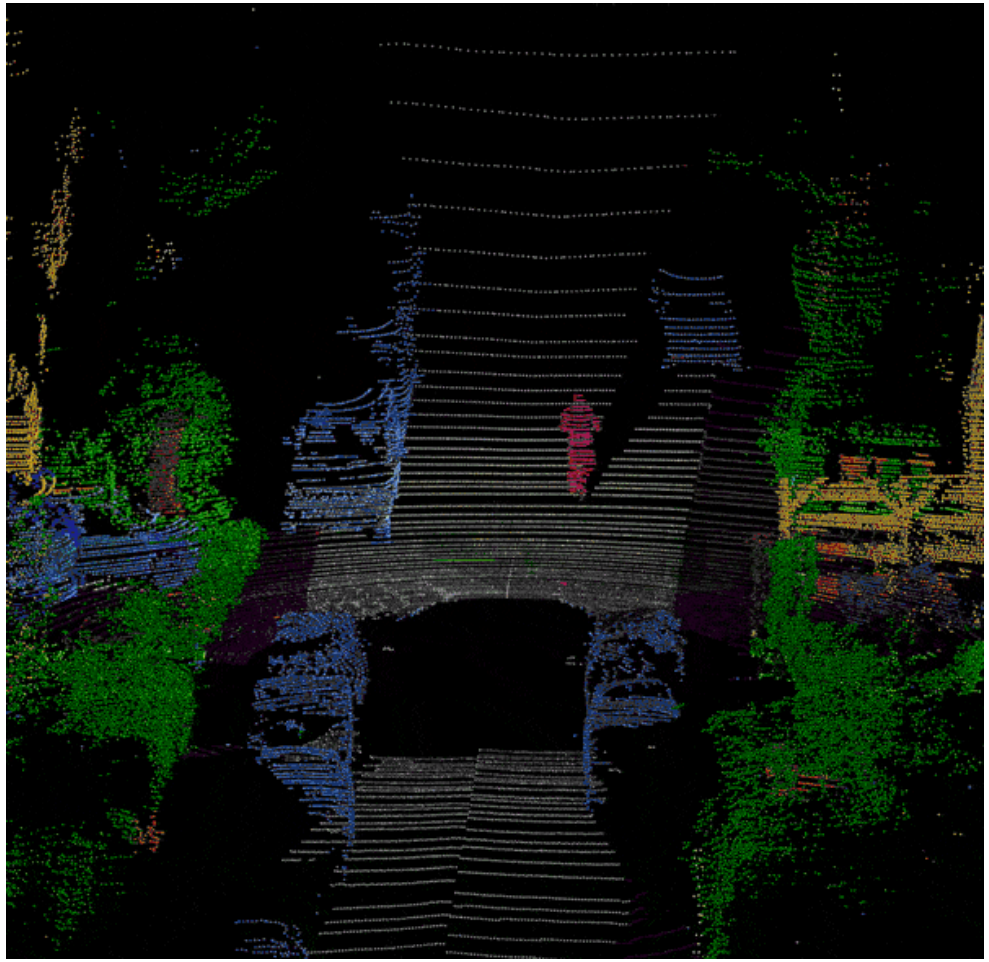
Deep Learning in Visual Odometry

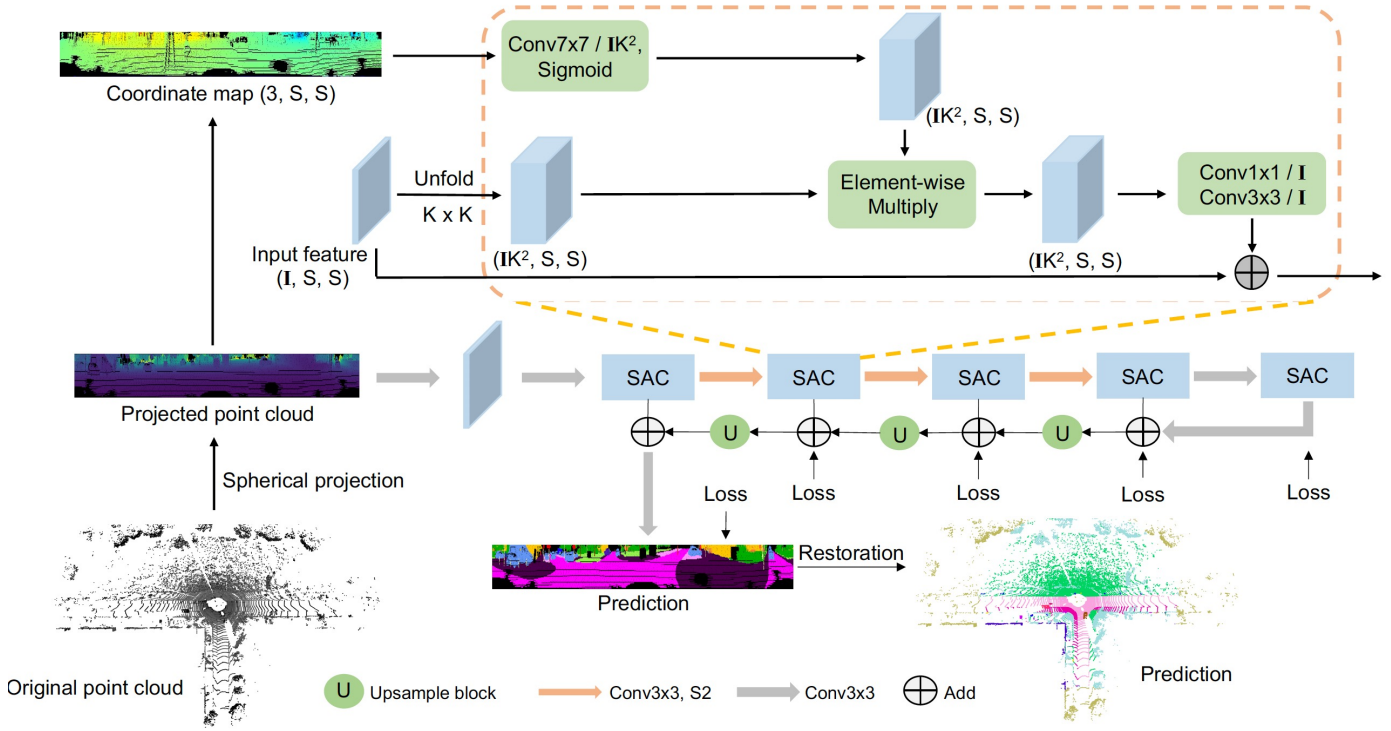


Zhan, Huangying, et al. "Visual odometry revisited: What should be learnt?." 2020 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2020.

Point Cloud Segmentation

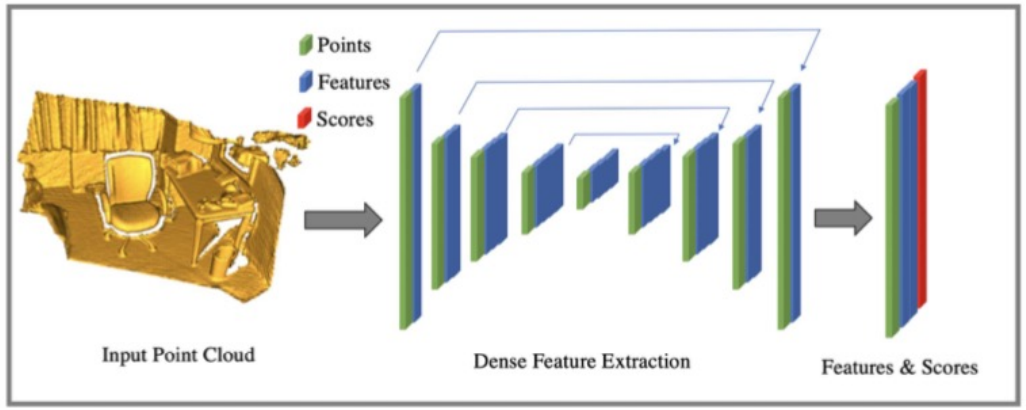
- ”SqueezeSegV3: Spatially-Adaptive Convolution for Efficient Point-Cloud Segmentation.”Chenfeng Xu, Bichen Wu, Zining Wang, Wei Zhan, Peter Vajda, Kurt Keutzer, and Masayoshi Tomizuka.

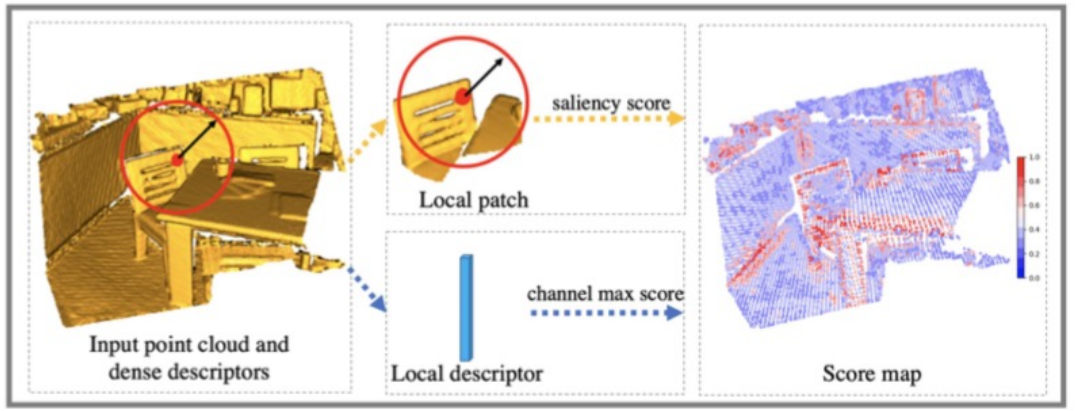


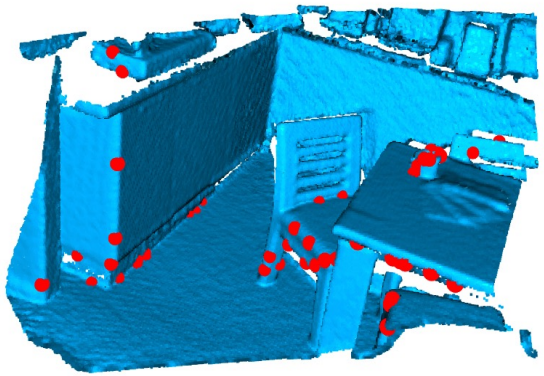


Feature Descriptors => Registration

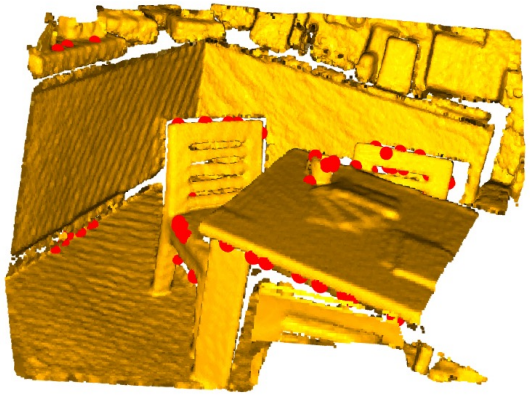
- "D3Feat: Joint Learning of Dense Detection and Description of 3D Local Features", by Xuyang Bai, Zixin Luo, Lei Zhou, Hongbo Fu, Long Quan and Chiew-Lan Tai.



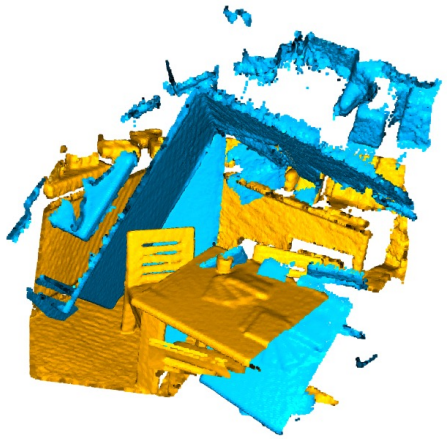




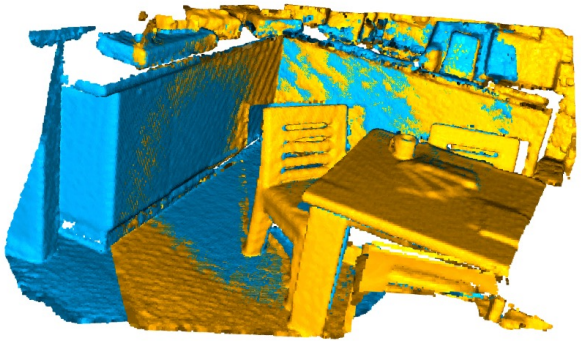
Source point cloud



Target point cloud



Initial State



Registered using D3Feat + RANSAC

Vision Language Models

Transformer encoders for vision

Mohit Iyyer

College of Information and Computer Sciences University of Massachusetts Amherst

https://people.cs.umass.edu/~miyyer/cs685/slides/vision_nlp.pdf

Image captioning

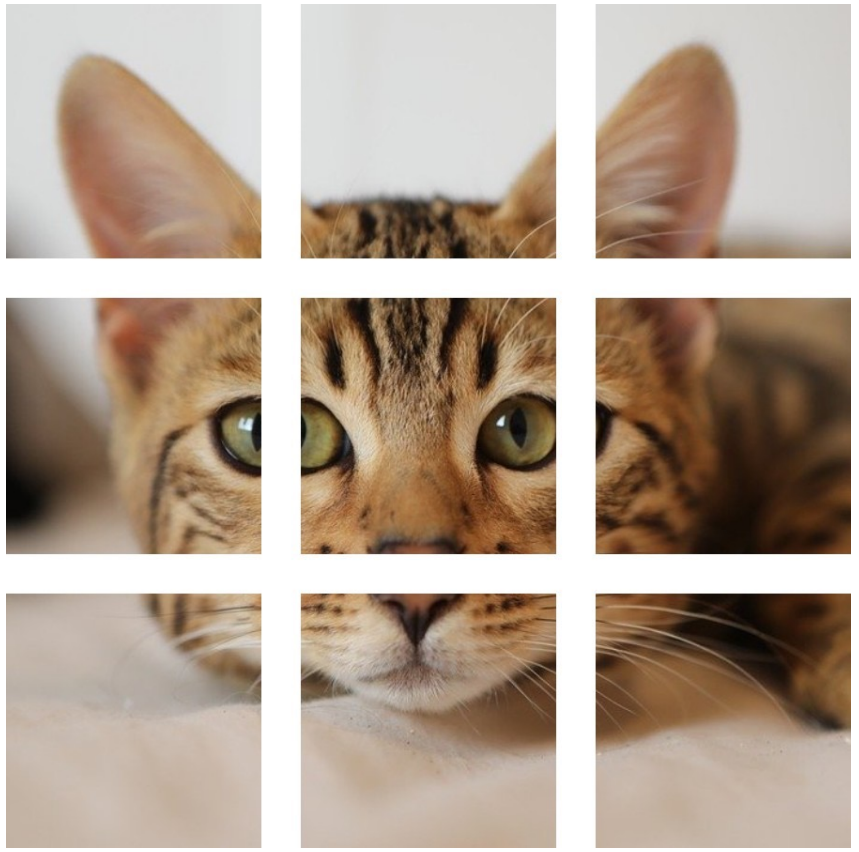
a red truck is
parked on a street
lined with trees

Visual question answering

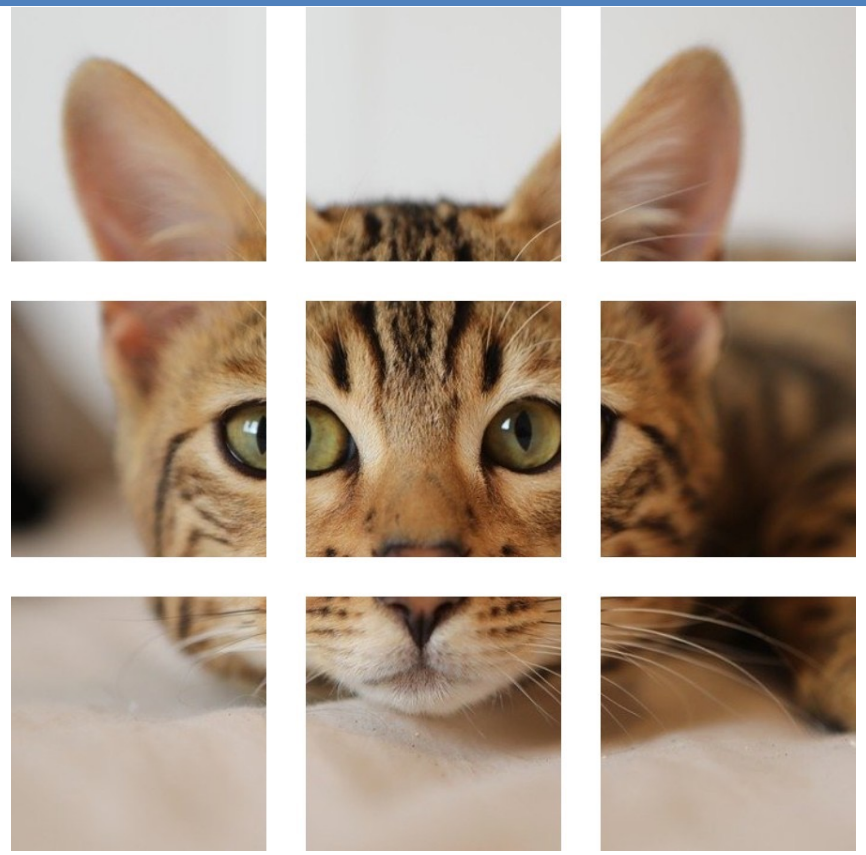
- Is this truck considered “vintage”?
- Does the road look new?
- What kind of tree is behind the truck?

Self-attention on pixels

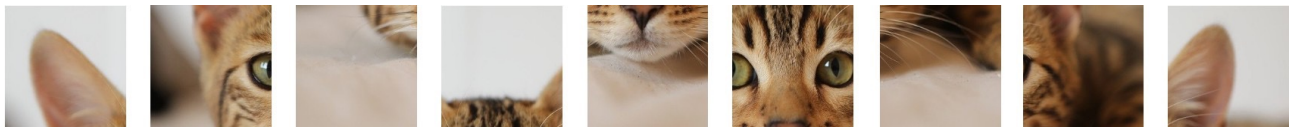
An Image is Worth 16x16 words, Dosovitskiy et al., ICLR 2021



Slides from Justin Johnson



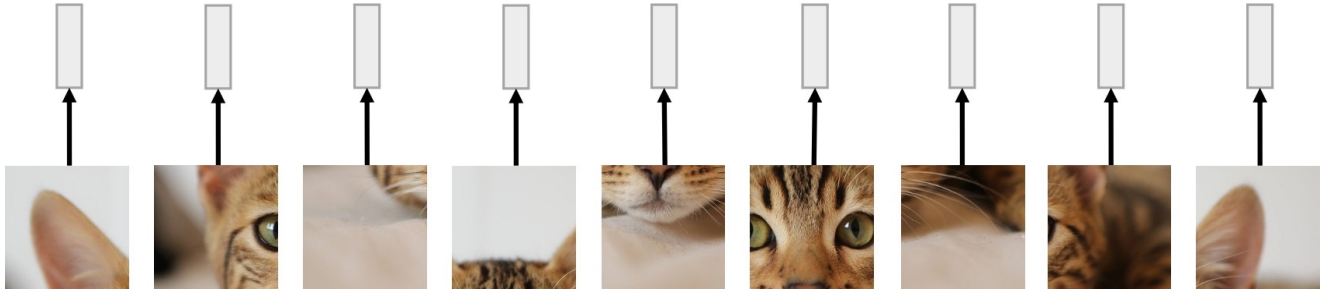
N input patches, each of shape 3x16x16



Slides from Justin Johnson

Linear projection to D-dimensional vector

N input patches, each of shape 3x16x16

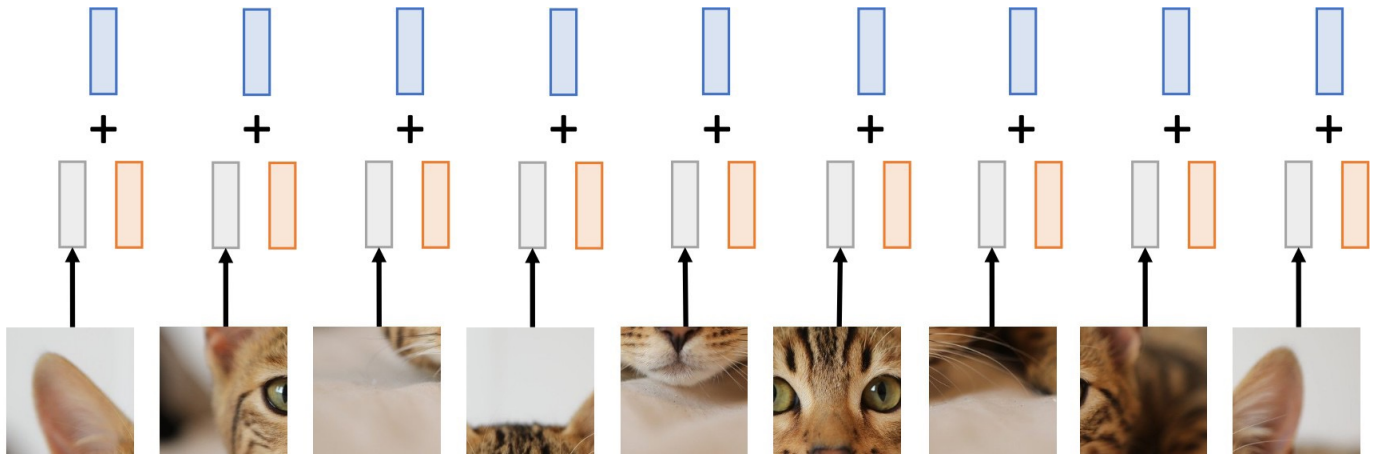


Slides from Justin Johnson

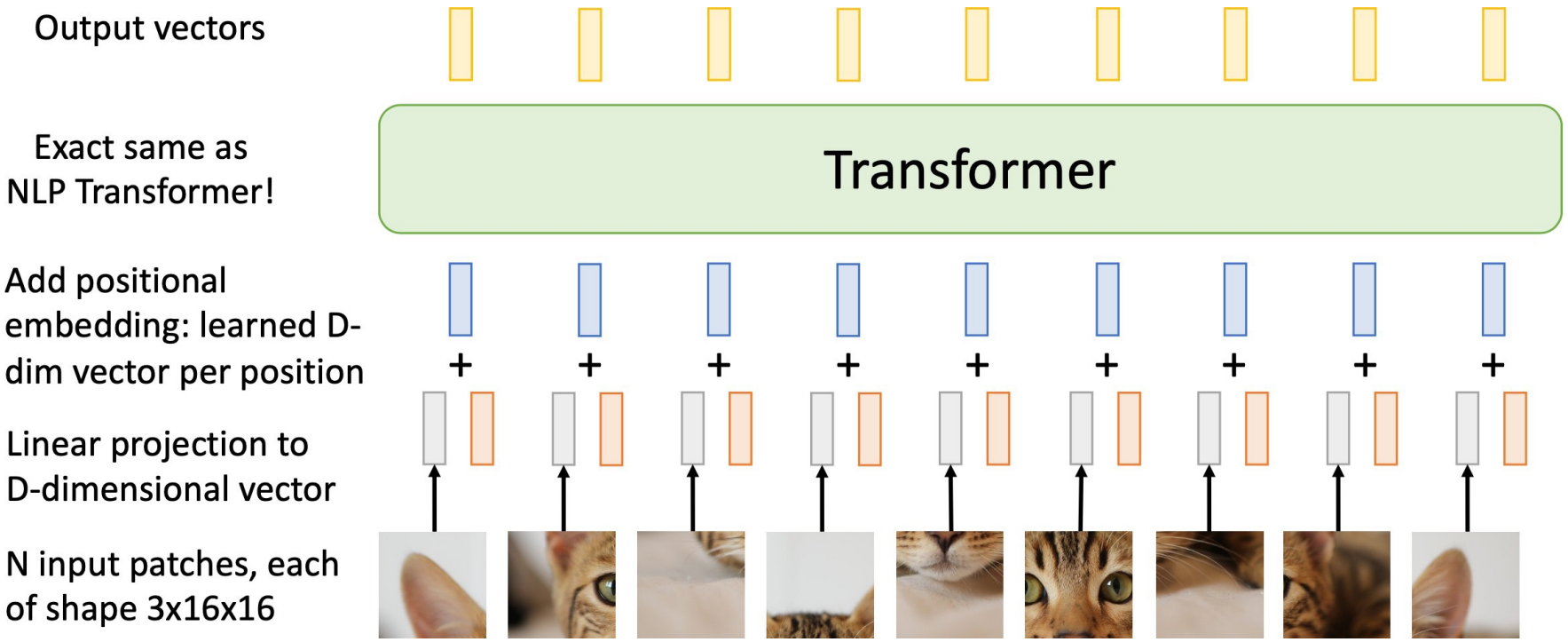
Add positional embedding: learned D-dim vector per position

Linear projection to D-dimensional vector

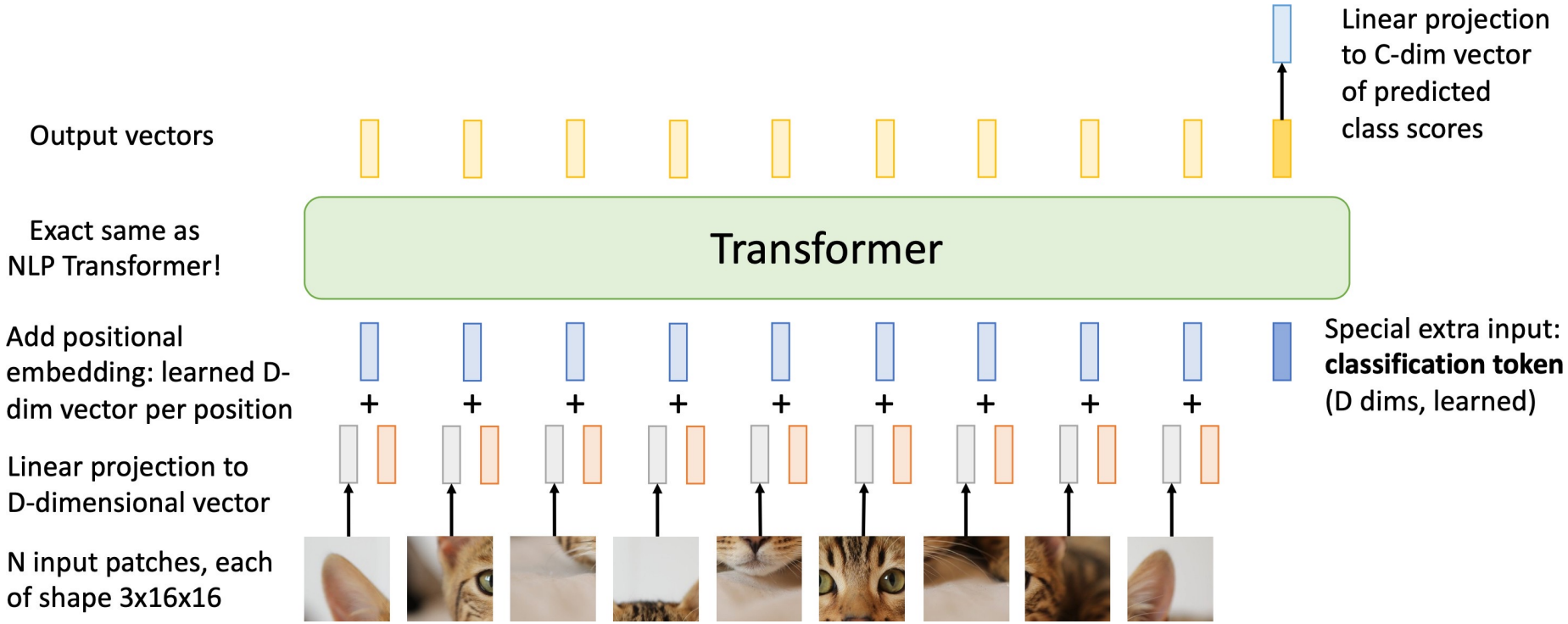
N input patches, each of shape 3x16x16



Slides from Justin Johnson



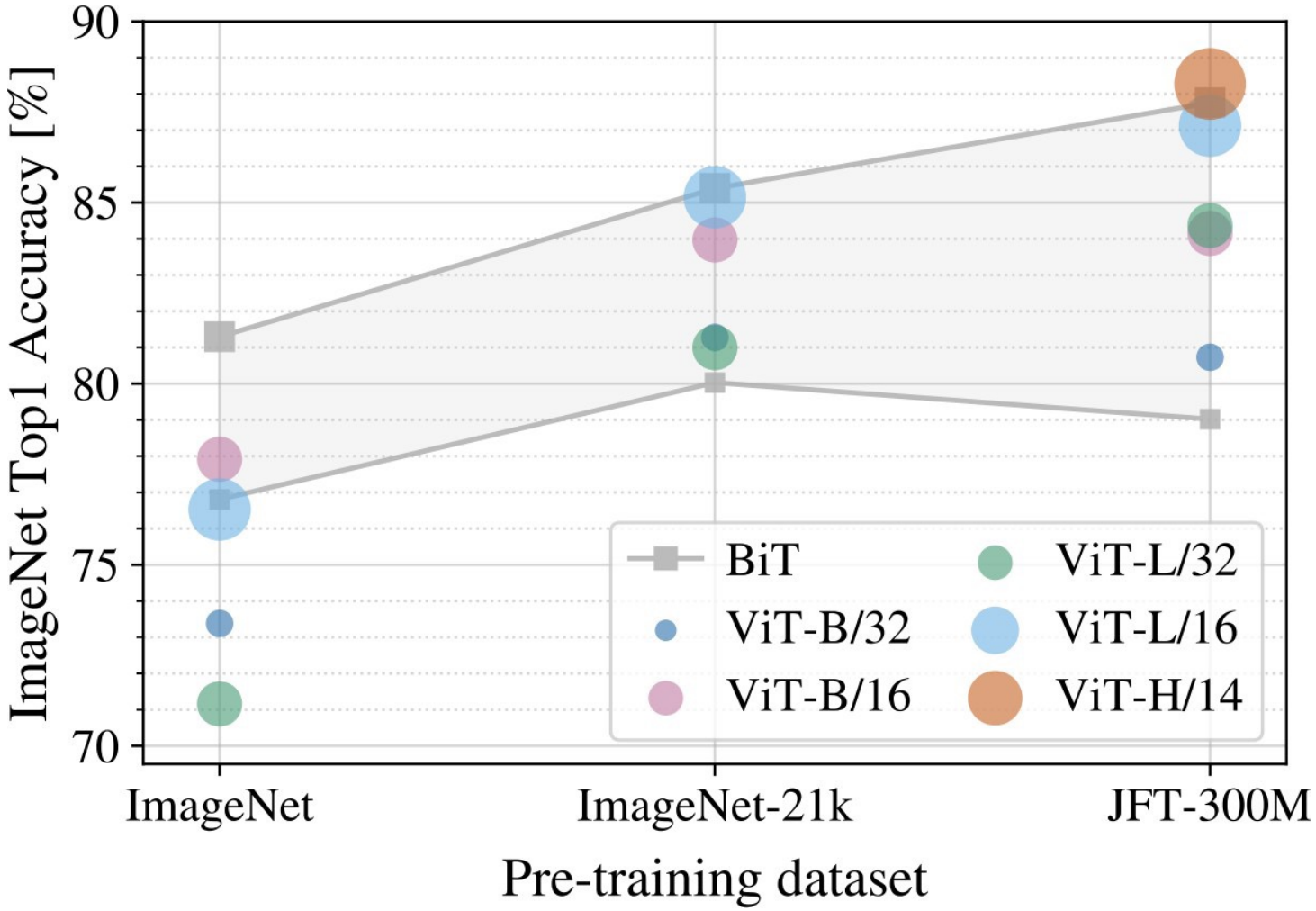
Slides from Justin Johnson



Slides from Justin Johnson

16x16 patches = 16*16*3 = 768d embedding

Vision Transformers (ViT)outperform ResNets with larger datasets

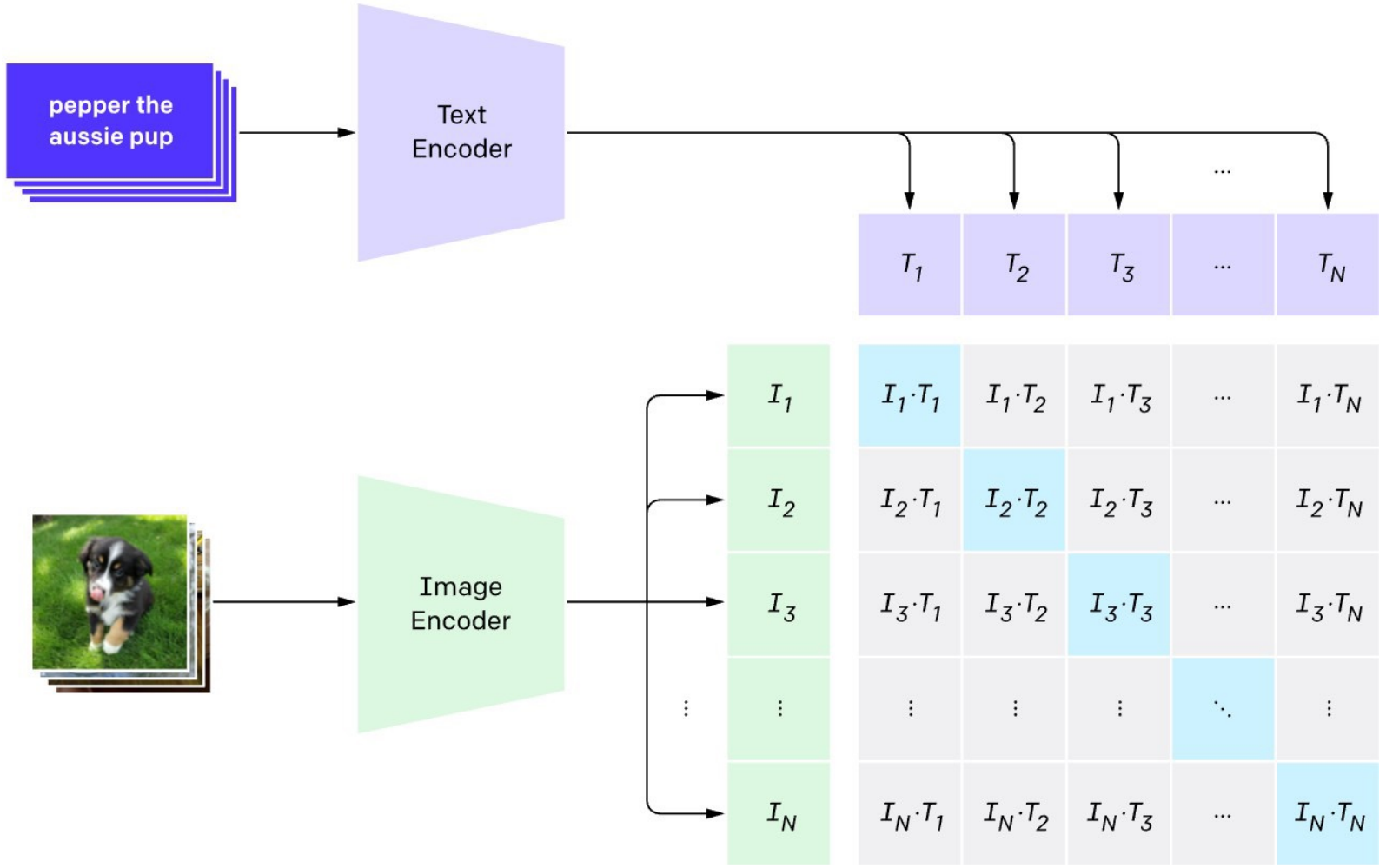


OpenAI's CLIP: Contrastive language-image pretraining

- OpenAI collect 400 million (image, text) pairs from the web
- Then, they train an image encoder and a text encoder with a simple contrastive loss: given a collection of images and text, predict which (image, text) pairs actually occurred in the dataset

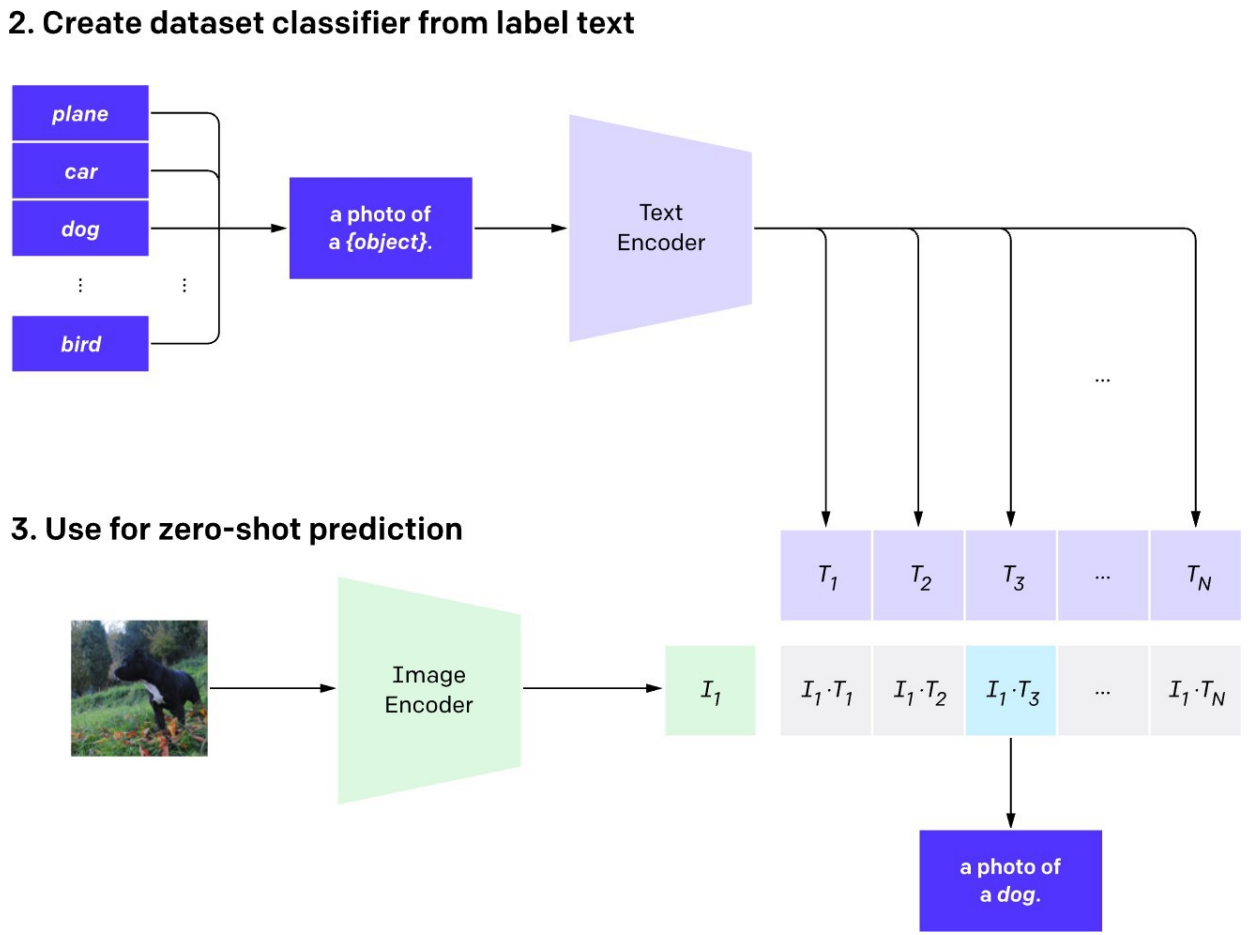
Radford et al., 2021 ("CLIP")

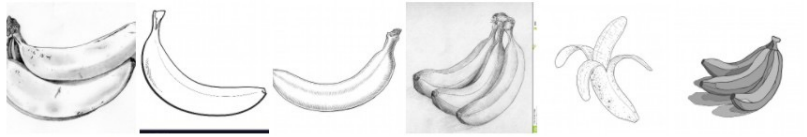
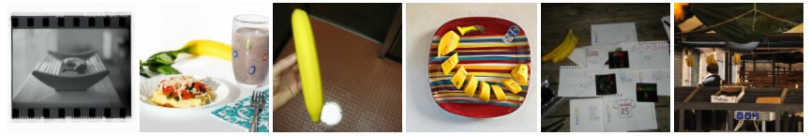
1. Contrastive pre-training



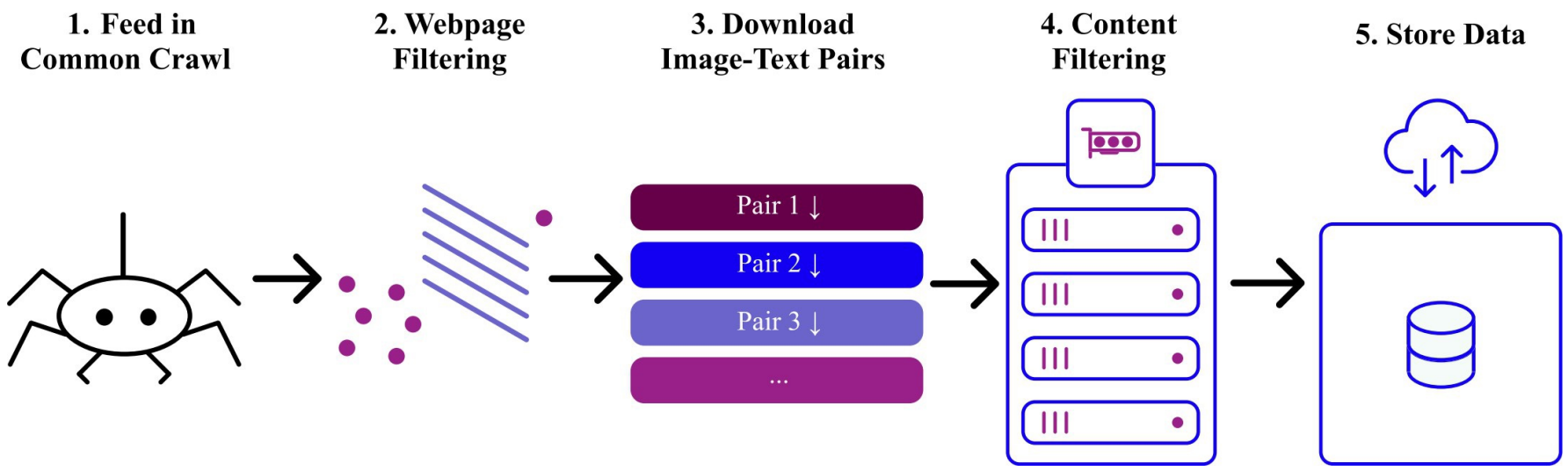
https://openai.com/blog/clip/

Similar to GPT-3, you can use CLIP for zero-shot learning



DATASET	IMAGENET RESNET101	CLIP VIT-L
 ImageNet	76.2%	76.2%
 ImageNet V2	64.3%	70.1%
 ImageNet Rendition	37.7%	88.9%
 ObjectNet	32.6%	72.3%
 ImageNet Sketch	25.2%	60.2%
 ImageNet Adversarial	2.7%	77.1%

LAION-5B: a dataset of 5 billion image/text pairs!

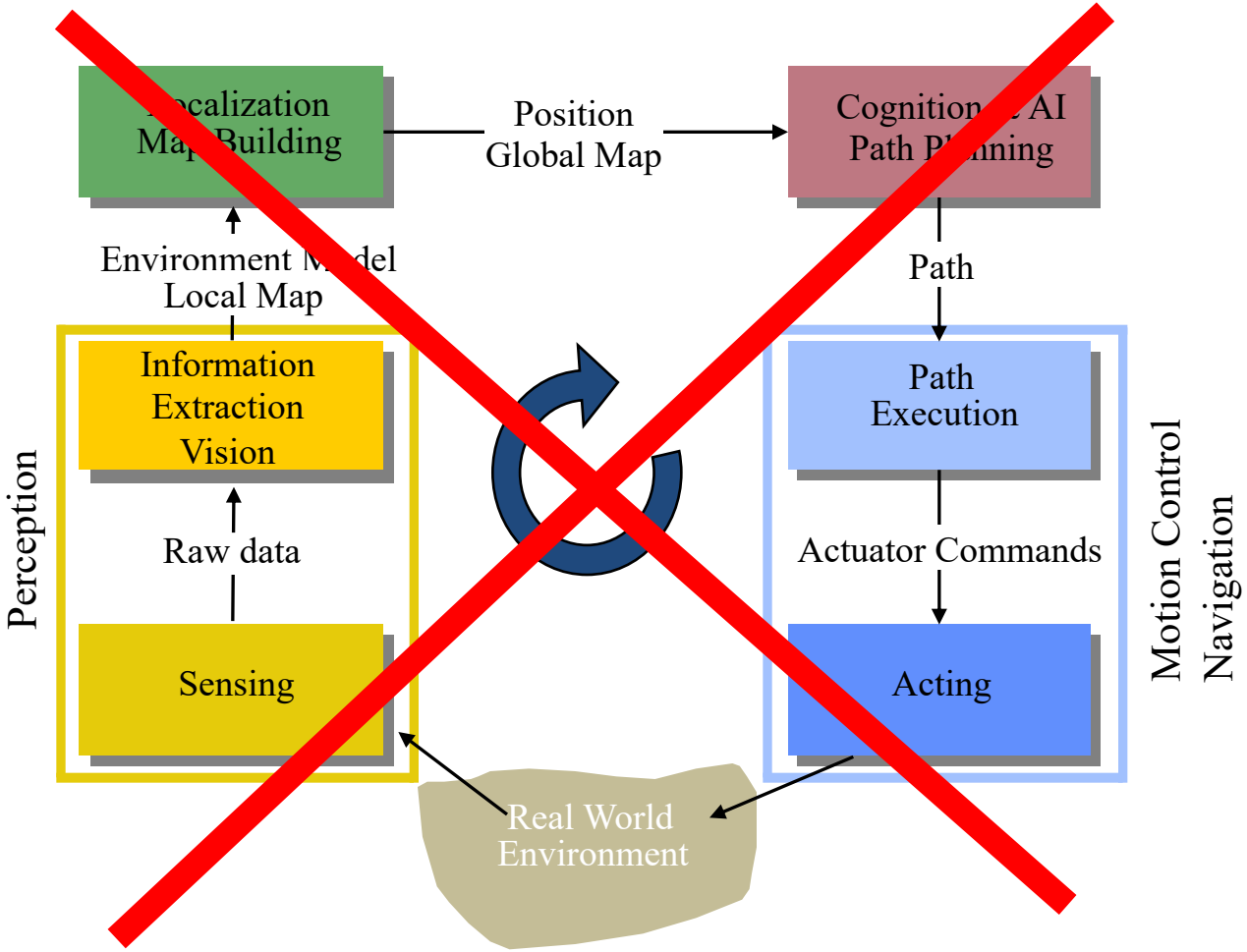


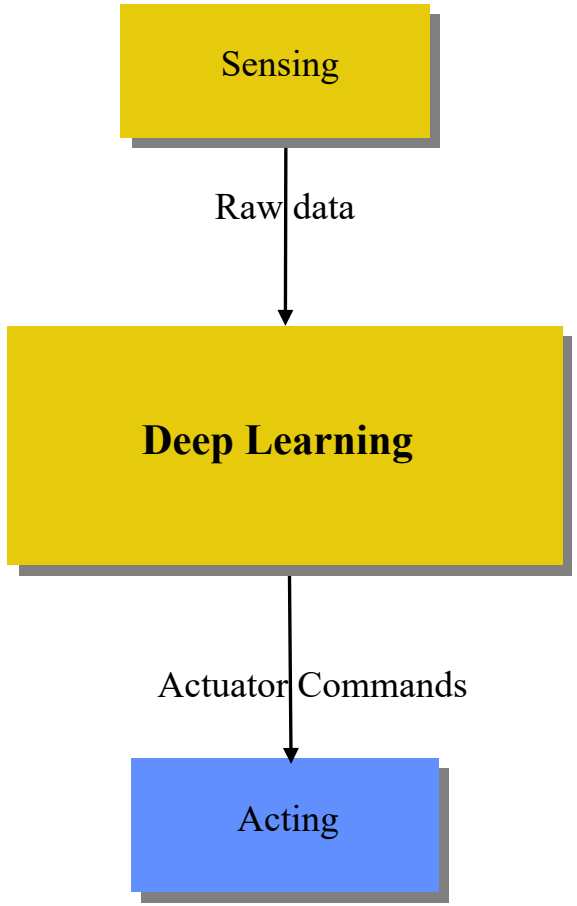
Schuhmann et al., 2022

Major copyright issues…

Stable Diffusion and other image-generating AI products could not exist without the work of painters, illustrators, photographers, sculptors, and other artists. Stable Diffusion was trained on the LAION-5B dataset. LAION-5B contains 5.85 billion image-text pairs. Most of the images contained in the dataset are copyrighted, and LAION claims no ownership in them. As it notes, “The images are under their copyright.”

End-to-End Deep Learning



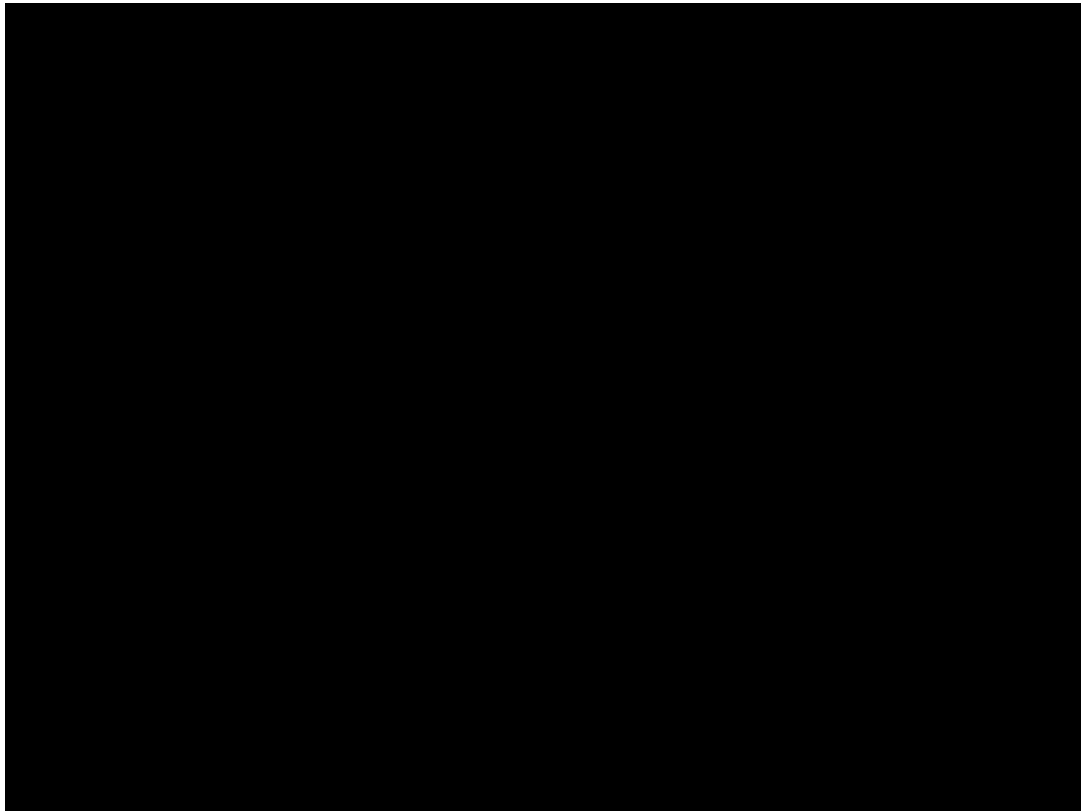


End-to-End Deep Reinforcement Learning

- From sensors to actuation: one layered or recurrent neural network! =>
  - NOT classical general control scheme (Perception, SLAM, Cognition & Planning, Navigation)

- Needs reward signal: sparse, noisy, delayed!
- Take time into account: input frames are related!

- Gained interest 2013 again with:
  - Deep Mind (google) playing ATARI 2600 games
  - Video: Breakout
  - Learned 7 games
  - Surpasses human expert in 3



https://www.cs.toronto.edu/~vmnih/docs/dqn.pdf

BRETT: Berkeley Robot for Learning Tedious Tasks: Deep Reinforcement Learning

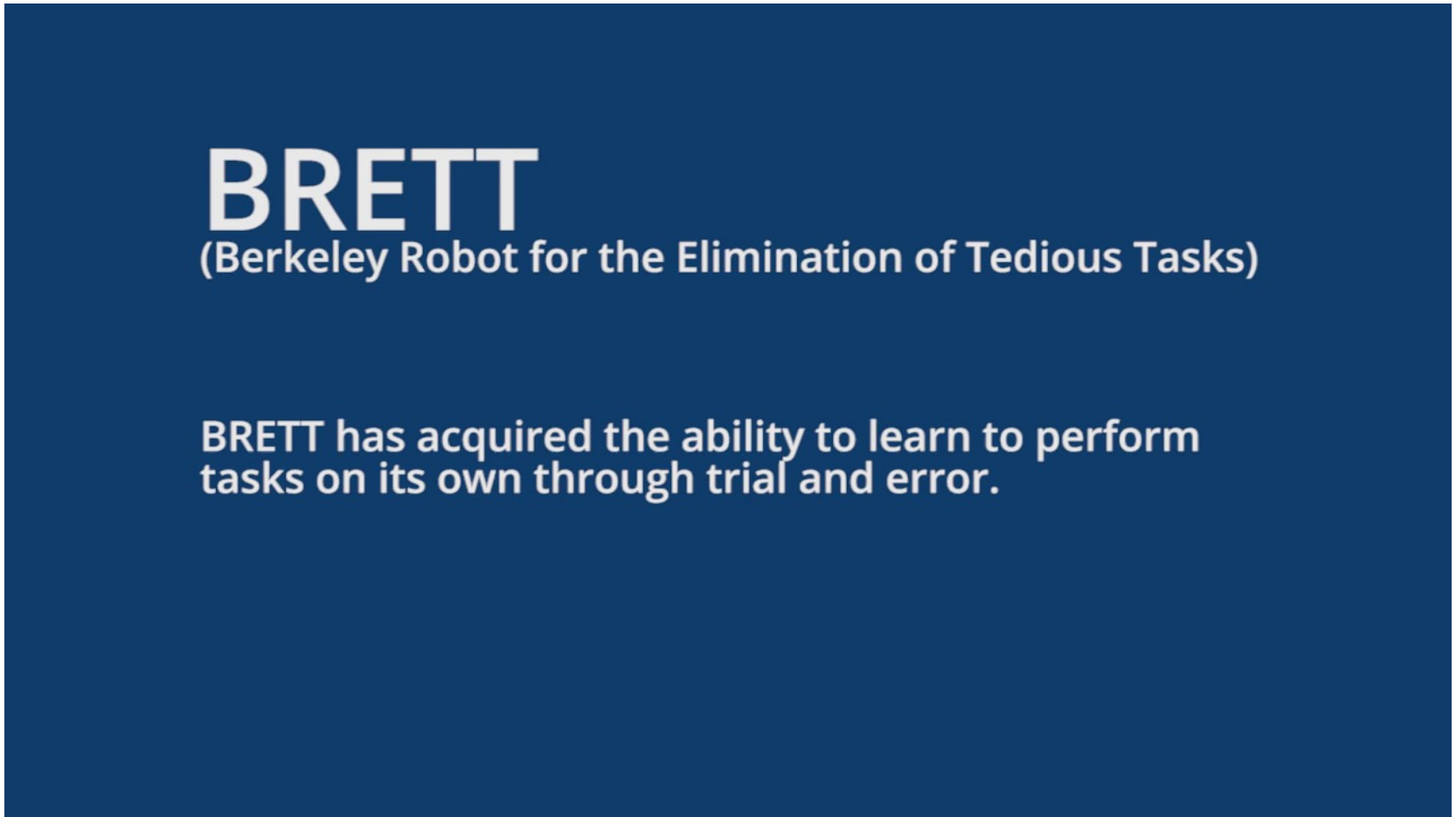
- ”There are no labeled directions, no examples of how to solve the problem in advance. There are no examples of the correct solution like one would have in speech and vision recognition programs”

- Learn simple tasks in 10 minutes;
  learn vision and control together in 3 hours

- Pieter Abbeel of UC Berkeley;

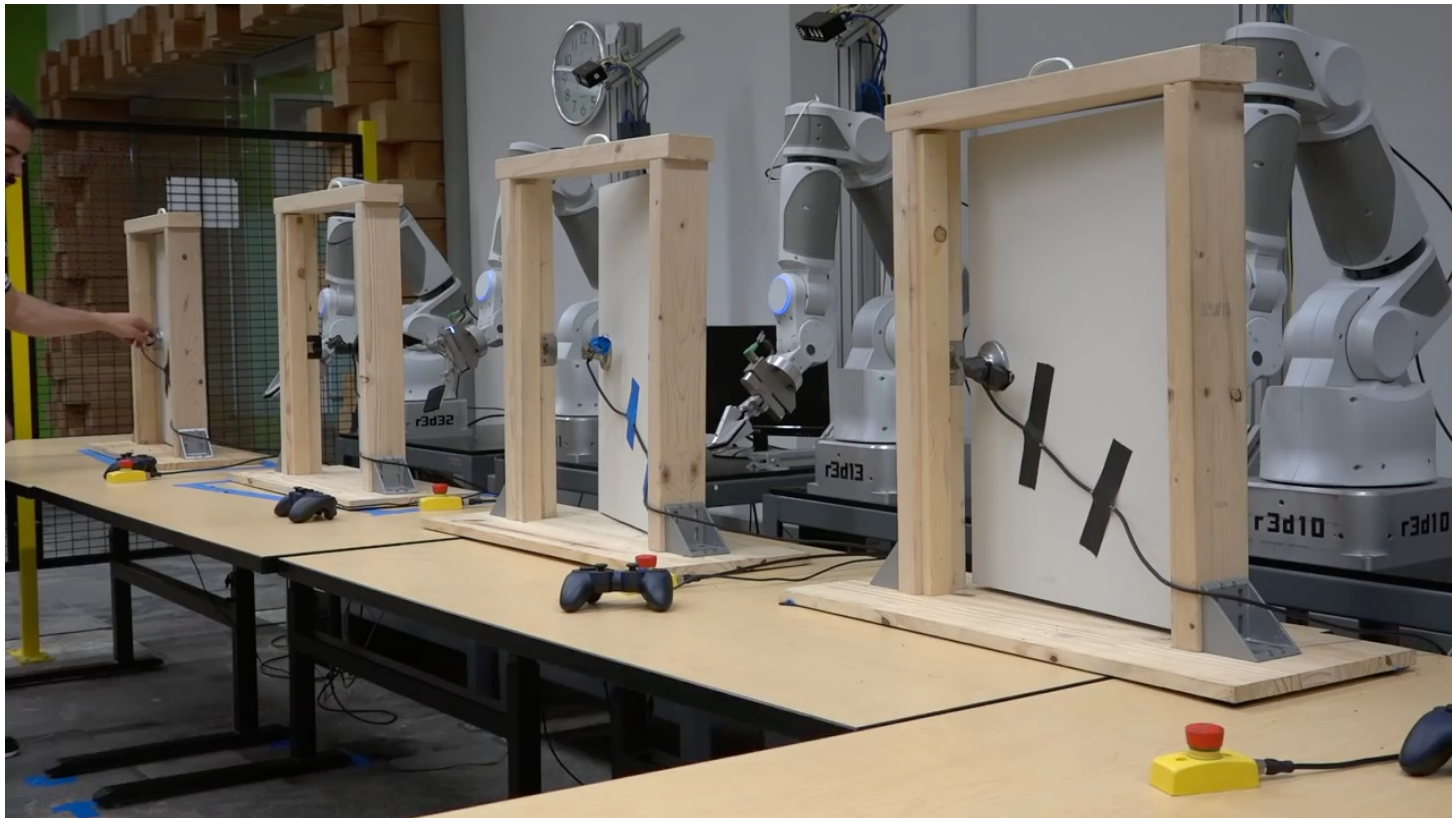
- 2015

http://news.berkeley.edu/2015/05/21/deep-learning-robot-masters-skills-via-trial-and-error/



Google Door Opening Project

- Learn to open doors using Reinforcement learning
  - Learning reward: opening the door
  - Much harder than purely digital learning: very slow iterations!
  - Simulation only helps a bit:
    real world much more complex



- Google and
  UC Berkeley Sergey Levine

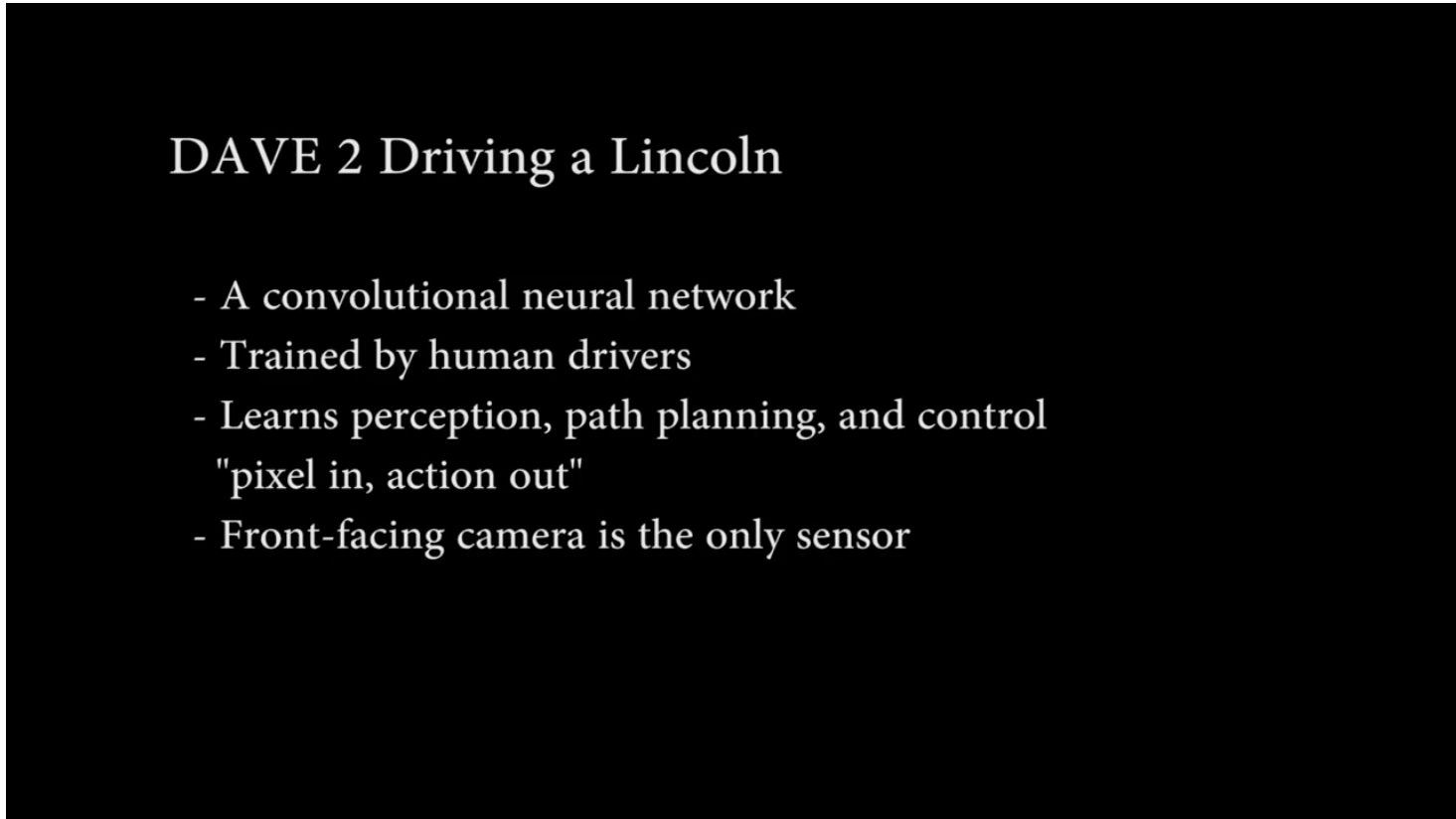
- Google very secretive …

https://www.wired.com/2017/01/googles-go-playing-machine-opens-door-robots-learn/

Nvidia end-to-end deep learning self driving car

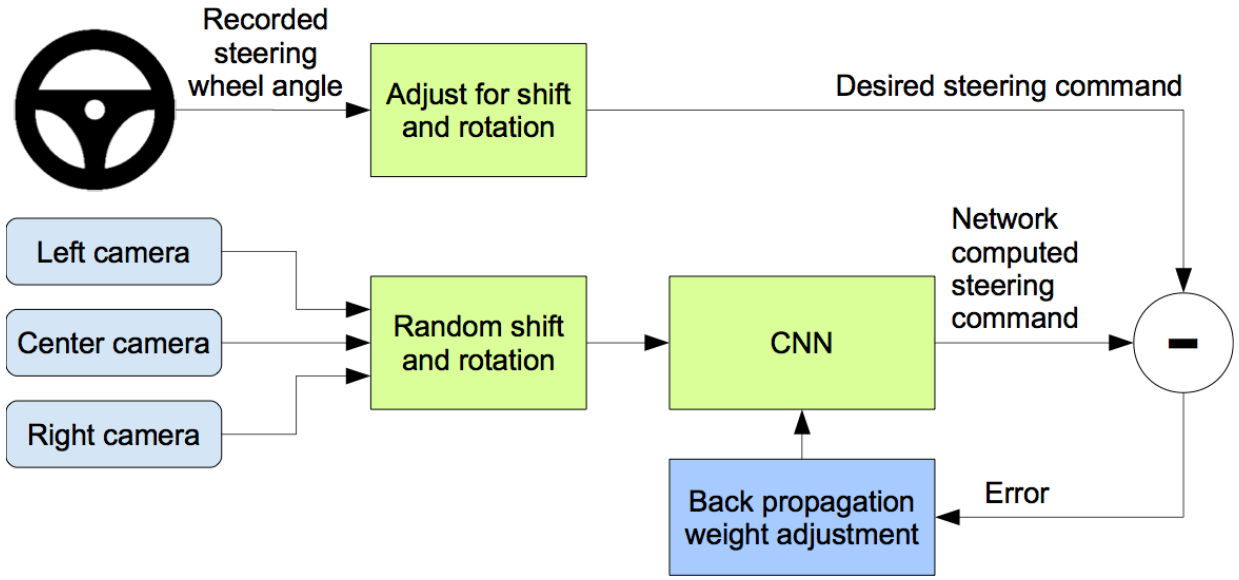
- Raw pixel of single camera => steering command of car
  - 30 FPS; single-image control (no history)
  - Nvidia Drive PX car computer (ARM cores and Nvidia GPU)
  - Human steering angle for training
- End-to-end: NO explicit:
  - Lane detection
  - Road detection
  - Obstacle detection

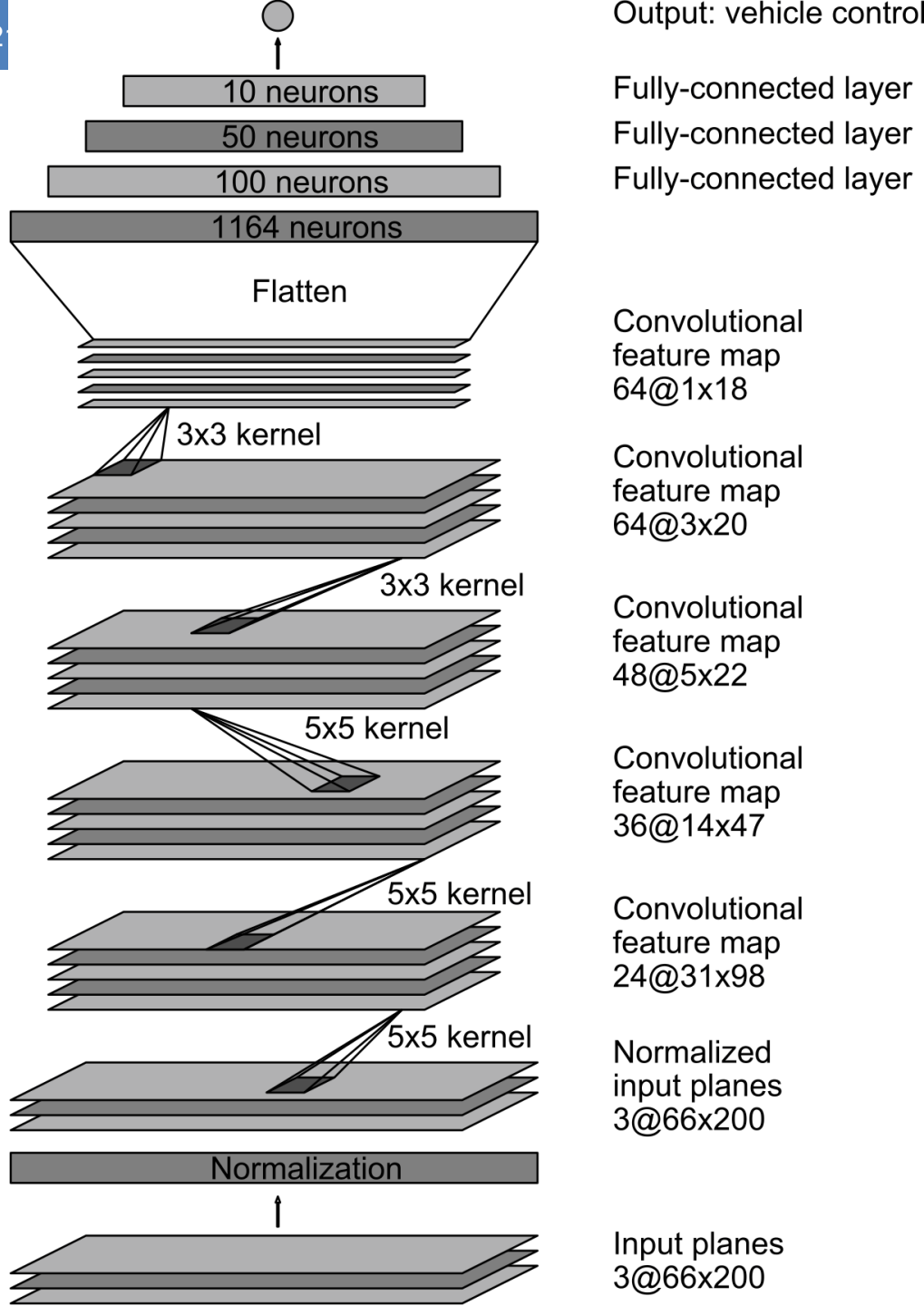
https://devblogs.nvidia.com/parallelforall/deep-learning-self-driving-cars/



- Training:
  - Additionally use left and right camera: negative examples!
  - Highway, residential roads, unpaved roads, car parks
  - Different weather conditions

- Result: Autonomous 98% of the time => 2% driver intervention





Problems with Deep Learning

- 99% success rate sounds good, but 1% failure is often unacceptable (e.g. autonomous car)
  - Failures are unavoidable =>
  - need quality estimate/ uncertainty of the result!
  - Often not available for DL ☹
- Lack of theory regarding deep learning
  - Acts like a black box…
- No introspection of how or why a DL system is behaving like it is =>
  - No safety guarantees possible
- Deep learning only part of an overall AI system
  - Hand-crafted methods can still be very powerful
  - Modelling useful (with input from DL)
  - Statistical methods
  - Reasoning
  - Planning