

# Spot The Differences Between Two Images

*Le Gia Khang*

*Le Duy Khang*

*Nguyen Hoang Tan*

## **CS231: Introduction to Computer Vision**

December 4, 2023

# Presentation Overview

## ① Problem Statement

Why it Matters

Input and Output Analysis

Examples

## ② Methodology

Pixel-Wise Comparison

Structural similarity index (SSIM)

Siamese Network

## ③ Experiments

# Presentation Overview

## ① Problem Statement

Why it Matters

Input and Output Analysis

Examples

## ② Methodology

Pixel-Wise Comparison

Structural similarity index (SSIM)

Siamese Network

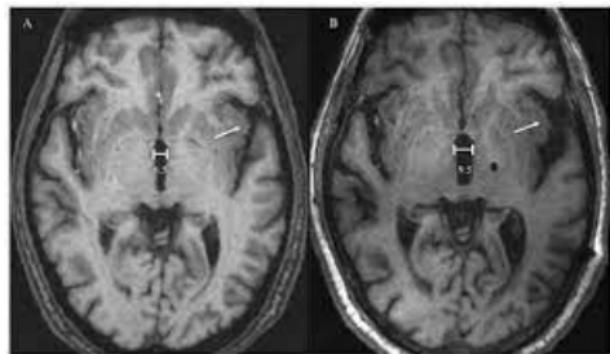
## ③ Experiments

# Why it Matters

## Applications

Detecting changes is a natural computer vision task:

- The “spot-the-difference” game
  - Facility monitoring
  - Medical imaging
  - Satellite surveillance
  - Counterfeit detection
- ... and many more.



Comparison of two MRI scans over 10-year period.

# Input and Output Analysis

## Input

Pair of images need to compare



# Input and Output Analysis

## Output

The provided images are marked with bounding boxes  
to indicate areas of distinction



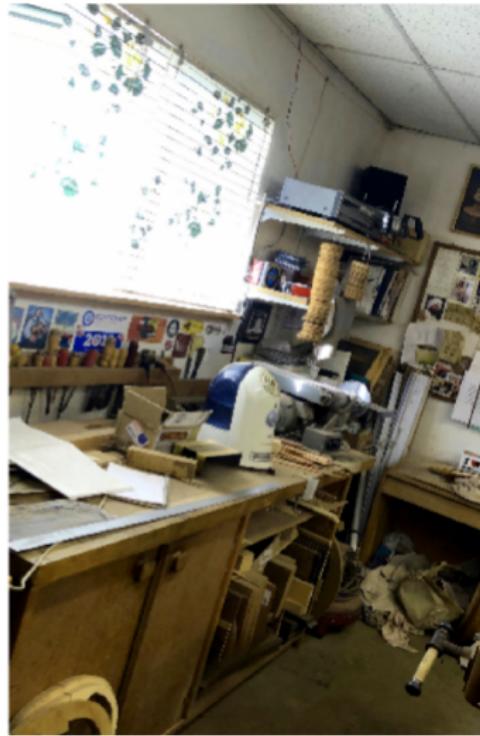
## Examples: Input



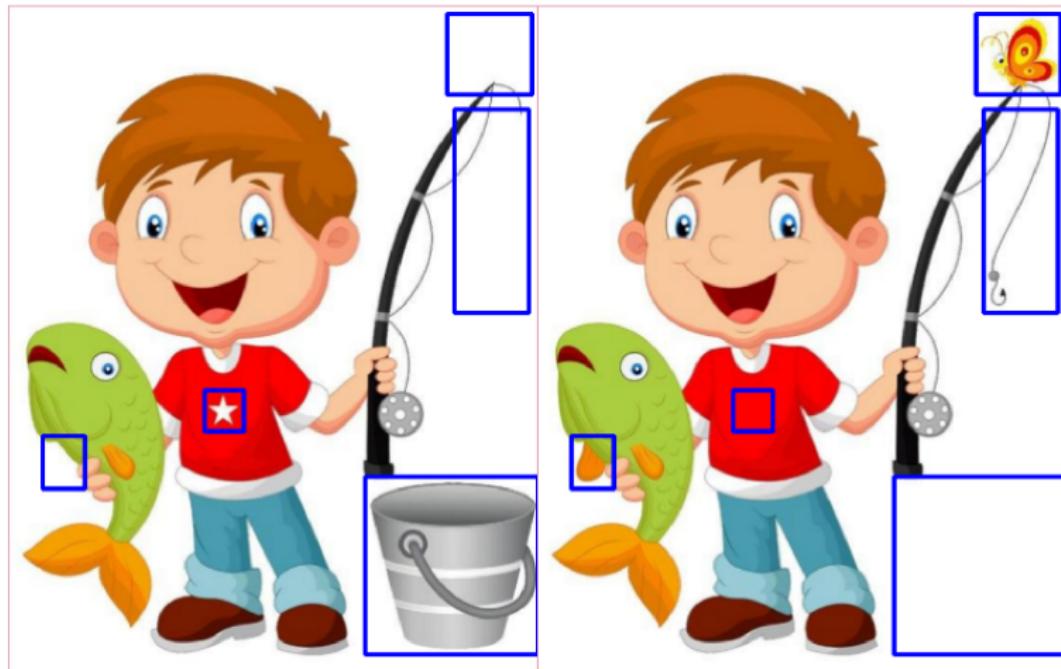
## Examples: Input



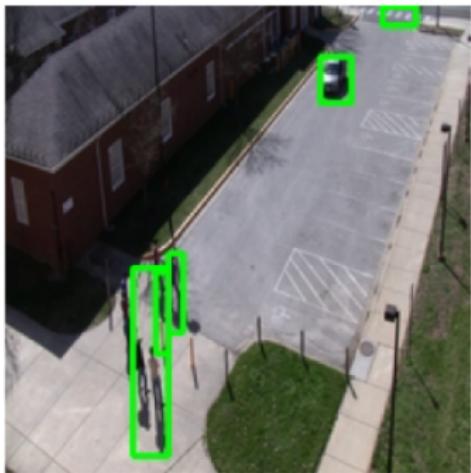
## Examples: Input



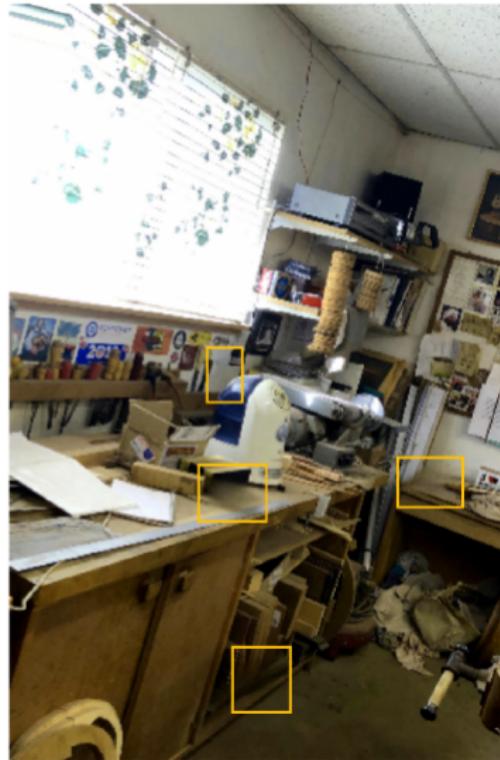
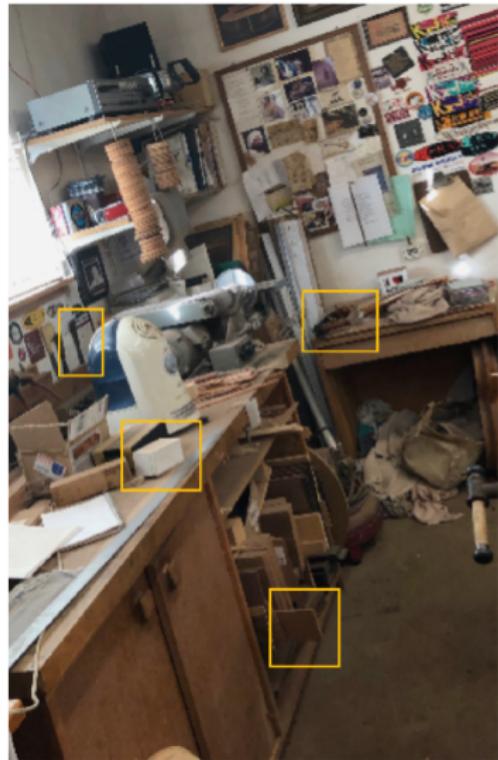
## Examples: Output



## Examples: Output



# Examples: Output



# Presentation Overview

## ① Problem Statement

Why it Matters

Input and Output Analysis

Examples

## ② Methodology

Pixel-Wise Comparison

Structural similarity index (SSIM)

Siamese Network

## ③ Experiments

# Presentation Overview

## ① Problem Statement

Why it Matters

Input and Output Analysis

Examples

## ② Methodology

Pixel-Wise Comparison

Structural similarity index (SSIM)

Siamese Network

## ③ Experiments

# Pixel-Wise

## Pixel-Wise Comparison

Compare the input image pair by first detecting which **pixels** have **changed** between the first and second image and then **segmenting** those pixels into **clusters** that *approximately* represent the objects that have changed.

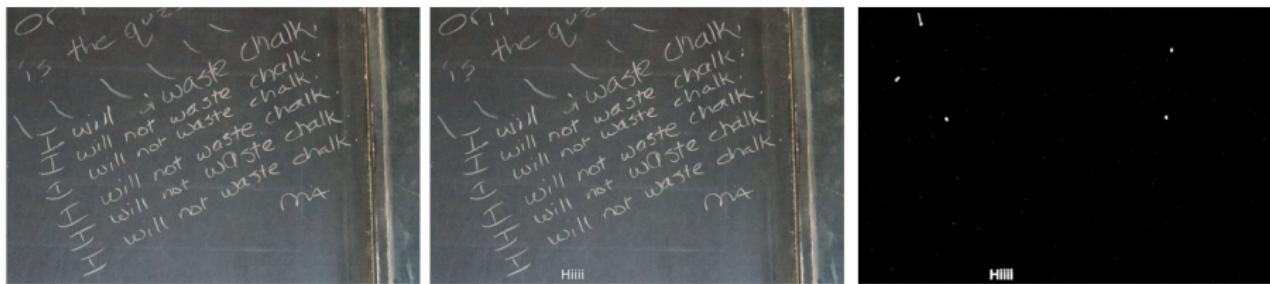
Or, in short:

- ① Detect pixels that have changed.
- ② Perform clustering on those pixels.

# Pixel-Wise: Detect changes in pixels

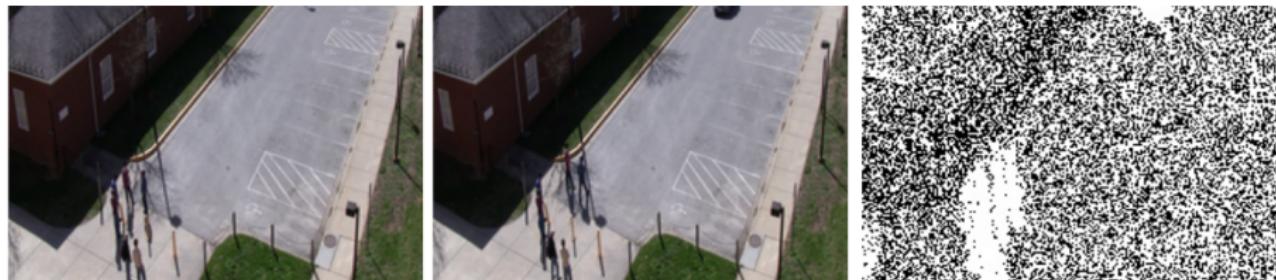
**Base approach:** pixel changes when its value is not the same.

Well, it works fine... with digital changes (e.g. photoshop,...)



## Pixel-Wise: Detect changes in pixels

But fails horribly with non-digital changes (e.g. camera's noises, environmental changes, . . . )



## Pixel-Wise: Detect changes in pixels

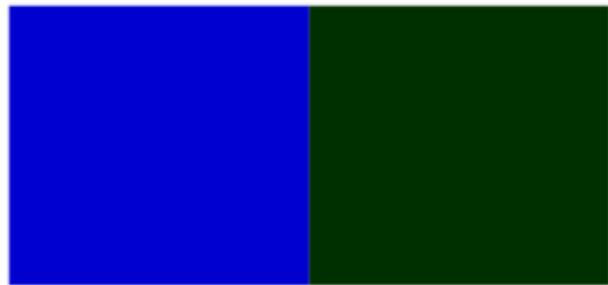
So, **some changes are tolerable**. Hence, we will need to compute the **distance** between two pixels and compare that against a **threshold** value.



The most popular method of computing that distance is to compute the difference between the two pixels' **luminosity** (convert images into grayscale and then compare its values).

## Pixel-Wise: Detect changes in pixels

However, some colours have similar **grayscale representation** despite not being the same color.



RGB(0, 0, 255) and RGB(0, 48, 0)



Convert to **grayscale**

## Pixel-Wise: Detect changes in pixels

Distance between colours  $C_1$  and  $C_2$ :

$$\bar{r} = \frac{C_{1,R} + C_{2,R}}{2}$$

$$\Delta R = C_{1,R} - C_{2,R}$$

$$\Delta G = C_{1,G} - C_{2,G}$$

$$\Delta B = C_{1,B} - C_{2,B}$$

$$\Delta C = \sqrt{\left(2 + \frac{\bar{r}}{256}\right) \times \Delta R^2 + 4 \times \Delta G^2 + \left(2 + \frac{255 - \bar{r}}{256}\right) \times \Delta B^2}$$

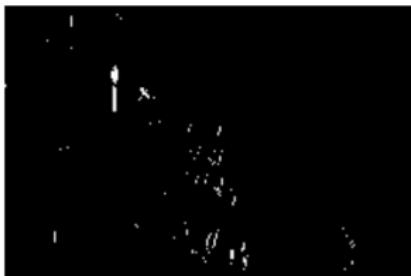
# Pixel-Wise: Detect changes in pixels



Figure: Result after applying the colour distance function

# Pixelwise: Clustering

As we only care about **objects** that have changed not changed pixels, we have to group those pixels **into clusters** that approximate those objects.



And, due to the characteristics of those pixels (denser area is more likely to be an object than noises) **density based clustering algorithms** like *DBScan* will perform well.

# Pixelwise: Clustering

Finally, encapsulate those clusters with bounding boxes.



## Pixel-wise optional: Remove non-interesting clusters

In images from a surveillance video, a lot of unnecessary changes are detected like slight movement, reflective area, ...

These clusters can be reduced by using a simple scoring system:

---

$$\text{number\_of\_changed\_pixels} + \text{height\_of\_cluster} \times 2 + \text{width\_of\_cluster} \geq \text{threshold}$$

---



# Presentation Overview

## ① Problem Statement

Why it Matters

Input and Output Analysis

Examples

## ② Methodology

Pixel-Wise Comparison

Structural similarity index (SSIM)

Siamese Network

## ③ Experiments

# Structural similarity index (SSIM)

## Structural similarity index (SSIM)

SSIM is used as a metric to measure the similarity between two given images which is a value between -1 and +1.

- Designed to mimic human perception in evaluating image quality.
- It takes into account *luminance*, *contrast*, and *structure*.

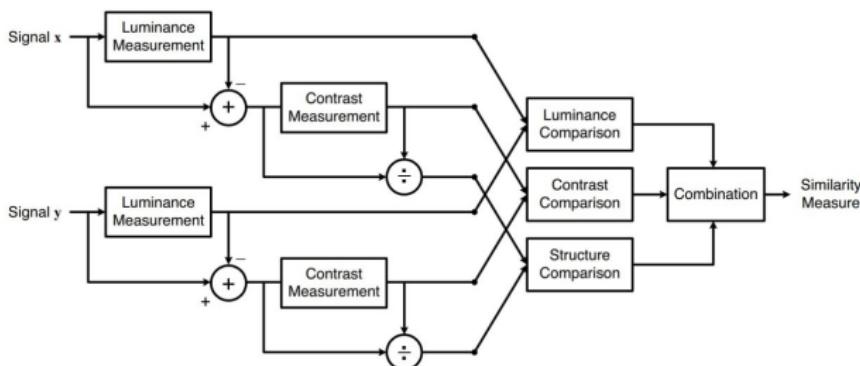


Figure: Diagram of the structural similarity (SSIM) measurement system

## SSIM: Luminance

Luminance is measured by **averaging** over all the pixel values. Its denoted by  $\mu$  and the formula is given below:

$$\mu_x = \frac{1}{N} \sum_{i=1}^N x_i$$

The luminance comparison function  $\ell(x, y)$

$$\ell(x, y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}$$

- 
- $\mu_x, \mu_y$ : Mean luminance of the first and second image.
  - $C_1 = (K_1 L)^2$ : where  $L = 255$  for 8-bit component images.
  - $K_1 \ll 1$ : Default value is .01

## SSIM: Contrast

**Contrast** is measured by taking the standard deviation (square root of variance) of all the pixel values. It is denoted by  $\sigma$

$$\sigma_x = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (x_i - \mu_x)^2}$$

The luminance comparison function  $(x, y)$

$$c(x, y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}$$

- 
- $\sigma_x, \sigma_y$ : The standard deviation of pixel values
  - $C_2 = (K_2 L)^2$ : where  $L = 255$  for 8-bit component images.
  - $K_2 \ll 1$ : Default value is .03

## SSIM: Structure

The formula for covariance  $\sigma_{xy}$  between  $x$  and  $y$  is given by:

$$\sigma_{xy} = \frac{1}{N-1} \sum_{i=1}^N (x_i - \mu_x)(y_i - \mu_y)$$

Structure comparison function  $s(x, y)$

$$s(x, y) = \frac{\sigma_{xy} + C_3}{\sigma_x \sigma_y + C_3}$$

- 
- $\sigma_{xy}$ : The covariance between  $x$  and  $y$ .
  - $C_3 = C_2/2$ .

## The SSIM score

And finally, the SSIM score is given by,

$$\text{SSIM}(x, y) = \left(l(x, y)\right)^{\alpha} \left(c(x, y)\right)^{\beta} \left(s(x, y)\right)^{\gamma}$$

Where  $\alpha > 0, \beta > 0, \gamma > 0$  denote the importance of each of the metrics.

To simplify the expression, if we assume,  $\alpha = \beta = \gamma = 1$ , we can get,

$$\text{SSIM}(x, y) = \frac{(2\mu_{\mathbf{A}}\mu_{\mathbf{B}} + C_1)(2\sigma_{\mathbf{AB}} + C_2)}{(\mu_{\mathbf{A}}^2 + \mu_{\mathbf{B}}^2 + C_1)(\sigma_{\mathbf{A}}^2 + \sigma_{\mathbf{B}}^2 + C_2)}.$$

# Mean Structural Similarity Index

Instead of applying the above metrics globally, it's better to apply the metrics regionally

## Mean Structural Similarity Index (MSSIM)

For image quality assessment, it is useful to apply the SSIM index locally rather than globally.

- Image statistical features are usually highly spatially nonstationary.
- Only a local area in the image can be perceived with high resolution by the human observer at one time instance

# Mean Structural Similarity Index

## Gaussian Weighing

We use an 11x11 circular-symmetric Gaussian Weighing function

- Moves pixel-by-pixel over the entire image.
- At each step, the local statistics and SSIM index are calculated within the local window.

$$\mu_x = \sum_{i=1}^N w_i x_i$$

$$\sigma_x = \left( \sum_{i=1}^N w_i (x_i - \mu_x)^2 \right)^{\frac{1}{2}}$$

Where  $w_i$  is the gaussian weighting function.

$$\sigma_{xy} = \sum_{i=1}^N w_i (x_i - \mu_x)(y_i - \mu_y).$$

# Mean Structural Similarity Index

Once computations are performed all over the image, we simply take the *mean of all the local SSIM values* and arrive at the global SSIM value.

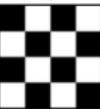
$$\text{MSSIM}(\mathbf{X}, \mathbf{Y}) = \frac{1}{M} \sum_{j=1}^M \text{SSIM}(\mathbf{x}_j, \mathbf{y}_j)$$

# SSIM: Examples

$$\text{SSIM} \left( \begin{array}{c} \text{[Solid Black]} \\ 0/255 \end{array}, \begin{array}{c} \text{[Solid White]} \\ 255/255 \end{array} \right) = \begin{array}{c} \text{[Solid Black]} \\ l = 0.0001 \end{array} \cdot \begin{array}{c} \text{[Solid White]} \\ c = 1 \end{array} \cdot \begin{array}{c} \text{[Solid White]} \\ s = 1 \end{array} = \begin{array}{c} \text{[Solid Black]} \\ 0.0001 \end{array}$$

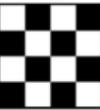
---

$$\text{SSIM} \left( \begin{array}{c} \text{[Grey]} \\ 128/255 \end{array}, \begin{array}{c} \text{[Grey]} \\ b|w \end{array} \right) = \begin{array}{c} \text{[White]} \\ l = 1 \end{array} \cdot \begin{array}{c} \text{[Solid Black]} \\ c = 0.0036 \end{array} \cdot \begin{array}{c} \text{[White]} \\ s = 1 \end{array} = \begin{array}{c} \text{[Solid Black]} \\ 0.0036 \end{array}$$

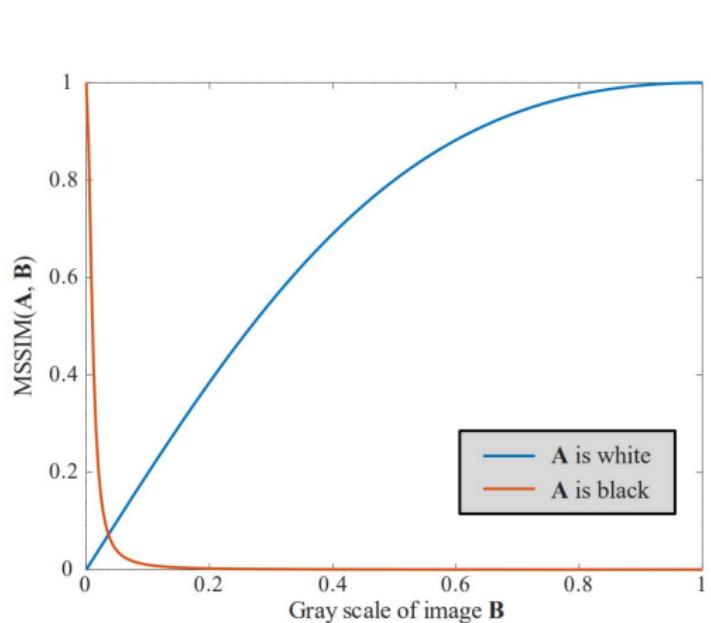
 

---

$$\text{SSIM} \left( \begin{array}{c} \text{[Grey]} \\ b|w \end{array}, \begin{array}{c} \text{[Grey]} \\ w|b \end{array} \right) = \begin{array}{c} \text{[White]} \\ l = 1 \end{array} \cdot \begin{array}{c} \text{[White]} \\ c = 1 \end{array} \cdot \begin{array}{c} \text{[Red]} \\ s = -0.9964 \end{array} = \begin{array}{c} \text{[Red]} \\ -0.9964 \end{array}$$

# SSIM: Examples



$$\text{MSSIM} \left( \begin{matrix} \text{white} \\ 255/255 \end{matrix}, \begin{matrix} \text{white} \\ 255/255 \end{matrix} \right) = 0.99997$$

$$\text{MSSIM} \left( \begin{matrix} \text{gray} \\ 128/255 \end{matrix}, \begin{matrix} \text{gray} \\ 130/255 \end{matrix} \right) = 0.99988$$

$$\text{MSSIM} \left( \begin{matrix} \text{black} \\ 0 \end{matrix}, \begin{matrix} \text{black} \\ 2/255 \end{matrix} \right) = 0.61914$$

$$\text{MSSIM} \left( \begin{matrix} \text{light gray} \\ 222/255 \end{matrix}, \begin{matrix} \text{white} \\ 255/255 \end{matrix} \right) = 0.99047$$

$$\text{MSSIM} \left( \begin{matrix} \text{black} \\ 0/255 \end{matrix}, \begin{matrix} \text{black} \\ 26/255 \end{matrix} \right) = 0.00953$$

MSSIM behavior for constant grayscale images.

# Presentation Overview

## ① Problem Statement

Why it Matters

Input and Output Analysis

Examples

## ② Methodology

Pixel-Wise Comparison

Structural similarity index (SSIM)

Siamese Network

## ③ Experiments

# Siamese Network

Proposed in the article "The Change You Want To See" accepted at the IEEE/CVF Winter Conference on Application of Computer Vision 2023.



Ragav Sachdeva

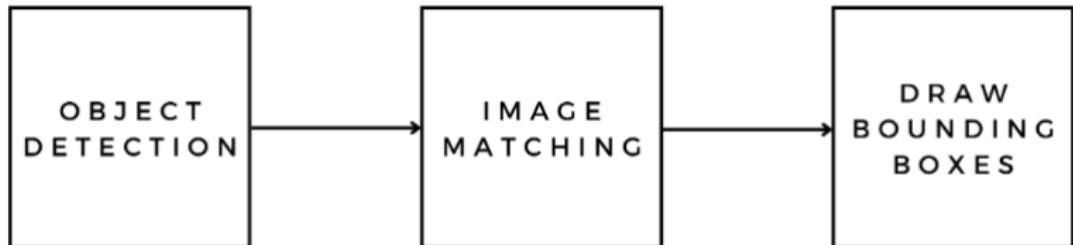


Andrew Zisserman

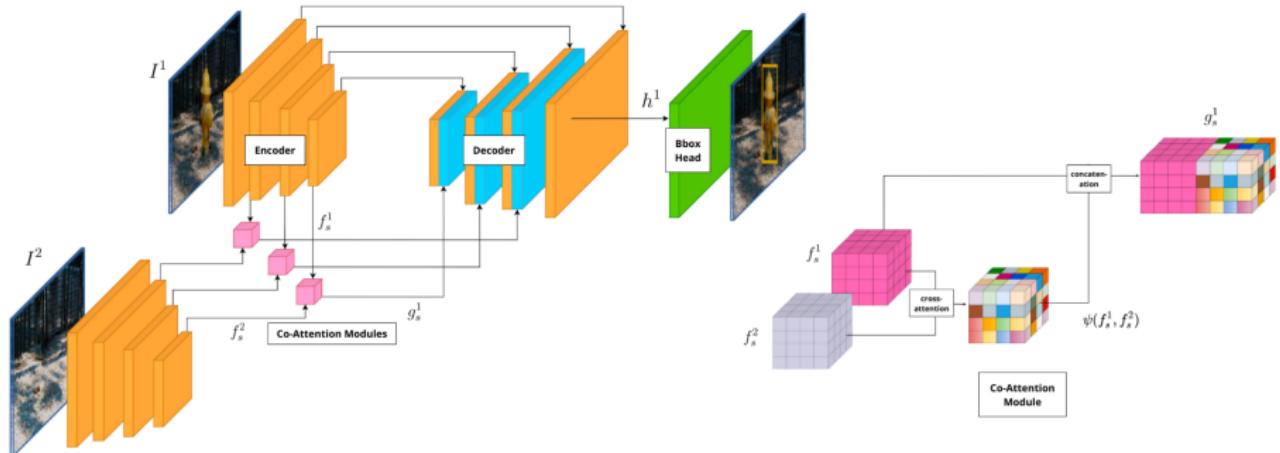
Visual Geometry Group at the University of Oxford

# Siamese Network: Overview

- Enable “object-level” change prediction and simplify counting the number of changes between two images.
- Use an architecture that operates on two images with geometric (scale, rotation,...) and photometric changes.
- Designed to be class-agnostic, it can detect changes irrespective of the object classes involved.



# Model Architecture Overview



**Figure: Architecture:** Utilizing a dual-image encoder, feature maps ( $f_1^s, f_2^s$ ) are generated. A co-attention module aligns and conditions these maps ( $g_1^s, g_2^s$ ). Subsequently, a U-Net decoder processes the original and conditioned maps to yield final feature maps ( $h_1, h_2$ ). The bounding box detector head employs  $h_1$  and  $h_2$  to generate bounding boxes for images  $I_1$  and  $I_2$ , respectively

# Siamese Network: Overview

## Siamese Network

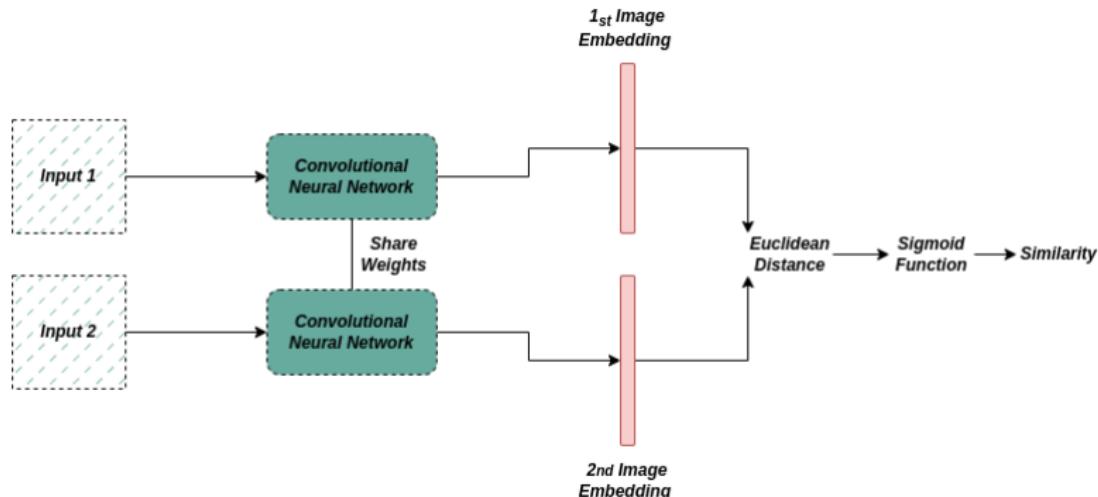
A type of neural network architecture designed for tasks involving similarity or distance measurement between input pairs.

- Consists of two identical subnetworks (or twins) that share the same set of weights and parameters.
- The name originates from Chang (left) and Eng Bunker (right).



# Siamese Network: Architecture

- Consists of two identical subnetworks.
- Extract feature vectors from both networks using a common set of convolutional and fully connected layers.
- Feature vectors from both networks are compared using a loss function  $L$ .

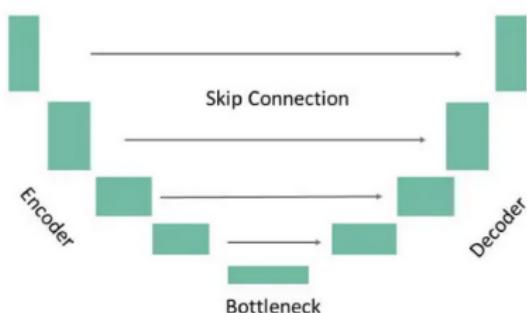


# U-Net Encoder-Decoder Network

U-Net is a type of convolutional neural network (CNN) architecture commonly used for image segmentation tasks.

Consists of an encoder-decoder structure:

- **Encoder:** capturing features from the input image.
- **Decoder:** upsampling and producing a segmented output.



The authors employed **ResNet50** as the CNN for the encoder.

# A UNet encoder-decoder with CoAM

## CoAM Attention Module

We wish to concatenate features from both images in order to condition the model on both input images.

- However, for a given spatial location, the relevant feature in the other image may not be at the same spatial location.

As a result, we use an attention mechanism to model long range dependencies.

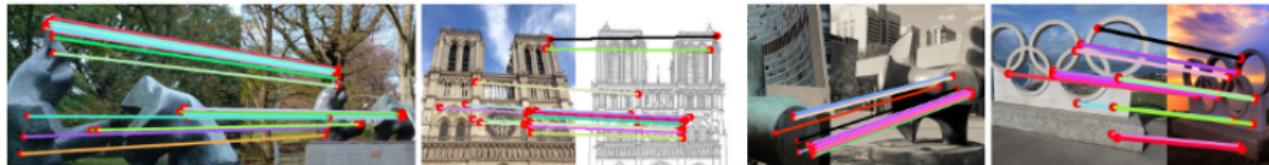
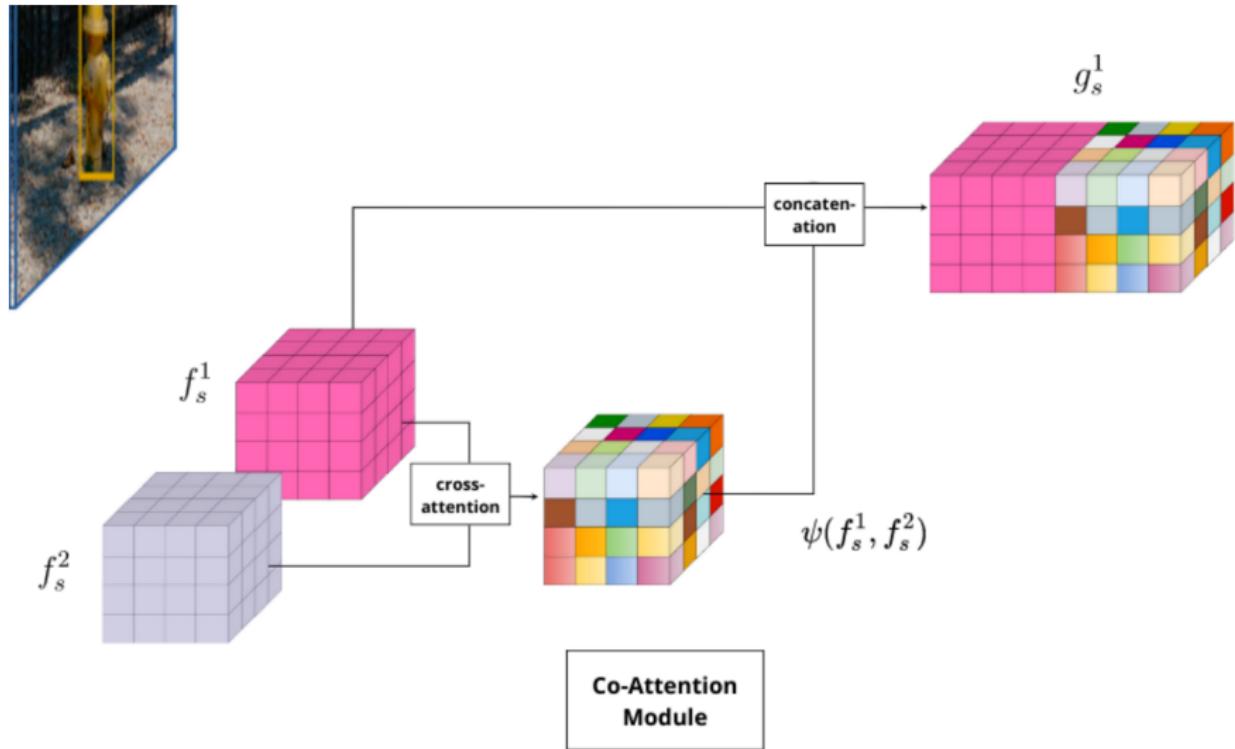
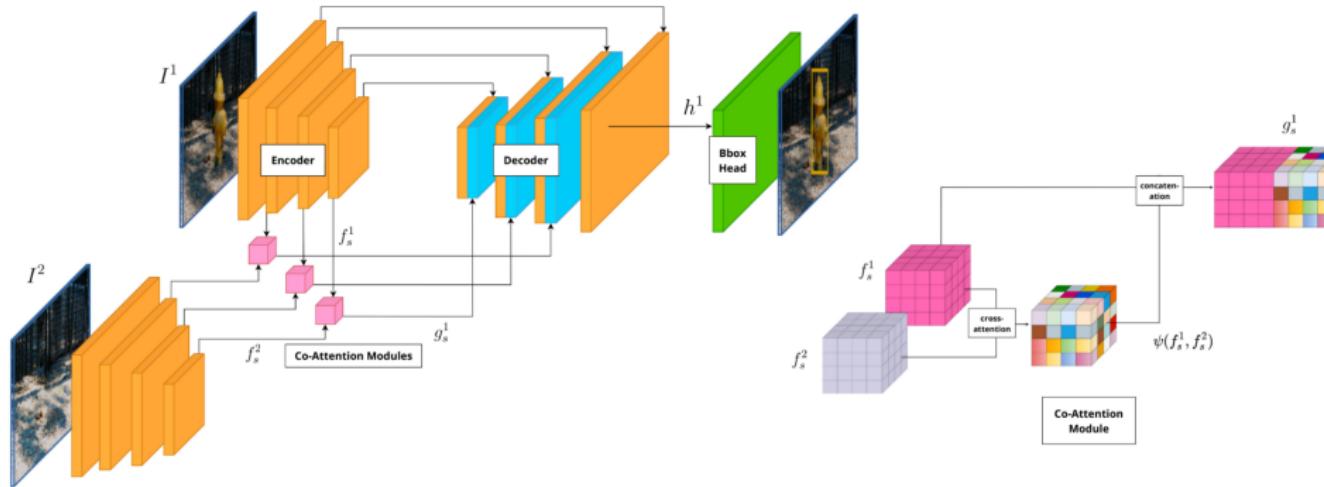


Figure: Correspondences obtained with the CoAM model, which is augmented with an attention mechanism.

# A UNet encoder-decoder with CoAM

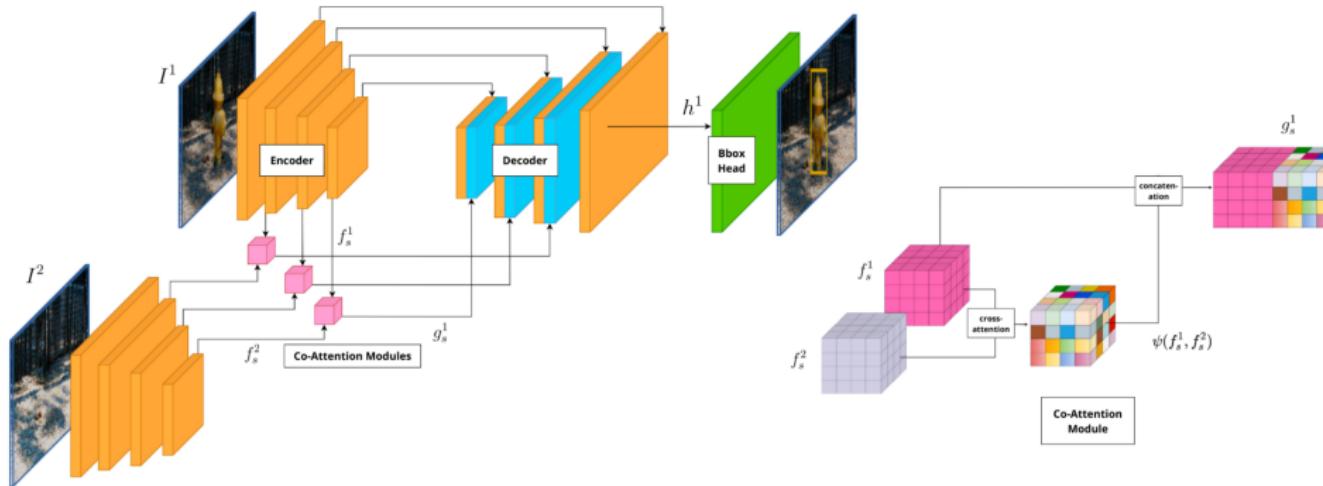


# Use Siamese network to detect changes



- First obtaining a set of dense feature descriptors for each image using a CNN-based (ResNet50) encoder.
- These features are then conditioned on each other using a co-attention mechanism that implicitly supplies the correspondences.

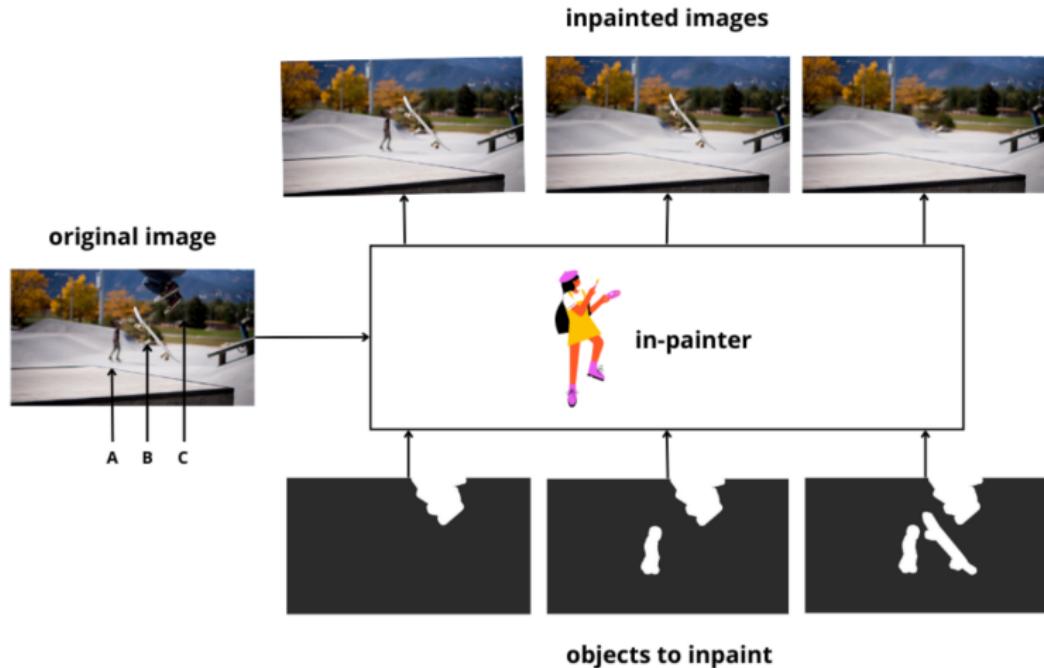
# Use Siamese network to detect changes



- Next, feature are passed through a decoder to obtain high resolution conditioned image descriptors which are used by a bounding box detection head to localise the changes.

# Siamese Network: Dataset

For this method, we make use of a state-of-the-art image inpainting method, **LaMa**, to make the objects *disappear*.



# Siamese Network: Dataset

- We also apply random affine transformations to the images along with colour jittering or add random text to “background” images.
- Datasets: COCO-Inpainted, Synthtext-Change, VIRAT-STD, Kubric-Change.

COCO-Inpainted



Kubric-Change



VIRAT-STD



Synthtext-Change



# Presentation Overview

## ① Problem Statement

Why it Matters

Input and Output Analysis

Examples

## ② Methodology

Pixel-Wise Comparison

Structural similarity index (SSIM)

Siamese Network

## ③ Experiments

# VIRAT Video Dataset

## VIRAT Video Dataset

A large-scale surveillance video dataset includes videos collected from stationary ground cameras and aerial vehicles. It has several event types:

- **Single Person Events (8):** walking, running, standing, throwing, gesturing, carrying, loitering, picking up
- **Person and Vehicle Events (7):** getting into or out of vehicle, opening or closing trunk, loading, unloading, dropping off, bicycling
- **Person and Facility Events (2):** entering or exiting facility



# Evaluation Metrics

To assess the effectiveness of the proposed methods, we will employ key performance metrics: *precision*, *recall*, and *F1 score*.

We calculate these metrics for a set of ground truth and predicted bounding boxes corresponding to specific image.

Table: Performance Evaluation of Different Approaches

Approach	Precision	Recall	F1 Score
Pixel-wise	.534	.417	.468
SSIM	.314	.169	.22
Siamese Network	.354	.584	.441

# Output Examples: Siamese Network



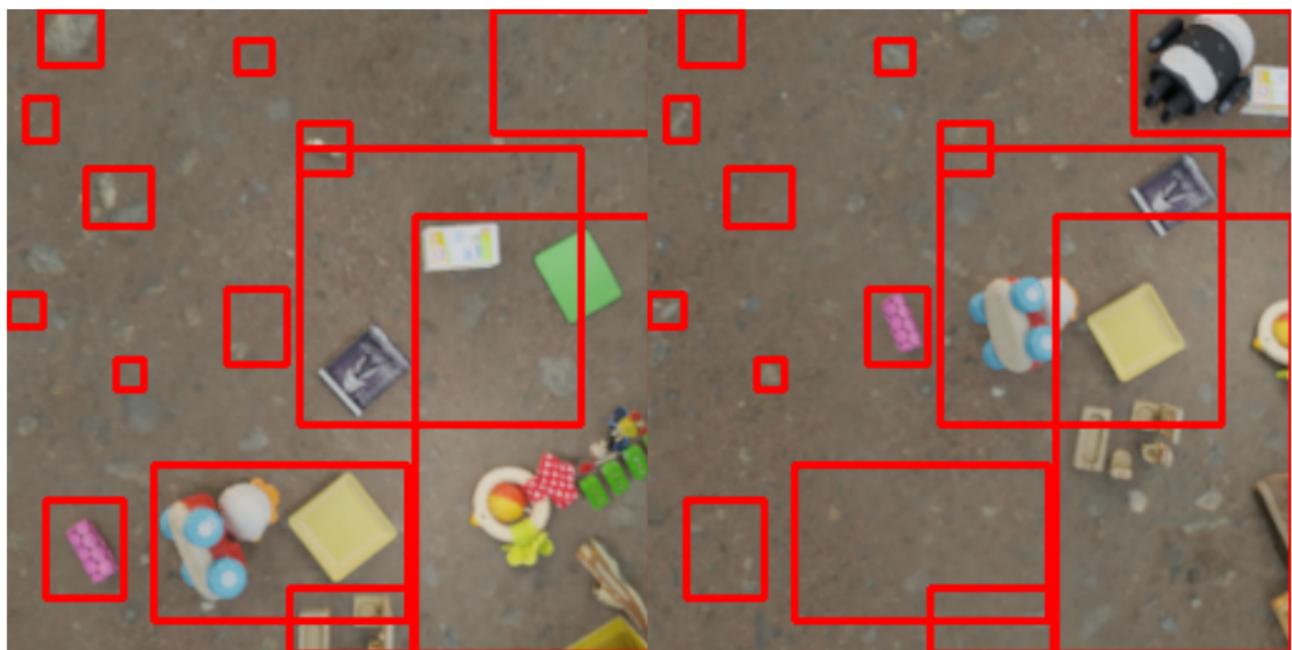
# Output Examples: Siamese Network



# Output Examples: Siamese Network



## Output Examples: SSIM



## Output Examples: Pixel-wise



# Reference



The Change You Want To See

Ragav Sachdeva and Andrew Zisserman



Image Quality Assessment: From Error Visibility to Structural Similarity

Zhou Wang, Alan Conrad Bovik, Hamid Rahim Sheikh and Eero P. Simoncelli



Datasets: [The Change You Want to See \(WACV 2023\)](#).

Ragav Sachdeva and Andrew Zisserman

# Thanks for listening!

## Q&A section