

Open in app ↗

Medium

 Search

★ Get unlimited access to the best of Medium for less than \$1/week. [Become a member](#)



Outlier detection using the RANSAC algorithm

saurabh dasgupta · [Follow](#)

6 min read · Feb 1, 2020



Listen



Share



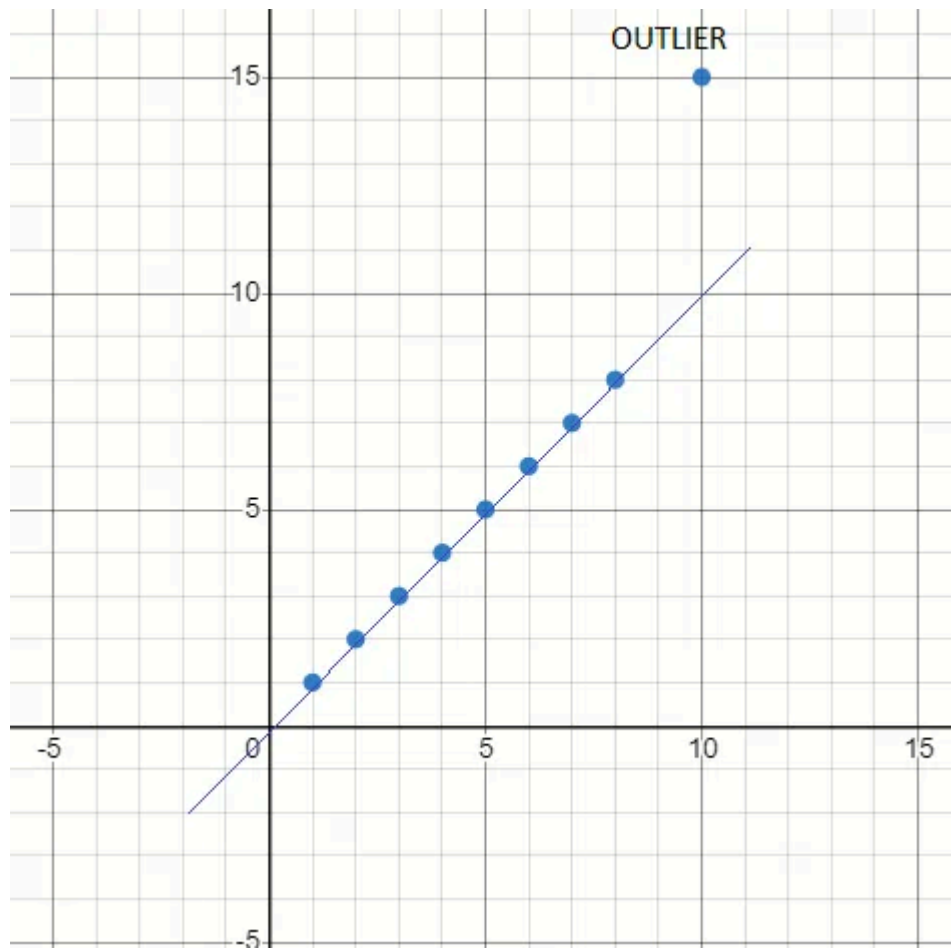
More

Introduction

In this article we will explore the **Random Sample Consensus** algorithm — more popularly known by the acronym RANSAC. This is an iterative and a non-deterministic algorithm that helps in eliminating outliers. This algorithm is commonly used to solve computer vision challenges. In this article I have presented the motivation for the RANSAC algorithm and the source code for a simplistic implementation using **Python**.

Problem definition

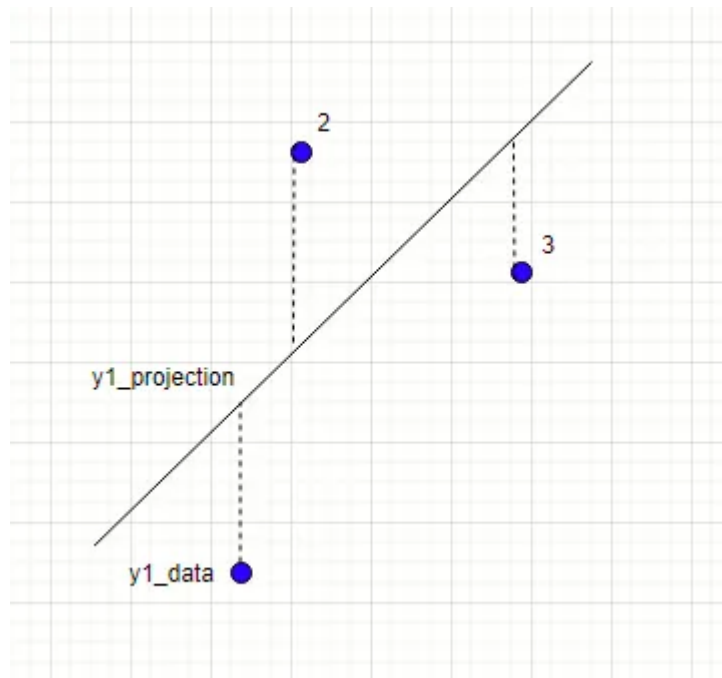
Consider the distribution of points in the following diagram.



Data points free from noise

The human mind can immediately spot that all the points in this distribution but for one is aligned in a straight line and the mind has no difficulty in distinguishing the inliers from the outliers. How can we make the computer emulate this aspect of the human behavior? The RANSAC algorithm attempts to address this challenge.

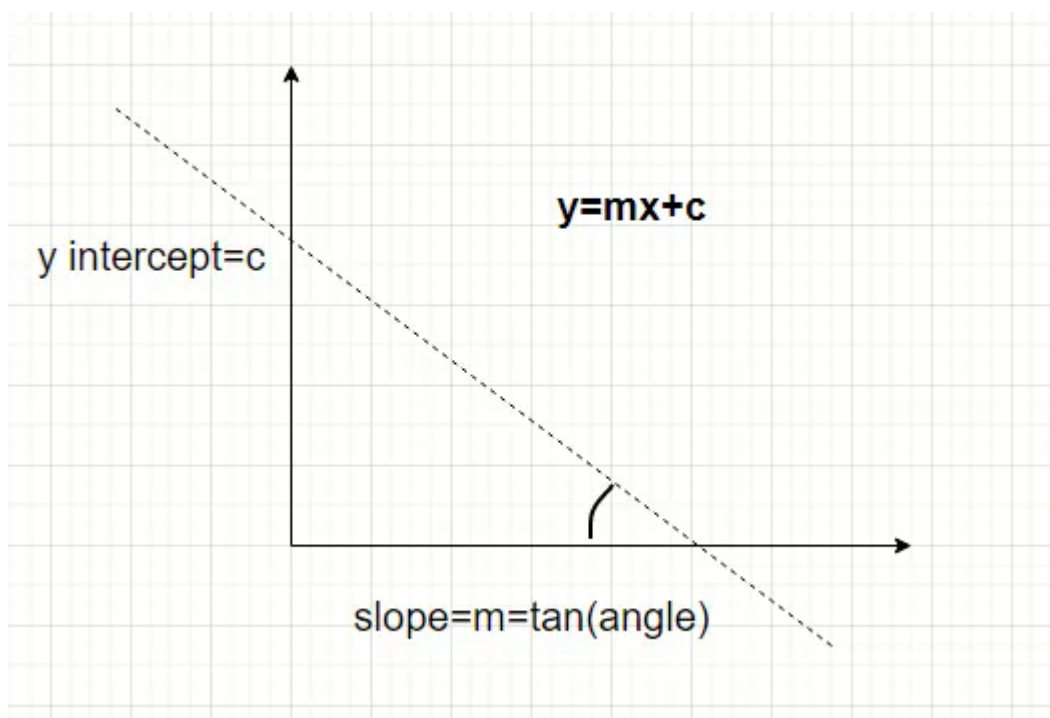
Traditional approach — Fitting a straight line



Consider the points above. How do we find a line which fits this distribution? One of the popular approaches is the least square distance method. In this approach we:

1. Create a cost function which sum up the distance of all points from the line
2. Iteratively tinker with the equation of the line and evaluate the cost function
3. Select the line which yields the lowest cost function

How do we build a cost function?

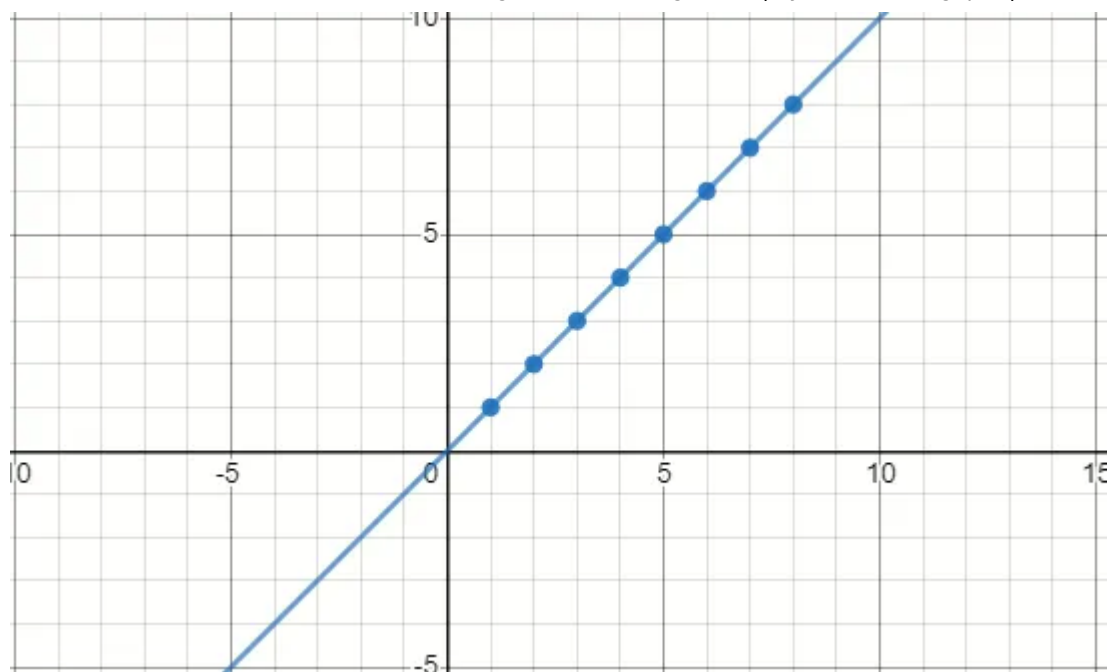


- Consider any point P_i with coordinates X_i, Y_i

- Consider a straight line with the equation $y=m.x+c$ where m is the slope and the c is the Y-intercept
- The distance of P_i from the point where the vertical projection intersects this line is given by $d(i)=(m.X_i+c) - Y_i$
- We do not want to be worried about negative values. Therefore let us square the above distance. $d(i)^2 = ((m.X_i+c) - Y_i)^2$
- The summation of the square of the vertical distance of all N points is given by $\text{Sum} = \sum ((m.X_i+c) - Y_i)^2$
- We can express the summation as a function which is dependent on two variables — The slope m and the Y intercept c
- The cost function $f(m,c)$ can now be expressed as $\sum ((m.X_i+c) - Y_i)^2$
- Since we have 2 variables (m and c) we need 2 equations to determine their values.
- The maxima/minima of a function can be determined by using derivatives. The point where a function achieves maxima/minima the derivative of the function at that point is zero.
- We will use partial differentiation to find the values of m and c which yield the lowest value
- The partial derivatives of $f(m,c)$ with respect to the variables m and c would have to be zero to give us the lowest cost value
- $df/dm = 2\sum ((m.X_i+c) - Y_i) * X_i$
- $df/dc = 2\sum ((m.X_i+c) - Y_i)$
- In the interest of time, I will skip the derivation of the least squares distance formula and straight away present the solution
- $m = (N.\sum(x.y) - \sum x.\sum y) / (N.\sum(x^2) + (\sum x).2)$
- $c = (\sum y - m.\sum x) / N$

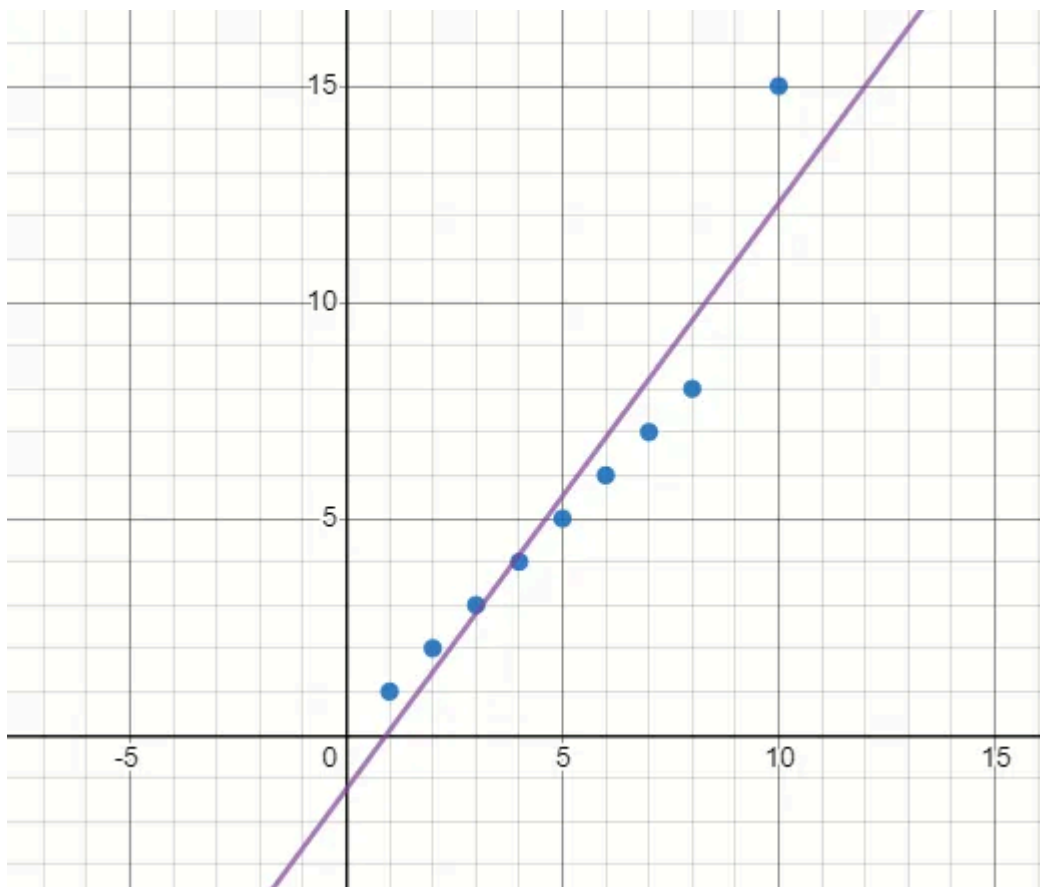
Challenges with least squares method

Scenario — No noise



Perfectly aligned data points

Scenario — A few noisy data points



Data points which are mostly linear and with 1 outlier

Understanding RANSAC — Overview

Before getting into the full details, I have presented a distilled version of RANSAC in this section

- Randomly select a smaller set of points (**n**) from the entire distribution (**N**)
- Use least squares regression to determine the linear equation which fits the **n** points
- Determine the average of the distance of every point **N** from this line. This score can be considered to be a measure of the goodness of the line.
- Keep track of the **score**. If this score is less than the best score obtained from previous iterations then discard the older linear equation and select the current linear equation.
- Go back the first step and continue iterating till you have completed a predetermined number of iterations
- Stop the algorithm when a predetermined number of iterations have been completed
- The linear equation available at the end of the iterations is possibly the best candidate line

We can see that the algorithm is not deterministic and hence the name *Random* in the acronym RANSAC. It is possible that you may not get the best model.

Understanding RANSAC — Detailed

In this section I have presented the algorithm from the Wikipedia page of RANSAC

1. **MAX** = max iterations
2. **n**= number of points to pick in every iteration. Could be initialized to 2
3. **best_model** = equation of the line with best_error . Initialize to NULL
4. **best_error**= The lowest error (average distance) obtained so far. Initialize to a large number
5. **threshold_error**=if the distance of a point from a line is below this value then the point is classified as an inlier otherwise outlier
6. **threshold_inliers**=minimum number of inliers for a model to be selected
7. **k**= count of iterations completed. Initialize to 0

Start

- $k = k + 1$
- if ($k > \text{MAX}$) then stop the algorithm
- select n random points from entire population. Denote this set by **random_points**
- Use least square regression to find the line which fits **random_points**. Denote this equation by **current_model**
- Determine **inliers** which is the set of all the points within **threshold_inliers** distance of **current_model**
- **count_of_inliers** = count of points in the **inliers**
- if **count_of_inliers** is less than **threshold_inliers** then abandon this sample and go to **Start**
- Extend the sample by combining **inliers** and **random_points**.
- Use least squares regression to find the line which fits **inliers** and **random_points**. This equation is denoted as **better_model**
- Determine the average distance of all points from **better_model**. This is denoted by **current_error**
- if (**current_error** > **best_error**) then go to **Start**
- **best_model** = **current_model**
- **best_error** = **current_error**
- Go to **Start**

End

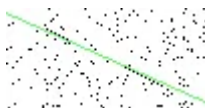
Results

Test 1

Input



Output



Test 2

Input



Output



Test 3

Input



Output



Overview of the code

Source code location

I have used the following tools to author the Python scripts that accompany this article.

1. Visual Studio 2019
2. with Python project templates
3. Python 3.7 engine

The source code can be found at <https://github.com/sdg002/RANSAC> You do not have to use Visual Studio. I am quite certain that the Python code should work as it is.

List of Python files and folders

- **RANSAC.py** — Outermost Python script which can be executed from the command line
- **GenerateNoisyLine.py** — Outermost Python script which will generate a random straight line with salt-pepper noise
- **LineModel.py** — Implements a class that represents the equation of a straight line
- **Point.py** — Implements a class which represents a 2d point
- **RansacHelper.py** — Implements the core RANSAC algorithm
- **Util.py** — Utility functions
- **test_*.py** — These are unit test classes
- **.\input** — The folder containing input files
- **.\output** — The folder where the resulting images are published

Quick start — Generating an image of a noisy line

- Run the script **GenerateNoisyLine.py** to generate a rectangular image with 1 line in a random orientation and salt-pepper noise
- The resulting image will be generated in the subfolder **.\out**

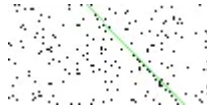


Straight line with salt and pepper noise

Quick start — Perform RANSAC on a noisy image

- Run the script **RANSAC.py** to find the best fitting line in a noisy image
- The input file is controlled by a variable inside **RANSAC.py** and the this file should be placed in the subdirectory **.\input**

- The output is generated in the form of a new image which has the RANSAC line superimposed over the original line



after running RANSAC the detected line

References and further reading

- Youtube lecture (<https://www.youtube.com/watch?v=BpOKB3OzQBQ>)
- Wikipedia article on RANSAC (https://en.wikipedia.org/wiki/Random_sample_consensus)
- Deriving the least squares regression (<https://online.stat.psu.edu/stat414/node/278/>)
- Weighted least squares (<https://towardsdatascience.com/when-and-how-to-use-weighted-least-squares-wls-models-a68808b1a89d>)
- Hough transform (https://en.wikipedia.org/wiki/Hough_transform)
- Finding the maxima and minima ([http://clas.sa.ucsb.edu/staff/lee/Max and Min's.htm](http://clas.sa.ucsb.edu/staff/lee/Max_and_Min's.htm))

Machine Learning

Algorithms

Computer Vision



Follow

Written by saurabh dasgupta

243 Followers · 12 Following

Over 22 years experience in software development. Porting C and Fortran code from UNIX to Windows NT.
My book on Neural Network: <http://amzn.eu/8G4erDQ>

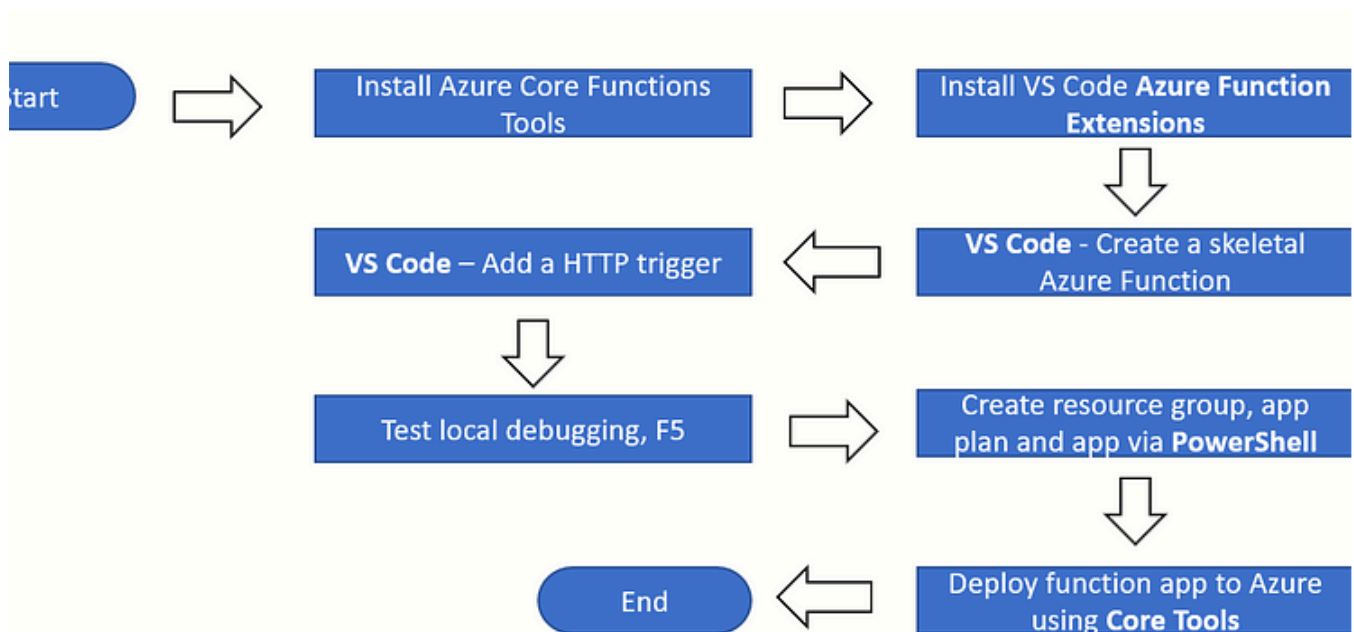



No responses yet

What are your thoughts?

Respond

More from saurabh dasgupta



 saurabh dasgupta


Developing and deploying a Python Azure Function—Step by step

Why did I write this?

Apr 16, 2022  68





 Maximilian Vogel

The ChatGPT list of lists: A collection of 3000+ prompts, GPTs, use-cases, tools, APIs, extensions...

Updated Dec-08, 2024. Added New Introductions, Prompts, Lists and Tools

Feb 8, 2023  13K  157



 Maximilian Vogel

The 10 Best Free Prompt Engineering Courses & Resources for ChatGPT, Midjourney & Co.

Updated Jan-21, 2024: Added a bonus resource.

Sep 7, 2023



1.5K



22



saurabh dasgupta

Power BI line chart with multiple years of Sales/Time series Data—So many options!

Problem statement

Jun 5, 2022



7

[See all from saurabh dasgupta](#)

Recommended from Medium



Fraidoon Omarzai

Image Segmentation In-Depth

Image segmentation is a crucial task in computer vision that involves dividing an image into meaningful segments to simplify or change its...

Aug 18, 2024 🖱 7



Eran Feit

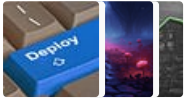
U-net Image Segmentation - How to segment persons in images 👤

Summary :

★ Jan 3 🖱 10



Lists



Predictive Modeling w/ Python

20 stories · 1774 saves



Practical Guides to Machine Learning

10 stories · 2142 saves



Natural Language Processing

1887 stories · 1536 saves



The New Chatbots: ChatGPT, Bard, and Beyond

12 stories · 537 saves



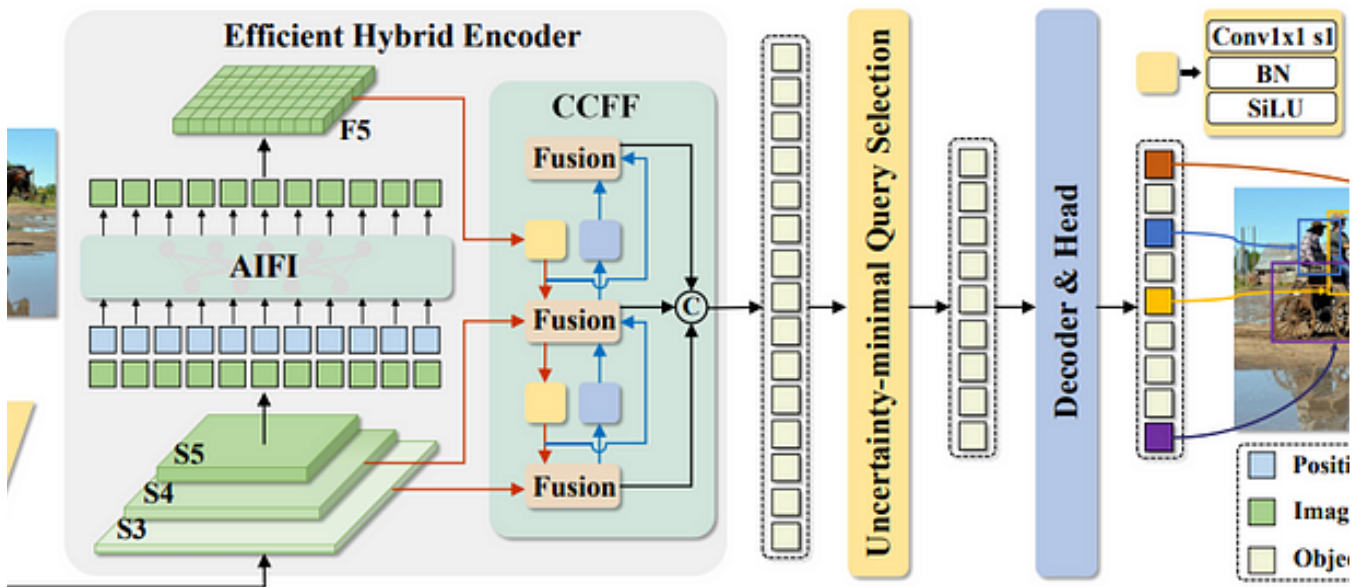
Abisha

Image Feature Extraction using Python - Part I

Basics of Image feature extraction techniques using python

★ Sep 5, 2024 🖱 312 💬 7





Antonio Consiglio

RT-DETR: A Faster Alternative to YOLO for Real-Time Object Detection (with Code)

Object detection has always faced a major challenge—balancing speed and accuracy. Traditional models like YOLO have been fast but...



Oct 28, 2024



240



1

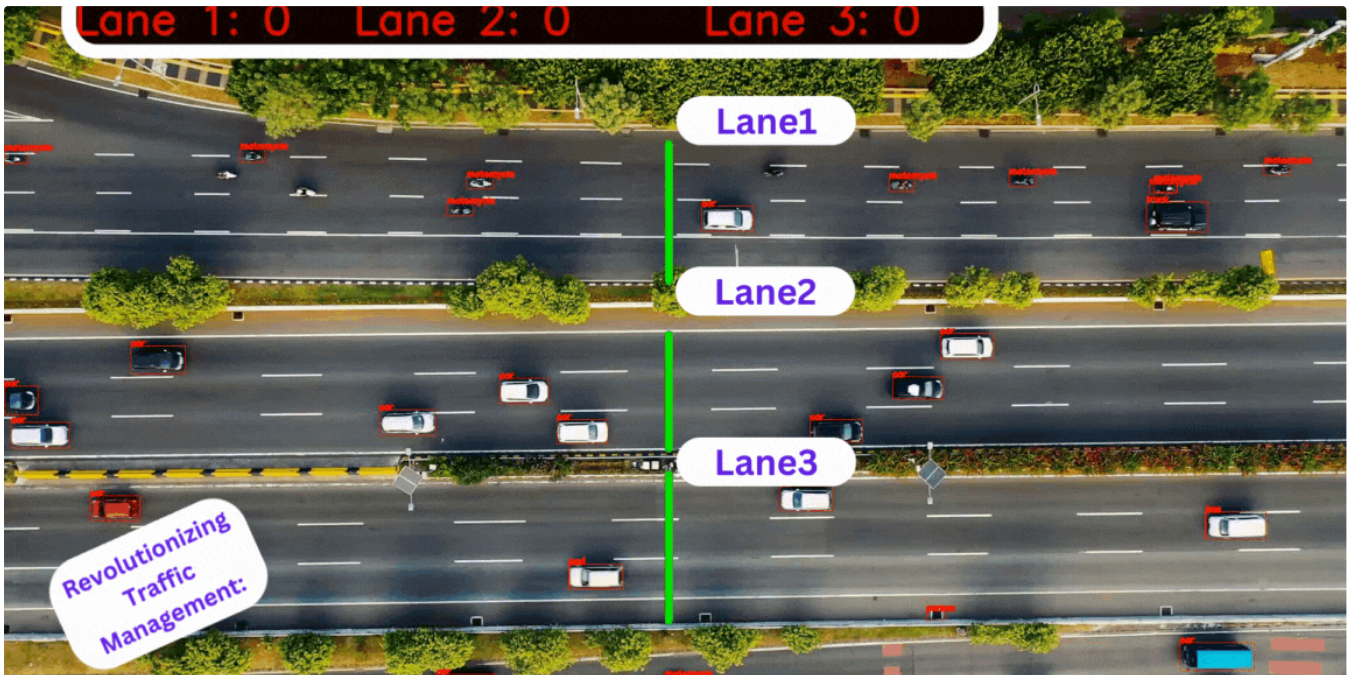



In Tech Spectrum by Aarafat Islam

Top 10 Object Detection Models in 2024

Object detection is a fundamental task in computer vision that involves identifying and localizing objects within an image. Deep learning...

★ Sep 30, 2024 🖱️ 374 💬 7



 Sunny Kumar

deepsort and yolo for object tracking and object counting.

DeepSORT (Deep Simple Online and Realtime Tracking) and YOLO (You Only Look Once) are commonly paired for real-time object tracking and...

★ Nov 11, 2024 🖱️ 4



See more recommendations