

Is it true that webcams with filters removed can see more at night?

Sources

Old DSLR w/ IR filter removed? : r/NightVision - Reddit

I am confused - Webcam IR filters - Imaging - Stargazers Lounge

Does Polarization ruin night

 reddit · 1

 stargazerslounge · 2

 dashcamtalk · 3

Answer

Removing the IR filter from a webcam or camera can indeed enhance its ability to see in low-light conditions, particularly at night. Here's why:

1. Increased light sensitivity: By removing the IR filter, the camera's sensor becomes sensitive to both visible light and near-infrared light. This allows more light to reach the sensor, improving its performance in low-light situations ¹⁻⁴.
2. Infrared detection: Without the IR filter, the camera can detect infrared light, which is invisible to the human eye but present in many light sources, including moonlight and artificial lighting ⁵. This enables the camera to "see" more in dark environments.
3. Night vision capabilities: When combined with an infrared light source, such as an IR flashlight or IR LEDs, a camera without an IR filter can effectively function as a night vision device ¹⁻⁵.

However, there are some important considerations and trade-offs to keep in mind:

1. Image quality: Removing the IR filter can affect daytime image quality, often resulting in a pink or reddish tint to images due to the sensor's increased sensitivity to infrared light ².
2. Black and white imagery: In many cases, images captured without an IR filter in low-light conditions appear black and white, as the infrared light overwhelms other wavelengths ⁴.
3. Lens compatibility: Some camera lenses have coatings that may block IR light, potentially limiting the effectiveness of removing the IR filter ¹.
4. Difficulty of removal: Properly removing the IR filter from a camera or webcam can be challenging and may require specialized tools and expertise ¹.
5. Full spectrum modification: Removing the IR filter often results in a "full spectrum" camera that is sensitive to both infrared and ultraviolet light, which may require additional filters for specific applications ¹.

While removing the IR filter can indeed improve a camera's ability to see in low-light conditions, it's important to weigh the benefits against the potential drawbacks and consider the specific use case before modifying a camera or webcam.

 Share  Rewrite

  ...

Is there a particular type of webcam for which the filter is especially easy to remove for those without much dexterity / that would make the process very easy for the most people?

Sources

Easy Solution! Remove IR filter PS3Eye - FreeTrack Forum

Best way to remove broken filter w/o damaging lens? - Reddit

Old DSLR w/ IR filter removed? : r/NightVision

 forum · 1

 Ask follow-up

While removing IR filters from webcams can enhance low-light performance, it's generally not a simple process for most models. However, some webcams are known to be easier to modify than others:

1. PlayStation Eye camera: This webcam is popular for IR filter removal due to its relatively straightforward disassembly process. The filter is often accessible without requiring extensive disassembly of the entire device.
2. Microsoft LifeCam series: Some models in this line, particularly older versions, have been reported to have easily removable IR filters.
3. Logitech C920: While not the easiest, this popular webcam has been successfully modified by many users and has extensive online tutorials available.
4. Generic, low-cost webcams: Inexpensive, no-name brand webcams often have simpler construction, making filter removal potentially easier.

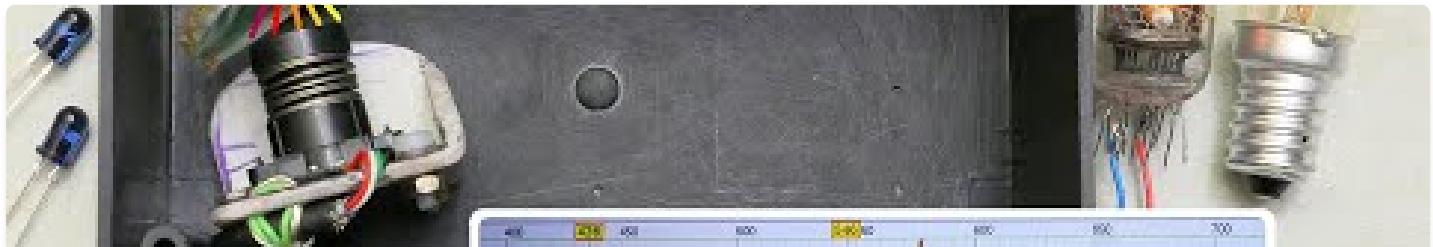
When considering IR filter removal, keep in mind:

- The process still requires some technical skill and can void warranties.
- Improper removal can damage the webcam or affect its functionality.
- Some webcams may not require physical filter removal; instead, you might be able to use software solutions or external IR pass filters.

For those with limited dexterity or technical experience, it's often safer and more practical to purchase a webcam specifically designed for low-light or IR use, rather than attempting to modify a standard webcam.

[Share](#) [Rewrite](#)

[Edit](#) [...](#)



Thank you. I've been discouraged because I'm trying to help a project and there are two problems that I would like to solve, both requiring very different approaches. I don't have an expensive FLIR camera and most people would not. In order to be successful, the project needs at least thousands of people aiming cameras at the sky at night.

Sources

[How to Convert a Webcam to a Near-Infrared Camera - YouTube](#)

[IR filter removal from new RPi camera - Raspberry Pi Forums](#)

[Remove IR-filter From Micros](#)

youtube · 1

forums.raspberrypi · 2

instructables · 3

Answer

I understand your project aims to involve thousands of people using cameras to observe the night sky, but you're facing challenges due to the lack of expensive specialized equipment like FLIR cameras. Here are some potential solutions based on the information provided:

1. Modified webcams: Removing the IR filter from standard webcams can significantly enhance their low-light performance. The PlayStation Eye camera is often recommended for this purpose due to its relatively simple modification process [1](#) [2](#). This could be a cost-effective option for many participants.
2. Logitech C920 or C615: These popular webcams have been successfully modified by many users with extensive online tutorials available [1](#) [2](#). While not the easiest, they offer a good alternative.
3. Microsoft LifeCam series: Some models in this line, particularly older versions, have been reported to have easily removable IR filters.

5. DIY IR pass filter: For those who successfully remove the IR filter, using developed but unexposed color film as a makeshift IR pass filter can be a budget-friendly option ^{1①}.
6. Leave the case open: After modifying the camera, consider leaving the case partially open to allow for lens adjustments as needed ³.
7. Software solutions: Some participants might be able to use software enhancements or settings adjustments to improve low-light performance without physical modifications.
8. Community workshops: Organizing local workshops to help people modify their cameras could increase participation and ensure proper modification techniques are used.
9. Partnering with astronomy clubs: Local astronomy clubs might have members with the technical skills to assist in camera modifications or may even have suitable equipment to contribute to the project.

By offering a range of options and support, you can make the project more accessible to a wider audience, increasing the likelihood of achieving the large-scale participation you need for success.

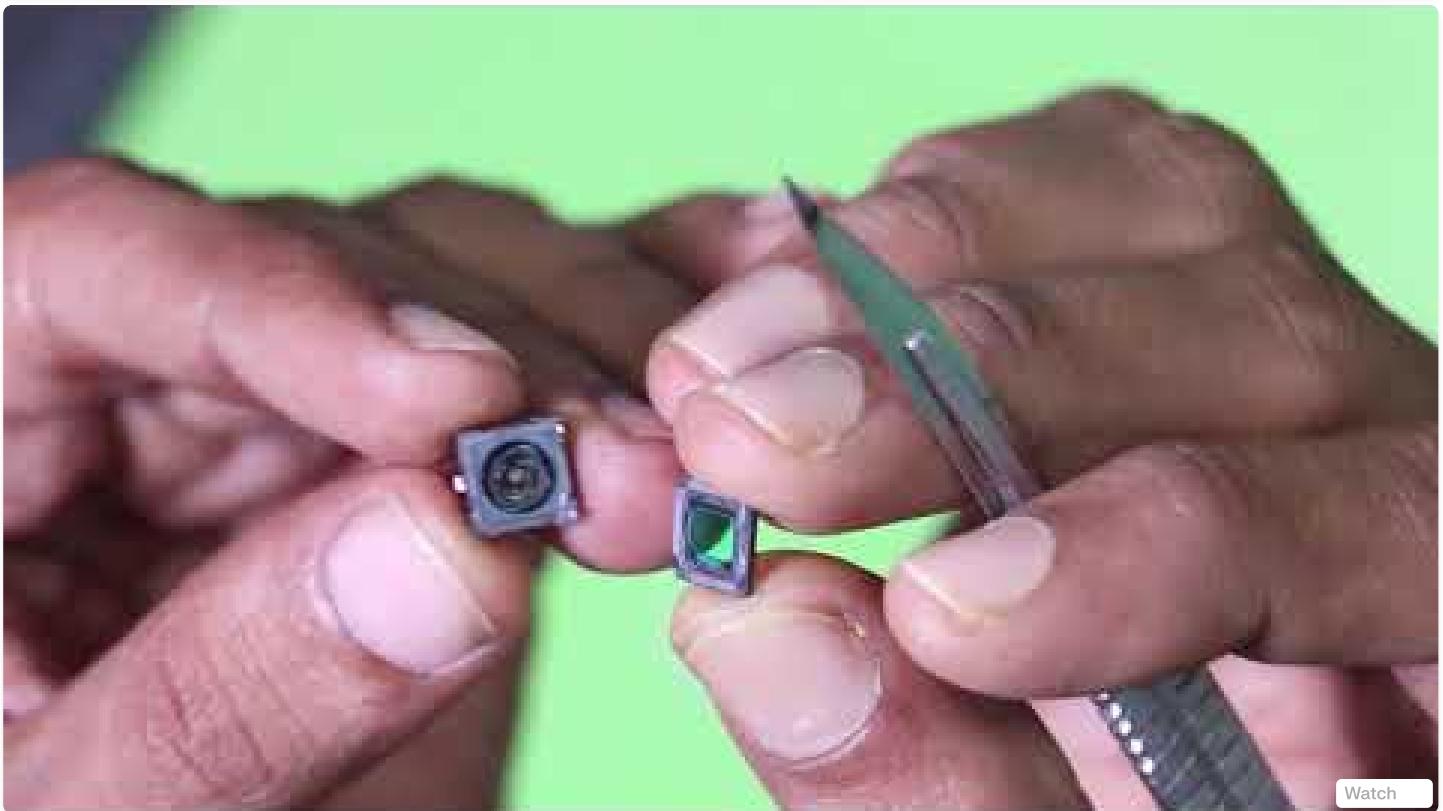
Share Rewrite

Bookmark Edit ...



Watch





I wrote a YOLO labeller using Python and Pygame and it works very well and I'm happy with it. However, training a model requires thousands of hours labelling on my part and I've been discouraged from doing that because a computer science department spent a year doing it already. I'm getting poor results from their model. Should I do it anyway?

Answer

Your situation is understandable, and it's great that you've developed a useful YOLO labeller. Here are some considerations to help you decide whether to proceed with your own labeling:

1. Quality of existing dataset:

- If the results from the existing model are poor, it might indicate issues with the original dataset's quality or relevance.
- Your own labeling could potentially produce better results if done carefully and consistently.

2. Specificity of your use case

- The existing dataset may not be tailored to your specific application or environment

3. Time investment vs. potential improvement:

- Consider whether the time spent labeling could yield significant improvements over the existing model.
- Even a smaller, but highly accurate dataset might outperform a larger, less precise one.

4. Incremental approach:

- Instead of relabeling everything, you could start by labeling a small subset and compare the results.
- This allows you to assess the potential benefit before committing to a full-scale labeling effort.

5. Augmenting existing data:

- You could potentially use the existing dataset as a starting point and augment it with your own labeled data.
- This approach combines the benefits of the existing work with your own targeted improvements.

6. Collaboration possibilities:

- Reach out to the computer science department that created the original dataset. They might be interested in collaborating or providing insights.
- Their experience could be valuable, and they might appreciate feedback on real-world application of their work.

7. Community involvement:

- Consider involving the community in the labeling process. Distributed effort could significantly reduce your personal time investment.

8. Automated assistance:

- Explore semi-automated labeling techniques that use the existing model as a starting point, which you then refine manually. Given your investment in creating a labeller and your dissatisfaction with current results, it may be worthwhile to proceed with your own labeling, at least on a limited scale initially. This approach allows you to evaluate the potential improvements without committing to a full relabeling effort upfront. Remember, the quality and relevance of the training data often have a more significant impact on model performance than the quantity alone.

Share Rewrite

...

I'm going to try to train a CNN to recognize patterns of stars such that it knows the common constellations and positions of stars. Depending on position of the observer on this planet, these patterns will vary. I have downloaded star catalogues and have been looking at Eleanor Lutz's github repo and work. I need to gather images in a way that agrees with the sensors used so it has good results though. Any ideas?

Sources

An Efficient and Robust Star Identification Algorithm Based on ...

Real-Time Convolutional Neural Network-Based Star Detection and ...

Real-Time Convolutional Neu ...

mdpi · 1

arxiv · 2

arxiv · 3

Answer

Your project to train a CNN to recognize star patterns and constellations is ambitious and exciting. Here are some suggestions to help you gather appropriate training data and improve your chances of success:

1. Simulated sky images:

Generate synthetic images of the night sky using star catalogs and astronomical software. This allows you to create a large dataset with precise control.

2. Sensor-specific simulation:

3. Data augmentation:

Implement techniques like random rotations, translations, and adding noise to increase the diversity of your training data and improve the model's robustness ^{1 3}.

4. Multi-band imaging:

Consider using multiple photometric bands (e.g., F336W, F438W, F814W) to provide more information to the CNN, similar to approaches used in star cluster analysis ^{4 5}.

5. Background variation:

Include realistic backgrounds in your training data, such as light pollution, atmospheric effects, and celestial objects like the Moon or planets ^{1 3}.

6. Observer location variation:

Generate training data for multiple observer locations on Earth to account for the changes in star positions and visibility ¹.

7. Time-based variation:

Include images representing different times of the year to account for seasonal changes in constellation visibility ¹.

8. Real-world data:

Supplement your synthetic data with a subset of real astronomical images to help the model generalize to actual observations ².

9. Magnitude thresholds:

Vary the magnitude limit in your simulated images to mimic different observing conditions and equipment sensitivities ³.

10. Constellation labeling:

Develop a consistent method for labeling constellations in your training data, considering both official boundaries and recognizable patterns.

11. Star identification:

Include individual star identification in your training data to allow the model to learn both constellation patterns and specific bright stars ^{1 2}.

12. Validation dataset:

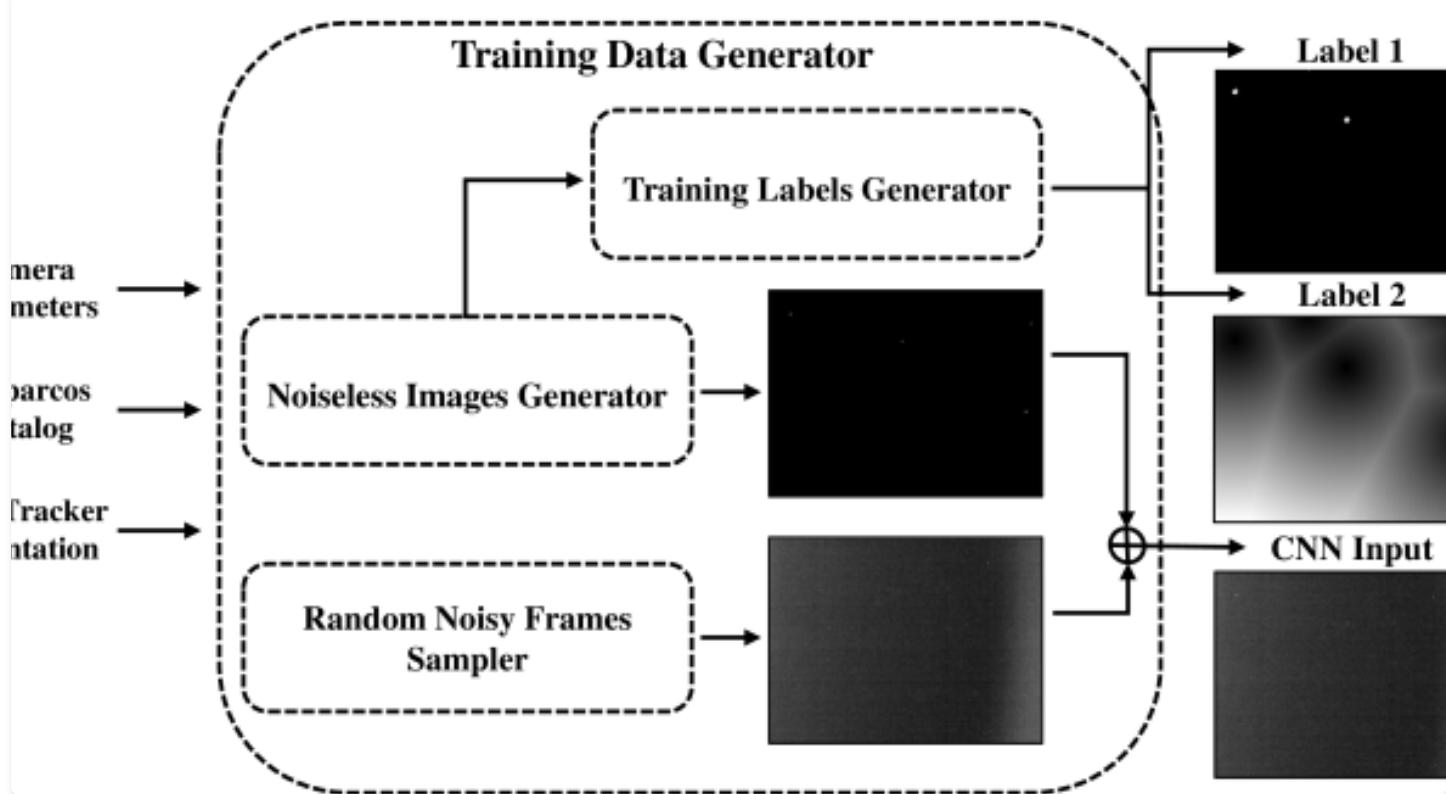
Create a separate validation dataset using real astronomical images to test the model's performance on actual night sky observations.

By combining these approaches, you can create a comprehensive training dataset that closely matches the expected input from your sensors while covering a wide range of observing conditions and locations. This should help your CNN learn to recognize star patterns and constellations more effectively across various real-world scenarios.

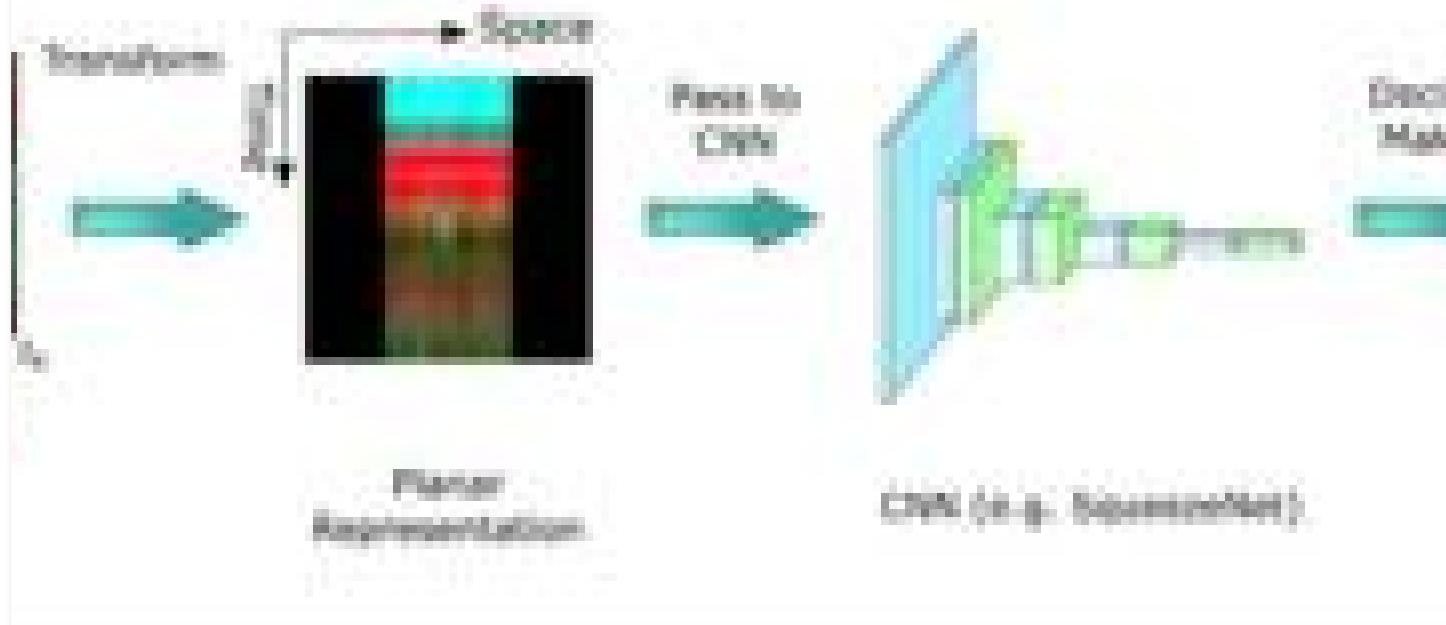
Share Rewrite

...





otion of image time series based on spatio-temporal representations,
Julio Krestin, Agustín Pachano and Nicola Vicentini

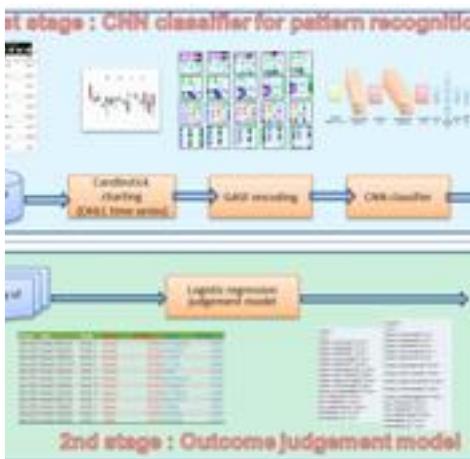


Luisrosario2604/Pattern-recognition-4



CNN (Convolutional Neural Network) for 4 types of images

A 1 Contributors 0 History ⚡ 1 Star 0 Forks



View More

I know Stellarium has an API using an http server that can be accessed by a Python module. If I could write a trainer for the CNN that went back and forth between changing ground position / camera vector, observing Stellarium sky output, and attempting to recognize using the CNN, it's poss

» Sources

Answer

Your idea of using Stellarium's API in conjunction with a CNN for an iterative training process is innovative and could potentially yield excellent results. Here's an approach to implement this concept:

1. Stellarium API Integration:

Use the RemoteControl plugin's HTTP-based interface to programmatically control Stellarium ¹. You can set the observer's location, time, and viewing direction using the API.

2. Image Generation:

Utilize Stellarium to generate sky images for various locations, times, and viewing angles. The MainService and ObjectService can be particularly useful for this purpose ¹.

3. CNN Architecture:

Design your CNN to take these generated images as input and predict the observer's location and/or the camera's pointing direction.

4. Iterative Training Process:

Implement a loop that does the following:

- Generate a sky image using Stellarium with random location/time/direction parameters.
- Feed this image to your CNN for prediction.
- Compare the CNN's prediction with the actual parameters used in Stellarium.
- Use this comparison to calculate loss and update the CNN weights.
- Repeat the process, gradually refining the CNN's ability to recognize patterns and make accurate predictions.

5. Data Augmentation:

Incorporate techniques like adding noise, adjusting brightness, or simulating atmospheric effects to make your model more robust to real-world conditions.

6. Validation:

Periodically test your model on a separate validation set of Stellarium-generated images to monitor its performance and prevent overfitting.

This approach combines the benefits of supervised learning (since you know the ground truth from Stellarium) with the flexibility of generating a vast amount of training data on-the-fly. It's similar to the method used by Tozzi in his deep learning celestial navigation project ², but with the added advantage of real-time interaction with Stellarium.

To further enhance your model:

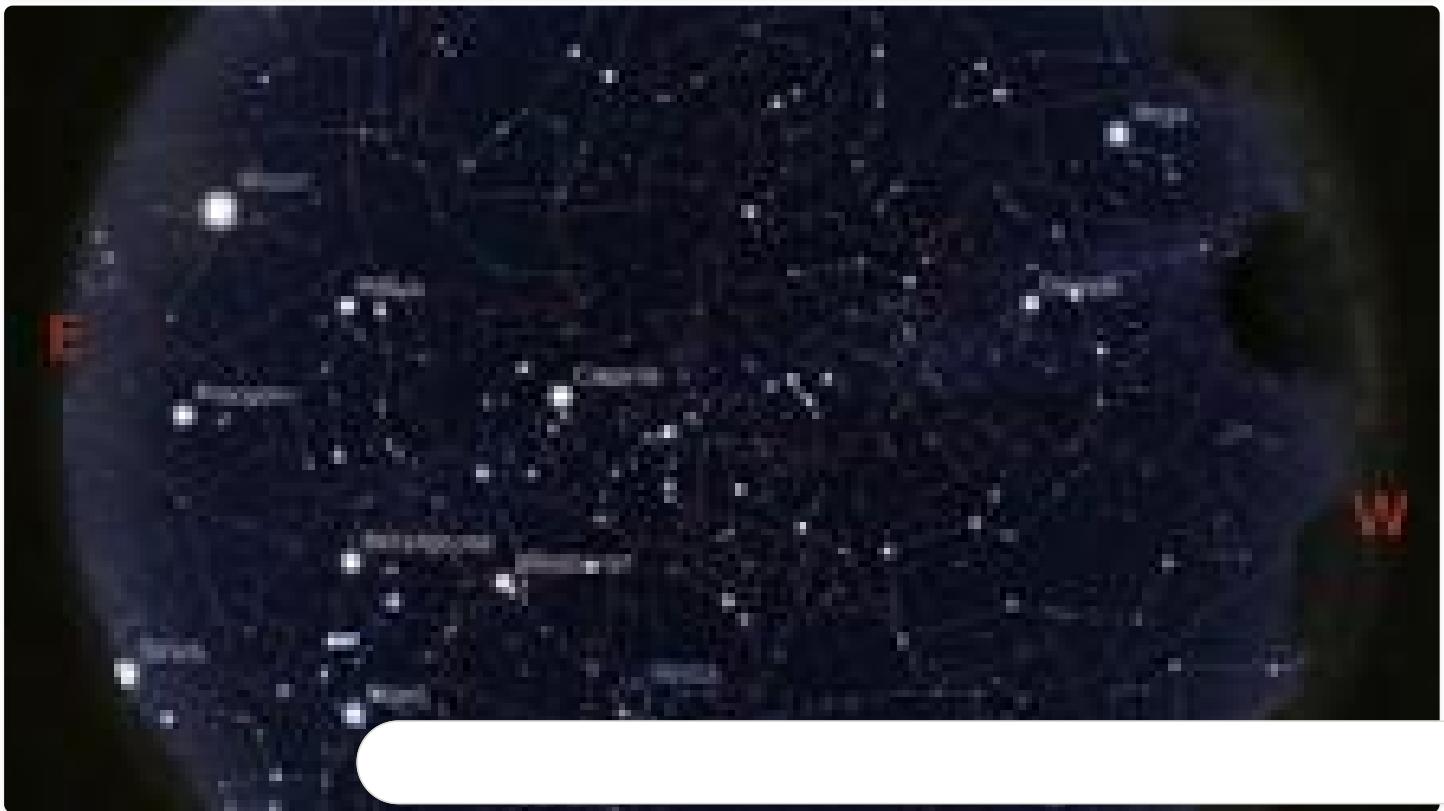
- Incorporate real-world data: Occasionally mix in real sky images (if available) to help the model generalize better to actual conditions.
- Time-based variations: Include images from different times of the year to account for seasonal changes in star positions.
- Multiple wavelengths: If Stellarium supports it, generate images in different wavelengths to mimic multi-spectral imaging capabilities.
- Constellation recognition: Train your model to not just predict location but also identify constellations, which could serve as an intermediate step in the learning process.
- Transfer learning: Start with a pre-trained model on a large dataset of astronomical images and fine-tune it with your Stellarium-generated data.

This approach could potentially create a powerful and flexible model for star pattern recognition and celestial navigation. The iterative nature of the training process, coupled with the vast possibilities for data generation through Stellarium, could lead to a robust and accurate system.

 Share  Rewrite

  ...

CNN implementation to detect constellations in sky

**Name:****Stellarium****Type:****Folder (inode/directory)****Contents:****200,020 items, totalling
42.3 GB****Parent folder:** /home/blech/Pictures**Modified:** Wed 17 Aug 2022 20:01:18**Free space:** 197.6 GB



#1.3 | How to use Stellarium?



I've been thinking. Considering that for this application, we're not exactly navigating, rather searching for unexpected objects in the sky, if we're comparing Stellarium's output to a picture gathered from a webcam (or other type of camera), knowing the position and time exactly, would ?

Sources

Answer

Your idea of comparing Stellarium's output to actual camera images for detecting unexpected objects in the sky is intriguing and could indeed be useful. Here's how a CNN could be beneficial in this scenario:

1. Pattern Matching:

A CNN could learn to recognize the expected star patterns from Stellarium and compare them to the actual images. This would allow for robust matching even with slight variations in camera orientation or atmospheric conditions.

2. Anomaly Detection:

By training on pairs of Stellarium-generated and real images, the CNN could learn to identify discrepancies. Anything present in the real image but not in the Stellarium output could be flagged as a potential object of interest.

3. Noise and Artifact Handling:

CNNs are excellent at handling noise and artifacts in images. This could help in distinguishing between actual celestial objects and camera or atmospheric artifacts.

4. Feature Extraction:

The CNN could learn to extract relevant features from both Stellarium and real images, making the comparison more robust to minor differences in brightness, contrast, or color.

5. Multi-spectral Analysis:

If using cameras sensitive to different wavelengths, a CNN could be trained to compare multi-spectral data, potentially revealing objects not visible in standard visible light.

6. Time-series Analysis:

By incorporating multiple frames over time, a CNN could learn to detect moving objects or transient phenomena that differ from the static Stellarium output.

7. Adaptive Thresholding:

The CNN could learn to apply adaptive thresholding techniques, adjusting for varying sky conditions and light pollution levels across different images.

8. Position and Time Correlation:

Given the exact position and time, the CNN could learn to correlate subtle changes in star positions and brightness with expected variations, flagging unexpected changes.

To implement this approach:

1. Generate a dataset of paired Stellarium and real sky images for various locations, times, and sky conditions.
2. Train a CNN to compare these image pairs, focusing on identifying differences.
3. Use transfer learning from existing astronomical image processing models to improve performance.

 + New

⌚ 4h

...



Share

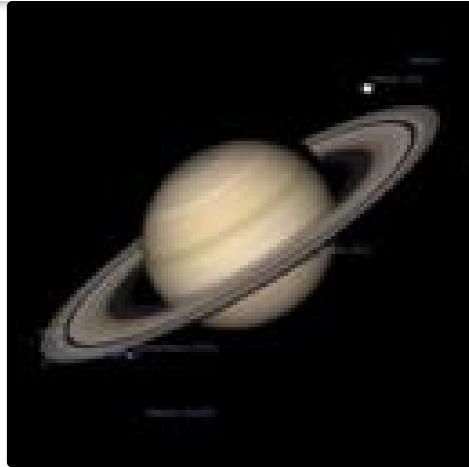
5. Develop a post-processing step to analyze and classify any detected anomalies.

This approach could be particularly effective for citizen science projects or large-scale sky surveys looking for transient phenomena, near-Earth objects, or other unexpected celestial events. The CNN's ability to process large amounts of data quickly could make it a valuable tool for initial screening, allowing human experts to focus on the most promising candidates for further investigation.

 Share  Rewrite

  ...



[View More](#)[Watch](#)

What other models might be useful? For instance, would something like an adversarial model between two CNNs, one around truth from Stellarium, and or

GitHub - gregtozzi/deep_learning_celnav: Toward automating celestial navigation with deep learning and edge inferencing

Real-Time Convolutional Neural Network-Based Star Detection and Centroiding Method for CubeSat Star Tracker

Real-Time Convolutional Neu Centroiding Method for Cube

 github · 1

 arxiv · 2

 arxiv · 3

Answer

Your idea of using an adversarial model with two CNNs is innovative and could potentially yield excellent results. Here's how this approach could be implemented and why it might be beneficial:

1. Generative Adversarial Network (GAN) Structure:

- Generator: CNN trained on webcam images
- Discriminator: CNN trained on Stellarium-generated images

2. Training Process:

The generator would try to produce images that match the Stellarium output, while the discriminator would attempt to distinguish between real Stellarium images and those produced by the generator. This process could help in:

- Reducing noise and artifacts in webcam images
- Enhancing image quality to match Stellarium's clarity
- Learning to compensate for atmospheric effects and light pollution

3. Anomaly Detection:

Once trained, discrepancies between the generator's output and the actual webcam image could indicate the presence of unexpected objects or phenomena.

4. Transfer Learning:

Start with pre-trained models on astronomical datasets to accelerate the learning process and improve generalization.

5. Multi-modal Input:

Incorporate additional data such as time, location, and sensor characteristics into both networks to improve accuracy.

6. Attention Mechanisms:

Implement attention layers to help the model focus on relevant parts of the image, potentially improving detection of faint or small objects.

7. Time Series Analysis:

Extend the model to process sequences of images, allowing for detection of moving objects or transient phenomena.

8. Uncertainty Quantification:

Implement techniques like Monte Carlo Dropout to provide uncertainty estimates for detections, which is crucial for scientific applications.

Other useful models to consider:

1. Siamese Networks:

These could be used to directly compare Stellarium and webcam images, learning a similarity metric that accounts for real-world distortions.

2. Variational Autoencoders (VAEs):

These could help in learning a compact representation of sky images, potentially making it easier to detect anomalies.

3. Transformer-based Models:

While typically used for sequence data, adapters like Vision Transformer (ViT) could be effective for processing sky images, especially when considering temporal sequences.

4. Ensemble Methods:

Combining predictions from multiple model types (e.g., CNNs, GANs, and traditional computer vision techniques) could provide more robust results.

5. Few-shot Learning Models:

These could be particularly useful for identifying new objects.

Implementing an adversarial approach could potentially lead to more robust and accurate detection of unexpected objects. It would allow the model to learn the nuances of real-

world variations and anomalies, improving its overall performance in low-light conditions.

[Share](#) [Rewrite](#) [...](#)

As far as transfer learning, do you know of any public / readily available astronomical CNNs out there?

• Sources

World's largest digital camera will be a 'game-changer' for astronomy

cnn · 1

Deriving star cluster parameters with convolutional neural networks
- I. Age, mass, and size

aanda · 2

Deriving star cluster parameters with convolutional neural networks
- II. Extinction and cluster-based

aanda · 3

❖ Answer

There are indeed several publicly available CNNs and datasets that could be useful for transfer learning in astronomical applications. Here are some notable examples:

1. AstroNet:

Developed by Google and NASA, AstroNet is a CNN trained to identify exoplanets in Kepler data. It could be a good starting point for transfer learning in tasks related to time-series analysis of astronomical data.

2. Galaxy Zoo:

The Galaxy Zoo project has released datasets and models for galaxy morphology classification. These could be valuable for transfer learning in tasks involving galaxy image analysis.

3. LSST (Legacy Survey of Space and Time) Datasets:

While not a CNN itself, the LSST project provides large datasets that have been used to train various models, including CNNs, for tasks like transient detection and classification.

4. Astro-MNIST:

This is an astronomical version of the MNIST dataset, containing images of galaxies, stars, and quasars. It's often used for benchmarking machine learning algorithms in astronomy.

5. StarNet:

This is a neural network for stellar spectral analysis. While not specifically a CNN, it could be useful for transfer learning in spectroscopic tasks.

6. Deep-HiTS:

This is a CNN trained for real-time detection of transients in astronomical images. It could be valuable for transfer learning in similar detection tasks.

7. VGG16 and ResNet pre-trained on astronomical data:

Some researchers have fine-tuned popular CNN architectures like VGG16 and ResNet on astronomical datasets. These models could serve as good starting points for various astronomical image analysis tasks.

8. Radio Galaxy Zoo:

Similar to Galaxy Zoo, but focused on radio astronomy. It provides datasets and models for classifying radio galaxies.

9. Supernova Classification Model

Various CNNs have been developed for supernova classification.

Time-series Classification Challenge (PLASSTIC).

earlier layers and retraining later layers to adapt to your task.

Remember to check the licensing and usage terms for any pre-trained models or datasets you decide to use in your project.

Share Rewrite

Copy Edit ...

How can I access the data from the Vera C. Rubin Observatory for training CNNs

Answer

The Vera C. Rubin Observatory, formerly known as the Large Synoptic Survey Telescope (LSST), is still in its construction and commissioning phase. However, there are ways to access simulated data and some precursor datasets that can be useful for training CNNs. Here's how you can access relevant data:

1. LSST Science Pipelines:

The LSST Science Pipelines software provides tools for processing and analyzing astronomical images. It includes simulated data that can be used for training purposes.

2. LSST Science Platform:

Once operational, this platform will provide access to Rubin Observatory data. Currently, it offers access to precursor datasets and simulations.

3. LSST Dark Energy Science Collaboration (DESC):

DESC provides simulated LSST data, including catalogs and images, which can be used for machine learning projects.

4. LSST Stack Club:

This is a community-driven project that provides tutorials and examples for working with LSST data and software.

5. Rubin Observatory Legacy Survey of Space and Time (LSST) Data Preview:

As the observatory nears operation, data previews are being released to help researchers prepare for the full survey.

To access these resources:

1. Visit the Rubin Observatory website (<https://www.lsst.org/>) and create an account.
2. Explore the Data Management section for information on data access and processing tools.
3. Join relevant working groups or collaborations to gain access to specific datasets.
4. Use the LSST Science Pipelines (<https://pipelines.lsst.io/>) to process and analyze data.
5. Check the LSST Community Forum (<https://community.lsst.org/>) for updates and discussions on data availability.
6. Explore the LSST DESC Data Portal (<https://lsstdesc.org/data-portal/>) for simulated datasets.
7. Consider attending LSST-related workshops or summer schools to learn more about working with the data.

Remember that while the full Rubin Observatory dataset is not yet available, the simulated data and precursor datasets can be valuable for developing and testing CNN models. These resources can help you prepare your algorithms for the wealth of data that will become available once the observatory is fully operational.

As the project progresses, more data and resources will become available, so it's a good idea to stay connected with the Rubin Observatory community for the latest updates and opportunities.

Share Rewrite

Copy Edit ...

Is there a good example of a deep learning model trained on astronomical data?

Stellarium?

Sources

3 axis USB accelerometer from electromake on Tindie

Input Shaping Made easy with a USB accelerometer! - YouTube

Removing IR filter from Logitech C920

 tindie · 1

 youtube · 2

 youtube · 3

Answer

Based on the search results and your query, there are a few USB accelerometer options that could potentially work for your project to get parameters for a Python script controlling Stellarium:

1. 3-axis USB accelerometer from electromake:

This is a compact USB accelerometer based on the ADXL345 IC that can easily connect to a PC, Mac or Raspberry Pi ¹. It streams data in CSV format, making it easy to read in Python. It allows you to control sampling frequency and measurement range via commands.

2. RECOVIB Feel USB Accelerometer:

This is designed for vibration analysis and connects to Android devices or PCs via USB ⁴. It comes with software for real-time vibration data streaming and analysis. While not specifically designed for astronomy, its features could potentially be adapted.

3. KUSBA: Klipper USB Accelerometer:

Though designed for 3D printer calibration, this USB-C connected accelerometer board could potentially be repurposed ⁵. It's open source hardware and comes pre-flashed with firmware.

For integrating with a camera and Stellarium:

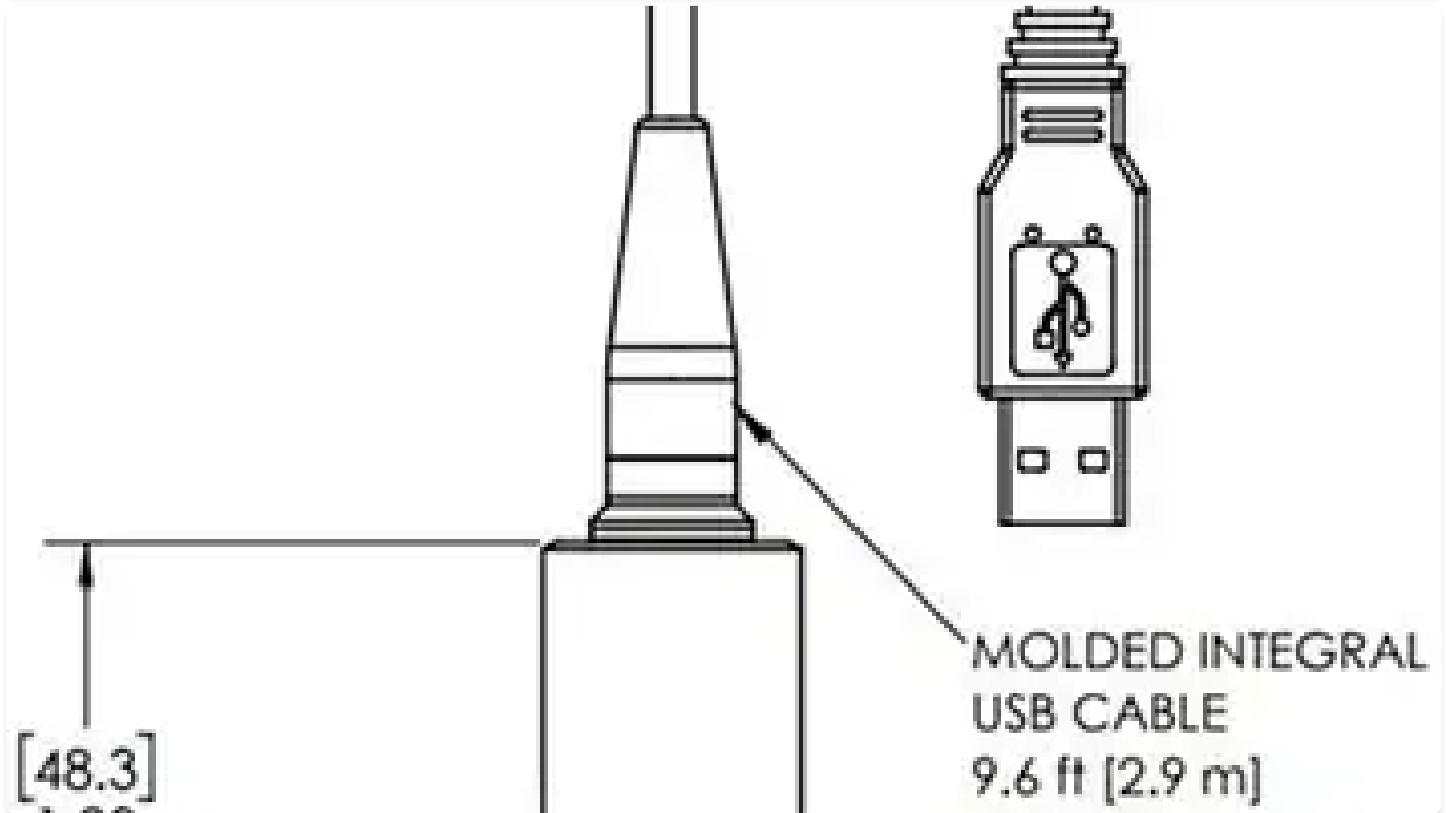
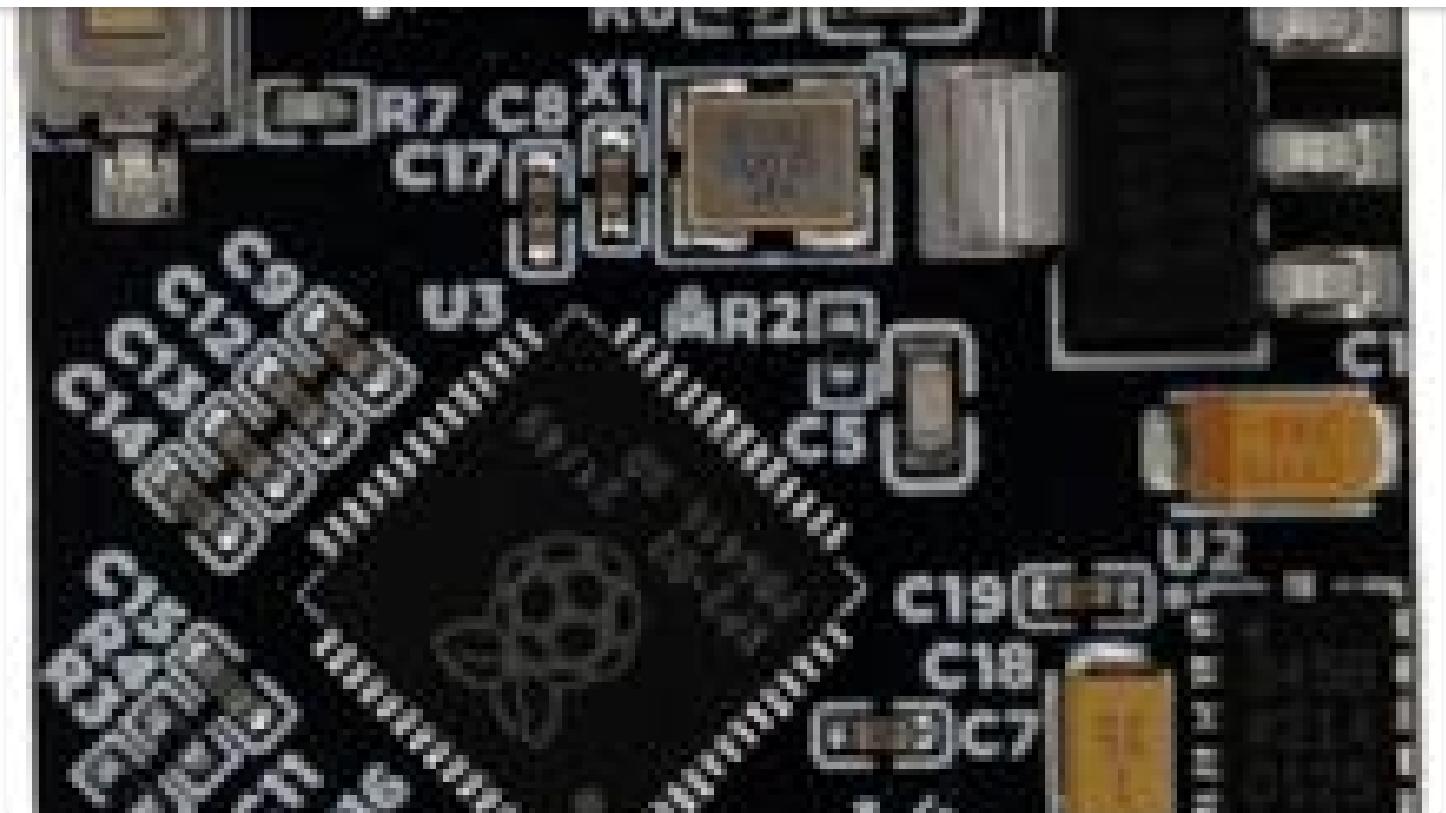
1. You could mount one of these accelerometers to your camera to measure its orientation/movement.
2. Write a Python script to read the accelerometer data via USB.
3. Use the accelerometer data to calculate the camera's pointing direction.
4. Use Stellarium's HTTP API (which you mentioned being familiar with) to update the view based on the calculated camera direction.
5. Correlate the accelerometer data with star positions from Stellarium to refine your calculations.

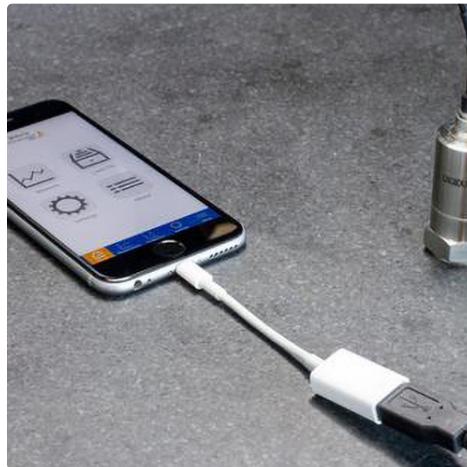
Keep in mind that while these accelerometers aren't specifically designed for astronomical use, their ability to provide orientation data could be useful for your project. You may need to do some custom calibration and data processing to get accurate results for astronomy applications.

 Share  Rewrite

  ...







View More

If you have two 2D point clouds which don't match exactly, is there a model type which would be easiest to use with little training to show where they best fit?

 arxiv · 1 forum.image · 2 answers.opencv · 3

Answer

Based on your query about matching two 2D point clouds that don't align exactly, there are a few approaches that could work well with minimal training:

1. Iterative Closest Point (ICP):

While you mentioned ICP may not be ideal due to noise, it can still be effective with some modifications:

- Use a robust error metric to handle outliers
- Implement a multi-scale ICP approach starting with coarse alignment
- Combine with RANSAC for outlier rejection

2. Feature-based matching:

- Extract keypoints and descriptors (e.g. SIFT, SURF, ORB) from both point clouds
- Match descriptors to find corresponding points
- Use RANSAC to estimate the transformation between matched points