

Alice Kuznetsov
10/21/2025

Contrastive Learning for Enhancing Scene Transfer

Contrastive Learning for Enhancing Robust Scene Transfer in Vision-based Agile Flight

Jiaxu Xing, Leonard Bauersfeld*, Yunlong Song, Chunwei Xing, and Davide Scaramuzza*



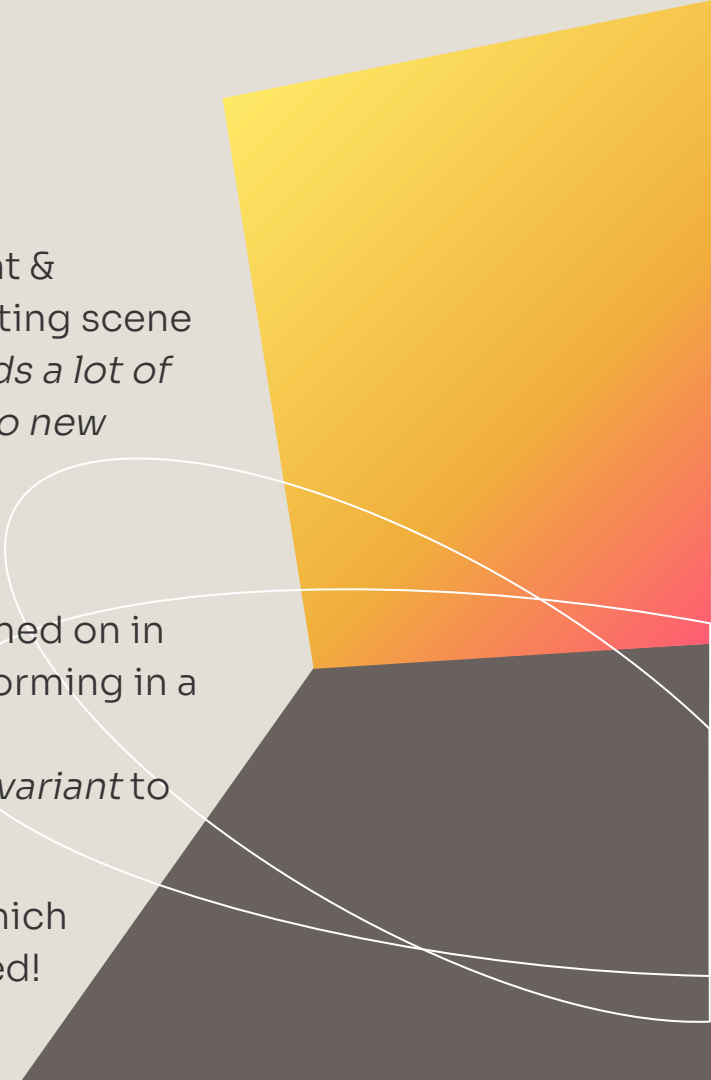
Overview

For vision-based mobile robotics, **Scene Transfer** is a relevant & challenging problem in creating **zero-shot** applications. Existing scene transfer approaches suffer from *poor sample efficiency (needs a lot of data to learn a task)* or *limited generalization (cannot adapt to new conditions)*.

Scene Transfer

- Refers to the ability to perform the same task it was trained on in an environment it was not trained on. Like an actor performing in a different theater!
- Ask: How can we train *task-related perception* that is *invariant* to the environment?

Their solution: **Adaptive Multi-pair Contrastive Learning**, which outperformed current solutions when scenes were transferred!

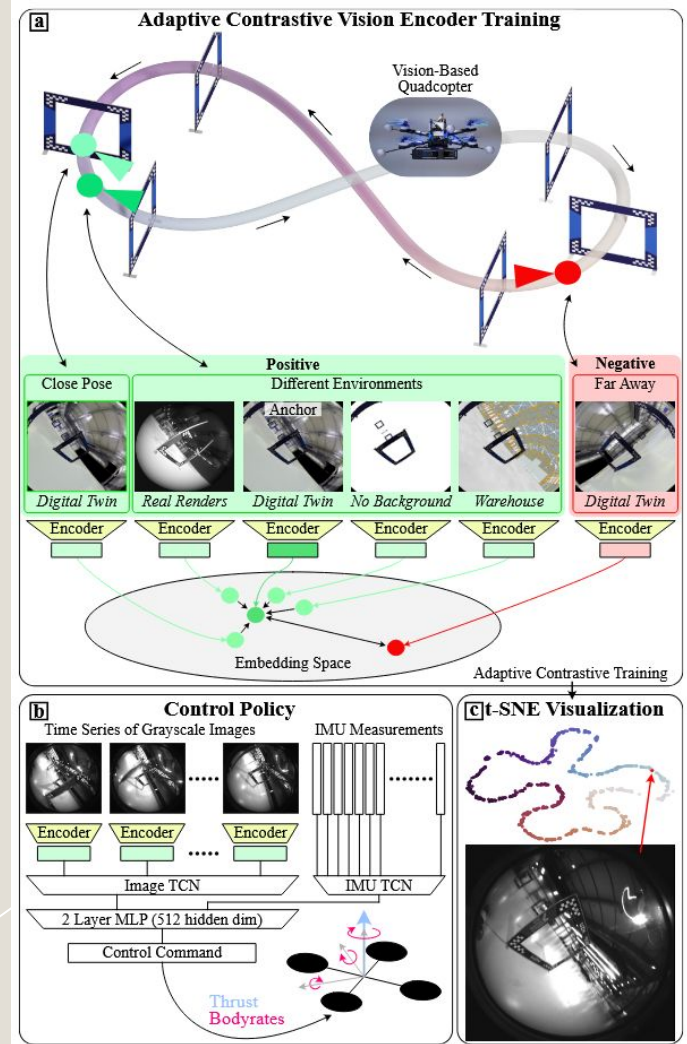


Their Approach

Contrastive Learning – “a Machine Learning paradigm where unlabeled data points are juxtaposed against each other to teach a model which points are similar and which are different.” ([Link](#))

Control Policy – Time Series of grayscale images are fed into *vision encoders* to map the perception into an embedding space. This time series is then combined with IMU measurements to output control commands.

Vision Encoder Training – This is where the contrastive training happens! Parts of their pipeline include randomly changing the scene through contrast & brightness changes, ensuring the same pose does not depend on background. Then, a teacher/student policy was implemented, rewarding the drone for passing gates and penalizing crashes.



Their Approach Cont.

Vision Encoder Training – Similarity from Time

- Trained using 24 negative samples & 12 positive samples, each with normalized track progress.
- Fig. 2–Similarity as a function of track progress difference for different models
- Only theirs considers that if 2 images are samples closely to each other, they should be very similar
- Fig. 3 shows Inter-scene (same scene) consistency, showing how if the robots pos remains unchanged, many models still report high differences.

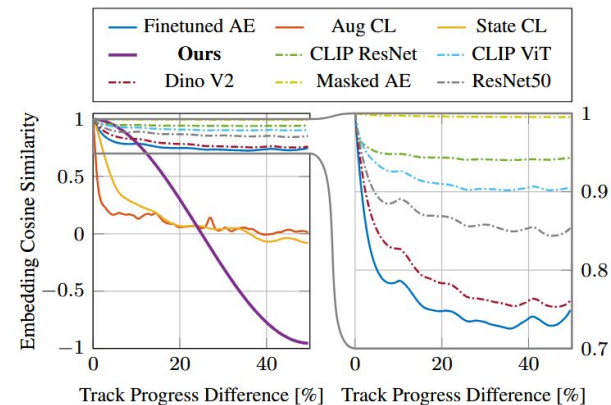
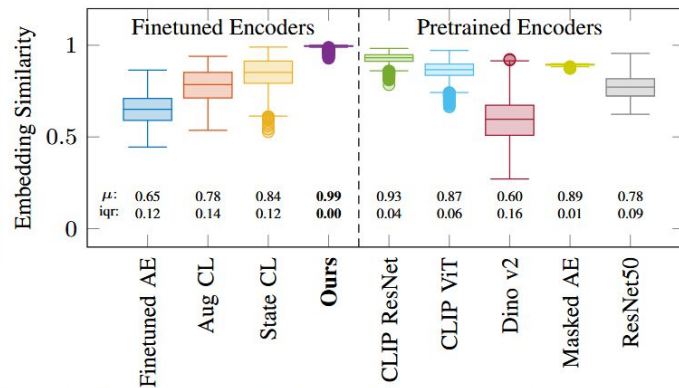


Fig. 2. This figure shows the intra-consistency of the different methods. Solid lines represent methods that have been finetuned and dash-dotted lines represent pretrained methods. One can clearly see that pretrained methods show a much more similar embedding, even at opposite ends of the track (50%). Our proposed method produces an embedding that is very distinctive, as far points on the track have very dissimilar embedding.



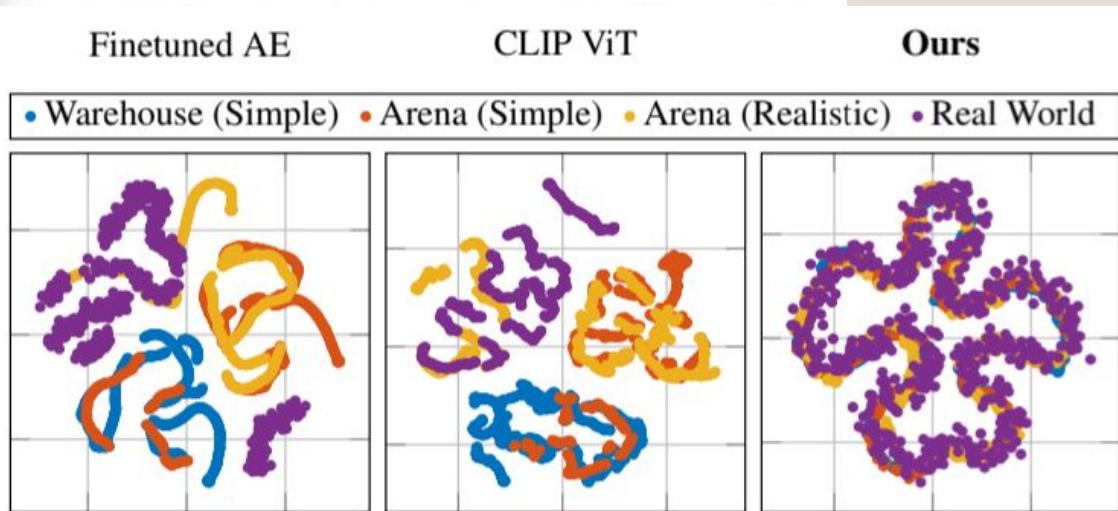
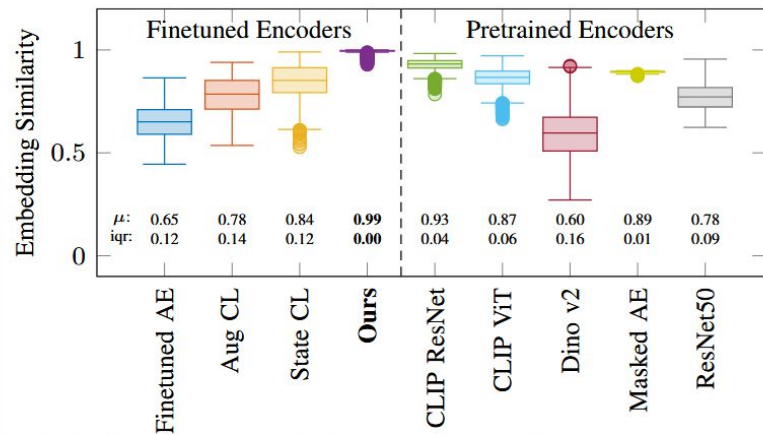


Fig 4. Shows how the ones that reported similar embeddings **just tend to always generate similar embeddings**.

Takeaway: Only their method provided a task-related descriptive embedding while being scene-invariant!

Cool, but does it work? (yes)

“In the real world, all methods degrade significantly, but our method still outperforms the next-best by 25%.”
(Table 1)

Table 2 does not include real-world results, as their onboard computer couldn’t run fast enough to stabilize their vehicle :(

TABLE I

Action Error of the different methods evaluated in four different settings: the training environment, a simple unseen test environment, a highly realistic (unseen) blender render using a digital twin of the real world, and real-world flight data. Our method outperforms all baselines.

Methods	Custom Dataset	AE Train (Simple)	AE Test (Simple)	AE Test (Realistic)	AE Real World
CLIP ResNet	✗	0.016	0.043	0.142	0.116
CLIP ViT [37]	✗	0.019	0.057	0.154	0.115
MAE [38]	✗	0.019	0.063	0.217	0.136
Resnet [60]	✗	0.018	0.071	0.112	0.087
Dino V2 [63]	✗	0.019	0.062	0.124	0.098
Autoencoder	✓	0.015	0.026	0.068	0.088
Aug CL [18]	✓	0.011	0.036	0.091	0.078
State CL [43]	✓	0.009	0.031	0.069	0.073
Ours w/o pose	✓	0.010	0.022	0.041	0.043
Ours	✓	0.008	0.012	0.039	0.036

TABLE II

Success rate (SR), Action Error, and Average Gate Passing (AGP) of the different methods when evaluated in closed-loop operation for the drone racing task. When the control policy is deployed in an environment it has been trained in, all methods perform very well. When deployed in an environment that is unseen during the training of the action net, only our approach is able to perform the task, highlighting its ability to transfer.

Methods	Custom Dataset	Train (Simple)			Test (Simple)			Test (Realistic)		
		SR [%] ↑	Act Err ↓	AGP ↑	SR [%] ↑	Act Err ↓	AGP ↑	SR [%] ↑	Act Err ↓	AGP ↑
CLIP Rn50 [37]	✗	93.8	0.014	9.42±0.20	0.0	0.078	2.85±0.48	0.0	0.194	1.55±0.16
CLIP ViT [37]	✗	95.3	0.015	9.29±0.43	0.0	0.052	2.93±0.29	0.0	0.203	1.46±0.16
MAE [38]	✗	89.1	0.017	9.39±0.28	0.0	0.043	3.02±0.27	0.0	0.210	1.32±0.14
Resnet [60]	✗	95.3	0.013	9.51±0.16	0.0	0.072	2.83±0.46	0.0	0.112	2.08±0.33
Dino V2 [63]	✗	95.3	0.015	9.05±0.19	0.0	0.058	2.98±0.30	0.0	0.153	1.54±0.15
Autoencoder	✓	98.4	0.015	9.31±0.38	0.0	0.136	1.67±0.19	0.0	0.139	1.67±0.19
Aug CL [18]	✓	96.8	0.011	9.51±0.44	0.0	0.091	3.11±0.31	0.0	0.124	1.91±0.21
State CL [43]	✓	98.4	0.010	9.82±0.32	0.0	0.040	3.68±0.32	0.0	0.082	2.45±0.12
Ours w/o pose	✓	96.8	0.013	9.57±0.42	48.4	0.035	5.01±0.23	31.3	0.073	4.43±0.10
Ours	✓	98.4	0.012	9.65±0.20	64.1	0.023	7.42±0.42	43.8	0.047	6.03±0.56

Looking toward the Future

Applications to Mobile Robotics:

- Pose information is not as common for mobile robotics as it is for vision-based agile flight tasks. So, they tested with only time-series data, and they still outperformed baselines and are only 10% worse in terms of accuracy compared to privileged encoder training

For future work, they state the following:

“Future work could include more prior knowledge of the physical world to further enhance policies’ robustness and safety.”

- AKA: Feeding in more data regarding physical constants, such as physics, could enhance the pipeline further.