Final Internship Report;

TouchDesigner & Luxonis Cameras

Noah Kornberg

Overview

Over the course of my internship at the Sensor and Computation Lab at Concordia University, I focused on bridging the gap between emerging machine learning tools and visual programming environments. Specifically, my work involved integrating Mediapipe and Luxonis hardware into TouchDesigner (TD) to create student-friendly tools and learning environments. The core idea behind the project was simple: machine learning, real-time interaction, and creative coding shouldn't be locked behind steep learning curves or expensive equipment.

Introduction to TouchDesigner and Machine Learning

TouchDesigner, developed by Derivative, is a node-based visual programming language used for real-time interactive multimedia content. It's widely used by artists, designers, and researchers to create audiovisual installations, projection mapping, interactive exhibits, and more. While powerful, its learning curve can be steep, especially when attempting to integrate external machine learning models.

Machine learning (ML), on the other hand, is an umbrella term for algorithms that enable computers to learn from and act on data without being explicitly programmed for specific tasks. Tools like PyTorch provide deep learning capabilities, allowing users to train and deploy neural networks for tasks such as pose estimation, object detection, and gesture recognition.

Combining these two fields, TD for real-time visualization and ML for data interpretation, can open the door to highly interactive, intelligent installations. However, there’s currently a lack of clear, accessible examples and documentation for beginners interested in combining them. That’s where this internship came in.

Why Machine Learning in TouchDesigner?

While TD is already an advanced platform for interactivity, ML integration brings additional layers of responsiveness and adaptability. ML tools can recognize and respond to human gestures, facial expressions, and body movements in real-time. These kinds of inputs can be valuable in educational contexts, public installations, performances, and even therapeutic settings.

However, running machine learning pipelines locally, especially those involving video and 3D data, can be computationally expensive. Many students don’t have high-end hardware, so finding a workaround was essential. Luxonis OAK-D cameras offered a practical solution.

These cameras come equipped with on-board processing powered by Intel's Myriad X VPU, allowing them to run computer vision models, like those used in Mediapipe, directly on the

device. This offloads the processing from the student's machine, making ML-powered interactivity more accessible to everyone regardless of their hardware.

Mediapipe vs. PyTorch: Why Simplicity Matters

I initially considered PyTorch for model integration but quickly shifted focus to Mediapipe, developed by Google. PyTorch is a powerful deep learning framework widely used in research and industry for building, training, and deploying custom neural networks. It offers full flexibility for designing complex model architectures and working with large datasets, making it ideal for high-level projects in fields like computer vision, natural language processing, and robotics. However, the flip side of this flexibility is that PyTorch demands a solid foundation in Python, mathematics, and machine learning theory. For someone without prior experience, just setting up an environment, loading a pretrained model, and feeding it data can be an overwhelming task. This level of complexity creates a major barrier for undergraduate students or artists who are more focused on interactive media than on coding or data science.

In contrast, Mediapipe provides a much more accessible entry point for working with machine learning. Developed by Google, it comes with a set of pre-built, real-time ML pipelines designed specifically for tasks like hand tracking, face mesh estimation, pose detection, and object tracking. Rather than requiring users to define model architectures or manually manage inference scripts, Mediapipe handles the heavy lifting internally. The pipelines are modular and can be deployed with minimal setup, often just a few lines of code, making it easy to integrate into interactive software environments. Importantly, TouchDesigner has pre-built integration with Mediapipe, making it quite simple to use in a TouchDesigner environment.

Throughout this project, I kept returning to one core issue: accessibility. TouchDesigner, PyTorch, and even Mediapipe, even though it's relatively user-friendly, often assume a baseline of technical knowledge that many students simply don't have. Online documentation, while often comprehensive, tends to be written for users who already understand the underlying systems. YouTube tutorials and forum posts often skip over the setup steps, fail to explain core concepts, or jump straight into advanced features. This leaves beginners stuck between learning entire programming languages from scratch or trying to reverse-engineer someone else's code without a clear understanding of what each piece does.

My response was to go beyond just creating example patches. I focused on building clear, interactive environments that both demonstrated the integration and acted as learning tools. Each patch not only includes the functioning pipeline, such as face tracking or hand detection, but also includes a 5-minute demo video about the specific TouchDesigner operators used, how external data is brought into the network, and where key parameters can be adjusted. I annotated components, organized the network layout for clarity, and in some cases, added notes directly in the TD interface to guide users through the logic.

The goal was to create a learning experience that feels like play, something that students could open up, experiment with, break, and rebuild. This approach is especially important in the context of creative technology education, where exploration and trial-and-error are vital parts of the process. By lowering the entry barrier and demystifying the integration process, I aimed to make these powerful tools more accessible to a broader range of students, including those without a background in programming or machine learning.

Creating Examples for Students

One of my main goals was to create a repository of simple, modular, and easy-to-understand TD patches that integrate ML features via Mediapipe. These patches are designed for students to explore, remix, and build upon, providing both technical function and a creative playground.

Example 1: Face Mesh Patch

The first example used Mediapipe's face mesh tracking, which outputs 468 facial landmarks in real-time. The TD patch pulls this data either from a regular webcam or directly from the Luxonis camera. It then maps the face's contours onto a visual grid, creating a distorted but responsive facial outline.

This patch was inspired by HOLO by Eric Prydz, a live VJ show that used 3D visuals and face tracking for surreal, immersive effects. The idea here was to show how creative coding could connect to real-world inspirations, helping students feel excited about the possibilities.

Example 2: Hand Tracking and Interactive Grid

The second patch focused on Mediapipe's hand tracking module, which is often the most requested feature among TD learners. In this example, hand position and finger movements affect a field of dynamic lines, like strumming a digital guitar.

This interactive environment was loosely based on a popular short demo video by Alvie Stars, which demonstrated a similar hand-controlled setup using Kinect. The video is widely circulated in TD forums, so many students already recognize the effect. Recreating a version of this within Mediapipe and Luxonis helped show how similar results could be achieved more easily and affordably.

Luxonis and Kinect in TD

The Luxonis OAK-D cameras, particularly the OAK-D Time-of-Flight (TOF) variant, are a major advancement in the field of computer vision. These cameras include multiple RGB and depth sensors and are powered by an onboard Myriad X VPU, which allows machine learning models to run directly on the device without relying on the host computer. This is especially valuable for

students or creators working with limited hardware, as the camera itself handles intensive operations like spatial AI, object tracking, and gesture recognition. It effectively decentralizes the workload, making real-time ML applications more feasible on lower-end systems.

Compared to earlier consumer-grade computer vision tools like the Microsoft Kinect, the OAK-D cameras offer a much higher degree of customization. While Kinect devices were user-friendly and had wide community support, Microsoft has since discontinued them, and their SDKs are no longer actively updated. Kinect still has a strong legacy in interactive media art, but it lacks the flexibility and forward compatibility that newer hardware like Luxonis provides. The OAK-D can be programmed with custom neural networks, supports a wide range of depth sensing modes, and integrates with open-source tools like DepthAI and OpenCV.

That said, the OAK-D cameras are not without their challenges. They are relatively new on the market, and documentation is still catching up to their full range of capabilities. TouchDesigner has introduced a set of components for Luxonis integration, but these are outdated and often buggy. In practice, I encountered multiple issues, including TouchDesigner crashing when attempting to connect with the camera or process its data. To get things working properly, I had to go into the pre-built components and manually rewrite parts of the scripts. The lack of stable documentation or consistent community support makes it difficult for beginners to work with these tools. There's clearly a need for more resources and best practices as AI and creative coding continue to intersect.

One of the more advanced features I worked on during my internship was developing a point cloud visualization using the Luxonis camera. Point clouds are a popular feature in TouchDesigner, particularly for creating 3D environments or reactive visuals based on real-world input. Traditionally, point clouds in creative coding are generated using devices like the Kinect, which project infrared dots into a scene and triangulate depth directly. Luxonis, however, uses a different approach. Rather than projecting structured light, it creates a depth map using stereo vision or time-of-flight sensing, then converts that data into a point cloud using software.

This workflow presented several complications. Unlike the Kinect, which has well-documented and widely supported TouchDesigner integrations for point clouds, the Luxonis approach requires additional processing. The camera provides depth data, but that data must be mapped to real-world coordinates and reshaped into a point cloud format. I had to rewrite and adapt parts of the integration scripts to access and correctly format this data inside TouchDesigner. It was the most technically difficult part of the project, but also the most rewarding. Point clouds allow for a level of spatial interactivity that can be expanded in endless directions—virtual environments, motion-based effects, or volumetric visualization. Including this feature felt essential, both because of its potential and its relevance to how I personally use TD in my own work.

Conclusion

Looking ahead, one area I was particularly interested in exploring further was SLAM, or Simultaneous Localization and Mapping. SLAM is a method used in computer vision and robotics to build a map of an unknown environment while simultaneously tracking the device's location within it. For interactive installations and mixed-reality applications, SLAM can enable spatial awareness, allowing a system to respond not just to the user's gestures or movements, but to their position in a physical space. Incorporating SLAM into TouchDesigner through the OAK-D camera could open up new possibilities for immersive experiences, such as room-scale AR or site-specific projection mapping that reacts to movement in real-time. I also hoped to begin experimenting with uploading custom machine learning models directly onto the OAK-D hardware. This would allow for tailored applications beyond the standard Mediapipe features, such as gesture-specific recognition or more complex pose classifications. However, both of these paths require more time, deeper technical understanding, and more stable tooling around the camera and its SDK.

This internship gave me a hands-on opportunity to explore the intersection of creative coding and real-time machine learning. By working with TouchDesigner and integrating Luxonis OAK-D cameras through Mediapipe, I was able to build examples that prioritize accessibility and usability for students. I encountered a number of technical challenges, outdated components, limited documentation, and system crashes, but these also highlighted the gaps that still exist between high-potential tools and their ease of use in educational settings. My work focused on creating a bridge: building patches that lower the barrier to entry, while showing how machine learning and interactive design can come together. There's still a lot of room for growth, especially in areas like SLAM and custom model deployment, but I hope this project serves as a helpful starting point for students and creators looking to explore this space.

Relevant Articles & Links:

https://docs.luxonis.com/software/

https://derivative.ca/UserGuide/OAK-D

https://github.com/luxonis/depthai

https://docs.luxonis.com/software/depthai/examples/

https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-computer-vision#object-classification

https://pytorch.org/

https://docs.luxonis.com/software/depthai-components/nodes/tof/

A note about the Sensor and Computation Lab;

The Sensor and Computation Lab at Concordia has been more than just a workspace, it's been foundational to my university experience. The insights and progress I've made during this internship wouldn't have been possible without the Lab's support, and truthfully, neither would a lot of the friendships, projects, and confidence I've built over the years.

Elio and Sabine have created something rare: a space where students don't just learn, they thrive. It's not just a lab. It's a community hub where students meet, collaborate, and share ideas. Whether it's through casual conversations, troubleshooting tech issues, or brainstorming wild new project concepts (Which are then shot down by Elio), the Lab is a space where curiosity is welcomed and creativity is supported.

More than anything, it's a place where students can come for one-on-one help, the kind of guidance that's often missing in larger university settings. But beyond the academic and technical support, it's a space where we've laughed, vented, cried a little, and supported each other through university life. It is a family.

As I finish my degree and get ready to move on to the next chapter, I just want to say thank you. Elio and Sabine, you've created an environment where I've been able to grow into who I am today. Your impact on me, and on so many others, runs deep. You've truly changed my life.

Thank you.

Noah