RetroActivity: Rapidly Deployable Live Task Guidance Experiences

Andrey Konin, Shahram Najam Syed, Shakeeb Siddiqui, Sateesh Kumar, Quoc-Huy Tran, M. Zeeshan Zia
Retrocausal Inc.

ABSTRACT

RetroActivity is a system for building computational models of a goal-driven physical task, that learns from a handful of video demonstrations of the task, without the need for any text-based computer programming effort. Once such a model is built, RetroActivity analyzes live demonstrations of the activity and guides a hands-on worker through the task, providing feedback when a mistake is made.

Keywords: Activity Understanding, Frontline workers, Human understanding.

1 INTRODUCTION

We present a system for automatically building computational models of a complex physical task, such as a medical procedure or a manufacturing assembly task performed by human workers, given only a handful of recorded correct demonstrations of the task. Once such a model is built, our system can finely analyse the same task being performed in live video, to provide measurements and analytics, improve efficiency, guide a hands-on worker through the task, or provide just-in-time training.

The illustration in Figure 1 shows such a system in action where a worker performs a manufacturing assembly task. Our system “watches” the procedure in real-time, and guides the worker step-by-step, while proactively warning for any mistakes.

Existing AI platforms offer capabilities to estimate human poses, locate objects, and even classify or segment simple actions in video (such as “take” or “twist”). Similarly, human poses or 3D objects can also be estimated using sensors such as an electro-magnetic transducer or a Motion Capture setup. However, in order to understand a certain complex activity with multiple conditional steps (e.g. a medical procedure), one still needs a team of PhD-level computer vision or IoT engineers, who write customized code to represent that specific activity, relating human pose changes with object movements over time and build this temporal causation graph on top of the visual recognition capabilities provided by existing platforms. From our experience, an average physical procedure comprising of 20 steps [1][2] needs 6 person-months to construct such a customized model, in addition to requiring 100s of recorded demonstrations to train machine learning models on. Yet another representative challenge for applications such as manufacturing, is the need to accommodate a large set of customizations in each workstation-level activity.

RetroActivity allows building compositional video understanding pipelines, which can readily adapt to such dynamic customization needs. Customers can build such complex quality control and guidance experiences within hours instead of months, and with 10s instead of 100s of training data points with our solution.

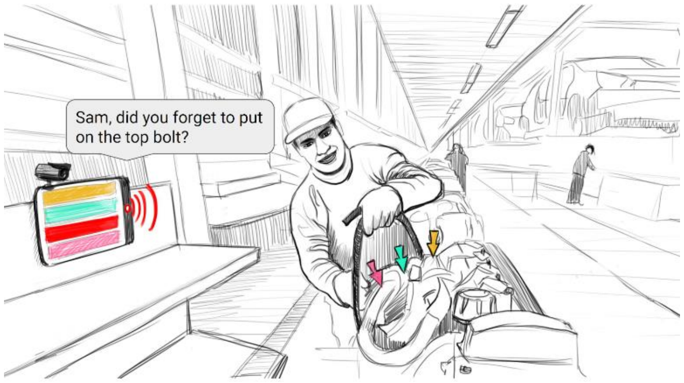

Figure 1: RetroActivity helps a worker avoid assembly mistakes.

2 SYSTEM

We present an overall system that combines task learning, task evaluation, human annotation, and continuous adaptation for understanding human activities in video data.

The two advantages of our approach include:

1. We bring the time to build live human-centered task guidance and quality assurance experiences from several weeks or months to hours.
2. We allow building compositional activity models by setting up “questions” that the live video performance will be evaluated on. We call these questions our “query operations”, which can be specified in an easy to use visual interface.

2.1 Setup

Our software lives in a high-end tablet computer, which is connected to one or more cameras by a bluetooth link. We expect an industrial engineer or an IT staff member (in the case of manufacturing use case) to mount the camera(s) using easy-to-use flexible mounts that we provide, such that they have line-of-sight to the activity with a resolution sufficient to capture the objects of interest in detail. The person setting up the system records about 25 correct demonstration videos of the activity in question, specifies a name for each step (Standard Operating Procedures, SOP) and labels at least one out of the 25 videos. We depict the overall process in Figure 2 and the part of the user interface in Figure 3.

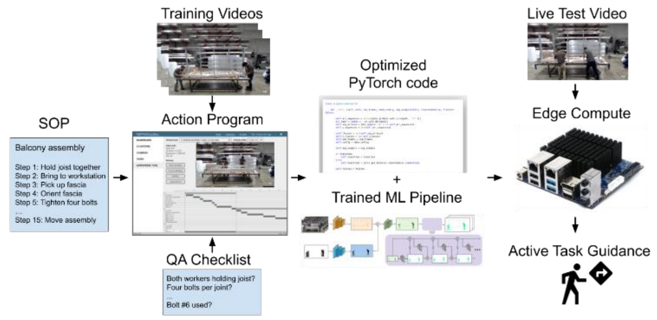

Figure 2: An Action Program visually represents the sub-actions in the activity being modeled. Our system constructs multi-module ML pipelines, trains them end-to-end, and optimizes them to run on edge compute hardware.

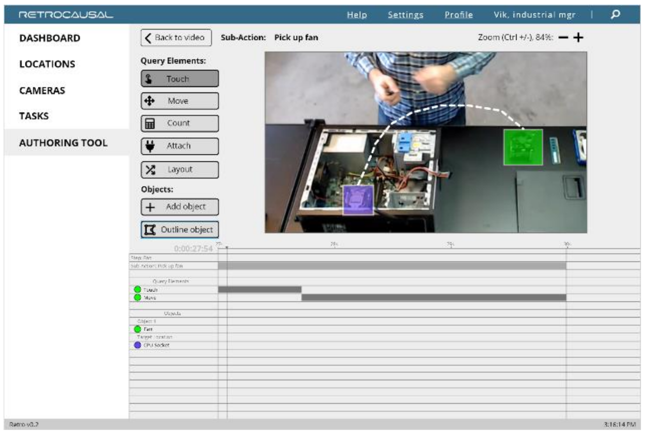

Figure 3: Compositional Action Recognition. User specifies that the "CPU cooling fan" must be attached over the processor, to properly complete "pick up fan" sub-action.

2.2 Video Understanding Model

We use a semantic video alignment module that transfers labels from the one or few labeled videos to the entire training set. Here we draw on ideas from Differentiable Dynamic Time Warping and Temporal Cycle Consistency Learning [3] literature. Next, we generate a significantly larger training set by augmenting the given videos in several ways: we use ideas from deep fake video generation to transfer person appearance [4], perform background reconstruction to augment background appearance, add or delete sub-steps in a controlled manner in the provided videos. This provides us with enough variations to train a video clip classification model [5]. We visualize this in the upper half of Figure 4.

2.3 Visual Query Operations on Live Video

Our querying interface is built on top of action localization in videos. Our system trains a machine learning model to detect individual action steps in real-time. We provide the user with "*query operations*" that can be utilized to set up questions that will be asked of the video during live performance of the activity (Figure 3).

Here is the set of *query operations* that we provide in RetroActivity:

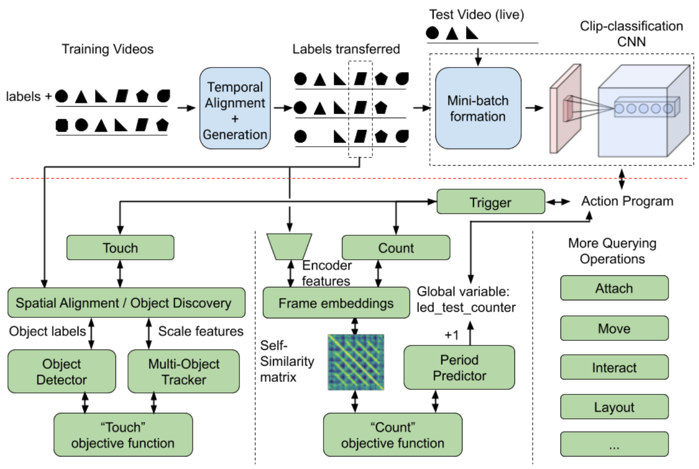

Figure 4: Reconfigurable multi-module vision pipeline represented by an action program defined on a set of activity videos.

1. Trigger: A *trigger* transfers control flow from the action classification network (CNN) to any of the rest of the operations.
2. Touch: Did a specified tool or hand touch specific points on a 3D object?
3. Attach: Were specific parts placed on specific regions of an object?
4. Move: Did the person move an object in 3D?
5. Count: Count the number of times a certain (sub-)action is performed.
6. Match: Match an object against a template image.
7. Permute: Specify which steps in the activity can be performed out-of-order.
8. Human interaction: Confirm that two workers are holding two sides on an object simultaneously.
9. Handed-ness: Specify invariance to handedness (left versus right).
10. Query workspace Layout: Allow manually specifying a 3D coarse layout for the scene, which the analytics engine will populate and compare against a dataset of alternatives for continuous improvement. E.g. alternative table top layouts to improve process efficiency.

We visualize how query operations interact with the action recognition model through "trigger" points in Figure 4. The components described above allow a user to build re-configurable visual task guidance experiences without any manual programming effort for a variety of use cases.

REFERENCES

[1] Roodaki et al., A Surgical Guidance System for Big-Bubble Deep Anterior Lamellar Keratoplasty. *MICCAI 2016*.

[2] Rice et al . Comparing Three Task Guidance Interfaces for Wire Harness Assembly. CHI EA 2016.

[3] Dwibedi et al. Temporal Cycle-Consistency Learning. CVPR 2019

[4] Chan et al. Everybody Dance Now. ICCV 2019

[5] Tran et al. Learning Spatiotemporal Features with 3D Convolutional Networks. ICCV 2015.