
//ALEGION
Alegion
Building A Video Annotation Platform

Building a Video Annotation Platform

Introduction

Major advances in computer vision (CV) are common and frequent compared to other fields; research is growing **exponentially**. Video is clearly the fastest growing data type in the CV space. It's hard to comprehend the variety of use cases that data scientists have found for video analytics and processing.

About a year ago we took on the challenge of video annotation (VA). We saw that the tools in the market weren't up to snuff, and in response, developed and launched Alegion Control, our next generation annotation platform built explicitly to simplify and accelerate video annotation. It boasts an amazing U/X with unmatched support for video length, annotation density, and ontology complexity without sacrificing performance.

As a follow-up to the **piece on our Design thinking** for Alegion Control, we thought we'd share some of the hard problems we've had to solve on the engineering side of the house as well.

This piece examines the challenges of building a super-responsive user experience for video annotation, one that must operate across numerous annotators with potentially unstable internet connections, and scale on the back-end.

TABLE OF CONTENTS

The Transition from Image to Video Annotation

Most video annotation solutions were built upon image annotation solutions. Early implementations processed videos as a sequence of images and used image annotation techniques. An incoming video was expanded to a sequence of frames and distributed, usually in parallel, to be annotated. Afterwards, the annotations were zipped back up based on the frame number. This approach worked...until it didn't.

In recent years, we've seen perception models move from object detection to object tracking. This involves identifying individual instances of objects and giving them a unique identifier that remains constant over time (car_7, player_3, hotdog_18) what we call "entity persistence." You can imagine the challenges around "localizing", drawing a bounding box or placing a keypoint, all of the players in a 1000-frame hockey game clip. Afterwards, because there were twenty annotators working in parallel, you'd have to figure out that Chip's player_3 is the same as Kenneth's player_17… for every frame. Sounds like the worst game of Telephone, ever.

Video also gives us the temporal context to identify events (periods of time) in which an object is in different states, e.g. understanding the periods when a car is accelerating, decelerating, parking, or parked. This temporal information is needed to build semantic scene understanding and event recognition solutions.

Long story short, video needs to be treated as a first-class data type.

What Is So Hard about Video as a Data Type?

More data

Video assets are simply larger and more information dense than images. Copying, processing, viewing, and storing all become much more expensive operations. It requires a ton of work to make a video annotation tool feel as responsive as an image annotation tool.

Other video annotation platforms store all the frame data in the browser. Consequently users would experience their browsers crashing and lose their unsaved work, an unacceptable and expensive problem.

Number of annotations

As noted above, the most difficult part of video annotation is the amount of labeled data customers need. Typically, every single frame of the video is labeled with multiple localizations. Most customer videos are thousands of frames in length, so memory and performance problems are easy to encounter.

Imagine building a perception model based on analysing hockey games. A two minute hockey video captured at 60 frames per second could easily have 100K+ annotations.

Players need to be tracked, each with a localization (bounding box, keypoint, etc.). Each player has multiple labels, some that are static (player number, team), some that change over time (in-frame/out-of-frame, offense/defense). The puck's relationship attribute needs to be continually updated to reflect which player is in current possession of it. Additionally, scene classifications are captured in order to provide semantic richness. For example, these scene classifications can feed features that detect the state of play (regular play vs. time-out vs. foul).

Video variables

There are many, many factors that affect the quality of annotations that are rooted in the quality and characteristics of the original video data to be annotated. Because annotations need to be accurate, it's paramount to have a robust video processing pipeline that accounts for all of the variables without adding unnecessary burdens on customers.

Solving for Seven Customer Needs

Early on in the process of building this video annotation platform, we articulated the end goals. We wanted to create a solution that enabled:

1. Long videos - minimum of an hour
2. High resolution video – minimum of 4K
3. Large number of annotations – 500K minimum
4. Responsive UI – no perceptible degradation regardless of video length or annotation count
5. Efficient UI – great U/X even with 1000s of objects
6. No video pre-processing – support what the browser supports while detecting and managing underlaying video issues
7. Scalability – support > 1K simultaneous annotators without performance degradation

All of these goals addressed clear pain points we were seeing in the marketplace and hearing about from our customers. We knew that if we could meet these goals for VA, the benefits would be tremendous for all our customers, across use cases and verticals.

An intense year of experimentation worked, and we met our goals. When we launched Alegion Control, our final solution knocked it out of the park. Let's dive into the key problems we were dealing with, how we solved them, and what our platform now offers customers.

Solving for Seven Customer Needs

Front-end problems and solutions

UX/UI and high speed rendering

In addition to looking and feeling great, the annotation experience needs to be the same regardless of the complexity of the use case. Alegion's Head of Design covered much of the design process involved in the annotation U/X in a separate article. One of the biggest challenges here is to create a smooth playback and review experience.

Problem

Initially we tried overlaying SVGs on top of an HTML5 video element. This approach made development easy, but performance slowed down with lots of annotations on the screen at once. During use, the painting of the annotations caused video to lag, creating a jerky playback and scrubbing experience. This was especially fatiguing for quality control (QC) work or reviewing an annotated video.

Solution

In our final iteration, regardless of the count (tested to 1K), we are always able to paint all annotations in under 10ms. As you can see below, this allows us to have a very smooth playback experience up to 100fps. This solution helped us meet goals 3, 4, and 5.

How we solved

The basic explanation is that we experimented with different JS libraries. We clearly needed a new approach so we investigated three different options, ultimately selecting one that gave us WebGL performance, played nice with our React/Redux front-end, used a familiar syntax (TSX), and supported our preferred approach to testing.

Solving for Seven Customer Needs

Memory management

Problem

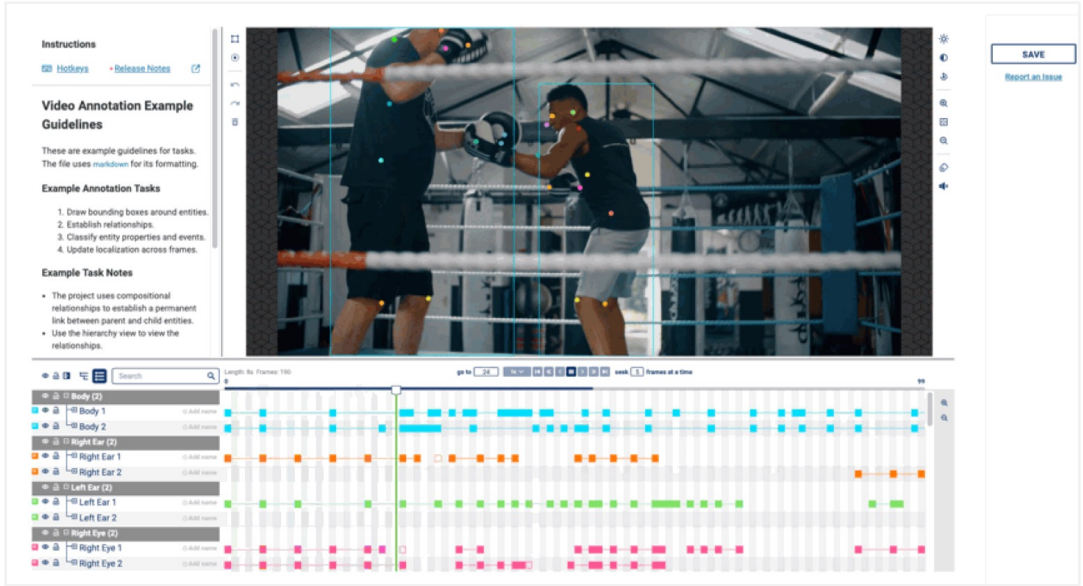
Many solutions, including our naive implementation, simply store all the annotations for every frame in memory. It's straightforward, but memory usage scales with the annotation count and clip length leading to unpredictable browser performance or even losing hours of work due to a browser crash.

Solution

We figured out how to keep the UI responsive while enabling support for long videos and very high annotation counts.This helped us meet goals 1-6.

How we solved

Without going into too much detail, we solved this problem by taking a functional approach to storing annotation data, essentially just storing keyframes. The addition of a caching layer between the browser and back-end allows us to minimize the amount of annotation data we have to keep in the browser and ensures that data is always being pushed to persistent storage.



Solving for Seven Customer Needs

Back-end problems and solutions

Stable and scalable infrastructure

Problem

We needed to figure out how to support >1K simultaneous annotators without performance degradation or work replication. Users are often not able to label an entire video in one session and needed an easy way to query the state of all previous sessions by all annotators, including themselves.

Solution

We built a new microservice for video annotation whose primary purpose is to store and retrieve keyframes for users as they are annotating videos. This dynamic scalability allows us to handle thousands of simultaneous annotators. Our implementation allows us to easily query the state of each video annotation session while including any previous sessions performed for the same video. This solution helped us meet goal 7.

How we solved

In the back-end, we use dedicated video annotation pods in our Kubernetes clusters to allow us to scale up and down based on utilization. The new microservice we built stores and retrieves keyframes. We chose Amazon Aurora for our database which gives us great scalability in addition to being based on the ubiquitous Postgres. We broke up the data logically for each “iteration” that is performed by one or more users.

Solving for Seven Customer Needs

Final output generation

Problem

If you recall, we switched the front-end to use a functional approach for determining the values for each keyframe (see section on Memory Management). This works great during the annotation process but we still needed to generate a final output with the values for every frame of the video. We also wanted to guarantee that outputs matched the same results as the browser showed the users.

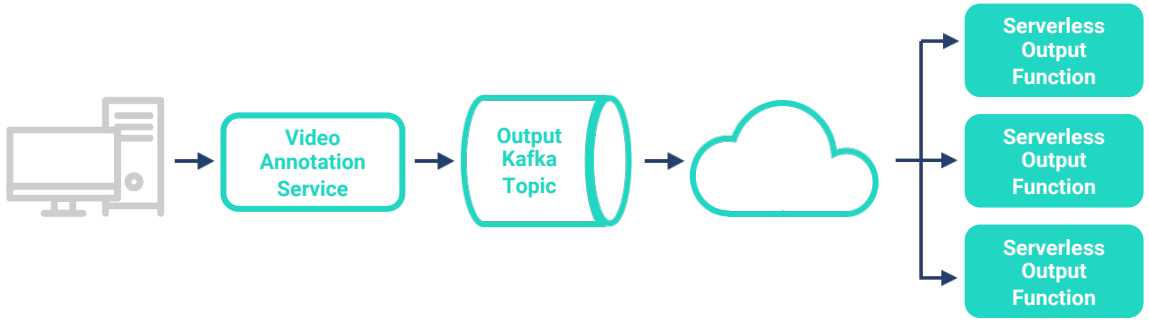
Solution

To accomplish this we decided to move the generation of the output to a OpenFaaS serverless solution. The design allows us to publish to a Kafka topic after submission takes place which asynchronously generates the full output for the video. This solution helped us meet goals 1-7.

How we solved

This design scales very well because OpenFaaS manages a queue of messages for a Kafka Topic. It ensures the cluster does not get overloaded by making sure tasks flow out of the queue in a sustainable manner.

Because OpenFaaS offers language flexibility, it allows us to share JavaScript code from the front-end using a Node.js serverless function. Importantly, this guarantees the same results as the browser showed the users. Sharing front-end and back-end code when possible allows us to maximize code, and equally important, test reuse.



Flow of Data through for Alegion Video Annotation

Solving for Common Challenges with Video Data

Creating a video-based computer vision (CV) application is hard. Video annotation is hard. After spending countless amounts of time and money, data science teams often find that errors or corruption in their ground truth video data has created a spiral of problems throughout their model training process.

When we were building Alegion Control, we knew we needed to help CV teams validate video quality first, in order to deliver the high quality, fast, accurate annotation data they need.

Video data errors and annotation

All video data goes through a process of compression and uncompression as it moves from collection point to playback; this basic process is called encoding and decoding.

The majority of our customers rely on streaming IP cameras to power their solutions. These cameras are small, low cost, and only require a network connection to stream video to a centralized location.

The objective function of these IP cameras is to ensure continuous streaming even under poor network conditions. When a network bandwidth issue or hiccup occurs, some encoding systems will duplicate frames to maintain a constant frame rate while others will drop frames or reduce the encoding bitrate.

Solving for Common Challenges with Video Data

Meanwhile, decoders can choose to handle underlying issues with the stream differently. For example, playing back a clip missing an I-frame may look different in Chrome than it does in Safari due to a difference in the decoders used by different browsers.

This tension between the encoding standard and the multiplicity of decoding options creates big problems when it comes to data labeling. Unless you are one of the few with pristine video sources, understanding the underlying encoding and decoding process for your processing pipeline is key to ensuring that your labels match up with your annotations.

How to cope with degraded video data

Here, I'm exploring two of the most common issues with video data and how decoders (and subsequently annotators) cope with them.

Missing frames

Missing frames are commonly caused by lost packets due to network interruptions while streaming from the camera to the centralized collection point.

When encountering missing frames, the decoder may duplicate previous frames, or show black frames to maintain a constant framerate (thus temporarily adding new content that did not exist in the source video), or simply skip ahead to the next frame.

If frames are duplicated, this causes problems during labeling because the annotator is working on a frame that doesn't exist in the original video and is not preserved after the annotation session. This is most problematic when you get further into the dataset serialization process, and you end up with more annotated frame data than actual frames. This scenario will cause annotations to be out of sync with the image data which in the end has the same negative effect as really inaccurate annotations.

Solving for Common Challenges with Video Data

Fixing missing frames

Once detected (usually by using presentation timestamp (PTS) metadata), fixing individual clips requires a straightforward re-encoding step. After re-encoding, you'll still be missing frames, but the video will be internally consistent and the number (and order) of annotated frames will match the new re-encoded reference video. The video is essentially "re-striped" meaning every frame will have a new pts_time with the expected interval.

Missing frames aren't a big deal as long as you are able to identify when they occur and re-stripe the video.

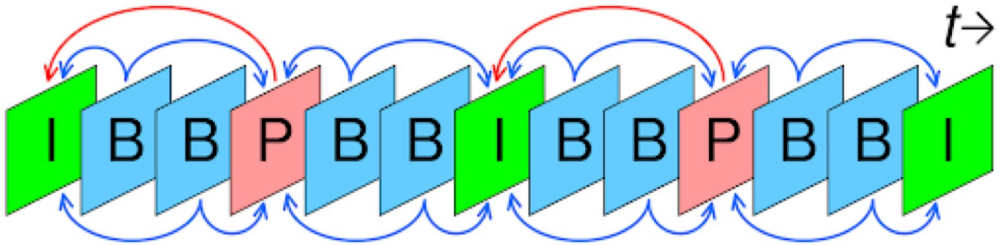
Compression artifacts

Lossy compression algorithms typically rely on a GOP (group of pictures) mechanism in order to reduce the data required to represent the video. There are three major picture types in video compression:

1. I-frames (Intra-coded picture): higher fidelity I-frames can essentially stand alone as "full" frames, what we would think of as a complete image.

2. P-frames (Predicted picture): P-frames store interpolated data from the previous I or P-frames, each new P-frame is essentially just the changes in the image from the previous frame.

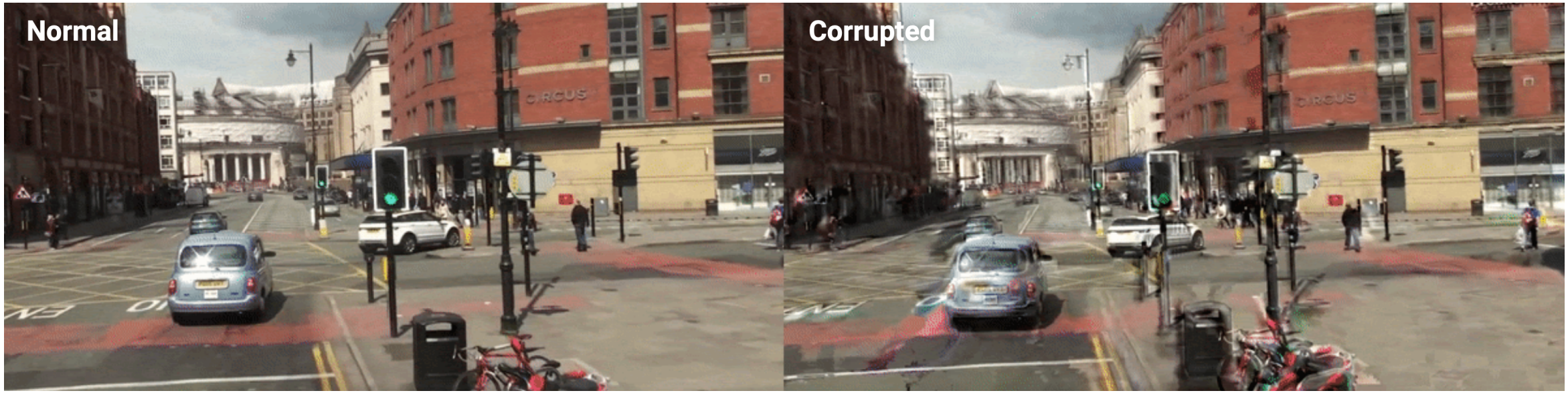
3. B-frames (Bidirectional predicted picture): B-frames also use intra-frame compression, but are informed by both previous and future I-frames and P-frames.



Solving for Common Challenges with Video Data

Problems with compression artifacts

The important thing to understand is that an error in one frame can cause artifacts in the entire GOP, essentially the frames between I-frame boundaries. Unfortunately, it's very common to see portions of a clip contain encoder induced compression artifacts that are attributable to corruption instead of bitrate or color-depth-induced quantization.



Normal video on the left side, visible P-frame corruption on the right side. Observe color and shape irregularities, increased posterization. (Ignore the lack of perfect synchronization between sides for this example.)

Most VA Platforms Handle Issues By Forcing Pre-processing. Alegion Doesn't.

Many competing VA platforms force a pre-processing step for video data in order to address issues with ground truth video files.

This forced pre-processing step is problematic for three big reasons:

1 **It's time-consuming.**

Choosing to treat the video as sequences of images or forcing a pre-processing step adds a HUGE amount of time, complexity, and compute overhead when working at production scale.

2 **You lose quality.**

Any time a video is re-encoded (transcoded), there will be a loss in quality. Additionally, most customers want to control the encoding profile, and many competing video annotation systems do not provide this flexibility.

3 **You risk mismatched data.**

On other systems, if your videos are pre-processed by their system and you are not given a new reference video along with your labels, you are not guaranteed that the annotations will match up with your original videos.

Alegion provides a video validation tool that can be integrated into your processing pipeline that not only identifies these issues (and many more) but handles the creation of specific remediation commands to ensure your video is pristine before beginning annotation.

Landing “Generations Ahead” of the VA Competition

We launched Alegion Control, our self-service video annotation offering in November 2020 to great acclaim. One market response described Control as “generations ahead” of the competition.

The launch of Alegion Control was informed by over a year of learnings built upon real world customer engagements. Our ability to produce solutions that advance the field is the result of Alegion’s company culture and our commitment to solve for our customers, both for their present needs and in anticipation of future needs.

I hope this deep dive has provided you with some insight into the technical and engineering challenges that come with building a world class, scalable, and highly responsive video annotation solution.

We're proud to work with some of the biggest innovators in the space as they build their next-generation video-based vision platforms. If you have questions or want to learn more, please reach out to solutions@alegion.com and get in touch.

As we say in Texas, “we'd love to talk with y'all.”