Prompt:  
  
I have a sscd model running lambda functions to analize monitored videos and watched videos (millions of them) so lambda is getting expensive in aws. So I would like to know if there is a way of fast filtering the videos that are very different, I already have in my s3 bucket video storage the frames extracted (1 frame per second) of all videos that must be analysed. So any massive transformation I rather doing it before so then our first filtering model can get data fast and filter 60% 70% or even 80% of the watched videos and only run sscd model with the resting videos, so we reduce expenses significantly. My partner told me something about embedding search but not sure how it would work. Also tell me which would be a good strategy for reducing the dataset significantly by using I don't know, average colors, forms, vector, I don't know, maybe even adding keywords to the videos, but for that we would need to run a model that extract the keywords from the videos and store it in a json file, but at least is done once and then we use the video's keywords to filter most of the dataset. What do you think?

Answer:  
  
Your approach to pre-filter the videos before running your expensive SSCd model is spot on. Here are a few strategies you might consider:

### **1. Embedding-Based Filtering**

**Video Embeddings:**

* **Frame Embeddings:** Use a pre-trained CNN (e.g., MobileNet, ResNet) to extract embeddings from each frame. Since you have one frame per second, you can compute embeddings for each frame and then aggregate (e.g., by averaging or pooling) them to get a compact representation for each video.
* **Embedding Search:** With these aggregated vectors, you can use approximate nearest neighbor search libraries like FAISS or Annoy. This will let you quickly identify clusters or outliers and filter out videos that are very different from your target patterns.

**Benefits:**

* Fast similarity checks allow you to quickly rule out videos that are not of interest.
* Reduces the number of videos passed to the heavy SSCd analysis, significantly cutting costs.

### **2. Lightweight Feature Extraction**

**Low-Level Features:**

* **Color Histograms & Average Colors:** Compute simple statistics such as average color values or histograms per video. Videos with drastically different color distributions can be filtered out quickly.
* **Shape and Form Features:** You might extract edge or contour features using classical computer vision techniques (e.g., Canny edge detection) to differentiate between visually similar and dissimilar content.

**Benefits:**

* These features are computationally inexpensive and can serve as a quick pre-filter.

### **3. Metadata & Keywords Extraction**

**Keyword Extraction:**

* **Semantic Labeling:** Use an offline model to run object detection or scene classification on your frames. Extract keywords or tags that describe the video's content.
* **JSON Metadata:** Store these keywords in a JSON file for each video. Then, at runtime, you can filter videos based on whether they contain relevant keywords.

**Benefits:**

* This approach is especially useful if you already have some semantic criteria for what constitutes an “interesting” video.
* The extraction is done once, and subsequent filtering can be as simple as a keyword lookup.

### **4. Combining Strategies**

A hybrid approach can often yield the best results:

* **Primary Filtering:** Start with lightweight features (average colors, simple statistics) and metadata keywords to discard obviously irrelevant videos.
* **Secondary Filtering:** For the remaining videos, use embedding search to further refine your dataset, ensuring that only the most ambiguous or relevant videos are sent to your SSCd model.

### **Implementation Considerations**

* **Preprocessing Pipeline:** Build an offline batch processing pipeline to extract embeddings, features, and keywords from your video frames and store the results. This avoids repeated computation.
* **Thresholding:** Experiment with different similarity thresholds for both embeddings and low-level features to balance between filtering out too many videos and letting through unnecessary ones.
* **Cost vs. Accuracy Trade-off:** Monitor how much of the dataset is filtered at each stage, and adjust your model parameters to achieve a significant reduction (e.g., 60–80%) without losing important cases.

By integrating these methods, you can achieve a robust pre-filtering mechanism that leverages fast, lightweight computations to minimize the load on your more expensive SSCd analysis. This not only reduces AWS Lambda costs but also speeds up the overall processing time.