

Experimental and Analysis Section

In this phase, we evaluate the behavior of our SVD-based novelty filter and analyze how different parameter settings affect its decisions and performance. We focus on the testing logic, how the filter operates, and what parameters influence the outcomes.

1. Filter Logic Overview

We use Singular Value Decomposition (SVD) to generate a compressed representation of each image. By reconstructing the image from this compressed form, part of the visual information is intentionally lost, creating a measurable Reconstruction Error (RE). As explained in methodology part, images that are visually similar tend to produce close RE values, while more different images produce RE values that diverge significantly.

Because high-FPS video typically contains minimal frame-to-frame changes, we compare each frame's RE not with the immediately preceding frame but with the Reference Frame, which is always the last frame classified as novel.

Example:

let the Reference frame: frame_0 RE = 0.50, and the threshold = 0.10

frame_1 RE = 0.52 → skipped

frame_2 RE = 0.55 → skipped

frame_3 RE = 0.58 → skipped

frame_4 RE = 0.61 → forwarded as novel ($0.61 - 0.50 > 0.10$)

and then **frame_4** is the **reference frame now**.

Parameters

There are two parameters that can be changed to control the how the filter behaves.

- **Compression Rank**

Determines how much compression occurs when generating the SVD reconstruction.

- Higher rank → less compression → reconstruction closer to original → RE sensitive to very small differences → more frames may appear novel
- Lower rank → stronger compression → only more significant changes survive → fewer frames forwarded to the AI model

Since extremely high ranks (~80+) almost reproduce the original image and sharply increase computational cost, we test ranks from 1 to 80.

- **Threshold**

The threshold defines the minimum RE difference needed to classify a frame as novel. If the RE difference exceeds the threshold, the frame is forwarded, If it is below, frame is ignored.

The threshold directly controls the sensitivity of the filter.

Evaluation Metrics

These are the metrics we collect during testing to understand the performance and practical effect of the filter, these values depend directly on the compression rank and threshold used in each test.

A. System Performance Metrics

- Frames Per Second (FPS) of the filtering stage
- Processing time per frame (ms/frame)
- Memory usage
- CPU utilization during continuous filtering

B. Novelty Detection Metrics

- Number of forwarded (novel) frames
- Number of skipped frames
- Forwarded Ratio

$$\text{forwarded ratio} = \frac{\text{novel frames}}{\text{total frames}}$$

- Reduction Percentage

$$\text{reduction} = 1 - \text{forwarded ratio}$$

C. Cost Saved by Using the Filter

By comparing Computation per frame if the AI model runs on all frames versus Computation when it only runs on forwarded frames

we calculate:

- Time saved

- Model calls avoided
- Resource load reduced (CPU/memory)

2. Test Case Scenarios

• Test Data

We made our test on a set of successive shots while walking from our faculty gate to a classroom (*data found in section 5 [1]*) simulating one of the visually impaired activities.

Due to time limitations and low computational power we have, we didn't test on a video; as it will take a huge time for running due to large number of frames, so we took a successive shots resembling video flow as much as we could.

Moreover, as we have a lot of parameters which change in parallel, we couldn't get a precise equation represents their relations, but we depended on observations of the results and descriptive statistics using graphs and averages.

To test our filter on a real AI model, we used YOLO Model.

Test Case 1: Low Rank (<10)

In this scenario, the system is evaluated using very small compression ranks. For each rank, the threshold is swept across the interval 0.01–0.21, and the average metric values corresponding to each threshold level are recorded.

observations:

- Lowest frame per second (fps) processing, and highest time consuming
- CPU usage is the highest, so there is only small saving in CPU usage from the baseline (pure AI Running on all frames)
- Lowest filtering efficiency, most images are forwarded to the AI
- The memory usage increases more than the baseline, (additionally all other test cases have the same observation)

Conclusions:

At very low compression ranks, the gatekeeper becomes so sensitive due to the severe loss of visual information in reconstructed frames. As a result, even small natural variations between frames appear significant, causing the filter to label the majority of frames as novel and forward them to the AI model. This leads to poor filtering efficiency and limited computational savings.

Test Case 2: Low Threshold (<0.025)

In this scenario, the system is evaluated using very small threshold. For each threshold, the rank is swept across the interval 1–80 with steps of 5, and the average metric values corresponding to each rank level are recorded.

observations:

- Lowest frame per second (fps) processing values and highest time consuming
- The CPU usage is the highest, so there is only small saving in CPU usage from the baseline pure AI Running on all frames
- Lowest filtering efficiency, most images are forwarded to the AI
- The memory usage increases more than the baseline

Conclusions:

At low threshold, the sensitivity of the gatekeeper filter is so high, as any small difference in the RE becomes above the threshold, so the gate sees most frames as novel ones and passes them to the AI model.

Test Case 3: Medium Rank (10-40)

In this scenario, the system is evaluated using medium values of compression ranks. For each rank, the threshold is swept across the interval 0.01–0.21, and the average metric values corresponding to each threshold level are recorded.

observations:

- The fps and time saving graphs have sharp strikes after rank 10, making the performance almost reaches the peek, and stabilize at the peek until slightly more than rank 40, behaving like a constant function in this interval
- The CPU usage saving graph has a peak at about rank 10 also, and changes semi-linearly until reaching the peek at about rank 50, making the Power saving is the best at the interval of ranks (10-50)
- Its filtering sensitivity is medium, it doesn't forward most images to the AI (like the low ranks) nor ignoring most of them (like at high ranks), making this range suitable for the least probability for false decisions.
- The memory usage increases more than the baseline and more than the lower ranks

Conclusions:

At medium compression ranks, the gatekeeper becomes more accurate in decisions, as it avoids the too low sensitivity like (high ranks) and too high sensitivity (like low ranks), while almost reaching the peek for all performance metrics.

Test Case 4: Medium Thresholds (0.025-0.05)

In this scenario, the system is evaluated using medium values of thresholds. For each threshold, the rank is swept across the interval 1–80 with step of 5, and the average metric values corresponding to each rank level are recorded.

observations:

- The performance over all (except memory usage) almost reaches the peek at this interval and continue almost as constant function after that interval.
- The decisions are convenient in this range, as lower than that range almost everything is forwarded, and higher than this range, almost nothing is forwarded to the AI

Conclusions:

At medium threshold values the performance-accuracy relation are the best for the system.

Test Case 5: High Rank (40-80)

In this scenario, the system is evaluated using high values of compression ranks. For each rank, the threshold is swept across the interval 0.01–0.21, and the average metric values corresponding to each threshold level are recorded.

observations:

- After rank of about 50 the fps decreases and time-consuming increases comparing to the medium ranks
- The CPU usage saved increases slightly from the medium ranks
- The accuracy drops as almost all the frames are ignored from being forwarded to the AI

Conclusions:

At high ranks the performance is worse than the medium ranks and the accuracy is lost by ignoring almost all frames, while the power saving is the best.

Test Case 6: High Thresholds (0.05-0.2)

In this scenario, the system is evaluated using high values of thresholds. For each threshold, the rank is swept across the interval 1–80 with step of 5, and the average metric values corresponding to each rank level are recorded.

observations:

- At this range after 0.05 the fps and time saved are nearly similar to those resulted from the medium range, and behave like a constant function
- The CPU usage saved and power saving enhance slightly from the medium ranks
- The gatekeeper sensitivity drops, and almost ignore all frames

Conclusions:

High thresholds ($\sim > 0.1$) do not enhance the performance nor the power saving with an obvious change, while making the accuracy drops.

3. Graphs

1.fps: Rank vs fps

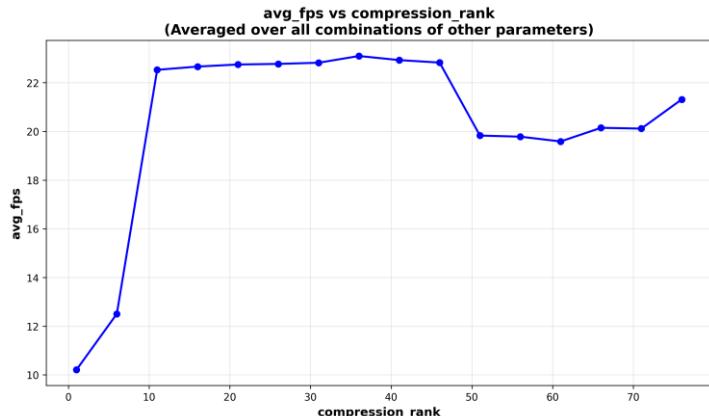


Figure 1: showing the relation between the compression rank and the frame per seconds (fps)

Threshold vs fps

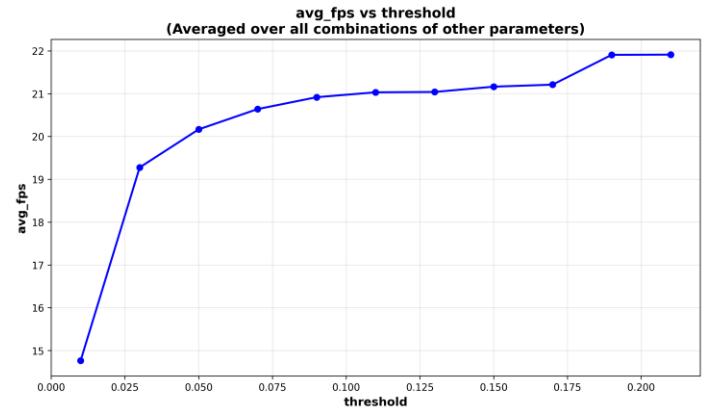


Figure 2: showing the relation between the threshold and the frame per seconds (fps)

2.Processing time saved percent:

Rank vs time saved percent

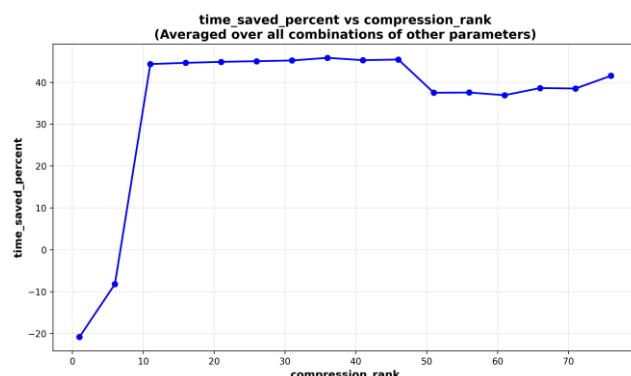


Figure 3: compression rank with processing time saved percent

Threshold vs time saved percent

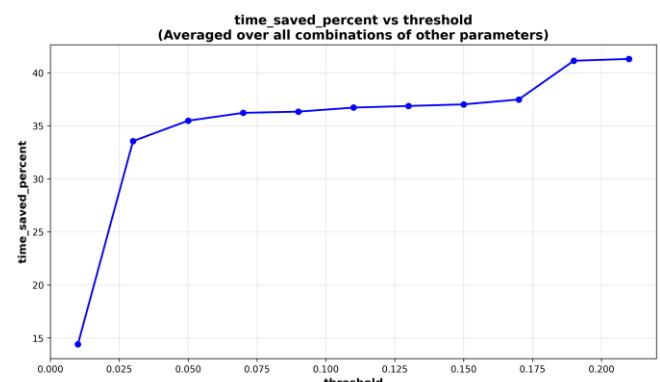


Figure 4: Threshold with processing time saved percent

3.CPU usage saved percent: Rank vs CPU usage saved

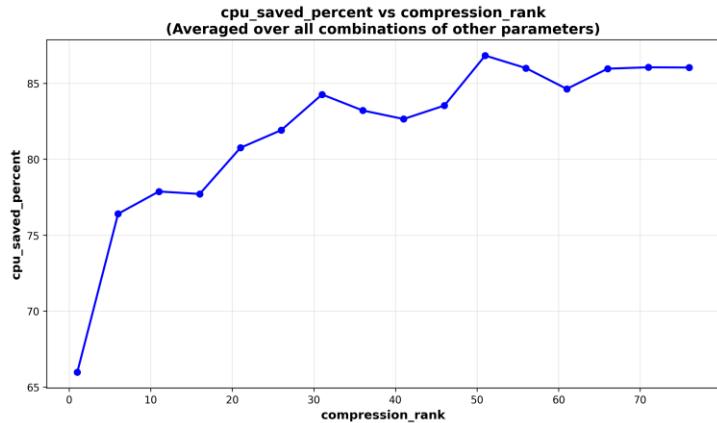


Figure 5: Compression rank with CPU usage saved percent

Threshold vs CPU usage saved

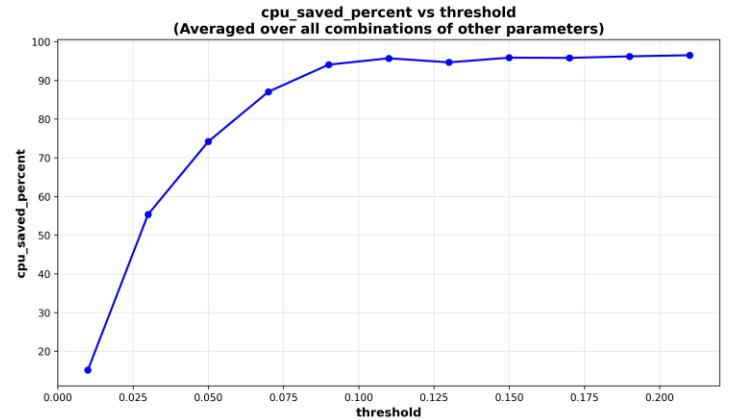


Figure 6: Threshold with CPU usage saved percent

4.Reduction Percentage in forwarding frames: compression rank vs Reduction Percent

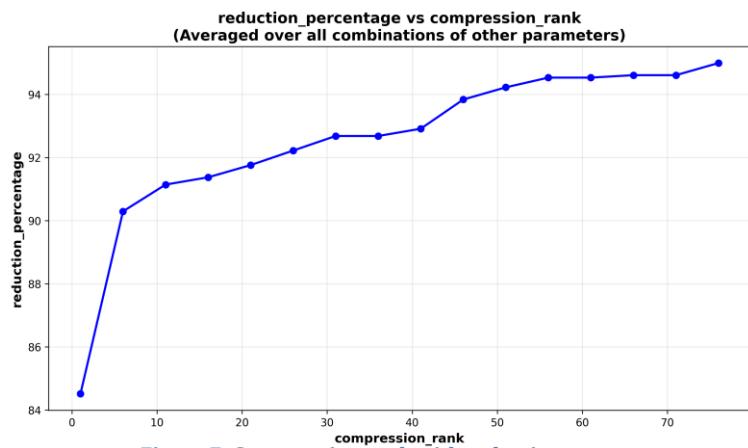


Figure 7: Compression rank with reduction percentage

Threshold vs Reduction Percent

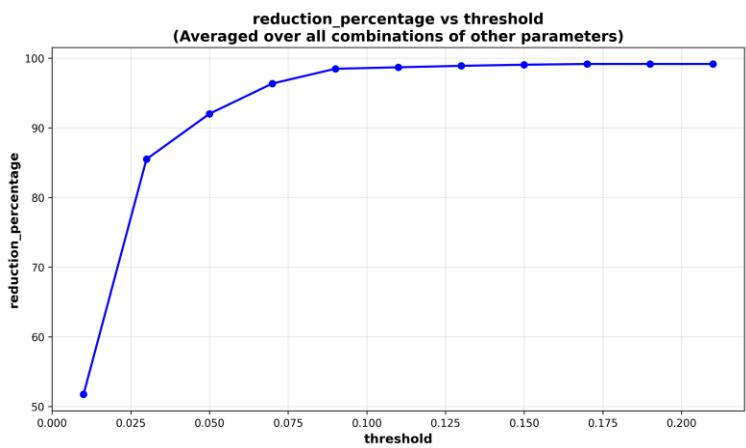


Figure 8: Threshold with reduction percentage

5.Memory Saved: Compression rank vs Memory saved

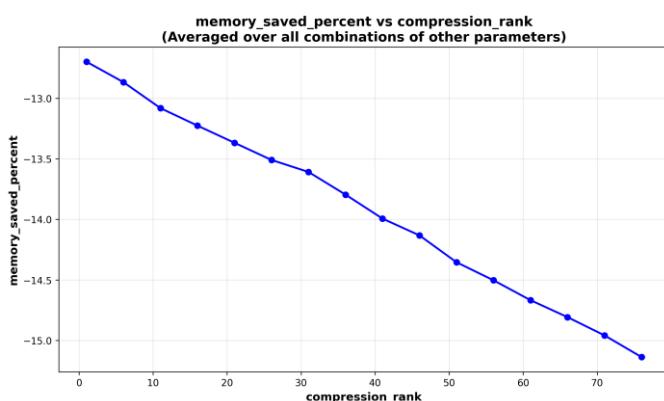


Figure9: Compression rank with the saved memory percent as the saved memory is in negative showing that the memory usage increases by increasing the rank

Threshold vs Memory saved

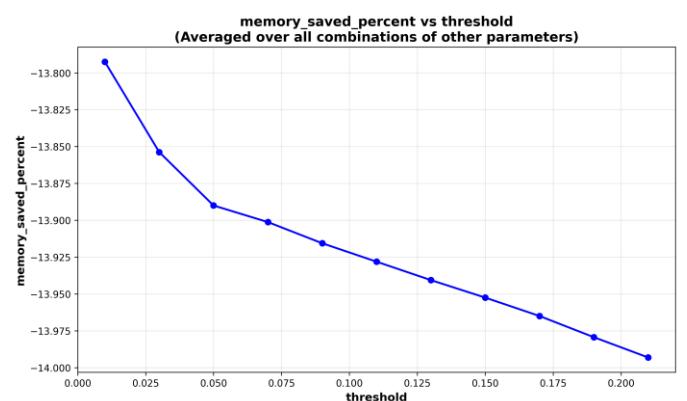


Figure10: Threshold with the saved memory percent as the saved memory is in negative showing that the memory usage increases by increasing the threshold

6. 3D graph plotting Compression Rank vs Threshold vs Average fps

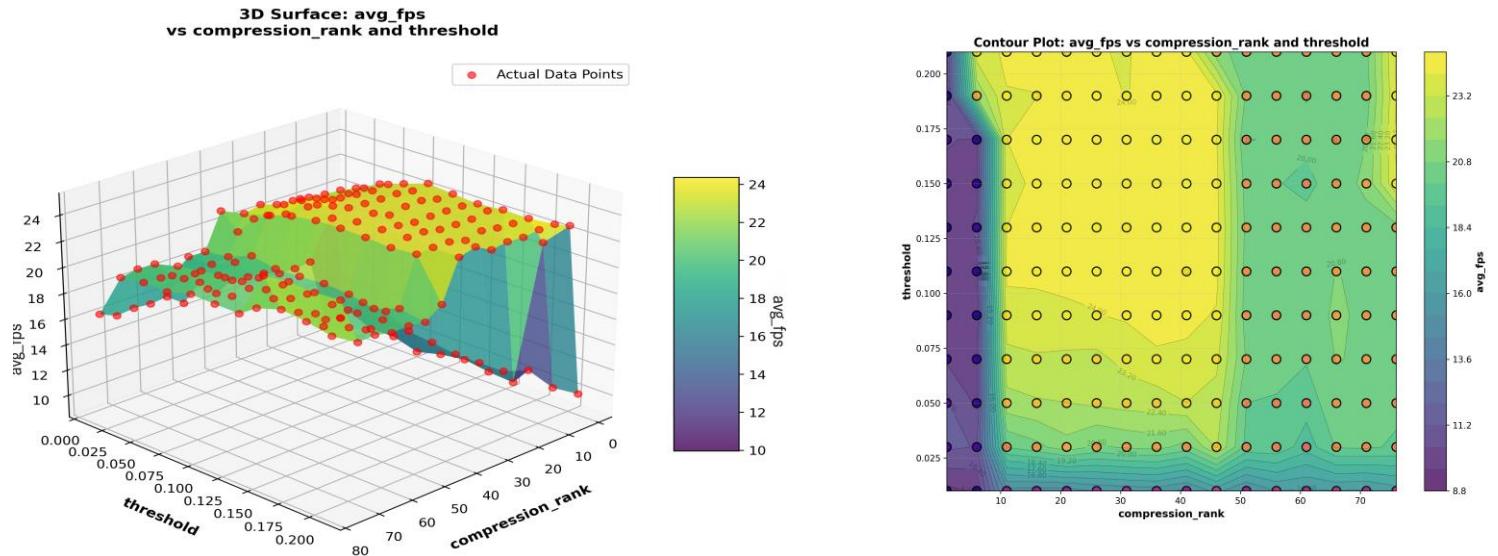


Figure 11: 3D graph showing the relation between the compression rank and threshold combined along with their effect on the average fps

7. 3D graph plotting Compression Rank vs Threshold vs CPU usage saved percent

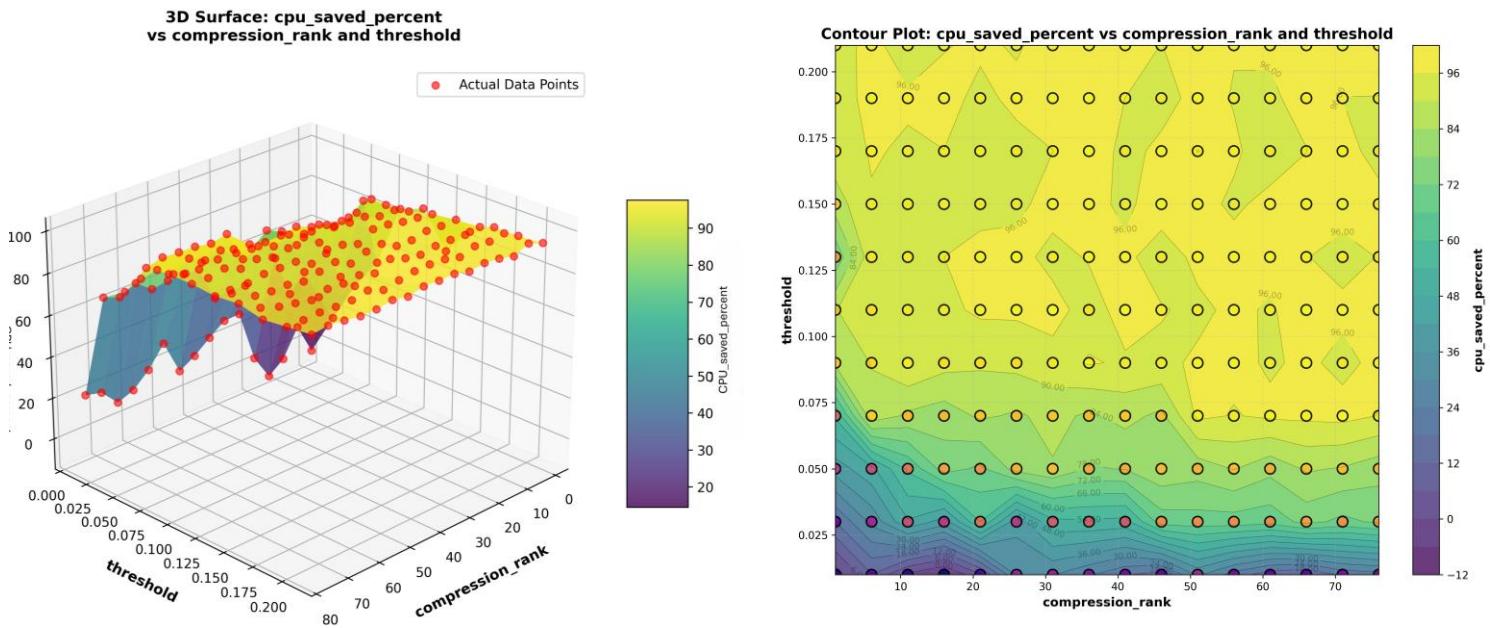


Figure 12: 3D graph showing the relation between the compression rank and threshold combined along with their effect on the CPU usage saved percent from the baseline

8. 3D graph plotting Compression Rank vs Threshold vs Reduction percent in forwarded frames

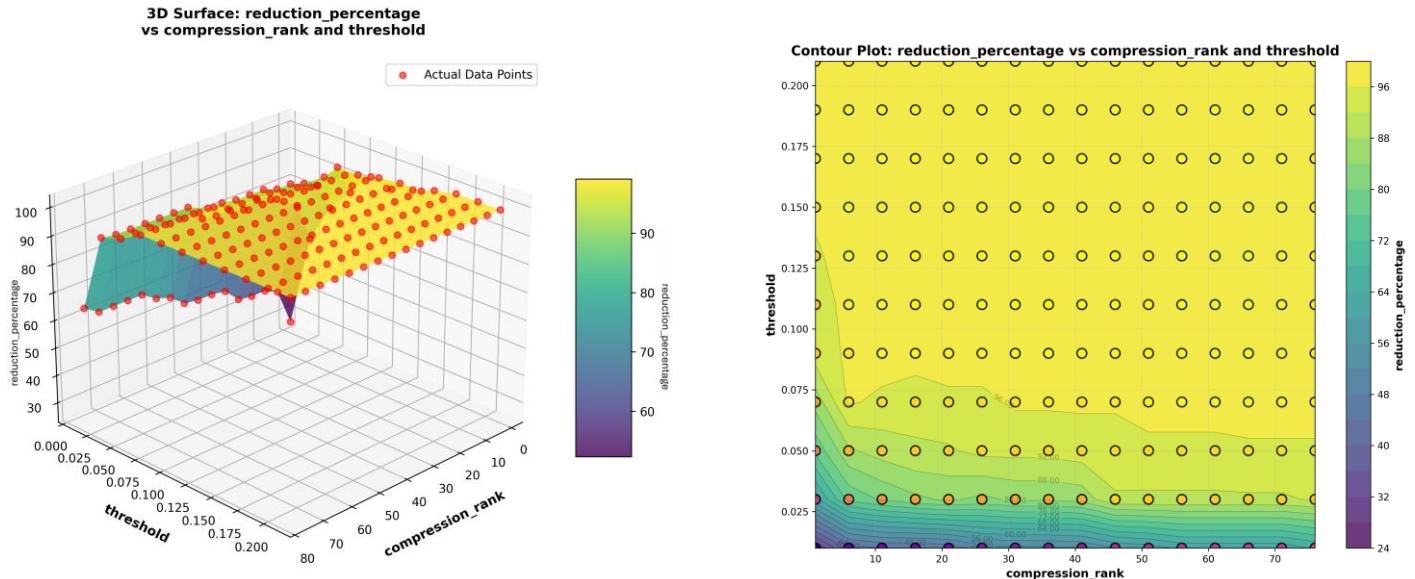


Figure 13: 3D graph showing the relation between the compression rank and threshold combined along with their effect on the reduction percent of forwarded frames after using the filter

9. 3D graph plotting Compression Rank vs Threshold vs saved time

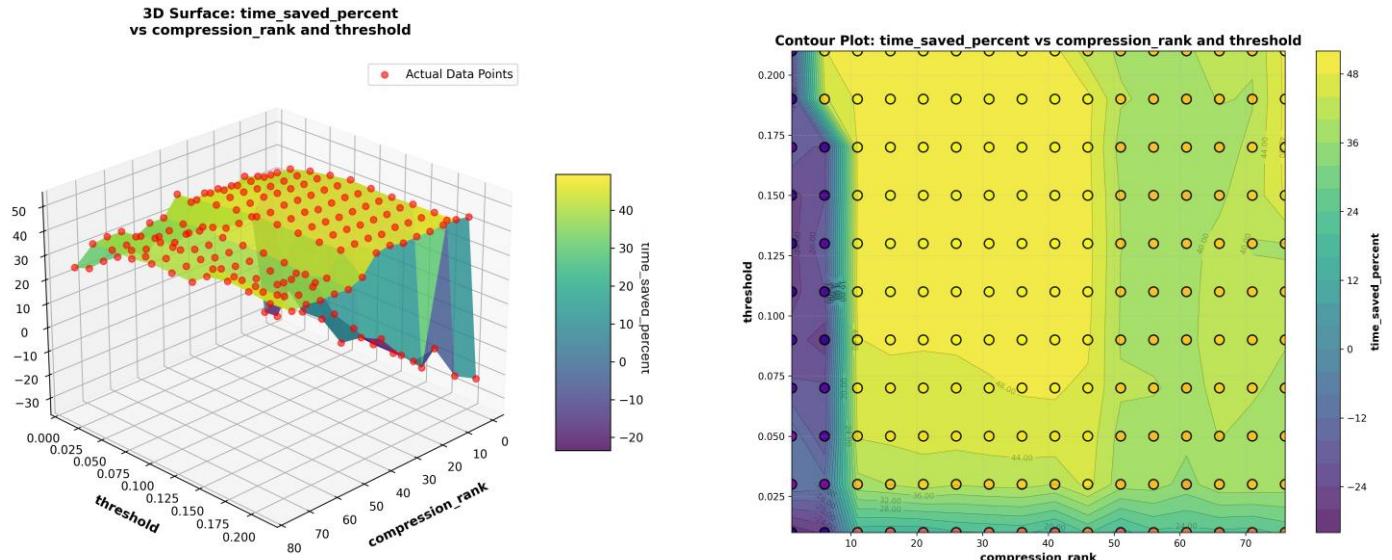


Figure 14: 3D graph showing the relation between the compression rank and threshold combined along with their effect on time saved from the baseline

10.3D graph plotting Compression Rank vs Threshold vs saved memory

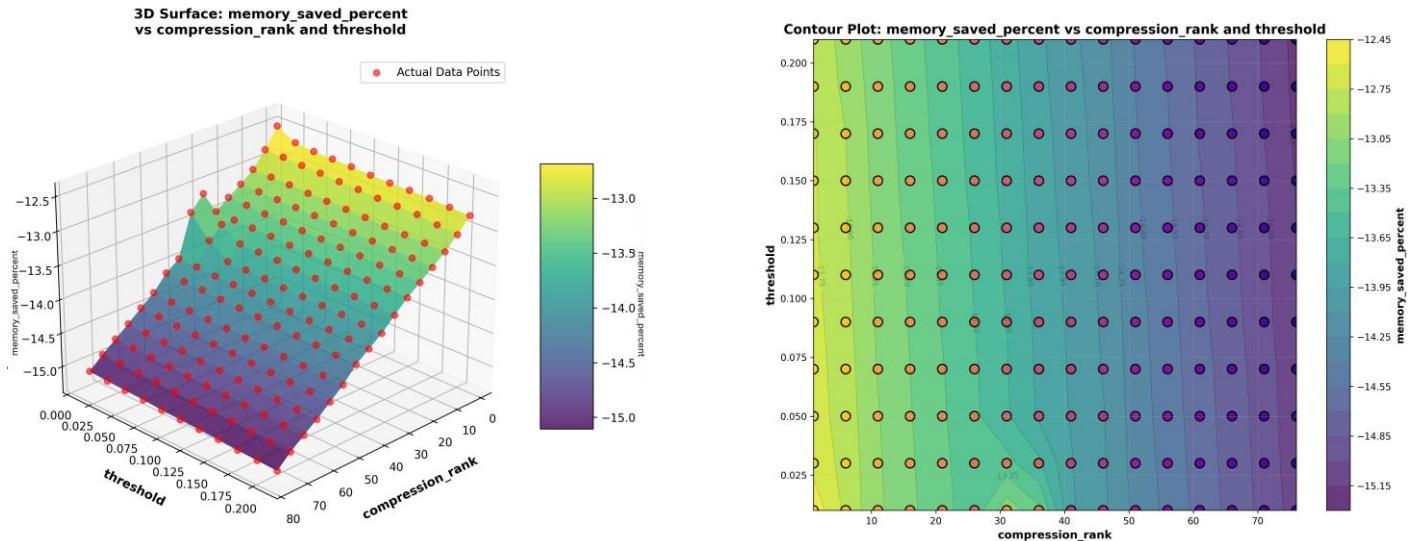


Figure15: 3D graph showing the relation between the compression rank and threshold combined along with the memory saved from the baseline

4.Conclusion

According to the test cases, we can find out that setting low compression and low threshold have big computational cost and forward most frames to the AI, while the high rank and high threshold cause ignoring almost all frames, so it is not serving our idea of detecting the environment around the blind people even if the saved computational power is large.

Accordingly, using compression rank ranged from 10 to 40 and threshold ranged from 0.025 to 0.05 can give best performance-accuracy balance, by ignoring a lot of frames that need no detection, along with detecting the most novel frames helping the visually impaired people to know their environment.

Also, we can notice, that the memory usage will increase slightly when applying the filter before the AI, this increase can be handled by clearing the memory from time to time while running. And as the increase in memory usage is small (from 10% to 20%), so it causes no big problems.

Points to be considered in coming researches

- more tests need to be run on different and larger datasets with different conditions (sunlight, dim light, different camera qualities, different moving speeds, etc.)
- more tests need to be run on different hardware (mobile, smart glass, special kit, etc.) to figure out exact computational power needed for such device to be wearable

5.Data and resources

[1] Taken images from Cairo University faculty of engineering, simulating walking from the entrance gate into the classroom at the noon, moving from sun light into dim light while normal behaviors like camera shaking, zoom changes are found.

Available: [https://drive.google.com/drive/folders/1ak8W-Zh0GXye0QJinstKNoT1pzl8e1yM?usp=drive link](https://drive.google.com/drive/folders/1ak8W-Zh0GXye0QJinstKNoT1pzl8e1yM?usp=drive_link)

[2] Code used in running tests is made by the research team. It is open source and can be used for any research purposes

Available: https://github.com/Hakeem-Taha-06/SVD_ImageComparison.git