Smart Classroom UROP Report

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Keywords

Smart classroom, intelligent classroom, sensors, automation.

Overview

This project aims to understand the existing challenges to classroom learning and invigilation, as well as to develop software solutions to address some of the identified challenges. These solutions would then be evaluated for their effectiveness, efficiency, efficacy, performance and usefulness. Our hope is to enhance or modify existing pedagogical practices so as to further support learning and invigilation. This is particularly relevant in the current climate of virtual classes due to the COVID-19 pandemic, since the attention rate of students is harder to estimate and monitor online using virtual means compared to physical classes.

Data

Emotion Detection with Webcam

Using pre-trained neural networks in Intel's OpenVINO toolkit, we were able to (1) detect faces and track their positions[1], (2) recognize emotions displayed on their faces[2].

Using the face-detection-adas-0001.bin model, we can get the bounding boxes for each face in the frame and hence use that to crop out just the faces present. The working demo produces the following cropped output:

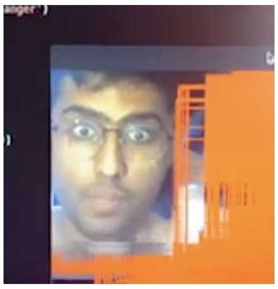


Figure 1: Face detection and extraction demo.

Using the extracted face and emotions-recognition-retail-0003.bin model, we can classify the emotions expressed by the face into 5 categories: ('neutral', 'happy', 'sad', 'surprise', 'anger'). Some classification examples from the demo are shown below:

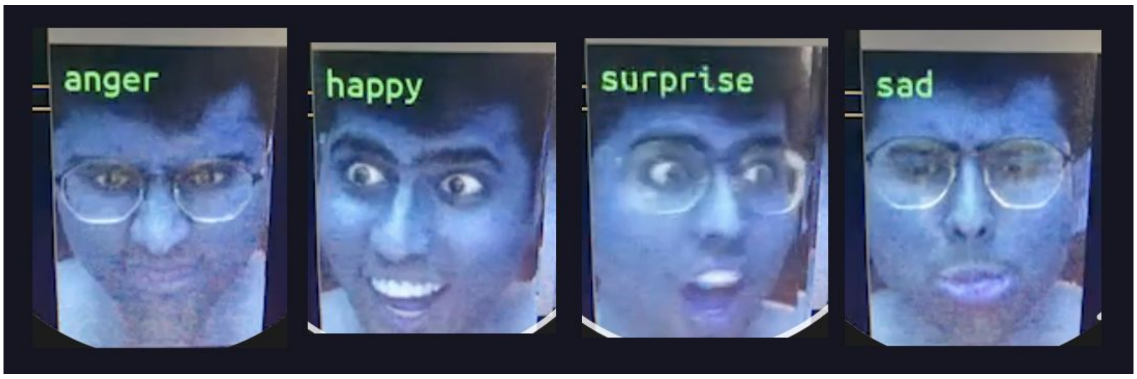


Figure 2: Emotion recognition from extracted face demo.

1 https://docs.openvinotoolkit.org/2018_R5/_docs_Transportation_object_detection_face_pruned_mobilenet_reduced_ssd_shared_weights_caffe_desc_face_detection_adas_0001.html

2 https://docs.openvinotoolkit.org/2019_R1/_emotions_recognition_retail_0003_description_emotions_recognition_retail_0003.html

Eye Tracking with Webcam

Due to the need for expensive hardware for individual participants to accurately track their gaze, we looked for alternative methods using hardware students already have.

GazeRecorder[3] filled that niche with a solution using a webcam. Users are required to perform a calibration using the GazePointer/GazeRecorder software. We found that the built-in calibration for GazeRecoder, with only 5 points, is insufficient to provide any meaningful gaze tracking information. The built-in calibration for GazePointer uses 16 points for gaze and 8 for head pose, providing much more accurate gaze tracking data.

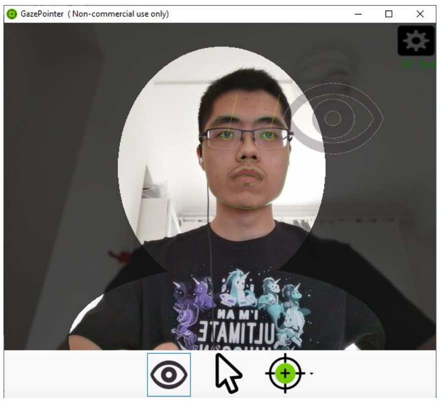
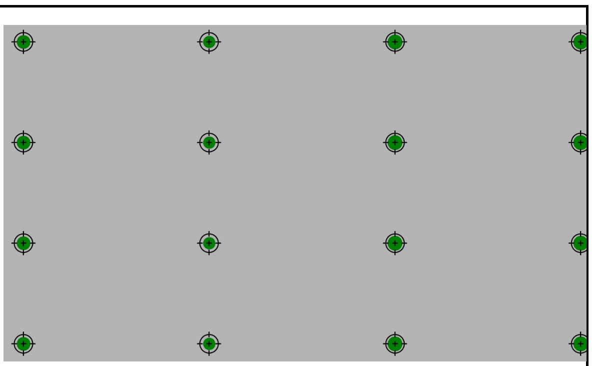
	

Figure 3: GazePointer.	Figure 4: 16 Gaze Tracking Calibration Points.

However, there is some drift when fixating at a fixed point on the screen. More research is needed to determine if the noisy output data due to drift is a concern.

The eye tracker comes with an API to query the 8 degrees of freedom. These data points are stored in a text file for further analysis.

Screen Space

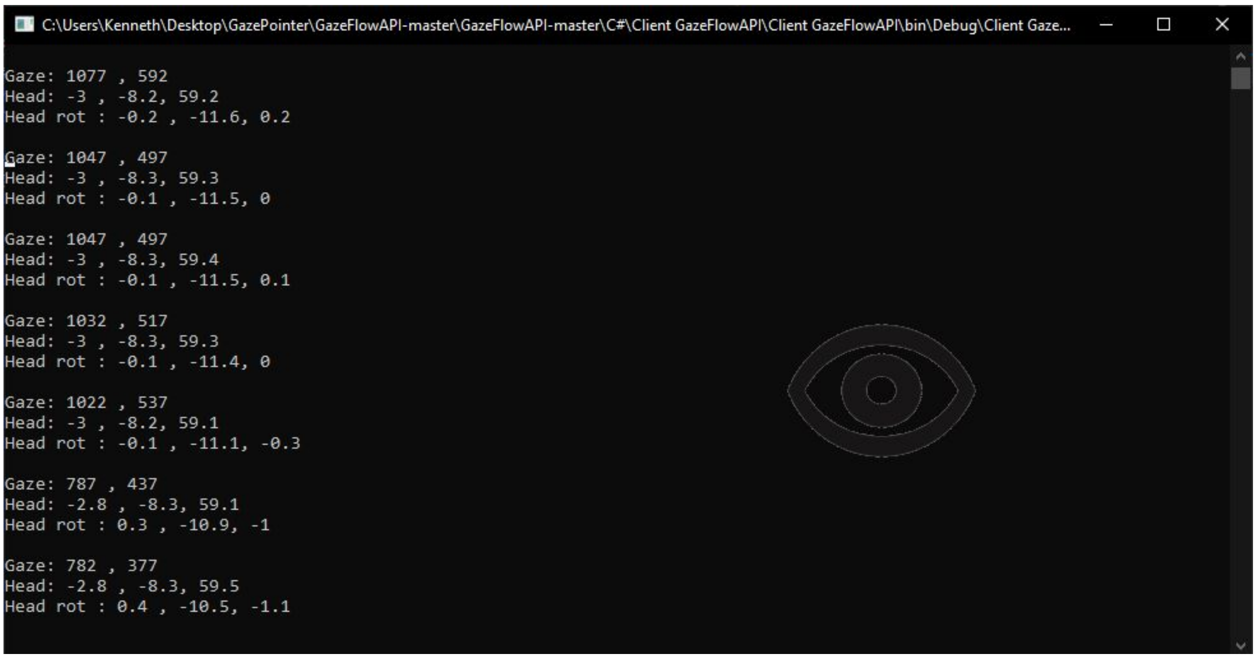
Gaze X: X position in pixels on display.
Gaze Y: Y position in pixels on display.

[3] https://gazerecorder.com/

World Space

Head X: X position measured from centre of screen.
Head Y: Y position measured from centre of screen.
Head Z: Z position measured from surface of screen (distance from screen).

Head Roll/Pitch/Yaw: Measures of head rotations.



```
C:\Users\Kenneth\Desktop\GazePointer\GazeFlowAPI-master\GazeFlowAPI-master\C#\Client GazeFlowAPI\Client GazeFlowAPI\bin\Debug\Client Gaze...

Gaze: 1077 , 592
Head: -3 , -8.2, 59.2
Head rot : -0.2 , -11.6, 0.2

Gaze: 1047 , 497
Head: -3 , -8.3, 59.3
Head rot : -0.1 , -11.5, 0

Gaze: 1047 , 497
Head: -3 , -8.3, 59.4
Head rot : -0.1 , -11.5, 0.1

Gaze: 1032 , 517
Head: -3 , -8.3, 59.3
Head rot : -0.1 , -11.4, 0

Gaze: 1022 , 537
Head: -3 , -8.2, 59.1
Head rot : -0.1 , -11.1, -0.3

Gaze: 787 , 437
Head: -2.8 , -8.3, 59.1
Head rot : 0.3 , -10.9, -1

Gaze: 782 , 377
Head: -2.8 , -8.3, 59.5
Head rot : 0.4 , -10.5, -1.1
```

Figure 5: Printed debug logging messages, displaying the corresponding positional and rotational coordinates.

Methodology

Currently, we have developed a technique to gather the video feed from online Zoom meetings using the livestream feature that the Zoom API has provided. We have outlined a step-by-step instruction that could be followed to set up the livestream feature on our GitHub repository here: https://github.com/jamestiotio/smart-classroom#zoom

To utilize the script, we would require an OAuth app access to the Zoom API endpoints. Follow the instructions provided to set up the NGINX server with the RTMP protocol and the appropriate dependencies installed. Set up the desired Zoom class meeting and populate settings.py as required. The zoom.py script would then handle sending the PATCH requests to Zoom's server necessary to initiate the livestream.

These are the headers of the PATCH requests being sent (with the access token being obtained beforehand through the act of logging in into the corresponding Zoom account):

```
headers = {
    "Content-Type": "application/json",
    "Authorization": f"Bearer {access_token}"
}
```

This is the JSON content of the first PATCH request being sent to set up the appropriate stream URL, stream key and page URL:

```
update_payload = {
    "stream_url": f"{settings.stream_url}",
    "stream_key": f"{settings.stream_key}",
    "page_url": f"{settings.page_url}"
}
```

This is the JSON content of the second PATCH request being sent to start the livestream:

```
start_live_payload = {
    "action": "start",
    "settings": {"active_speaker_name": True, "display_name": "inc"}
}
```

With the appropriate meeting ID, these are the URLs to which the PATCH requests are being sent respectively:

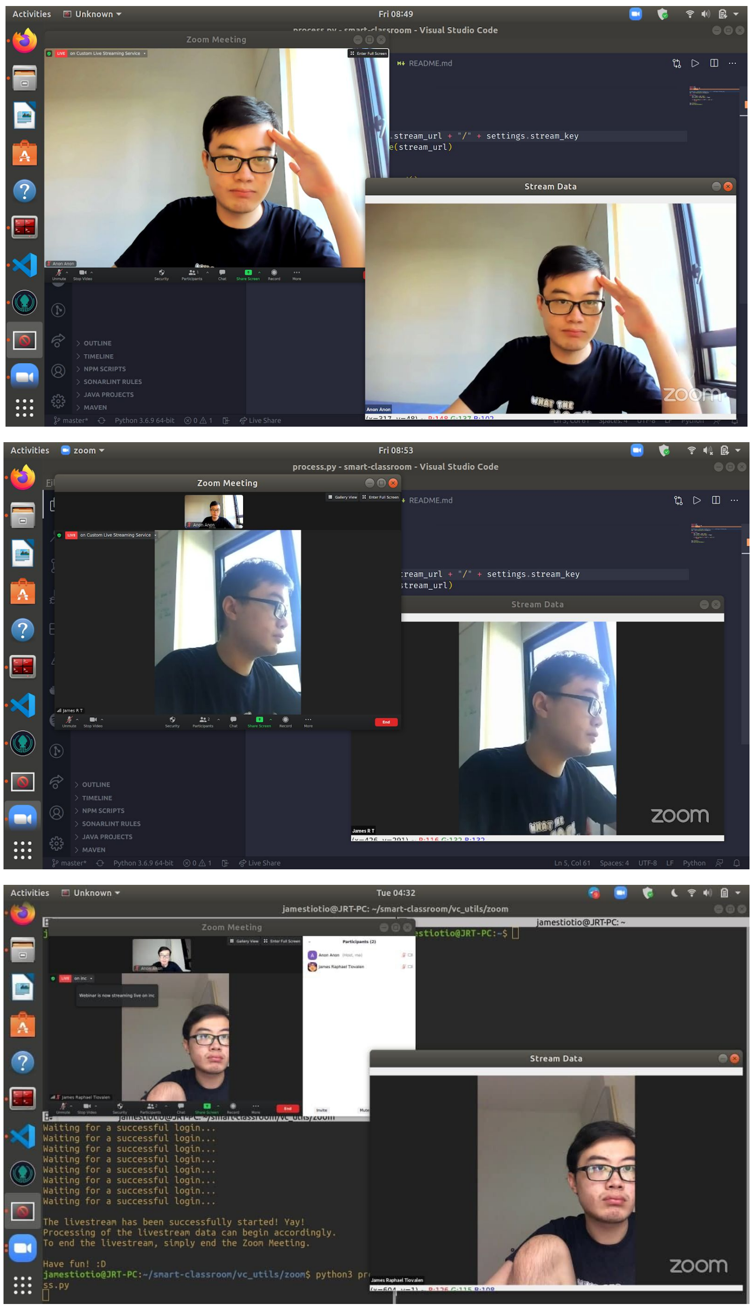
```
first_url = f"https://api.zoom.us/v2/meetings/{settings.meeting_id}/livestream"

second_url = f"https://api.zoom.us/v2/meetings/{settings.meeting_id}/livestream/status"
```

If the zoom.py script was successfully executed without any error messages or exceptions raised, then the livestream video data feed is ready to be processed by the facial analysis script. The process.py script is a mock-up of such a backend processing script that actually does the main facial analysis work.

The livestream video data feed would bear the Zoom watermark since it is an official livestream. More research needs to be conducted to determine whether this watermark would have an adverse effect on the final results or not. The video frames being collected depend on the actual content being shared in the Zoom meeting. By default, the livestream will send over the video frames of the current live speaker in the meeting. If there is any screen-sharing involved, the content of the screen being shared will be livestreamed instead, with a thumbnail of the sharer's camera video feed being displayed on the top-right corner. If the host switches to a gallery view, a gallery view would be livestreamed instead. The content being live-streamed does not depend on the resolution of the host's window (whether it is a current speaker view or a gallery view), and as such, it does not matter if the host's Zoom window was resized. For our purposes, the gallery view is of particular interest to us since such a view would be able to display up to 49 participants with their respective camera video feed ready to be processed.

Results

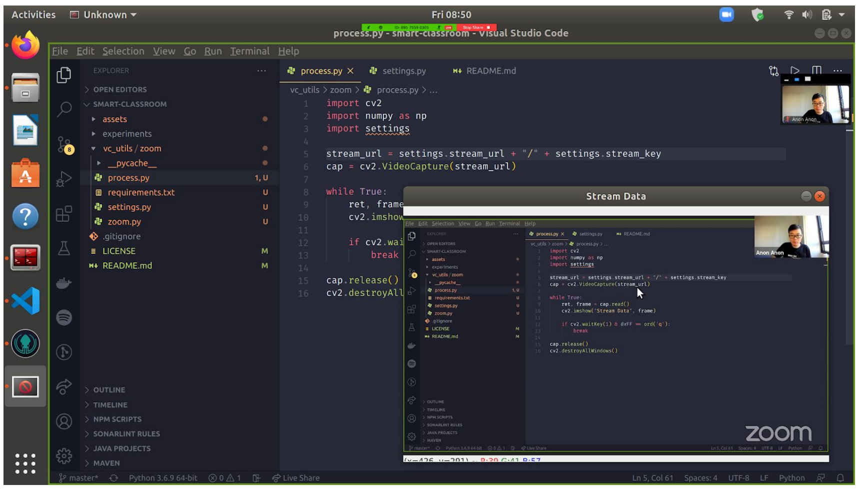
These are some evidence to show that the Zoom API script is working.



```
stream_url + "/" + settings.stream_key
e(stream_url)
```

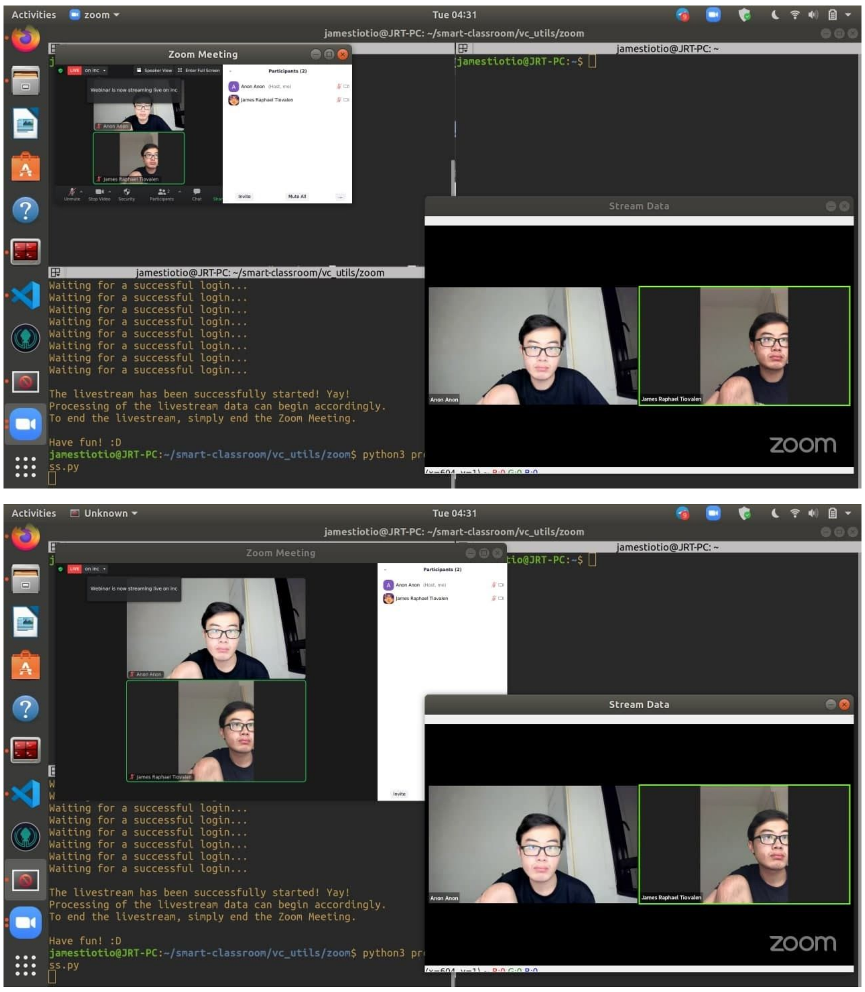
```
Waiting for a successful login...
Waiting for a successful login...
Waiting for a successful login...
Waiting for a successful login...
Waiting for a successful login...
Waiting for a successful login...
Waiting for a successful login...
Waiting for a successful login...

The livestream has been successfully started! Yay!
Processing of the livestream data can begin accordingly.
To end the livestream, simply end the Zoom Meeting.

Have fun! :D
jamestiotio@JRT-PC:~/smart-classroom/vc_utils/zoom$ python3 pr
ss.py
```

Figure 6: Active Speaker Display View Mode.



```python
import cv2
import numpy as np
import settings

stream_url = settings.stream_url + "/" + settings.stream_key
cap = cv2.VideoCapture(stream_url)

while True:
    ret, frame = cap.read()
    cv2.imshow('Stream Data', frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

Figure 7: Screen Share Display View Mode.



```
jamestiotio@JRT-PC: ~/smart-classroom/vc_utils/zoom
Waiting for a successful login...
Waiting for a successful login...
Waiting for a successful login...
Waiting for a successful login...
Waiting for a successful login...
Waiting for a successful login...
Waiting for a successful login...
Waiting for a successful login...

The livestream has been successfully started! Yay!
Processing of the livestream data can begin accordingly.
To end the livestream, simply end the Zoom Meeting.

Have fun! :D
jamestiotio@JRT-PC:~/smart-classroom/vc_utils/zoom$ python3 pr
ss.py
```

Figure 8: Gallery Display View Mode.

Furthermore, as seen from Figure 8, resizing of the Zoom app's window on the same computer and camera (with the same resolution and display zoom settings) does not change or alter the resolution of the actual live-streamed video frames. This factor-independence is important, especially if we were to conduct further analysis on the video frames.

Future Work

Further work could be done, including, but not limited to:

- Integration between the scraped video feed from the Zoom livestream and the facial analysis backend.
- Development of a dashboard to display the output in an intuitive, elegant, simple and beautiful manner to the instructor so as to provide real-time feedback on the class' current performance and analytics.
- Synchronization between the backend and the current status of the lesson's progress (such as the slides' pages) so as to plot and track a graph of attention rate against time as the class session progresses.
- A more diverse set of tools to scrape video feed from various other video conference tools and utilities, such as Google Meet and Microsoft Teams, as well as from physical sensors in a physical classroom layout (such as from multiple laptops of participating students). This is to accommodate more users who might utilize different video conference platforms.
- Actual instrumentation of a physical classroom with different sensors to gather the necessary data for the software to create the inferences. This requires further analysis on the physical layouts of classrooms and on the best possible method to create a non-intrusive monitoring system that is still conducive for students as a studying environment.
- Actual testing and deployment of a prototype of the system to ongoing classrooms (following the prevailing COVID-19 guidelines in Singapore). Through this deployment, we could conduct more studies and collect more data so as to improve the current prototype.
- Market analysis and demand estimation for such a solution with the university as the primary beneficiary, as well as other potential customers such as other educational institutions, both in Singapore and other countries.

Conclusion

In conclusion, we were able to conduct a few preliminary investigations, explore several options and possible avenues of interest and develop some prototypes of several software solutions to assist us in our goal of improving classroom learning and invigilation. More work is necessary to produce a more concrete outcome from actual students and a more practical output for instructors, but current results are promising.

Related Work

The Research on Anti-Cheating Strategy of Online Examination System
https://ieeexplore.ieee.org/document/6010689

Activity Recognition (Acoustic)
https://dl.acm.org/doi/abs/10.1145/2370216.2370269

Localisation (Inertial)
https://dl.acm.org/doi/10.1145/2370216.2370280

Stress (Voice)
https://dl.acm.org/doi/10.1145/2370216.2370270

Classroom Attention Analysis Based on Multiple Euler Angles Constraint and Head Pose Estimation
https://link.springer.com/chapter/10.1007/978-3-030-37731-1_27

Translating Head Motion into Attention - Towards Processing of Student's Body-Language
https://files.eric.ed.gov/fulltext/ED560534.pdf

Smart Classroom System Chinese Patent
https://patents.google.com/patent/CN107154202A/en

Chinese Classroom Surveillance Article
https://www.sixthtone.com/news/1003759/camera-above-the-classroom

Related Tools

Deep Head Pose
https://github.com/natanielruiz/deep-head-pose

Deepgaze
https://github.com/mpatacchiola/deepgaze

Intel OpenCV ML Models
https://download.01.org/opencv/2019/open_model_zoo/R4/20200117_150000_models_bin/

Intelligent Classroom
https://github.com/intel-iot-devkit/intelligent-classroom

List of RGBD Datasets
http://www.michaelfirman.co.uk/RGBDdatasets/

Webgazer - Brown University
https://github.com/brownhci/WebGazer

WebRTC
https://webrtc.org/

ParallelDots
https://github.com/ParallelDots/ParallelDots-Python-API

Microsoft Cognitive Services
https://azure.microsoft.com/en-us/services/cognitive-services/

Progress Log

Date	Progress
04 May 2020 - 03 Sep 2020	Ragul conducted some experiments on the Emotions Recognition script and its corresponding Machine Learning model using OpenVINO.
03 Sep 2020 - 18 Dec 2020	James worked on and developed the script to scrape the Zoom live-stream video frames by utilizing the Zoom API.
04 May 2020 - 18 Dec 2020	Kenneth researched on Eye Tracking algorithms and mechanisms utilizing the limited resources of a laptop webcam.