University of the Philippines Manila
College of Arts and Sciences
Department of Physical Sciences and Mathematics

Using Semi-auto Annotation and Optical Character Recognition for Transcription of Patient Monitor using Smartphone Camera

A special problem in partial fulfillment
of the requirements for the degree of
Bachelor of Science in Computer Science

Submitted by:

Jan Federico P. Coscolluela IV

June 2023

University of the Philippines Manila
College of Arts and Sciences
Department of Physical Sciences and Mathematics

Using Semi-auto Annotation and Optical Character Recognition for Transcription of Patient Monitor using Smartphone Camera

A special problem in partial fulfillment
of the requirements for the degree of
Bachelor of Science in Computer Science

Submitted by:

Jan Federico P. Coscolluela IV

June 2023

Permission is given for the following people to have access to this SP:

Available to the general public	Yes
Available only after consultation with author/SP adviser	No
Available only to those bound by confidentiality agreement	No

ACCEPTANCE SHEET

The Special Problem entitled "Using Semi-auto Annotation and Optical Character Recognition for Transcription of Patient Monitor using Smartphone Camera" prepared and submitted by Jan Federico P. Coscolluela IV in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science has been examined and is recommended for acceptance.

Marbert John C. Marasigan, M.Sc. (cand.)
Adviser

EXAMINERS:

	Approved	**Disapproved**
1. Avegail D. Carpio, M.Sc.		
2. Richard Bryann L. Chua, Ph.D. (cand.)		
3. Perlita E. Gasmen, M.Sc. (*cand.*)		
4. Ma. Sheila A. Magboo, Ph.D. (*cand.*)		
5. Vincent Peter C. Magboo, M.D.		
6. Geoffrey A. Solano, Ph.D.		

Accepted and approved as partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science.

Vio Jianu C. Mojica, M.Sc.
Unit Head
Mathematical and Computing Sciences Unit
Department of Physical Sciences
and Mathematics

Marie Josephine M. De Luna, Ph.D.
Chair
Department of Physical Sciences
and Mathematics

Maria Constancia O. Carrillo, Ph.D.
Dean
College of Arts and Sciences

Abstract

Vital signs monitoring is a key function in healthcare delivery to ensure immediate and precise evaluation of a patient's well-being. It is done by attaching monitor devices to patients which collect, store, and display values on a screen. In many low-to-medium-income countries (LMICs), hospitals still rely on manual observation and handwritten documentation of vital signs, which is susceptible to human errors, data tampering, process inefficiency, and limited opportunities for comprehensive data analysis. More advanced hospitals utilize interface engines which transmit data to electronic medical records but tend to be model-specific and are very costly. Optical character recognition (OCR) offers a cost-effective and non-invasive alternative to digitizing manual transcription of vital signs data in healthcare settings with low financial resources. An image preprocessing pipeline is proposed to perform contour-based screen extraction of the patient monitor captured by a camera, thus providing a well-defined region more suitable for subsequent tasks of object detection and data extraction. The study offers a newly accrued dataset of over 4000 images of Mindray Beneview T8 patient monitor with multi-parameter annotations. Results showed that screen extraction prior to object detection significantly improved the mean Average Precision (mAP) of the model from 68.55% to 93.65% at an IoU threshold of 0.7.

Keywords: patient monitor, optical character recognition, object detection, image preprocessing, annotation

Contents

List of Figures

List of Tables

I. Introduction

A. Background of the Study

Patient monitoring is a critical aspect of healthcare delivery, ensuring timely and accurate assessment of vital signs and overall patient well-being. Patient monitor devices serve as the primary tool to continuously measure and display vital signs such as heart rate, blood pressure, and oxygen saturation. This provides healthcare professionals with real-time information about a patient's physiological status for long-term observation and early medical interventions as needed [1]. Despite technological advancements, manual monitoring techniques, such as nurses manually observing or taking note of value, still persist in many hospitals. This approach poses several limitations, including the risk of human error, consumption of time, difficulties in data storage and retrieval, and hindered opportunities for comprehensive analysis.

Digitizing manual patient monitoring through the utilization of Optical Character Recognition (OCR) technology offers a promising solution to address these challenges as it grows to be a growing area of research [2]. This provides a cost-effective solution with reduced hardware costs and connectivity expenses using camera without the need for expensive cords or third party software. To this end, the UP Manila Standards and Interoperability Lab (UPM SILAB) can incorporate data interoperability across health institutions involving patient monitor data, specifically in the Philippine General Hospital (PGH).

B. Statement of the Problem

The inadequacy of publicly available patient monitor dataset poses challenge to the development of OCR system with specific context and focus of application. Despite third-party software offering simulated patient monitor videos, most of which are paid, these datasets are already high-quality and mainly designed for trend analysis studies.

Furthermore, one common challenge encountered when working with computer vision is the presence of noise in image datasets. The problem at hand is to develop effective techniques and methodologies to mitigate noise in image datasets, thereby enhancing the quality and reliability of the data for subsequent analysis and applications.

C. Objectives of the Study

This study intends to provide a newly accrued dataset of patient monitor images reflecting a realistic hospital environment. In particular, the goals of the paper are as follows:

- **Dataset Objectives**:

    1. Manually collect video recordings of patient monitor from the Post Anesthesia Care Unit of Philippine General Hospital
    2. Utilize smartphone camera to perform data collection in both natural and low lighting conditions with different camera angles
        - direct camera
        - skewed to the left
        - skewed to the right
        - skewed upwards
        - skewed downwards
    3. Perform frame extraction to obtain patient monitor image dataset from captured videos
    4. Implement a semi-auto annotation approach to expedite the manual dataset annotation procedure
    5. Fully annotate raw and preprocessed images in PascalVOC XML format
    6. Train a customized object detection model to locate vital signs from the newly accrued dataset
    7. Apply optical character recognition to extract vital signs from model detections

- **System Objectives**:

  1. Allow the user to capture an image or video of a patient monitor via device camera in real-time
  2. Allow the user to upload a pre-taken image or video of a patient monitor
  3. Implement frame extraction to retrieve individual images from video input type at 2-second intervals
  4. Implement a screen-extraction procedure using an image preprocessing pipeline
     (a) gamma correction
     (b) edge detection
     (c) skew correction
  5. Allow user to download preprocessed image output in ZIP and CSV format

D. Significance of the Project

A new dataset of camera-captured patient monitor images with multi-parameter annotations is a contribution to computer vision which can aid benchmarking research, validation of computer vision algorithms, and remote patient monitoring.

Creating a web application that allows access to camera and integrates image enhancement procedures can also be scaled for future work of text and digit recognition or waveform interpretation. In particular, the proposed study can serve as an initial step for UPM SILAB's OCR project for PGH, focusing on image optimization before the OCR process. By optimizing images prior to OCR, a more suitable image data for subsequent tasks is obtained. The workload on healthcare staff can be minimized, and valuable patient information can be preserved for research purposes.

E. Scope and Limitations

This study operates under the following conditions:

1. The dataset only considers Mindray Beneview Series patient monitor.

2. Camera angle during data collection is adjusted manually without the use of any software.

F. Assumptions

This project operates under the following assumption/s.

1. The monitor is not obstructed by any object during capture.

2. The monitor is sufficiently captured and not cropped.

II. Review of Related Literature

Computer vision is a field of artificial intelligence that has been extensively utilized across various domains such as real estate, businesses, and healthcare. It is used to simulate human visual abilities by enabling computers to analyze surroundings as humans do—or even more. A concept under computer vision known as optical character recognition (OCR) is a highly researched topic [2]. This is typically done by having a digital image of a document, performing image processing to remove unwanted information, training the computer to locate characters of interest, and finally segmenting the detected characters for identification [3]. For instance, self-driving cars incorporate OCR technology not only to facilitate detection of objects such as obstacles and nearby vehicles but also to perform corresponding actions to keep the car free from collision.

Majority of previous work utilizing OCR focused on number recognition, document analysis, and vehicular license plate recognition [4]. For instance, Zacharias et al [2] explored the extraction of Intermodal Loading Units (ILU) codes printed on the rear end of swap bodies (freight containers for road and rail transport) using a text recognition pipeline with the open-source Tesseract OCR engine. A small variation in illumination among the captured images was found to contribute to large errors in text recognition and thereby negatively affect model success metrics. Implementation of deep learning-based model can be promising to overcome the large fluctuation of model accuracy with scene text images [2].

Such recommendation for use of deep learning was explored by an optical character recognition post-correction study conducted by Karthikeyan et al. [5] which showed the feasibility of model accuracy improvement applied on medical reports. Correct transcription and recognition of documents is a key challenge identified in this paper due to presence of noise such as obscured, skewed, or illegible text. Specific medical terminologies deviating from general language lexicons were also found to compound the error rate of OCR process. Introducing medical terminologies to the vocabulary of employed OCR model is a highlighted technique that could be employed in patient monitor dataset since health parameters and their symbols may not syntactically align with that of general language. This apparent dependence of OCR model accuracy on dataset quality suggests the critical role of data collection and proper image preprocessing as applied to a specific context of data. In a medical study using lung MRI images as dataset, filtering techniques such as Wiener, median, and Gaussian reduced the time it takes to process the images [6]. Blur detection is a technique that can be explored in related works to assess the quality of captured medical image beforehand and retake the data collection phase prior to further processing.

Despite the extensive studies using OCR, work centered particularly on scanned documents ranging from business forms, receipts, and bibliographic data. Commercially available OCR tools are also optimized for scanner-captured documents which results to drastic decrease of the transcription accuracy for camera-captured images due to apparent noise and distortion from environmental factors [7]. Further, lack of open studies around these commercial tools leads to low system repeatability and assessment. The challenge thus remains on expanding the application of OCR primarily in the context of healthcare where captured images or video from data sources are not readily suitable for modeling. For instance, medical data may come in the form of prescriptions and patient records which are typically stored on paper with the possibility of content smudges, handwritten corrections, and writing style differences. Pronouncing this tendency of low-quality data is the fact that data collection in the medical context must be noninvasive and consensual which could translate to moving the capturing device at a distance or angle to avoid inconvenience.

Survey results showed that errors committed during data entry in clinical databases range from 2.3% to 26.9% which roots from data entry mistakes and misinterpretation of information [8]. Adriano et al. [3] aimed to reduce the high error rate of data entry using OCR applied on novel digital conversion model for hand-filled forms. Their dataset came from a selected special database that readily provided forms (containing handwritten text) to facilitate character recognition and training of classifiers. Their best-performing pipeline used feature extraction via AlexNet, a convolutional neural network architecture. They recommended other CNNs for exploration namely ResNet and Squeezenet, as well as using other SVM kernels like Gaussian and RBF— points of work that presents great utility. Exploring scanned medical prescriptions, a camera inside an IoT-enabled smart medicine box embedded with OCR technology was explored by the work of Rumi et al.[9]. This targets elderly patients who cannot monitor their medication by notifying an individual about the medication information extracted from their respective prescriptions. The paper's focus on scanned prescriptions can be further extended to another clinical setup like patient information displayed on medical devices which not only requires text recognition but also correct mapping of numerical health values to their corresponding health parameters (e.g., heart rate, blood pressure).

In a work by Xue et al.[10], a text detection and recognition pipeline considered two real-life scenarios in the medical scene: (1) multilingual laboratory reports, and (2) documents with many textual objects each occupying a very small region. The authors proposed a deep learning approach that performs a patch-based training strategy applied to a detector that outputs a set of bounding boxes containing texts. A concatenation structure is then inserted into a recognizer that takes the areas of bounding boxes in the original image as input, thereby outputting the recognized texts. The patch-based strategy enforced by the authors in text detection module achieves 99.5% recall and 98.6% precision, a desirable result given the average quality of images. Likewise, the concatenation structure effectively improved the recognition performance by being able to deal with images with different resolutions at 90% accuracy. Their patch-based strategy during text detection may be something worth looking into given its contribution to achieve desirable success outcomes in terms of recall and precision.

OCR works efficiently with printed text documents [11]. However, as mentioned above, medical data does not include textual forms alone but spans widely across different medical devices as such as blood pressure monitors and patient monitor systems among many others. Few datasets exist such as the Queensland [12] and VitalDB [13] dataset but they are both high-fidelity vital signs database designed for anesthesia monitoring research and biosignal analysis, respectively. To the best knowledge of the researchers, there is a lack of camera-captured patient monitor images reflecting actual environment conditions (e.g., illumination variation, background noise, etc.) which are essential in optical character recognition.

In the work of Kulkarni et al. [14], OCR was used to digitize camera-captured blood pressure readings through a mobile application. The paper underscored medical data transcription errors as well as relatively inadequate technologies in low- to middle-income countries (LMICs). The use of ubiquitous phone camera to detect LCD frame location provides a cost-effective solution to facilitate OCR without the need for expensive software or high-end capturing devices Their modular image enhancement algorithm including image binarization, LCD frame localization, and LCD frame normalization may also be used as reference when applied on a similar medical tool like patient monitor. Similar to previous works, low image quality was found to significantly degrade their model accuracy, hence post-OCR correction may be applied.

A study by Shenoy et al. [15] developed a smartphone-based system that automatically reads and records biometric monitor results from a camera-captured monitor reading. This was, however, limited to seven-segment displays and does not involve recognition of alphanumeric content as observed in a patient monitor screen. Its target device is also limited to Apple's HealthKit in iOS, which leads to less generalizability but poses points for open work.

Storage itself of extracted information is as equally important as text recognition to facilitate research, drive business decisions, and assist in forecasting and policy making. However, medical devices and screens may have limited hardware capabilities to store and export data for further clinical research and diagnosis. This is particularly the case for LMICs, where technology may not be as advanced as other countries [14].

Document archiving and record management was explored with application of optical character recognition in the paper of Jayoma et al. [16]. The authors of such document archival study focused on digitization of multiple forms of records in the Department of Social Worker and Development (DSWD) Caraga. Their general framework consolidating OCR and information storage used open-source technologies such as Django, MySQL, and Pytesseract which can be used as references to develop a system using similar technologies. This can further be extended in terms of a different dataset (i.e., images from medical devices).

In a work by Yadav et al.[17], a robust web application that uses OCR to extract information from handwritten and printed documents was developed. Their technical architecture comprised four sequential processes namely (1) adaptive thresholding, (2) connected component analysis, and (3) line and word detection, and (4) two-layer text recognition. Specifically, the use of adaptive thresholding to account for variations in illumination in the image dataset may serve as reference in image preprocessing of different dataset. The study showed the feasibility of text recognition hosted online.

A system built by Froese et al. [18] extracted the desired information from real-time pump monitor images. Their methodology mainly used scripting to extract images from a medication pump which is then fed to an OCR model. Recognized text and values are then transferred to a real-time monitoring software. It was underlined that future work is required for more universal application of such system which can be explored by superimposing their model on a different medical dataset and assessing the accuracy. Their data collection setup through a USB camera capturing images from the medical pump at 60 frames/second can be employed in my paper. By observation, data capture used in such paper was relatively near the pump (i.e., the USB camera is immediately in front of the device). Their capturing conditions can be extended in this study by incorporating more realistic scenarios such as the camera slightly tilted or skewed with respect to the patient monitor. Hence, further image optimization encompassing variation in camera face angles can be explored [18].

The feasibility of using OCR to extract information from a patient monitor screen was also shown in Bukhari's work [19]. Various image preprocessing such as binarization and bitwise masking were used on a high-quality dataset retrieved from SimCapture. The OCR pipeline used in such paper includes a script that extracts frames per second from input video and individually extracts health values eventually saved in a CSV file. This may serve as basis of the proposed system to implement a data export functionality in order to provide the user a downloadable file consisting of the extracted information in easily editable format. Future work was encouraged which can be summarized in three parts: (1) more image preprocessing to ensure that the model is dynamic, (2) automatic detection of all pixel color values of parameter for classification, since only 4 colors are considered in the paper, and (3) use of deep learning models in contrast to traditional image processing techniques. Given that high data quality is required to maintain the model accuracy [19], the study may be extended to be applied on patient monitor dataset taken from a real medical setting with environmental factors present such as brightness variation, blurring, distant capture, etc.

With all considered, data entry errors being committed in healthcare—let alone the tedious process of such task—slows down clinical procedures and leaves plenty of room for improvement. It was further pronounced that dataset quality is a key consideration in developing an accurate OCR model, upon which OCR post-correction methods and several image preprocessing techniques are possible workaround. To this end, the research aims to fill in the gap among previous studies through (1) use of smartphone camera to collect and curate realistic field image dataset of patient monitor, (2) creation of an image preprocessing pipeline to improve image quality, and (3) development of a system to utilize the image preprocessing pipeline to enhance raw image of patient monitor. Contributing a new set of patient monitor images and developing an image preprocessing pipeline to enhance such images would provide a benchmark dataset and development of real-time OCR applications in a similar domain.

III. Theoretical Framework

A. Patient Monitor Screen

Patient monitoring system was introduced by Venetian Doctor Santorio in 1965 through his publication of methods to measure body temperature using spirit thermometer and pendulum for counting heart rate. With the advent of integrated circuits and advancement of technology, computer-based patient monitoring systems with better computing power have been developed. A widely used medical device is a patient monitor screen which continuously monitors patient parameters such as oxygen level, heart rate, blood pressure, etc. These data are observed via non-intrusive sensors on human body to check the condition of the patient over time which facilitates prompt assessment and decision-making relative to real-time patient status such as those coming straight from surgery in Intensive Care Units (ICUs). A standard patient monitor [20] based from is shown in Figure 1.

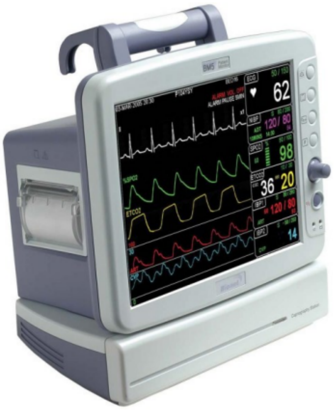


Figure 1: Standard Patient Monitor

A notable trend among patient monitors is that the numerical values are highly contrasted with a black background, with characters displayed in synthetic fonts. As of date, these medical devices are still widely used to monitor patients not just in the medical sector but also in social support such as retirement homes.

B. Image Annotation

Image annotation is the task of assigning labels to an image to create metadata for a training dataset in computer vision models. The model utilizes such annotations as its ground truth, and uses them to learn how to label or detect objects or images on its own. Image annotation is typically useful in object recognition, or object detection, which enables machines to identify a particular object in an image and apply the accurate label. An example is a self-driving car which labels its surroundings depending on whether vehicles and/or obstacles are nearby.

C. Image Preprocessing

The aim of image preprocessing is quality improvement by suppressing undesired distortions and enhancing some features to obtain more suitable data for further processing and analysis tasks.

1. OpenCV

Open-source computer vision (OpenCV) is an image preprocessing library that has gained popularity in computer vision given it is open-source. It was originally envisioned to support computer operations such as object identification, image recognition, and object movement tracking but has now expanded to over 2500 functions based on its documentation. This enables faster execution of tasks such as color conversion, image masking, and filter application. Furthermore, its interface flexibility allows for multiple programming languages such as Python, Java, and C as well as different platforms such as Mac and Windows. Figure 2 presents the architecture design of OpenCV in a mobile imaging work [21].

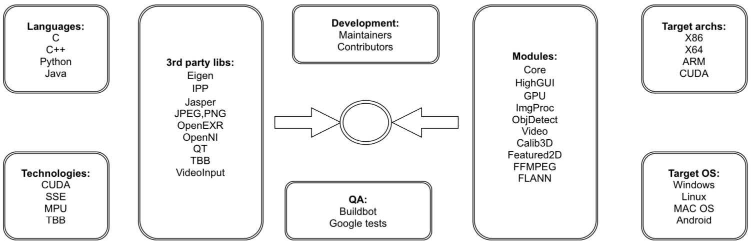


Figure 2: Architecture of OpenCV

In comparison with other similar tools like Matlab, OpenCV provides a relatively detailed toolbox for image processing instead of generic solutions. The wide array of functions in OpenCV also efficiently integrates common noise removal and image quality manipulation techniques in one library.

2. Grayscaling

Most OCR engines normally perform better with grayscaled images which refers to a color space with only one channel. Pixels in typical images are represented in Red-Green-Blue (RGB) format which gives them the color that the human eye perceives. There are three ways on how to compute the new value of pixel from RGB: average, lightness, and luminosity. The average method takes the simple arithmetical mean across the color channels of certain pixel. Lightness is computed by averaging the maximum and minimum value of pixel color channel. Lastly, luminosity works with the average of all color channels, with every single channel weighted. Formulas for these conversions are shown in formulas (1), (2), (3).

$$lightness = (max(R, G, B) + min(R, G, B))/2 \tag{1}$$

$$average = (R + G + B)/3 \tag{2}$$

$$luminosity = 0.299(R) + 0.587(G) + 0.115(B) \tag{3}$$

3. Gamma Correction

This technique can be used to control the brightness of an image. Such method is typically used in image preprocessing to adjust the image brightness depending on how it was captured. Gamma values less than 1 will shift the image towards the darker end of the spectrum while gamma values greater than 1 will make the image appear lighter. A gamma value exactly equal to one will result in no change in image [22].

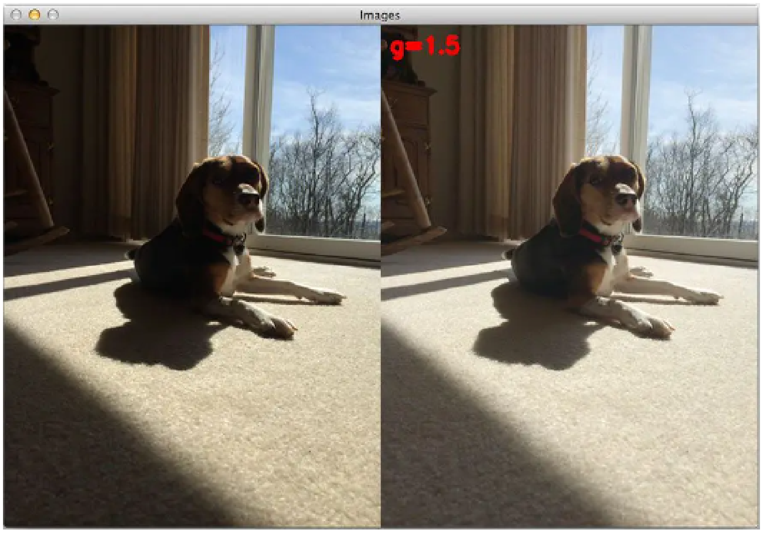


Figure 3: Gamma Correction

4. Canny Edge Detection

OCR generally performs better if the object of interest is narrowed down from the input image. For instance, a scanned receipt may be slightly skewed, with other non-essential objects included in the same image (e.g., pen, person, etc.). Edge detection is a technique that aims to extract the four corners of an object of interest such as documents or monitor display. One popular edge detection approach is Canny Edge Detection. The entire process of this detection [23] is summarized in Figure 4.

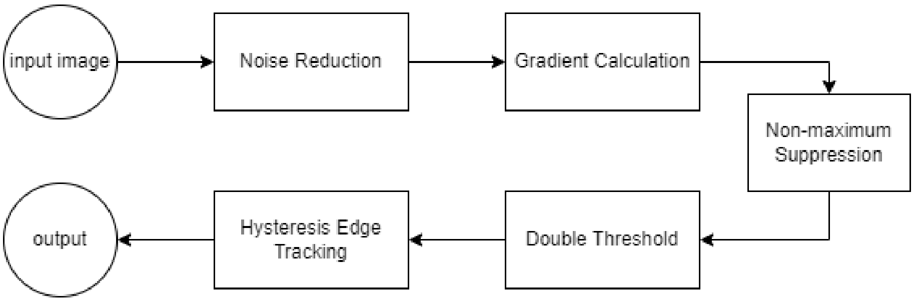


Figure 4: Canny Edge Detection

- **Noise Reduction via Blurring**

  Edge detection results are particularly sensitive to image noise and one way to address this is through the application of Gaussian blur to smooth an input image. To do so, image convolution technique is applied on an input image with a Gaussian Kernel which may have varying kernel size such as 3x3, 5x5, etc. The kernel size influences the intensity of blur, where higher value leads to more visible blur effect.

- **Gradient Calculation**

  This step detects the intensity of edges as well as direction via calculation of the gradient in the image using edge detection operators. A change of pixels' intensity represents an edge. Filters can be applied in order to highlight such intensity change in horizontal and vertical directions and easily detect the edges.

- **Non-maximum Suppression**

  Thin edges are ideal in the output images. Hence, presence of thick edges can be addressed through non-maximum suppression to thin them out. The algorithm essentially iterates through every point on the gradient intensity matrix and locates the pixels whose value in the edge directions is maximum.

- **Double Threshold**

  The goal of this step is to identify three kinds of pixels namely strong, weak, and non-relevant:

  - Strong pixels are those with relatively high intensity that assures as about their contribution to the final edge.
  - Weak pixels are those with intensity that is neither high or low enough to be considered strong or non-relevant, hence are still potentially contributors in the edge.
  - Any other pixel not classified under the two aforementioned types belong to this class.

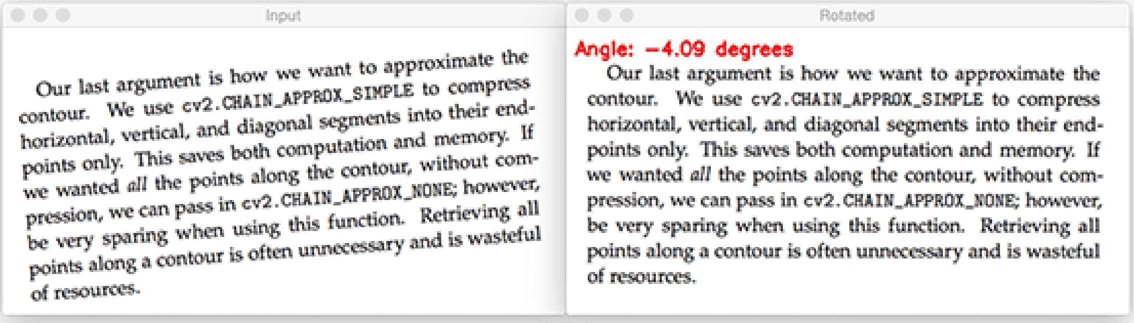
  With this considered, high threshold is used to identify the strong pixels while low threshold is used to identify the non-relevant ones. On the other hand, the rest of the pixels having intensity between both thresholds are identified as weak which are then further filtered out by the next step to delineate whether it ultimately belongs to strong or non-relevant.

- **Edge Tracking by Hysteresis**

  Based on the threshold results, the hysteresis consists of transforming weak pixels into strong ones, if and only if at least one of the pixels around the one being processed is a strong one [23].

5. Skew Correction

Raw image content, especially text, sometimes tend to be skewed or tilted at a certain angle. This is contributed by the point of capture where the camera is not leveled with that of the object. For computer vision tasks, skew correction is essential to improve model accuracy by ensuring as much *visually normal* input as possible. Python has libraries to implement correction of perspective like OpenCV [24].



Our last argument is how we want to approximate the contour. We use cv2.CHAIN_APPROX_SIMPLE to compress horizontal, vertical, and diagonal segments into their end-points only. This saves both computation and memory. If we wanted *all* the points along the contour, without compression, we can pass in cv2.CHAIN_APPROX_NONE; however, be very sparing when using this function. Retrieving all points along a contour is often unnecessary and is wasteful of resources.

Angle: −4.09 degrees

Our last argument is how we want to approximate the contour. We use cv2.CHAIN_APPROX_SIMPLE to compress horizontal, vertical, and diagonal segments into their end-points only. This saves both computation and memory. If we wanted *all* the points along the contour, without compression, we can pass in cv2.CHAIN_APPROX_NONE; however, be very sparing when using this function. Retrieving all points along a contour is often unnecessary and is wasteful of resources.

Figure 5: Skew Correction with OpenCV

In machine learning, especially computer vision, the quality of the data is just as important (if not more) as the model itself. Hence, performing necessary image preprocessing procedures on raw images have significant contribution toward noise reduction and overall positive effect on model training.

IV. Design and Implementation

Ethical approval from UP Manila Research Ethics Board (UPM REB) and Philippine General Hospital Expanded Hospital Research Office (PGH-EHRO) is obtained to proceed with the manual collection of dataset.

A. Data Collection Setup

A smartphone camera (iPhone 11) is mounted on a tripod to capture data from a Mindray Beneview T8 patient monitor attached to five (5) healthy volunteers at the Post-Anesthesia Care Unit (PACU) of PGH. A sample image of the monitor is illustrated in Figure 6.

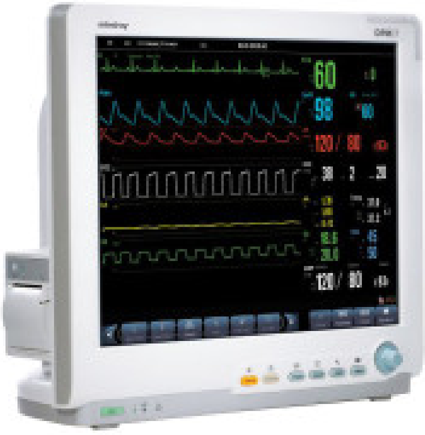


Figure 6: Beneview T8 Patient Monitor

The inclusion criteria for data acquisition were as follows:

- Aged 18 - 65 years old
- Student, or faculty from College of Arts and Sciences and/or College of Medicine

Vital signs data are recorded at a resolution of 1920 x 1080 at 60 frames/second. Every volunteer session lasts for 30 minutes and camera placement is adjusted every 3 minutes to account for different capture conditions. The tilt and angle to which the camera was skewed are manually adjusted up to a maximum of 45 degrees.

B. Dataset Annotation

In this paper, a practical heuristic for bounding box annotation on the proposed image dataset is presented through a trained object detection model to automate the manual approach. This intends to reduce workload by shifting the majority of human involvement to the correction stage only.

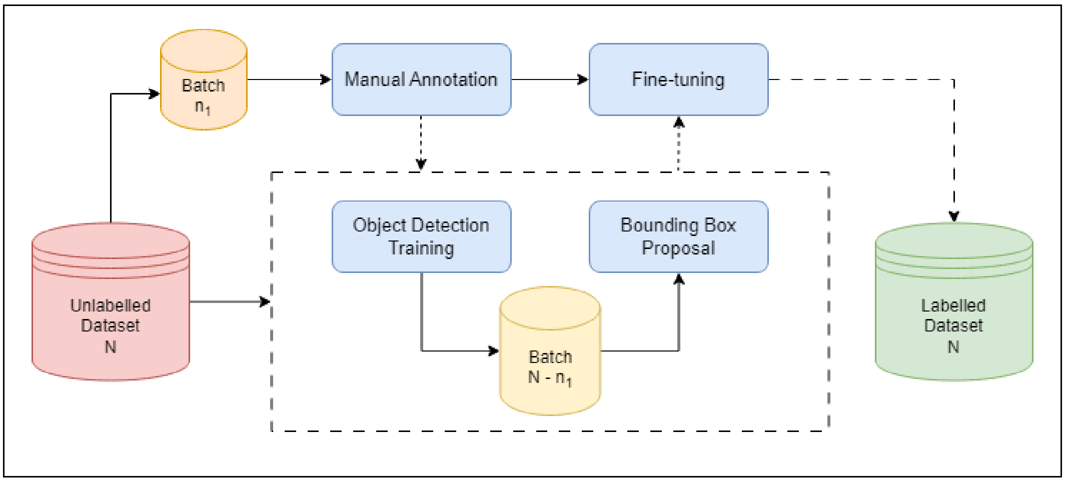


Figure 7: Semi-auto Annotation Methodoloy

1. Manual Annotation

The process begins with domain experts manually annotating a randomly selected batch of images (n_1) from the unlabelled dataset. The annotation involves full human involvement to draw bounding boxes around health parameters and provide their corresponding class labels. The open-source annotation software LabelImg is used with no speed-up procedures.

2. Object Detection Training

The next step is to train an object detection model. Transfer learning is applied by using a pre-trained SSD network and fine-tuning on the proposed dataset [25]. The single shot detector (SSD) network [26] proposed by Liu et al. is used for the detection architecture given its lightweight nature. It is pre-trained with MS COCO dataset and is typically the model of choice for resource-limited inference scenarios given that the detections are produced directly in a single forward pass of the network [27]. Furthermore, the Mobilenet V2 [28] is applied for the backbone.

3. Bounding Box Proposal

The trained model is used to predict bounding boxes for the unlabelled images with an associated confidence level for each detection. A confidence threshold value between 0 and 1 is used to define a true positive. In other words, the model will only draw bounding boxes around a detected parameter if and only if its associated confidence level is equal to or higher than the specified threshold.

4. Manual Fine-tuning

The resulting annotations proposed by the model are inspected and manually corrected by the domain experts through several corrective measures as follows:

- **Addition**: Missing bounding box is manually drawn around a parameter, if needed.
- **Removal**: Incorrectly predicted box is deleted from the annotation.
- **Label Correction**: Mislabeled class is corrected.
- **Box Adjustment**: If the predicted box is too wide or insufficiently encloses a parameter, the box is recalibrated accordingly.

5. Workload Estimation

We estimate human workload by comparative analysis of how much time is spent between the manual and semi-auto annotation strategies. For the manual approach, the total time (T) to complete the annotation as described in Section B.1 is measured with a timer. The average (t) is then calculated using the formula $\frac{T}{n_1}$ which corresponds to the estimated time to annotate a single image. This value is then multiplied to the total number of images in the dataset to estimate the overall duration to label the dataset exclusively through a manual approach. On the other hand, the semi-auto annotation strategy is measured by adding the time consumed both in Section B.1 and Section B.3.

6. Labelled Dataset

After the correction stage, the fully labeled image dataset is saved as a ZIP file containing the images in JPG format and their corresponding annotations in Pascal VOC XML format. This is done for both raw dataset and its preprocessed counterpart (screen-extracted).

To enable easier navigation of the dataset, files are named as follows: volunteer number_ file code _ frame count. For instance, the image with file name *01_01_1.jpg* corresponds to the first extracted frame from the first volunteer data with a direct camera and natural lighting condition.

File Code	Capture Orientation	Lighting Condition
01	Direct Camera	Natural
02	Direct Camera	Low
03	Skewed to Left	Natural
04	Skewed to Left	Low
05	Skewed to Right	Natural
06	Skewed to Right	Low
07	Skewed Upward	Natural
08	Skewed Upward	Low
09	Skewed Downward	Natural
10	Skewed Downward	Low

Table 1: Dataset File Naming

C. Image Preprocessing Flowchart

Figure 8 summarizes the proposed image preprocessing pipeline.

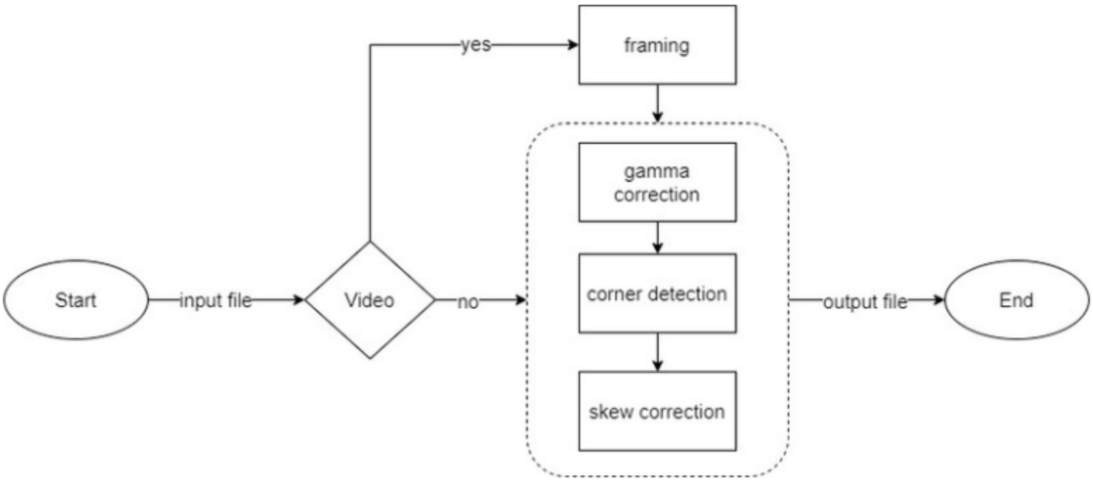


Figure 8: Image Preprocessing Pipeline

1. Framing

The OCR cannot process an input file in video format, hence frames are extracted. The video dataset is fragmented into individual images at 2-second intervals.

2. Image Preprocessing

Once framing is done, the brightness of an image is automatically adjusted using the concept of dynamic inverse gamma correction. Afterward, the brightness of an image is automatically fine-tuned using dynamic inverse gamma correction followed by image smoothing to blur the image. It is followed by edge detection to identify the edges of the patient monitor to be extracted. The proposed flow for this detection is shown below.

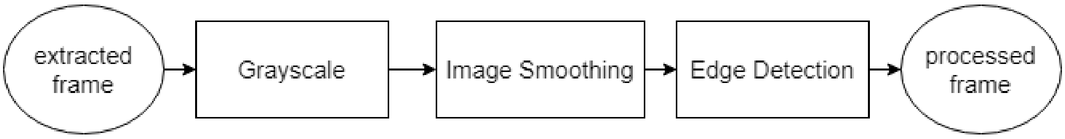


Figure 9: Edge Detection Sub-Explosion

Lastly, skew correction is applied using OpenCV python library to address any degree of skewness in the image information.

D. System Architecture

Monixor is a web application that uses PostgreSQL as the database server. It is developed using the Python-based framework Django to enable easier integration with machine learning, image preprocessing, and optical character recognition implementations.

E. Technical Architecture

The minimum requirements for the server machine include:

- Apache 2.4.23
- 1GB RAM
- PostgreSQL 14

The client-side must satisfy these minimum requirements:

- Google Chrome 57.0.2897
- Mozilla Firefox 43.0.1
- Windows 7 / Android 7.0+ / iOS 12.4+
- Intel Core i5-4200U
- 4GB RAM

V. Results

A. Dataset

A total of 4,674 images saved in JPG format were obtained after deleting extracted frames with visible human subject/s to maintain data anonymity. Table 2 presents image samples classified into one of the 10 classes.

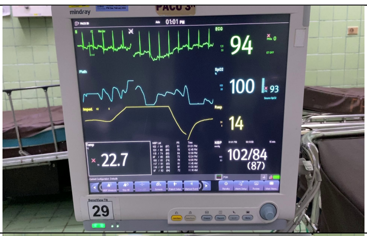
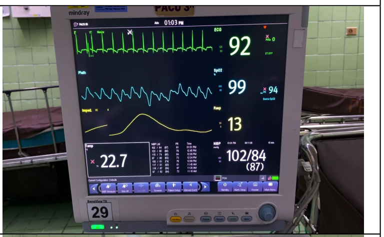
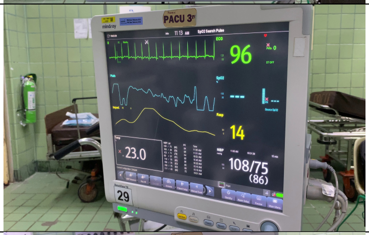
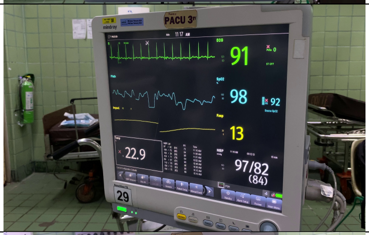
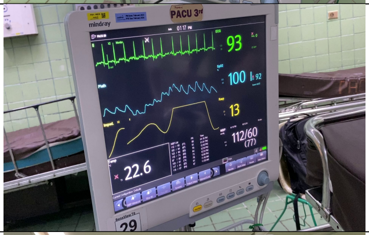
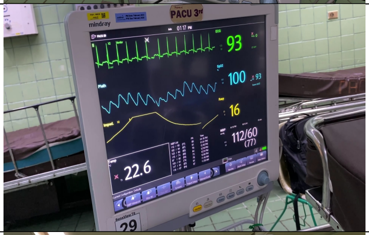
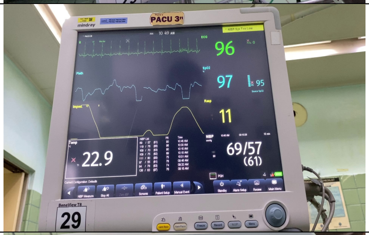
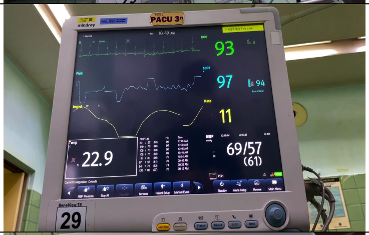
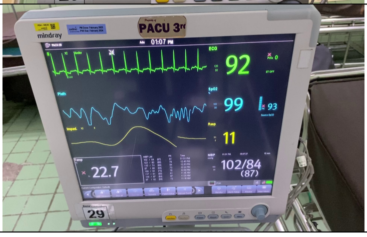
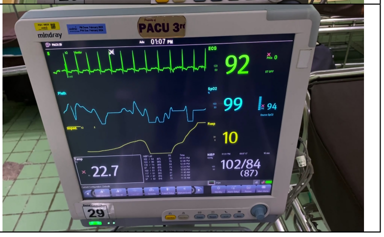
Monixor Dataset ($N = 4674$)		
Camera Orientation	Lighting Condition	
	Natural	Low
Direct ($n = 917$)		
Skewed to Left ($n = 933$)		
Skewed to Right ($n = 949$)		
Skewed Upwards ($n = 964$)		
Skewed Downwards ($n = 911$)		

Table 2: Dataset Image Classes Overview

1. Manual Annotation

A total of 250 images were randomly selected from the unlabelled dataset in which each class had 25 representatives. Seven health parameters were considered as objects and labeled as follows:

1. heart rate $< heartrate >$
2. oxygen saturation $< oxygensaturation >$
3. pulse rate $< pulserate >$
4. respiratory rate $< respiratoryrate >$
5. blood pressure $< bloodpressure >$
6. mean arterial pressure $< map >$
7. temperature $< temperature >$

These labels are in accordance with the official manual of Mindray Beneview T8 monitor [29] and as confirmed by a resident anesthesiologist in PGH.

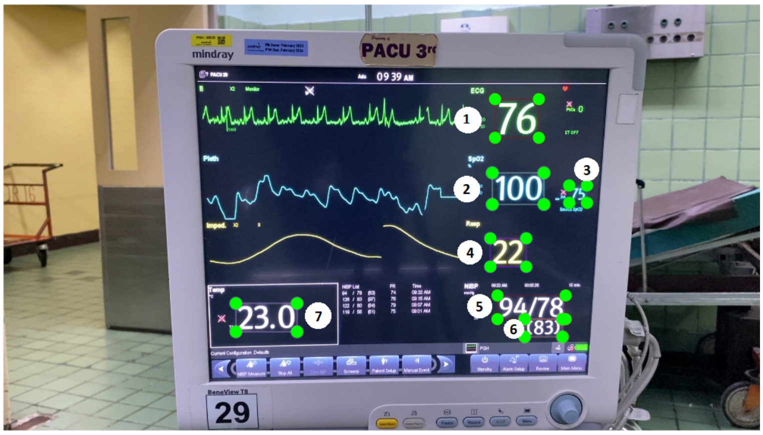


Figure 10: Manual Annotation using LabelImg

2. Semi-Auto Annotation

An 80-20 data split was applied for model training, allotting 200 images for train data and the remaining 50 for testing.

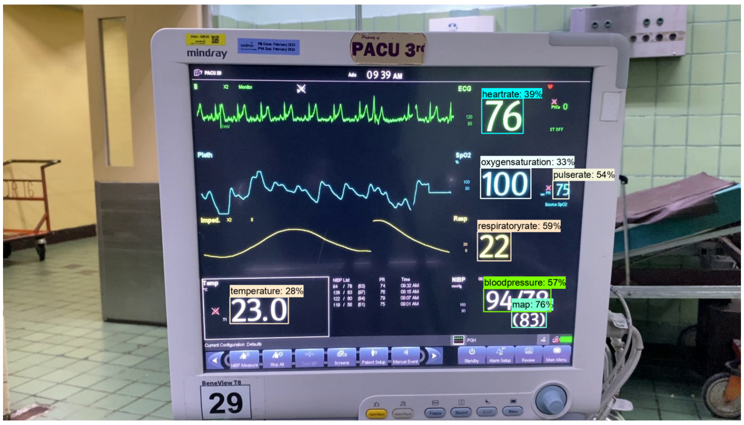


Figure 11: Automated Annotation using Object Detection

The trained model was then applied to automate the annotation for the rest of the unlabelled dataset as illustrated by Figure 11. To speed up the process, an auto-labeling tool was used in which a confidence score threshold of 0.2 was declared [30]. This means that any object detected by the model with at least 20% confidence will have bounding boxes drawn around it.

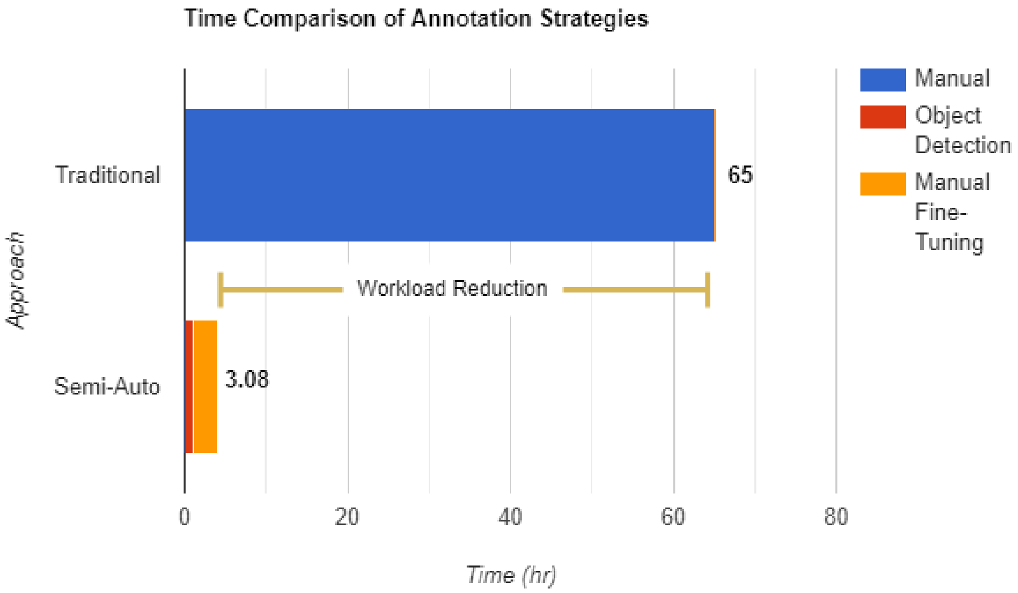


Figure 12: Semi-auto Annotation Workload Reduction.

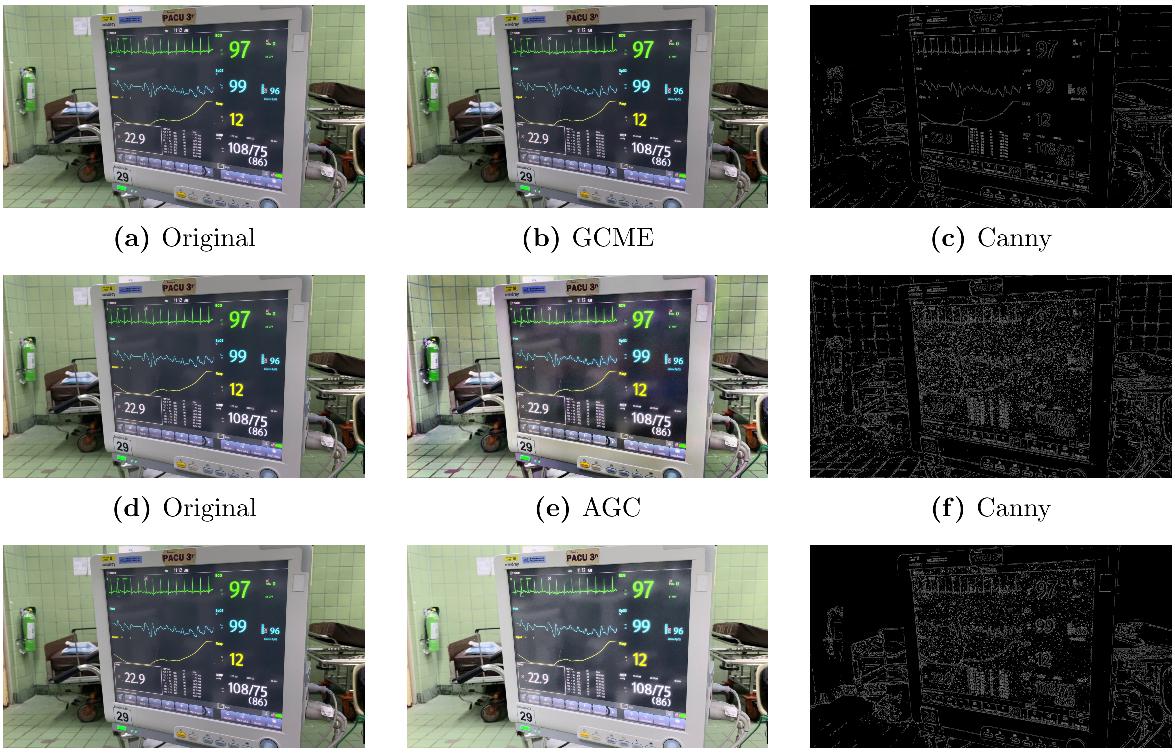
Figure 12 shows the workload reduction in terms of time. The traditional approach of manually annotating an image takes approximately 50 seconds. This translates to roughly 65 hours of projected time in order to annotate the entire dataset. On the other hand, the object detection model annotated the entire dataset at around 5 minutes only with an additional 3 hours incurred for manual fine-tuning of the results. The proposed semi-auto annotation method expedited the manual process by 22 times. The methodology for time measurement is discussed in Section B.5.

B. Image Preprocessing Pipeline

In addition to a newly accrued dataset with multi-parameter annotation, a preprocessing pipeline was created to extract the screen of the patient monitor. By doing so, non-textual elements which may hinder future tasks of optical character recognition were removed while providing a well-defined region containing only the necessary details. The preprocessing was divided into three stages namely (1) gamma correction, (2) edge detection, and (3) skew correction.

1. Gamma Correction

Three dynamic gamma correction techniques were compared namely Blind Inverse Gamma Correction with Maximized Differential Entropy (GCME) [31], Adaptive Gamma Correction (AGC) [32], and Improved Adaptive Gamma Correction with Weighting Distribution (IAGCWD) [33]. Two sample images (natural and low lighting) were judged whether visually satisfactory or not. Basic application of Canny Edge detection ($min_{thresh} = 40$) was also implemented without extra enhancement procedures to see an immediate effect on the detection of contours.



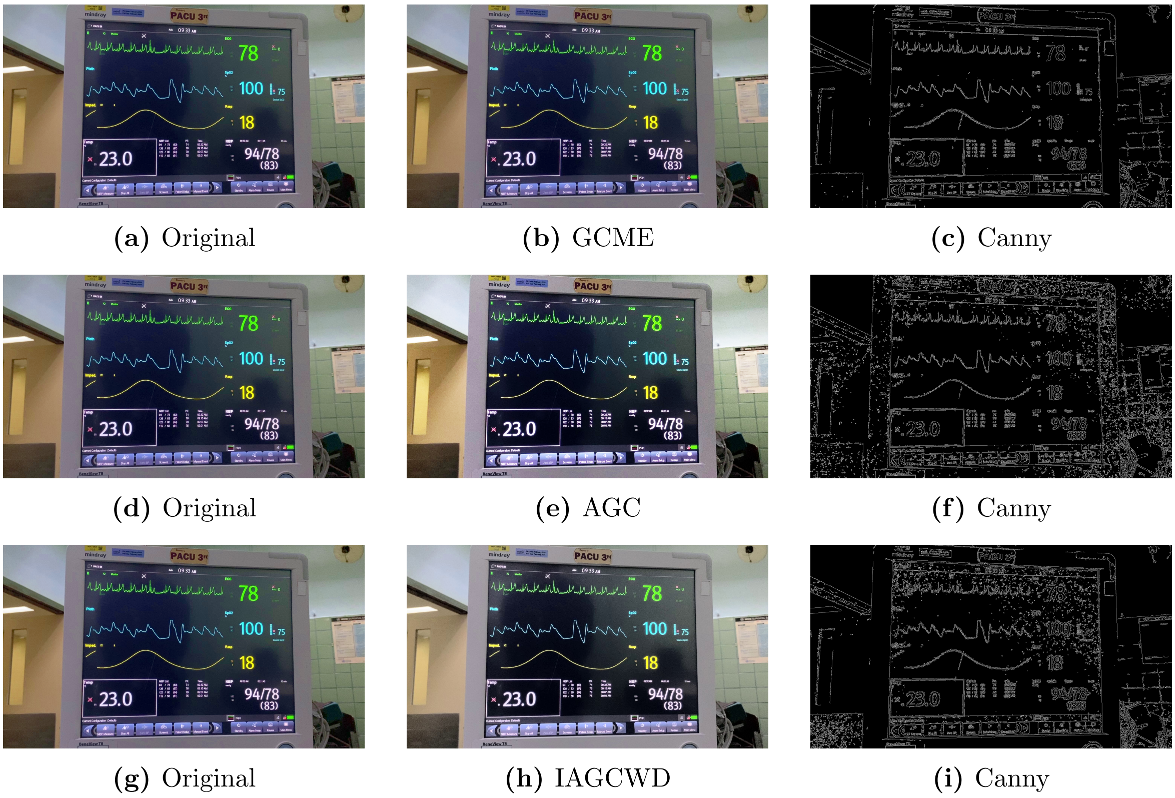
(a) Original **(b)** GCME **(c)** Canny

(d) Original **(e)** AGC **(f)** Canny

(g) Original **(h)** IAGCWD **(i)** Canny

Table 3: Comparison of Dynamic Gamma Correction on an Image with Natural Lighting

Table 3 shows the effect of different gamma correction techniques on an image with natural lighting. Ideally, gamma correction should be able to enhance the brightness of an image while improving the visibility of edges. As shown in (c), GCME best preserved the continuity of edges. On the other hand, AGC and IAGCWD produced irrelevant contours (or image artifacts) as presented by the application of edge detection despite the successful adjustment of image brightness.



(a) Original (b) GCME (c) Canny

(d) Original (e) AGC (f) Canny

(g) Original (h) IAGCWD (i) Canny

Table 4: Comparison of Dynamic Gamma Correction on an Image with Low Lighting

Table 4 shows a similar finding on a relatively darker image, where the GCME technique performed superior over the others in terms of minimizing image artifacts that may hinder successful edge detection. Table 5 shows a quantitative assessment of the two image samples after gamma correction similar to metrics used by Sara et. al [34].

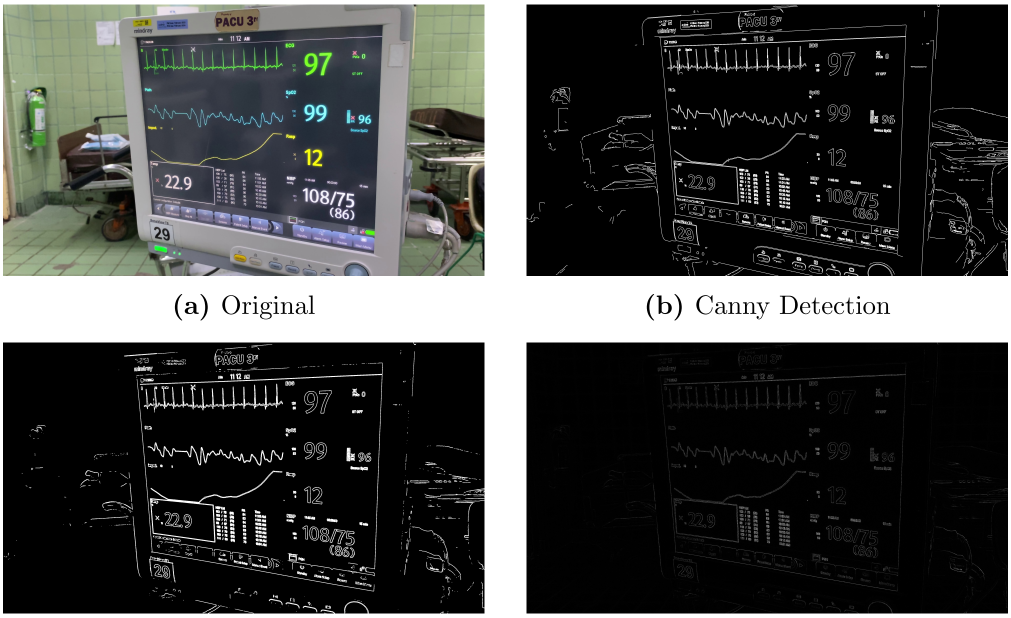
Image	Method	Quality Assessment Techniques		
		MSE	PSNR	SSIM
Natural Light	GCME	**71.4269**	**29.5922**	**0.99**
	AGC	520.7562	28.7108	0.9104
	IAGCWD	1476.1134	27.4494	0.9185
Low Light	GCME	**616.8686**	**28.1407**	**0.89467**
	AGC	753.3069	28.1144	0.82192
	IAGCWD	1908.7040	28.6277	0.8573

Table 5: Error Deduction Summary for Image Quality Metrics (MSE, PSNR, SSIM)

A lower MSE means that the processed image is closer to the original image in terms of pixel values. On the other hand, higher PSNR and SSIM mean that the processed image has less distortion relative to the original image. Since GCME performed better considering lower MSE values and higher PSNR and SSIM values for both image samples, such gamma correction technique was adopted for the pipeline.

2. Edge Detection

Recent studies found that Canny's algorithm is best suitable for object extraction in most contexts as it yields less number of false edges, especially with noisy images [35, 36]. Table 6 compares it with two other techniques namely Sobel [37], and Laplacian detection [38].



(a) Original

(b) Canny Detection

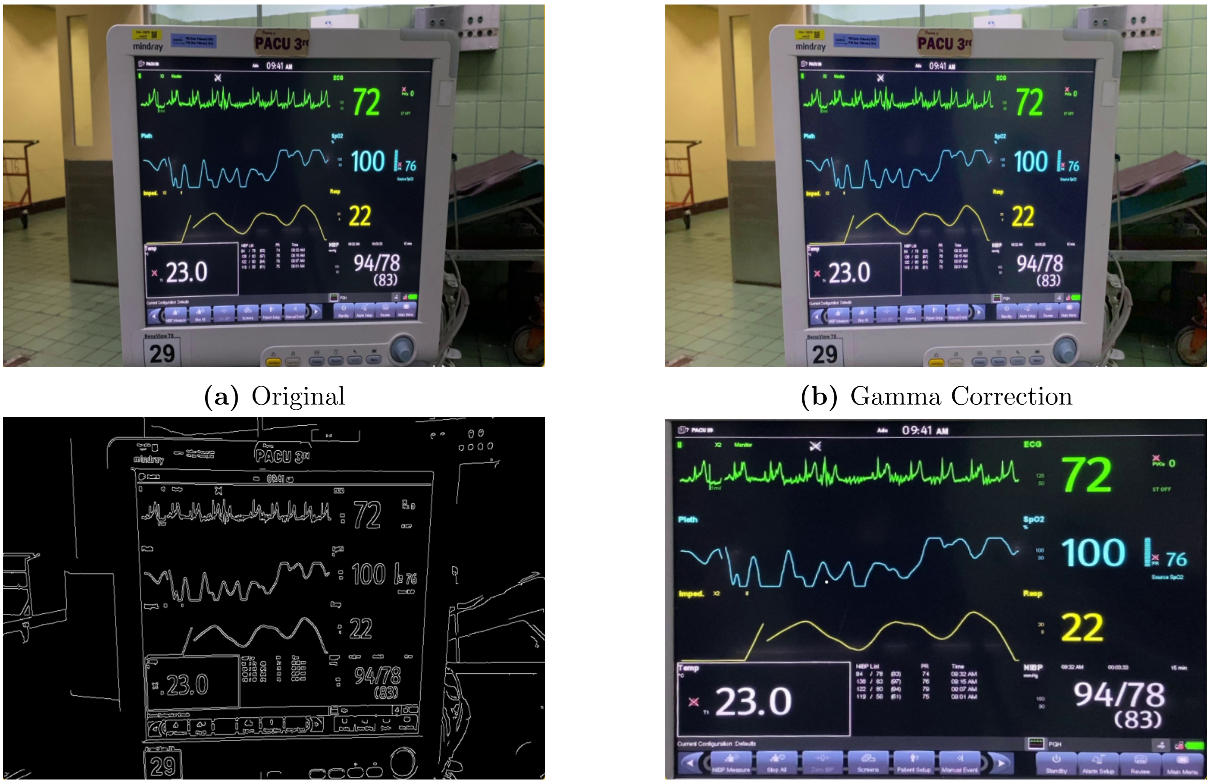
(c) Sobel Detection

(d) Laplacian Detection

Table 6: Edge Detection Algorithms Comparison ($min_{thresh} = 40$)

Close morphological transformation [39] was applied to the raw detected edges for enhancement and restoration of the shape of objects in the presence of edge gaps or discontinuity. The Canny approach performed best in preserving the edges of the patient monitor. On the other hand, Sobel failed to sufficiently detect the upper edge of the monitor. Laplacian was not able to identify the monitor edges at all.

Hence, Canny edge detection was adopted. Lastly, the corner coordinates obtained from edge detection stage were used for skew correction using the OpenCV library *PerspectiveTransform*. Perspective transformation involves mapping points from one perspective to another, thereby changing the perceived viewpoint of the extracted monitor region from the image. Figure 13 illustrates the proposed image preprocessing pipeline for screen extraction.



(a) Original

(b) Gamma Correction

(c) Edge Detection

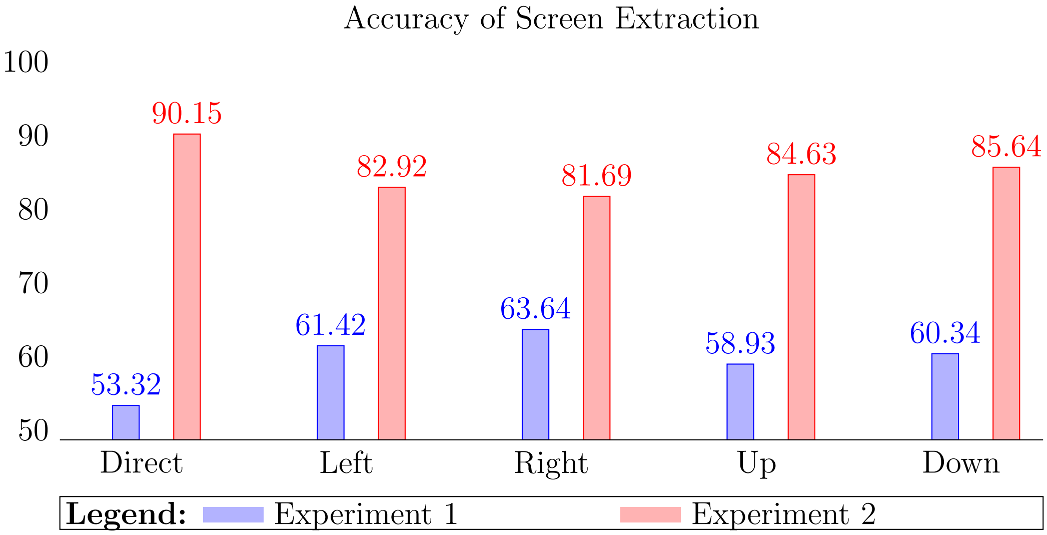
(d) Skew Correction

Figure 13: Image Preprocessing Pipeline Implementation

3. Accuracy

The 50 videos obtained from data collection were uploaded into the web application which implements the proposed pipeline. The accuracy metric is computed by dividing the number of successfully preprocessed images (i.e., screen-extracted) by the total number of frames. The average accuracy was then obtained as the final metric value. Two experiments were done as follows:

- **Experiment 1**. The images directly undergo Canny edge detection without image enhancement techniques other than skew correction for post-processing.
- **Experiment 2**. The proposed image preprocessing pipeline is applied. This includes GCME gamma correction, image smoothing via bilateral filter, close morphological transformation, and skew correction.



4. Processing time

On average, preprocessing an image takes 0.8 seconds. Image resolution used for this assessment is 1920 x 1080.

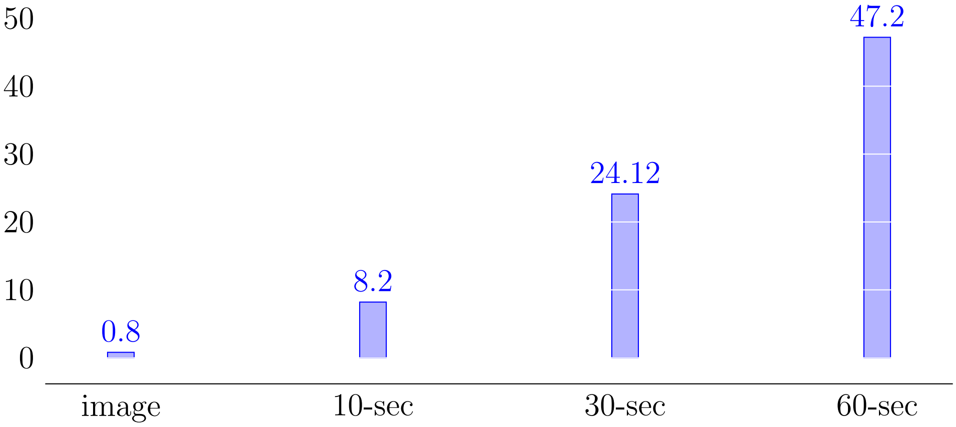
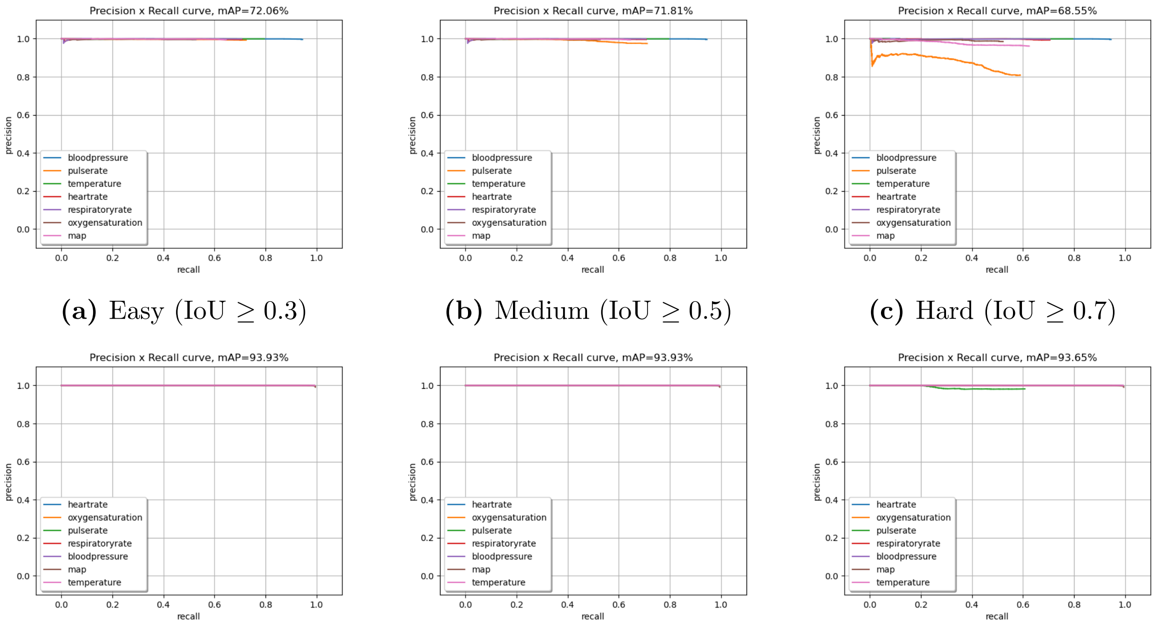


Figure 14: Image Preprocessing Time

C. Optical Character Recognition

This section explores the feasibility of object detection to locate health parameters after screen extraction and extract the values using OCR. For this purpose, the training data from Section B.2 was diversified by introducing a new batch of 250 screen-extracted images to improve the generalizing ability of the model. Following similar evaluation protocols as in the work of Bulatov et al. [40], three configurations of Mean Average Precision (MAP) with different Intersection over Union (IoU) values were used to evaluate the object detection method. The IoU threshold from 0.3 to 0.7 demonstrates the localization requirements from easy to hard. These metrics were calculated using a GUI-based tool for object detection assessment [41]. The semi-auto-annotated dataset served as the ground truth.



(a) Easy (IoU ≥ 0.3) **(b)** Medium (IoU ≥ 0.5) **(c)** Hard (IoU ≥ 0.7)

(d) Easy (IoU ≥ 0.3) **(e)** Medium (IoU ≥ 0.5) **(f)** Hard (IoU ≥ 0.7)

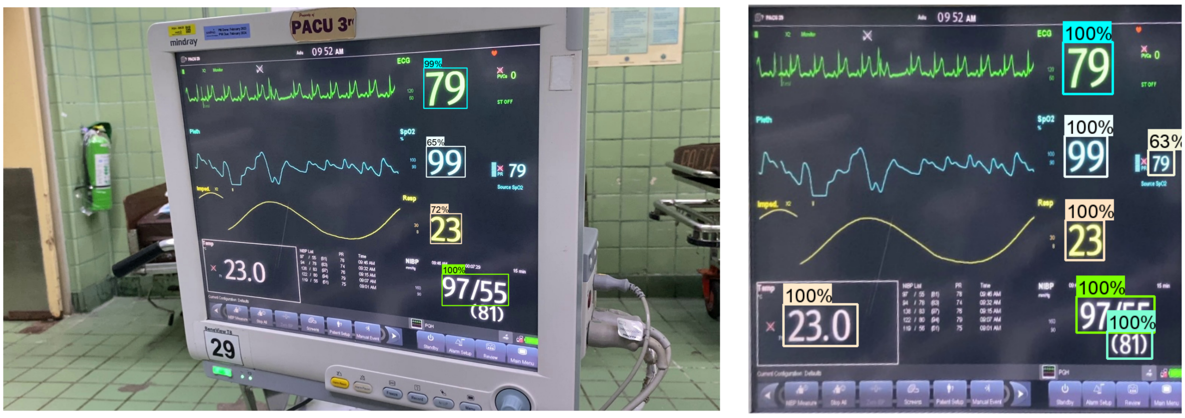
Figure 15: Precision-Recall Curves of Object Detection. a-c) Raw Dataset; d-f) Preprocessed Dataset

Figure 15 shows that the object detection model consistently performed better on preprocessed frames across the three different IoU thresholds. The model performance declined on raw dataset at IoU threshold ≥ 0.7, especially on detecting some parameters such as *pulse rate* and *map*, indicating that boxes could miss out a portion of the values.

	heart rate	oxy. sat.	pulse rate	resp. rate	blood press.	map	temp.
Raw	0.7044	0.5182	0.5214	0.7044	0.9442	0.6116	.7942
Screen	0.9927	0.9920	0.6010	0.9889	0.9929	0.9950	0.9930

Table 7: Average Precision (AP) across Health Parameters (IoU ≥ 0.70)

Table 7 presents the average precision of each health parameter at a strict threshold of 0.7. The object detection model performed generally better on preprocessed images than raw images, with lowest AP metric on the parameter *pulse rate*.



(a) Raw

(b) Preprocessed

Figure 16: Detected Health Parameters at $min_{conf.score} = 0.5$

Figure 16 demonstrates the performance of a more selective object detection model which differentiates true positives from false positives given a minimum confidence score threshold of 50%. Optical character recognition was then applied using EasyOCR library [20] to extract the values inside the detected bounding boxes.

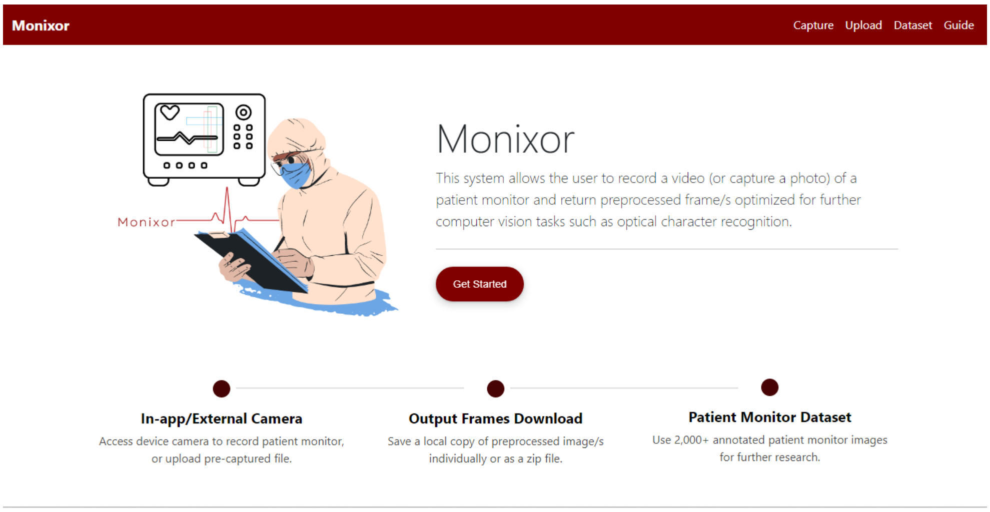
	heart rate	oxy. sat.	pulse rate	resp. rate	blood press.	map	temp.
Raw	’79’	’99’	–	’23’	’97/55’	–	–
Screen	’79’	’99’	’79’	’23’	’97/55’	’(81)’	’23.0’

Table 8: OCR-extracted Vital Signs Data

Table 8 shows that health values were completely extracted on a preprocessed image while three parameters were missed in the case of its raw image counterpart.

D. System

The home page shows an overview of the system's functionalities such as allowing in-app camera access, downloading of preprocessed images, and accessing the dataset. A *Get Started* button is provided to redirect the user to the capture mode.



Monixor

This system allows the user to record a video (or capture a photo) of a patient monitor and return preprocessed frame/s optimized for further computer vision tasks such as optical character recognition.

In-app/External Camera

Access device camera to record patient monitor, or upload pre-captured file.

Output Frames Download

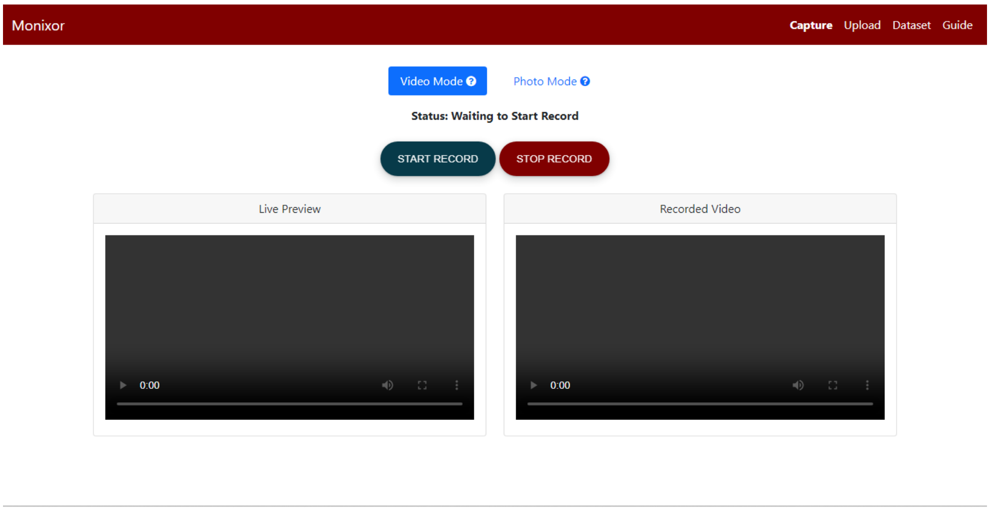
Save a local copy of preprocessed image/s individually or as a zip file.

Patient Monitor Dataset

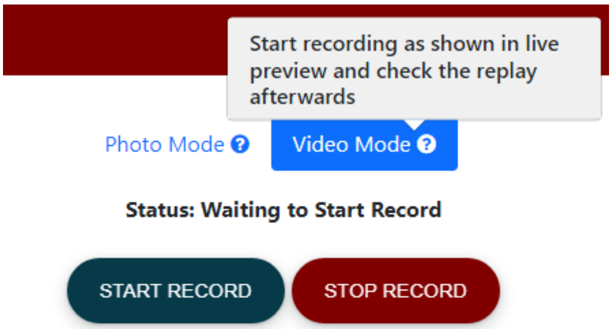
Use 2,000+ annotated patient monitor images for further research.

Figure 17: Home Page

In the *Capture* page, the user can access the device camera and capture an image or video of a patient monitor. A tooltip is provided on each capturing mode namely *Photo* and *Video*.



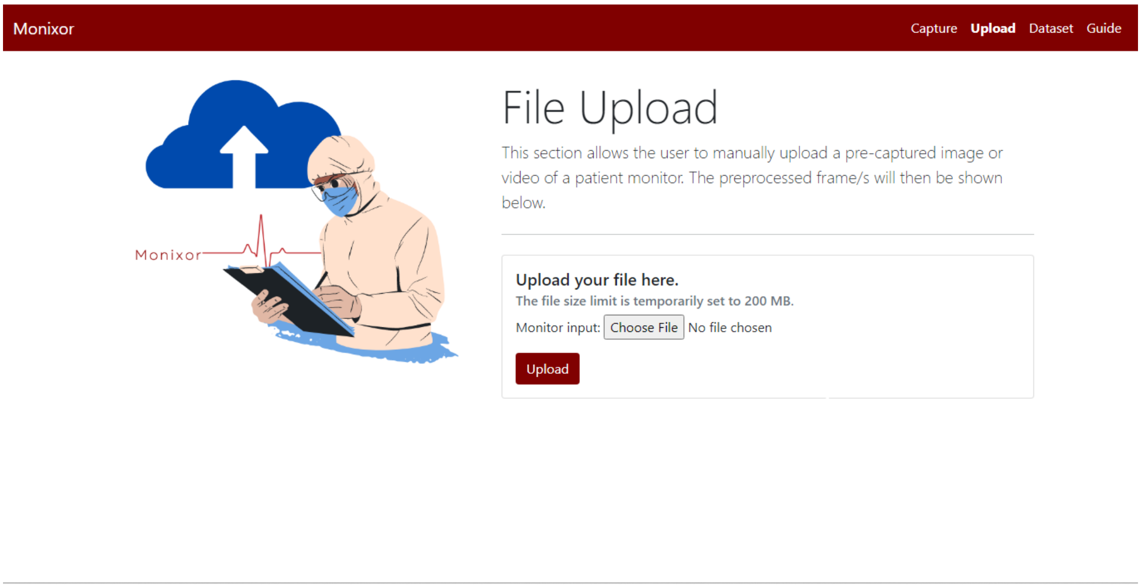
Status: Waiting to Start Record


Start recording as shown in live preview and check the replay afterwards
Photo Mode
Video Mode
START RECORD
STOP RECORD

Status: Waiting to Start Record

Figure 18: In-app Camera Access Page

In the *Upload* page, the user can submit a pre-captured input file. After the submission of input file, the resulting preprocessed image/s will be displayed, and available for download.



File Upload

This section allows the user to manually upload a pre-captured image or video of a patient monitor. The preprocessed frame/s will then be shown below.

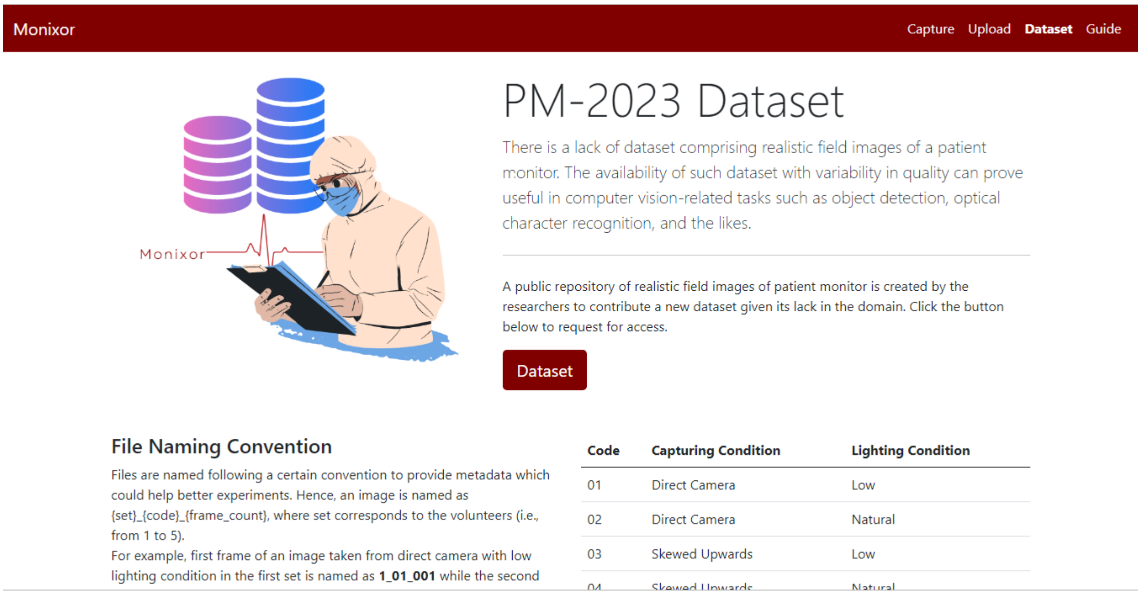
Upload your file here.

The file size limit is temporarily set to 200 MB.

Monitor input: No file chosen

Figure 19: File Upload Page

In the *Dataset* page, the user can access the patient monitor dataset with annotations in ZIP format. The file naming convention is also provided as a guide for navigating such dataset.



PM-2023 Dataset

There is a lack of dataset comprising realistic field images of a patient monitor. The availability of such dataset with variability in quality can prove useful in computer vision-related tasks such as object detection, optical character recognition, and the likes.

A public repository of realistic field images of patient monitor is created by the researchers to contribute a new dataset given its lack in the domain. Click the button below to request for access.

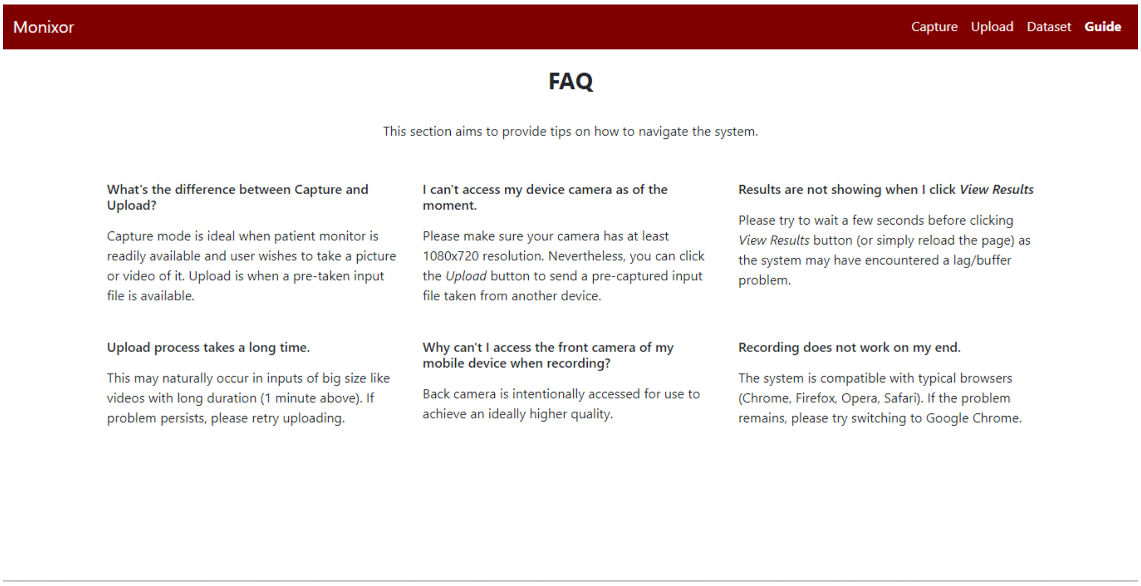
File Naming Convention

Files are named following a certain convention to provide metadata which could help better experiments. Hence, an image is named as {set}_{code}_{frame_count}, where set corresponds to the volunteers (i.e., from 1 to 5).
For example, first frame of an image taken from direct camera with low lighting condition in the first set is named as **1_01_001** while the second

Code	Capturing Condition	Lighting Condition
01	Direct Camera	Low
02	Direct Camera	Natural
03	Skewed Upwards	Low
04	Skewed Upwards	Natural

Figure 20: Dataset Page

The user has the option to navigate the *Guide* page in order to find answers to their questions as they encounter them while using the system.



FAQ

This section aims to provide tips on how to navigate the system.

What's the difference between Capture and Upload?

Capture mode is ideal when patient monitor is readily available and user wishes to take a picture or video of it. Upload is when a pre-taken input file is available.

I can't access my device camera as of the moment.

Please make sure your camera has at least 1080x720 resolution. Nevertheless, you can click the *Upload* button to send a pre-captured input file taken from another device.

Results are not showing when I click *View Results*

Please try to wait a few seconds before clicking *View Results* button (or simply reload the page) as the system may have encountered a lag/buffer problem.

Upload process takes a long time.

This may naturally occur in inputs of big size like videos with long duration (1 minute above). If problem persists, please retry uploading.

Why can't I access the front camera of my mobile device when recording?

Back camera is intentionally accessed for use to achieve an ideally higher quality.

Recording does not work on my end.

The system is compatible with typical browsers (Chrome, Firefox, Opera, Safari). If the problem remains, please try switching to Google Chrome.

Figure 21: Guide Page

VI. Discussions

The semi-auto annotation on a newly collected dataset of patient monitor was shown to expedite the manual approach using object detection model trained on a small subset of the original data. Further, the proposed image preprocessing pipeline to perform screen extraction of the patient monitor is not restricted to only one patient monitor model to crop the screen as it is a contour-based approach. This means it simply bases on the visibility of four corners of the screen in order to extract a well-defined region from the rest of the image. This, however, requires the recording device to sufficiently capture the monitor and has limitations on challenging camera angles that might affect the visibility of edges. Nevertheless, this allows the applicability of the screen extraction method for other models of patient monitors.

Screen extraction has also been shown to improve the accuracy of the object detection model to locate health parameters as it may contribute to (1) reduced complexity and background noise, (2) more consistent image characteristics, (3) enhanced object visibility. Results also showed that the performance of the object detection model declined on locating the parameters *pulse rate* and *mean arterial pressure (map)* which could be attributed to their small size relative to the other parameters as well as similarity of color with respect to the bigger values adjacent to them. This observation is pronounced on raw images since the point of capture is taken from a distance with varying degrees of skew.

VII. Conclusions

This paper presented an annotated dataset of patient monitor reflecting a real hospital environment, together with an image preprocessing pipeline for screen extraction. Such dataset can be instrumental in training and validating computer vision algorithms and models such as vital signs estimation, trend analysis, remote patient monitoring, and alert-aided anomaly detection. This can further enhance the accuracy of computer vision systems in healthcare settings. It can also aid benchmarking needs to enable evaluation of performance across different methods with respect to other similar datasets. The object detection model further showed the feasibility of performing OCR on such medical device even with relatively small training data.

Lastly, the non-invasive web application using camera shows that digitizing the acquisition and storage of vital signs from a patient monitor is possible without third party software and other expensive hardware to do so. Such tool offers a cost-effective solution to utilize vital signs data for real-time applications involving patient monitoring, further research, or policymaking purposes.

VIII. Recommendations

The provided dataset only considered one patient monitor model in the Philippine General Hospital namely the Mindray Beneview T8 model. Future work could expand such dataset by considering other models or have it complement other existing patient monitor datasets to create a better object detection model with higher generalizability for recognizing vital signs.

In terms of screen extraction, other approaches can be explored such as the application of deep learning or convolutional neural networks to improve the accuracy of edge detection. Image segmentation techniques to separate the foreground from the background prior to edge detection can also be studied. In addition, the object detection model used for OCR in this study only used 10% of the dataset as training data. Hence, future work can train a more complex object detection with larger data by utilizing the already-provided annotations. Saving the extracted values as a dataframe could further enable conversion of such data to waveforms represented by time series graph.

Future work is also encouraged to improve the web application by integrating the proposed object detection and optical character recognition steps after the preprocessing pipeline for complete data acquisition and extraction.

IX. Bibliography

[1] G. Iohom, "Basic patient monitoring during anesthesia." UpToDate, 2022 [Online].

[2] E. Zacharias, M. Teuchler, and B. Bernier, "Image processing based scene-text detection and recognition with tesseract," *ResearchGate*, 2020.

[3] J. Adriano, K. Calma, N. Lopez, J. Parado, L. Rabago, and J. Cabardo, "Digital conversion model for hand-filled forms using optical character recognition (ocr)," *IOP Conference Series: Materials Science and Engineering*, 2019.

[4] S. Babbar, S. Kesarwani, N. Dewan, K. Shangle, and S. Patel, "A new approach for vehicle number plate detection," *2018 Eleventh International Conference on Contemporary Computing*, 2018.

[5] S. Karthikeyan, A. S. de Herrera, F. Doctor, and A. Mirza, "An ocr post- correction approach using deep learning for processing medical reports," *IEEE Transactions on Circuits and Systems for Video Technology*, 2021.

[6] S. Perumal and V. Thambusamy, "Preprocessing by contrast enhancement techniques for medical images," *International Journal of Pure and Applied Mathematics*, 2018.

[7] J. Liang, D. Doermann, and H. Li, "Camera-based analysis of text and documents: a survey," *International Journal of Document Analysis and Recognition (IJDAR)*, 2005.

[8] S. Goldberg, A. Niemierko, and A. Turchin, "Analysis of data errors in clinical research databases," *AMIA Annual Symposium Proceedings*, 2008.

[9] R. I. Rumi, M. I. Pavel, E. Islam, M. B. Shakir, and M. A. Hossain, "Iot enabled prescription reading smart medicine dispenser implementing maximally stable extremal regions and ocr," *2019 Third International Conference on I-SMAC (IoT in Social, Mobile, Analytics, and Cloud)(I-SMAC))*, 2019.

[10] W. Xue, Q. Li, and Q. Xue, “Text detection and recognition for images of medical laboratory reports with a deep learning approach,” *IEEE Access*, 2019.

[11] N. Ramesh, A. Srivastava, and K. Deeba, “Improving optical character recognition techniques,” *International Journal of Engineering and Technology*, 2018.

[12] D. Liu, M. Gorges, and S. Jenkins, “Vitaldb, a high-fidelity multi-parameter vital signs database in surgical patients,” *PhySioNet*, 2022.

[13] H.-C. Lee and C.-W. Jung, “University of queensland vital signs dataset: development of an accessible repository of anesthesia patient monitoring data for research,” 2012.

[14] S. S. Kulkarni, N. Katebi, C. E. Valderrama, P. Rohloff, and G. D. Clifford, “Cnn-based lcd transcription of blood pressure from a mobile phone camera,” *Frontiers in Artificial Intelligence*, vol. 36, 2021.

[15] V. Shenoy and O. Aalami, “Utilizing smartphone-based machine learning in medical monitor data collection: Seven segment digit recognition,” *AMIA. Annual Symposium Proceedings. AMIA Symposium*, 2018.

[16] J. Jayoma, E. Moyon, and E. Morales, “Ocr based document archiving and indexing using pytesseract: A record management system for dswd caraga, philippines,” *2020 IEEE 12th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM)*, 2020.

[17] R. Yadav, “Optical character recognition based webapp,” *International Journal of Advanced Research in Science, Communication and Technology*, 2020.

[18] L. Froese, J. Dian, C. Batson, A. Gomez, A. S. Sainbhi, B. Unger, and F. Zeiler, “Computer vision for continuous bedside pharmacological data extraction: A novel

application of artificial intelligence for clinical data recording and biomedical research," *Frontiers in Big Data*, 2021.

[19] S. I. Bukhari, "Object character recognition from patient monitor screen," *Faculty of Science and Technology*, 2021.

[20] Jaidedai, "Ready-to-use ocr with 80+ supported languages and all popular writing scripts including latin, chinese, arabic, devanagari, cyrillic, and etc.," *Github*, 2021.

[21] Z. Chen and J. Chen, "Mobile imaging and computing for intelligent structural damage inspection," *Advances in Civil Engineering*, 2014.

[22] A. Rosebrock, "Opencv gamma correction." PyImageSearch, 2015 [Online].

[23] S. Sahir, "Canny edge detection step by step in python — computer vision," *Towards Data Science*, 2019.

[24] A. Rosebrock, "Text skew correction with opencv and python." PyImageSearch, 2017 [Online].

[25] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, 2010.

[26] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. Berg, "Ssd: single shot multibox detector," *European Conference on Computer Vision*, 2016.

[27] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and L. Zitnick, "Microsoft ´ coco: common objects in context," *Computing Research Repository*, 2014.

[28] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018.

[29] Mindray, "Beneview t5 t8 t9 operator's manual." Mindray, 2019 [Online].

[30] A. L. C. Carneiro, “Auto-labeling tool for object detection.” Towards Data Science, 2022 [Online].

[31] Y. Lee, S. Zhang, M. Li, and X. He, “Blind inverse gamma correction with maximized differential entropy,” *Electrical Engineering and Systems Science*, 2020.

[32] S. Rahman, M. M. Rahman, M. Abdullah-Al-Wadud, G. D. Al-Quaderi, and M. Shoyaib, “An adaptive gamma correction for image enhancement,” 2016.

[33] G. Cao, L. Huang, H. Tian, X. Huang, Y. Wang, and R. Zhi, “Contrast enhancement of brightness-distorted images by improved adaptive gamma correction,” 2018.

[34] U. Sara, M. Akter, and M. S. Uddin, “Image quality assessment through fsim, ssim, mse, and psnr-a comparative study,” 2019.

[35] S. K. Katiyar and P. Arun, “Comparative analysis of common edge detection techniques in context of object extraction,” 2012.

[36] B. K. Shah, V. Kedia, R. Raut, S. Ansari, and A. Shroff, “Evaluation and comparative study of edge detection techniques,” *IOSR Journal of Computer Engineering*, 2020.

[37] OpenCV, “Sobel derivatives.” OpenCV Open Source Computer Vision.

[38] OpenCV, “Laplace operator.” OpenCV Open Source Computer Vision.

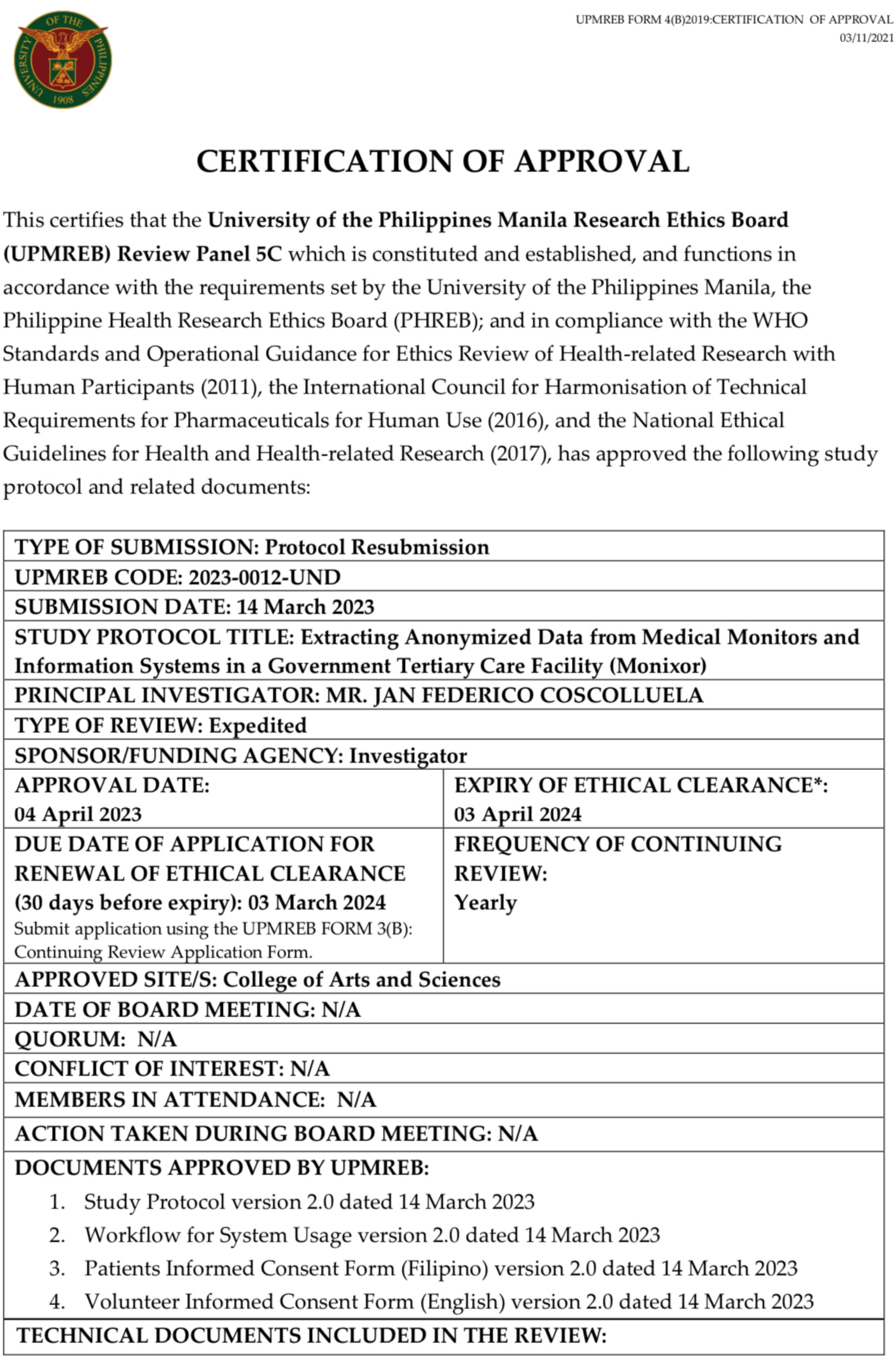
[39] OpenCV, “Morphological transformations.” OpenCV Open Source Computer Vision.

[40] K. B. Bulatov, E. Emelianova, D. V. Tropin, N. S. Skoryukina, Y. S. Chernyshova, A. V. Sheshkus, S. A. Usilin, Z. Ming, J.-C. Burie, M. M. Luqman, and V. V. Arlazarov, “Midv-2020: A comprehensive benchmark dataset for identity document analysis,” *ArXiv*, vol. abs/2107.00396, 2021.

[41] R. Padilla, W. L. Passos, T. L. B. Dias, S. L. Netto, and E. A. B. da Silva, “A comparative analysis of object detection metrics with a companion open-source toolkit,” *Electronics*, vol. 10, no. 3, 2021.

X. Appendix

A. Ethics Board Approval



UPMREB FORM 4(B)2019:CERTIFICATION OF APPROVAL
03/11/2021

CERTIFICATION OF APPROVAL

This certifies that the **University of the Philippines Manila Research Ethics Board (UPMREB) Review Panel 5C** which is constituted and established, and functions in accordance with the requirements set by the University of the Philippines Manila, the Philippine Health Research Ethics Board (PHREB); and in compliance with the WHO Standards and Operational Guidance for Ethics Review of Health-related Research with Human Participants (2011), the International Council for Harmonisation of Technical Requirements for Pharmaceuticals for Human Use (2016), and the National Ethical Guidelines for Health and Health-related Research (2017), has approved the following study protocol and related documents:

TYPE OF SUBMISSION: Protocol Resubmission	
UPMREB CODE: 2023-0012-UND	
SUBMISSION DATE: 14 March 2023	
STUDY PROTOCOL TITLE: Extracting Anonymized Data from Medical Monitors and Information Systems in a Government Tertiary Care Facility (Monixor)	
PRINCIPAL INVESTIGATOR: MR. JAN FEDERICO COSCOLLUELA	
TYPE OF REVIEW: Expedited	
SPONSOR/FUNDING AGENCY: Investigator	
APPROVAL DATE: **04 April 2023**	**EXPIRY OF ETHICAL CLEARANCE*:** **03 April 2024**
DUE DATE OF APPLICATION FOR RENEWAL OF ETHICAL CLEARANCE (30 days before expiry): 03 March 2024 Submit application using the UPMREB FORM 3(B): Continuing Review Application Form.	**FREQUENCY OF CONTINUING REVIEW:** **Yearly**
APPROVED SITE/S: College of Arts and Sciences	
DATE OF BOARD MEETING: N/A	
QUORUM: N/A	
CONFLICT OF INTEREST: N/A	
MEMBERS IN ATTENDANCE: N/A	
ACTION TAKEN DURING BOARD MEETING: N/A	
DOCUMENTS APPROVED BY UPMREB: 1. Study Protocol version 2.0 dated 14 March 2023 2. Workflow for System Usage version 2.0 dated 14 March 2023 3. Patients Informed Consent Form (Filipino) version 2.0 dated 14 March 2023 4. Volunteer Informed Consent Form (English) version 2.0 dated 14 March 2023	
TECHNICAL DOCUMENTS INCLUDED IN THE REVIEW:	

1. Curriculum vitae of principal investigator, Jan Federico Coscolluela, and certificate of completion in a six-hour course on Good Clinical Practices by NIDA Clinical Trials Network dated 07 December 2022
2. Curriculum vitae of co-investigator, Alvin Marcelo, MD, and certificate of completion of the e-learning course ICH Good Clinical Practice E6 (R2) dated 15 September 2022
3. Curriculum vitae of co-investigator, Marbert John Marasigan, and certificate of completion in a six-hour course on Good Clinical Practices by NIDA Clinical Trials Network dated 27 January 2022
4. Curriculum vitae of co-investigator, Miguel Sandino O. Aljibe, LME, MD, and certificate of completion in a six-hour course on Good Clinical Practices by NIDA Clinical Trials Network dated 16 March 2022
5. Budget Proposal version 2.0 dated 14 March 2023

RESPONSIBILITIES OF PRINCIPAL INVESTIGATOR WHILE STUDY IS IN PROGRESS (*Please note that forms may be downloaded from the UPMREB website:* ***reb.upm.edu.ph*****):**

1. Register research study in the Philippine Health Research Registry upon approval (http://registry.healthresearch.ph)
2. Progress report using the attached UPMREB FORM3(B)2012: Continuing Review Application Form, as indicated above, which includes the following: *(NOTE: In view of active ethical clearance, this report is mandatory even if the study has not started or is still awaiting release of funds.)*
   a. Date covered by the report
   b. Protocol summary and status report on the progress of the research
   c. Philippine Health Research Registry ID
   d. Number of participants accrued
   e. Withdrawal or termination of participants
   f. Complaints on the research since the last UPMREB review
   g. Summary of relevant recent research literature, interim findings and amendments since the last UPMREB review
   h. Any relevant multi-center research reports
   i. Any relevant information especially about risks associated with the research
   j. A copy of the informed consent document
3. Any amendment/s in the protocol, especially those that may adversely affect the safety of the participants during the conduct of the trial including changes in personnel, and revisions in the informed consent, must be submitted or reported using UPMREB FORM3(A)2012: Study Protocol Amendment Submission Form.

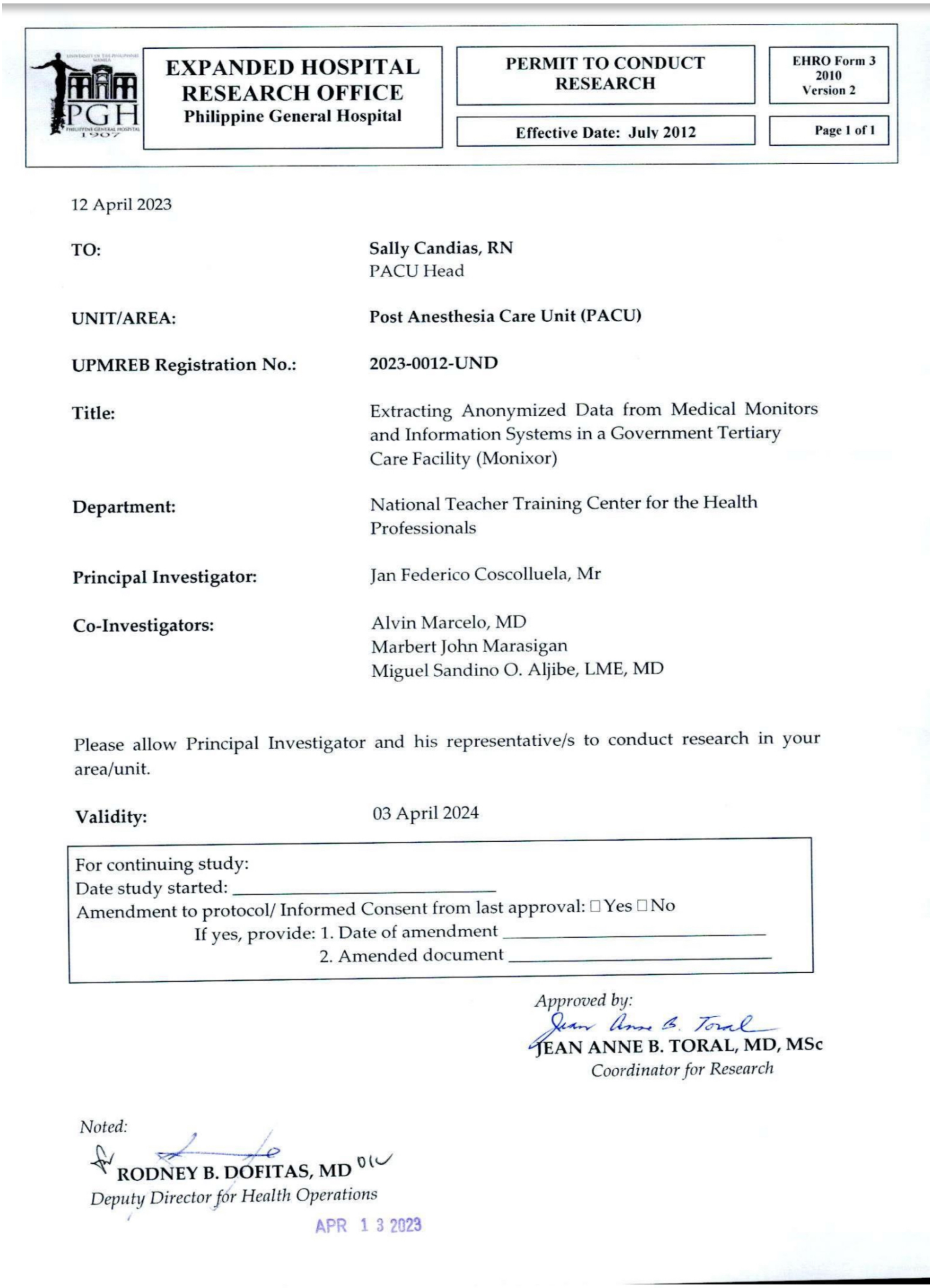
4. Report of non-compliance (deviation/violation), whether minor or major, at the soonest possible time up to six (6) months after the event, using UPMREB FORM 3(D)2012: Study Protocol Non-Compliance (Deviation/Violation) Report.
5. Reports of adverse events including from other study sites (national, international) using the UPMREB FORM 3(G)2012: Suspected, unexpected serious adverse event/reaction/s report, with timelines for submission guided by the GL 02 Version 2.0: Guideline on Reporting Serious Adverse Events; or list of reportable negative events using the UPMREB FORM 3(I)2012: Queries, Notification, and Complaints.
6. Notice of early termination of the study and reasons for such using UPMREB FORM 3(E)2012, or notice of time of completion of the study using UPMREB FORM 3(C)2012: Final Report Form.
7. Any event which may have ethical significance, and/or any information which is needed by the UPMREB to do ongoing review.

Tess de Guzman

MA. TERESA DE GUZMAN, PhD

Chair, UPMREB Review Panel 5C

B. Philippine General Hospital Approval



PGH

EXPANDED HOSPITAL RESEARCH OFFICE
Philippine General Hospital

PERMIT TO CONDUCT RESEARCH

Effective Date: July 2012

EHRO Form 3
2010
Version 2

Page 1 of 1

12 April 2023

TO: **Sally Candias, RN**
PACU Head

UNIT/AREA: **Post Anesthesia Care Unit (PACU)**

UPMREB Registration No.: **2023-0012-UND**

Title: Extracting Anonymized Data from Medical Monitors and Information Systems in a Government Tertiary Care Facility (Monixor)

Department: National Teacher Training Center for the Health Professionals

Principal Investigator: Jan Federico Coscolluela, Mr

Co-Investigators: Alvin Marcelo, MD
Marbert John Marasigan
Miguel Sandino O. Aljibe, LME, MD

Please allow Principal Investigator and his representative/s to conduct research in your area/unit.

Validity: 03 April 2024

For continuing study:
Date study started: ______________________
Amendment to protocol/ Informed Consent from last approval: □ Yes □ No
If yes, provide: 1. Date of amendment ______________________
2. Amended document ______________________

Approved by:
Jean Anne B. Toral
JEAN ANNE B. TORAL, MD, MSc
Coordinator for Research

Noted:
RODNEY B. DOFITAS, MD OIC
Deputy Director for Health Operations

APR 13 2023

C. Source Code

source–code/captureStyles.css

```
/* STYLES.CSS */
.button-17 {
  align-items: center;
  appearance: none;
  background-color: #073a49;
  border-radius: 24px;
  border-style: none;
  box-shadow: rgba(0, 0, 0, 0.2) 0 3px 5px -1px,
    rgba(0, 0, 0, 0.14) 0 6px 10px 0, rgba(0, 0, 0, 0.12) 0 1px 18px 0;
  box-sizing: border-box;
  color: white;
  cursor: pointer;
  display: inline-flex;
  fill: currentcolor;
  font-family: "Google Sans", Roboto, Arial, sans-serif;
  font-size: 14px;
  font-weight: 500;
  height: 48px;
  justify-content: center;
  letter-spacing: 0.25px;
  line-height: normal;
  max-width: 100%;
  overflow: visible;
  padding: 2px 24px;
  position: relative;
  text-align: center;
  text-transform: none;
  transition: box-shadow 280ms cubic-bezier(0.4, 0, 0.2, 1),
    opacity 15ms linear 30ms, transform 270ms cubic-bezier(0, 0, 0.2, 1) 0ms;
  user-select: none;
  -webkit-user-select: none;
  touch-action: manipulation;
  width: auto;
  will-change: transform, opacity;
  z-index: 0;
}

.button-17:hover {
  background: black;
  color: white;
}

.button-17:active {
  box-shadow: 0 4px 4px 0 rgb(60 64 67 / 30%),
    0 8px 12px 6px rgb(60 64 67 / 15%);
  outline: none;
}

.button-17:focus {
  outline: none;
  border: 2px solid #4285f4;
}

.button-17:not(:disabled) {
  box-shadow: rgba(60, 64, 67, 0.3) 0 1px 3px 0,
    rgba(60, 64, 67, 0.15) 0 4px 8px 3px;
}

.button-17:not(:disabled):hover {
  box-shadow: rgba(60, 64, 67, 0.3) 0 2px 3px 0,
    rgba(60, 64, 67, 0.15) 0 6px 10px 4px;
}

.button-17:not(:disabled):focus {
  box-shadow: rgba(60, 64, 67, 0.3) 0 1px 3px 0,
    rgba(60, 64, 67, 0.15) 0 4px 8px 3px;
}

.button-17:not(:disabled):active {
  box-shadow: rgba(60, 64, 67, 0.3) 0 4px 4px 0,
    rgba(60, 64, 67, 0.15) 0 8px 12px 6px;
}

.button-17:disabled {
  box-shadow: rgba(60, 64, 67, 0.3) 0 1px 3px 0,
    rgba(60, 64, 67, 0.15) 0 4px 8px 3px;
}

.button-18 {
  align-items: center;
  appearance: none;
  background-color: maroon;
  border-radius: 24px;
  border-style: none;
  box-shadow: rgba(0, 0, 0, 0.2) 0 3px 5px -1px,
    rgba(0, 0, 0, 0.14) 0 6px 10px 0, rgba(0, 0, 0, 0.12) 0 1px 18px 0;
  box-sizing: border-box;
  color: white;
  cursor: pointer;
  display: inline-flex;
  fill: currentcolor;
  font-family: "Google Sans", Roboto, Arial, sans-serif;
  font-size: 14px;
  font-weight: 500;
  height: 48px;
  justify-content: center;
  letter-spacing: 0.25px;
  line-height: normal;
  max-width: 100%;
  overflow: visible;
  padding: 2px 24px;
  position: relative;
  text-align: center;
  text-transform: none;
  transition: box-shadow 280ms cubic-bezier(0.4, 0, 0.2, 1),
    opacity 15ms linear 30ms, transform 270ms cubic-bezier(0, 0, 0.2, 1) 0ms;
  user-select: none;
  -webkit-user-select: none;
  touch-action: manipulation;
  width: auto;
  will-change: transform, opacity;
  z-index: 0;
}

.button-18:hover {
  background: black;
  color: white;
}

.button-18:active {
  box-shadow: 0 4px 4px 0 rgb(60 64 67 / 30%),
    0 8px 12px 6px rgb(60 64 67 / 15%);
  outline: none;
}

.button-18:focus {
  outline: none;
  border: 2px solid #4285f4;
}

.button-18:not(:disabled) {
  box-shadow: rgba(60, 64, 67, 0.3) 0 1px 3px 0,
    rgba(60, 64, 67, 0.15) 0 4px 8px 3px;
}

.button-18:not(:disabled):hover {
  box-shadow: rgba(60, 64, 67, 0.3) 0 2px 3px 0,
    rgba(60, 64, 67, 0.15) 0 6px 10px 4px;
}

.button-18:not(:disabled):focus {
  box-shadow: rgba(60, 64, 67, 0.3) 0 1px 3px 0,
    rgba(60, 64, 67, 0.15) 0 4px 8px 3px;
}

.button-18:not(:disabled):active {
  box-shadow: rgba(60, 64, 67, 0.3) 0 4px 4px 0,
    rgba(60, 64, 67, 0.15) 0 8px 12px 6px;
}

.button-18:disabled {
  box-shadow: rgba(60, 64, 67, 0.3) 0 1px 3px 0,
    rgba(60, 64, 67, 0.15) 0 4px 8px 3px;
}

.button-19 {
  align-items: center;
  appearance: none;
  background-color: green;
  border-radius: 24px;
  border-style: none;
  box-shadow: rgba(0, 0, 0, 0.2) 0 3px 5px -1px,
    rgba(0, 0, 0, 0.14) 0 6px 10px 0, rgba(0, 0, 0, 0.12) 0 1px 18px 0;
  box-sizing: border-box;
  color: white;
  cursor: pointer;
  display: inline-flex;
  fill: currentcolor;
  font-family: "Google Sans", Roboto, Arial, sans-serif;
  font-size: 14px;
  font-weight: 500;
  height: 48px;
  justify-content: center;
  letter-spacing: 0.25px;
  line-height: normal;
  max-width: 100%;
  overflow: visible;
  padding: 2px 24px;
  position: relative;
  text-align: center;
  text-transform: none;
  transition: box-shadow 280ms cubic-bezier(0.4, 0, 0.2, 1),
    opacity 15ms linear 30ms, transform 270ms cubic-bezier(0, 0, 0.2, 1) 0ms;
  user-select: none;
  -webkit-user-select: none;
  touch-action: manipulation;
  width: auto;
  will-change: transform, opacity;
  z-index: 0;
}

.button-19:hover {
  background: black;
  color: white;
}

.button-19:active {
  box-shadow: 0 4px 4px 0 rgb(60 64 67 / 30%),
    0 8px 12px 6px rgb(60 64 67 / 15%);
  outline: none;
}

.button-19:focus {
  outline: none;
  border: 2px solid #4285f4;
}

.button-19:not(:disabled) {
  box-shadow: rgba(60, 64, 67, 0.3) 0 1px 3px 0,
    rgba(60, 64, 67, 0.15) 0 4px 8px 3px;
}

.button-19:not(:disabled):hover {
  box-shadow: rgba(60, 64, 67, 0.3) 0 2px 3px 0,
    rgba(60, 64, 67, 0.15) 0 6px 10px 4px;
}

.button-19:not(:disabled):focus {
  box-shadow: rgba(60, 64, 67, 0.3) 0 1px 3px 0,
    rgba(60, 64, 67, 0.15) 0 4px 8px 3px;
}

.button-19:not(:disabled):active {
  box-shadow: rgba(60, 64, 67, 0.3) 0 4px 4px 0,
    rgba(60, 64, 67, 0.15) 0 8px 12px 6px;
}

.button-19:disabled {
  box-shadow: rgba(60, 64, 67, 0.3) 0 1px 3px 0,
    rgba(60, 64, 67, 0.15) 0 4px 8px 3px;
}

/* Stepper in Home Page */

:root {
  --circle-size: clamp(0.5rem, 2vw, 1.5rem);
  --spacing: clamp(0.25rem, 2vw, 0.5rem);
}

.c-stepper {
  display: flex;
}

.c-stepper__item {
  display: flex;
  flex-direction: column;
  flex: 1;
  text-align: center;
}

.c-stepper__item:before {
  --size: 2rem;
  content: "";
  display: block;
  width: var(--circle-size);
  height: var(--circle-size);
  border-radius: 50%;
  background-color: rgb(71, 3, 3);
  margin: 0 auto 1rem;
}

.c-stepper__item:not(:last-child):after {
  content: "";
  position: relative;
  top: calc(var(--circle-size) / 2);
  width: calc(100% - var(--circle-size) - calc(var(--spacing) * 2));
  left: calc(50% + calc(var(--circle-size) / 2 + var(--spacing)));
  height: 2px;
  background-color: #e0e0e0;
  order: -1;
}

.c-stepper__title {
  font-weight: bold;
  color: black;
  font-size: clamp(1rem, 4vw, 1.25rem);
  margin-bottom: 0.5rem;
}

.c-stepper__desc {
  color: rgb(73, 73, 73);
  font-size: clamp(0.85rem, 2vw, 1rem);
  padding-left: var(--spacing);
  padding-right: var(--spacing);
}
```

source-code/home.html

```
<!-- HOME.HTML -->

<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>Home Page</title>
    {% load static %}
    <link rel="icon" type="image/png" href="{% static '/images/home.ico' %}" />
    <meta name="viewport" content="width=device-width">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet"
        integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC" crossorigin="anonymous">
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"
        integrity="sha384-MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM"
        crossorigin="anonymous"></script>
    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"
        integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN"
        crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.12.9/dist/umd/popper.min.js"
        integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q"
        crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/js/bootstrap.min.js"
        integrity="sha384-JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl"
        crossorigin="anonymous"></script>
    <link rel="stylesheet" href="{% static 'css/captureStyles.css'%}">
</head>

<body>
    <nav class="navbar navbar-expand-lg navbar-light" style="background-color: maroon;">
        <div class="container-fluid">
            <a class="navbar-brand" href="{% url 'home' %}" style="color: white"><b>Monixor</b></a>
            <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarScroll"
                aria-controls="navbarScroll" aria-expanded="false" aria-label="Toggle navigation">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse" id="navbarScroll">
                <ul class="navbar-nav ms-auto my-2 my-lg-0 navbar-nav-scroll" style="--bs-scroll-height: 100px;">
                    <li class="nav-item">
                        <a class="nav-link active" aria-current="page" href="{% url 'capture' %}"
                            style="color: white">Capture</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link active" aria-current="page" href="{% url 'upload' %}"
```

```
                            style="color: white">Upload</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="{% url '
      dataset' %}" style="color: white">Dataset</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="{% url '
      guide' %}" style="color: white">Guide</a>
                    </li>
                </ul>
            </div>
        </div>
    </nav>
    <main>
        <div class="container my-auto mx-auto" style="
      padding-top: 10px;padding-bottom: 30px;">
            <div class="row">
                <div class="col-md-5">
                    <img src="{% static '/images/monixor.png'
       %}" alt="Monixor" style="display: block;
                    margin-left: auto;
                    margin-right: auto;
                    width: 95%;">
                </div>
                <div class="col-md-7 my-auto">
                    <div class="jumbotron my-auto">
                        <h1 class="display-4">Monixor</h1>
                        <p class="lead">This system allows the
       user to record a video (or capture a photo) of a patient
                            monitor and return preprocessed
      frame/s optimized for further computer vision tasks such
      as
                            optical character recognition.
                        </p>
                        <hr class="my-4">
                        <button class="button-18">
                            <a href="{% url 'capture' %}" style
      ="text-decoration: none; color: inherit;">Get
                                Started</a></button>
                    </div>
                </div>
            </div>
            <br>
            <div class="row c-stepper">
                <div class="col-md-4 c-stepper__item">
                    <h3 class="c-stepper__title">In-app/
      External Camera</h3>
                    <p class="c-stepper__desc">Access device
      camera to record patient monitor, or upload pre-captured
                        file.</p>
                </div>
                <div class="col-md-4 c-stepper__item">
                    <h3 class="c-stepper__title">Video Record
      /Image Copy</h3>
                    <p class="c-stepper__desc">Save local
      copy of the extracted frames individually or as
                        a zip file.</p>
                </div>
                <div class="col-md-4 c-stepper__item">
                    <h3 class="c-stepper__title">Patient
      Monitor Dataset</h3>
                    <p class="c-stepper__desc">Use over 4,000
       annotated images of camera-captured monitor.
                    </p>
                </div>
            </div>
        </div>
    </main>
</body>

</html>
```

source-code/capture.html

```
<!-- CAPTURE.HTML -->

<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>Capture</title>
    {% load static %}
    <link rel="icon" type="image/png" href="{% static '/
      images/capture.ico' %}" />
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/
      ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css
      ">
    <meta name="viewport" content="width=device-width">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/
      dist/css/bootstrap.min.css" rel="stylesheet"
        integrity="sha384-EVSTQN3/
      azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC
      " crossorigin="anonymous">
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5
      .0.2/dist/js/bootstrap.bundle.min.js"
        integrity="sha384-MrcW6ZMFYlzcLA8Nl+
      NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/
      tWtIaxVXM"
        crossorigin="anonymous"></script>
    <script src="https://code.jquery.com/jquery-3.2.1.slim.min
      .js"
        integrity="sha384-
      KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/
      Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN"
        crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1
      .12.9/dist/umd/popper.min.js"
        integrity="sha384-ApNbgh9B+
      Y1QKtv3Rn7W3mgPxhU9K/
      ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q"
        crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@4
      .0.0/dist/js/bootstrap.min.js"
        integrity="sha384-JZR6Spejh4U02d8jOt6vLEHfe/
      JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl"
        crossorigin="anonymous"></script>
    <script defer src={% static "js/capture.js" %}></script>
    <link rel="stylesheet" href="{% static 'css/captureStyles.
      css'%}">
    <script>
        $(document).ready(function () {
            $('[data-toggle="popover"]').popover();
        });
    </script>
</head>

<body>
    <nav class="navbar navbar-expand-lg navbar-light" style
      ="background-color: maroon;">
        <div class="container-fluid">
            <a class="navbar-brand" href="{% url 'home' %}"
       style="color: white">Monixor</a>
            <button class="navbar-toggler" type="button"
      data-bs-toggle="collapse" data-bs-target="#
      navbarScroll"
                aria-controls="navbarScroll" aria-expanded="
      false" aria-label="Toggle navigation">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse" id="
      navbarScroll">
                <ul class="navbar-nav ms-auto my-2 my-lg
      -0 navbar-nav-scroll" style="--bs-scroll-height: 100
      px;">
                    <li class="nav-item">
                        <a class="nav-link active" aria-
      current="page" href="{% url 'capture' %}"
                            style="color: white"><b>Capture
      </b></a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link active" aria-
      current="page" href="{% url 'upload' %}"
                            style="color: white">Upload</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="{% url '
      dataset' %}" style="color: white">Dataset</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="{% url '
      guide' %}" style="color: white">Guide</a>
                    </li>
                </ul>
            </div>
        </div>
    </nav>
    <main>
        <div class="container my-auto mx-auto" style="
      padding-top: 30px; text-align: center;">
            <div style="display: flex; justify-content: center
      ;">
                <ul class="nav nav-pills mb-3" id="pills-tab
      " role="tablist">
                    <li class="nav-item mx-auto" data-
      toggle="tooltip" data-placement="top"
                        title="Capture a Photo using Camera
      ">
                        <a class="nav-link active" id="pills-
      profile-tab" data-toggle="pill" href="#pills-profile"
                            role="tab" aria-controls="pills-
      profile" aria-selected="false">Photo Mode
                            <i class="fa fa-question-circle"
      aria-hidden="true" data-toggle="popover"
                                title="Take photo as seen in
```

```
the live preview below”
                data−content=”Some content inside the popover” data−bs−placement=”top”></i>
            </a>
          </li>
          <li class=”nav−item” style=”margin−right: 20px;”>
            <a class=”nav−link” id=”pills−home−tab” data−toggle=”pill” href=”#pills−home” role=”tab”
              aria−controls=”pills−home” aria−selected=”true”>Video Mode
              <i class=”fa fa−question−circle” aria−hidden=”true” data−toggle=”popover”
                title =”Start recording as shown in live preview and check the replay afterwards”
                data−content=”Some content inside the popover” data−bs−placement=”top”></i>
            </a>
          </li>
        </ul>
      </div>

      <b>
        <p id=”status” style=”display: inline”>Status: Waiting to Start Record</p>
      </b>
    </div>
    <br>
    <div class=”tab−content” id=”pills−tabContent”>
      <div class=”tab−pane fade” id=”pills−home” role=”tabpanel” aria−labelledby=”pills−home−tab”>
        <div class=”col−12 text−center”>
          <span style=”display: inline”>
            <button class=”button−17” id=”btnStart”>START
              RECORD</button>
            <button class=”button−18” id=”btnStop”>STOP RECORD</button>
          </span>
        </div>
        <br>
        <div class=”container”>
          <div class=”row”>
            <div class=”col−md−6 text−center mx−auto”>
              <div class=”card”>
                <div class=”card−header text−center”>
                  Live Preview
                </div>
                <div class=”card−body”>
                  <video id=”vid1” controls autoplay style=”width: 100%; height: 100%”></video>
                </div>
              </div>
              <br>
            </div>
            <div class=”col−md−6 text−center mx−auto”>
              <div class=”card”>
                <div class=”card−header text−center”>
                  Recorded Video
                </div>
                <div class=”card−body”>
                  <video id=”vid2” controls autoplay style=”width: 100%; height: 100%”></video>
                </div>
              </div>
              <br>
            </div>
          </div>
          <br>
          <div class=”col−12” style=”margin: auto;
          width: 50%; text−align: center;”>
            <button class=”btn btn−danger mx−auto” type=”button” id=”loadingBtnVid” style=”display: none”>
              <span class=”spinner−border spinner−border−sm” role=”status” aria−hidden=”true”></span>
              Extracting frames, please wait ...
            </button>
          </div>
          <div class=”col−12 text−center”>
            <span style=”display: inline”>
              <button class=”button−19” id=”proceed” style=”display: none;”>
                <a href=”{% url ’results’ %}” style=”text−decoration: none; color: white”>Extracted
                  Frames</a>
              </button>
            </span>
          </div>
          <br>
          <br>
        </div>
      </div>
      <div class=”tab−pane fade  show active” id=”pills−profile” role=”tabpanel”
        aria−labelledby=”pills−profile−tab”>
        <div class=”col−12 text−center”>
          <span style=”display: inline”>
            <button class=”button−17” id=”initiate”>OPEN CAMERA</button>
            <button class=”button−18” id=”startbutton”>TAKE PHOTO</button>
          </span>
        </div>
        <br>
        <div class=”container”>
          <div class=”row”>
            <div class=”col−md−6 text−center mx−auto”>
              <div class=”card”>
                <div class=”card−header text−center”>
                  Live Preview
                </div>
                <div class=”card−body”>
                  <div class=”camera”>
                    <video id=”video” controls style=”width: 100%; height: 100%”>Video stream not
                      available.</video>
                  </div>
                </div>
              </div>
              <br>
            </div>
            <div class=”col−md−6 text−center mx−auto”>
              <div class=”card”>
                <div class=”card−header text−center”>
                  Captured Photo
                </div>
                <div class=”card−body”>
                  <canvas id=”canvas” style=”display: none”>
                  </canvas>
                  <img src=”{% static ’/images/emptyPic.png’ %}” id=”canvasimg” alt=”” width=”58%”>
                  <div class=”output”>
                    <img id=”photo” style=”width: 100%; height: 100%” />
                  </div>
                </div>
              </div>
              <br>
            </div>
          </div>
          <br>
          <div class=”col−12” style=”margin: auto;
          width: 50%; text−align: center;”>
            <button class=”btn btn−danger mx−auto” type=”button” id=”loadingBtnPic” style=”display: none”>
              <span class=”spinner−border spinner−border−sm” role=”status” aria−hidden=”true”></span>
              Processing, please wait ...
            </button>
            <br>
          </div>
          <div class=”col−12 text−center”>
            <span style=”display: inline”>
              <button class=”button−17” id=”downloadImageContainer” style=”display: none;”>
                <a id=”downloadImage” download style=”display: none”>
                  Download Image
                </a>
              </button>
              <button class=”button−19 mx−auto” href id=”proceedPic” style=”display: none;”>
                <a href=” {% url ’resultPic’ %}” style=”text−decoration: none; color: white”>View
                  Results</a>
              </button>
            </span>
          </div>
          <br>
          <br>
        </div>
      </div>
    </div>
  </main>
</body>
```

```

</html>
```

source-code/index.html

```
<!-- INDEX.HTML (UPLOAD FUNCTIONALITY) -->

<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Upload</title>
    {% load static %}
    <link rel="icon" type="image/png" href="{% static '/images/upload.ico' %}" />
    {% load crispy_forms_tags %}
    <!-- Boostrap Dependencies -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet"
        integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC" crossorigin="anonymous">
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"
        integrity="sha384-MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM"
        crossorigin="anonymous"></script>
    <!-- <script defer src={% static "js/recordVid.js" %}></script> -->
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.0/jquery.min.js"></script>
    <!-- <script defer src={% static "js/try.js" %}></script> -->
</head>

<body>
    <!-- Navbar -->
    <nav class="navbar navbar-expand-lg navbar-light" style="background-color: maroon;">
        <div class="container-fluid">
            <a class="navbar-brand" href="{% url 'home' %}" style="color: white">Monixor</a>
            <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarScroll"
                aria-controls="navbarScroll" aria-expanded="false" aria-label="Toggle navigation">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse" id="navbarScroll">
                <ul class="navbar-nav ms-auto my-2 my-lg-0 navbar-nav-scroll" style="--bs-scroll-height: 100px;">
                    <li class="nav-item">
                        <a class="nav-link active" aria-current="page" href="{% url 'capture' %}"
                            style="color: white">Capture</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link active" aria-current="page" href="{% url 'upload' %}"
                            style="color: white"><b>Upload</b></a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="{% url 'dataset' %}" style="color: white">Dataset</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="{% url 'guide' %}" style="color: white">Guide</a>
                    </li>
                </ul>
            </div>
        </div>
    </nav>
    <main>
        <div class="container my-auto mx-auto" style="padding-top: 10px;">
            <div class="row">
                <div class="col-md-5">
                    <img src="{% static '/images/Upload.png' %}" alt="Monixor" style="display: block;
                    margin-left: auto;
                    margin-right: auto;
                    width: 95%;">
            </div>
            <div class="col-md-7 my-auto">
                <div class="jumbotron my-auto">
                    <h1 class="display-4">File Upload</h1>
                    <p class="lead">This section allows the user to manually upload a pre-captured image or video of
                        a
                        patient monitor. The preprocessed frame/s will then be
                        shown below.</p>
                    <hr class="my-4">
                    <div class="card">
                        <div class="card-body">
                            <h5 class="card-title">Upload your file here.</h5>
                            <h6 class="card-subtitle mb-2 text-muted">The file size limit is temporarily set to 500
                                MB.</h6>
                            <form action="" method="POST" enctype="multipart/form-data">
                                {% csrf_token %}
                                {{form.as_p}}
                                <button class="btn" type="submit" style="background-color:maroon; color: white">
                                    Upload </button>
                            </form>
                        </div>
                    </div>
                    <div>
                        {% for message in messages %}
                        {% if 'success' == message.tags %}
                        <div class="alert alert-success alert-dismissible fade show" role="alert">
                            <strong>Success!</strong> {{message | striptags}}
                            <button type="button" class="btn-close" data-bs-dismiss="alert"
                                aria-label="Close"></button>
                        </div>
                        {% else %}
                        <div class="alert alert-danger alert-dismissible fade show" role="alert">
                            <strong>Error!</strong> {{message | striptags}}
                            <button type="button" class="btn-close" data-bs-dismiss="alert"
                                aria-label="Close"></button>
                        </div>
                        {% endif %}
                        {% endfor %}
                    </div>
                </div>
            </div>
        </div>

        <br><br>

        <section class="mx-auto my-auto text-center">
            {% if outputImages %}
            <h3>Output</h3>
            <i>Image Count: {{outputImages | length}} (Kindly click an image to download) </i>
            <br>
            <br>
            <div class="col-12 text-center">
                <button class="btn btn-success mx-auto">
                    <a href="{% url 'downloadZipProcessed' %}"
                        style="text-decoration: none; color: inherit;">Download
                        Image/s</a>
                </button>
            </div>
            <br>
            <div style="overflow-y: auto; height:500px; margin-bottom: 50px">
                {% for outputImage in outputImages %}
                <a href="{{outputImage.preprocessed.url}}" download>
                    <img src="{{outputImage.preprocessed.url}}" alt="Output image" width="250px" height="190px"
                        style="padding: 10px;">
                </a>
                {% endfor %}
                <br>
                <br>
                {% else %}
                <p></p>
                {% endif %}
            </section>
        </div>
        </div>
    </main>
</body>

</html>
```

source–code/dataset.html

```
<!-- DATASET.HTML -->

<!DOCTYPE html>
<html lang=”en”>

<head>
    <meta charset=”UTF-8”>
    <meta http-equiv=”X-UA-Compatible” content=”IE=edge”>
    <meta name=”viewport” content=”width=device-width, initial-scale=1.0”>
    <title>Dataset</title>
    {% load static %}
    <link rel=”icon” type=”image/png” href=”{% static ’/images/dataset.ico’ %}” />

    <!-- Boostrap Dependencies -->
    <link href=”https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css” rel=”stylesheet”
        integrity=”sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC” crossorigin=”anonymous”>
    <script src=”https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js”
        integrity=”sha384-MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM”
        crossorigin=”anonymous”></script>
</head>

<body>
    <!-- Navbar -->
    <nav class=”navbar navbar-expand-lg navbar-light” style=”background-color: maroon;”>
        <div class=”container-fluid”>
            <a class=”navbar-brand” href=”{% url ’home’%}” style=”color: white”>Monixor</a>
            <button class=”navbar-toggler” type=”button” data-bs-toggle=”collapse” data-bs-target=”#navbarScroll”
                aria-controls=”navbarScroll” aria-expanded=”false” aria-label=”Toggle navigation”>
                <span class=”navbar-toggler-icon”></span>
            </button>
            <div class=”collapse navbar-collapse” id=”navbarScroll”>
                <ul class=”navbar-nav ms-auto my-2 my-lg-0 navbar-nav-scroll” style=”--bs-scroll-height: 100px;”>
                    <li class=”nav-item”>
                        <a class=”nav-link active” aria-current=”page” href=”{% url ’capture’ %}”
                            style=”color: white”>Capture</a>
                    </li>
                    <li class=”nav-item”>
                        <a class=”nav-link active” aria-current=”page” href=”{% url ’upload’ %}”
                            style=”color: white”>Upload</a>
                    </li>
                    <li class=”nav-item”>
                        <a class=”nav-link” href=”{% url ’dataset’ %}” style=”color: white”><b>Dataset</b></a>
                    </li>
                    <li class=”nav-item”>
                        <a class=”nav-link” href=”{% url ’guide’ %}” style=”color: white”>Guide</a>
                    </li>
                </ul>
            </div>
        </div>
    </nav>
    <div class=”container” style=”padding-top: 10px;”>
        <section>
            <div class=”container my-auto mx-auto”>
                <div class=”row”>
                    <div class=”col-md-5”>
                        <img src=”{% static ’/images/dataset.png’ %}” alt=”Monixor” style=”display: block;
                        margin-left: auto;
                        margin-right: auto;
                        width: 95%;”>
                    </div>
                    <div class=”col-md-7 my-auto”>
                        <div class=”jumbotron my-auto”>
                            <h1 class=”display-4”>PM-2023 Dataset</h1>
                            <p class=”lead”>There is a lack of dataset comprising realistic field images of a patient
                                monitor. The availability of such dataset with variability in quality can prove useful
                                in computer vision-related tasks such as object detection, optical character
                                recognition, and the likes.</p>
                            <hr class=”my-4”>
                            <p>A public repository of realistic field images of patient monitor is collected to
                                contribute a new dataset for computer vision.
                            </p>
                            <p class=”lead”>
                                <a class=”btn btn-lg”
                                    href=”https://drive.google.com/drive/folders/1-7GUeSjbOU8xQQJNE-LOj0bhutbLR_Q5?usp=sharing”
                                    role=”button” style=”background-color: maroon; color: white;”>Dataset</a>
                            </p>
                        </div>
                    </div>
                </div>
                <br>
                <div class=”row”>
                    <div class=”col-md-6”>
                        <h4>About the Dataset</h4>
                        <p>Files are named following a certain convention to provide metadata for easier navigation of
                            files . An image is named as {volunteer number}_{file code}_{frame_count}, where volunteer
                            number
                            corresponds
                            to the 5 study participants (i.e., from 1 to 5).
                            <br>
                            For example, first frame of an image taken from direct camera with low lighting condition in
                            the first volunteer data is named as <b>01_01_1.jpg</b> while the second frame is named
                            <b>01_01_2</b>
                            and so on.
                        </p>
                        <p>
                            The dataset is divided into two folders namely (1) <u>raw</u> and (2) <u>preprocessed</u>.
                            The raw dataset
                            corresponds to extracted frames from the collected
                            videos (2-sec interval) while the preprocessed dataset corresponds to the screen-extracted
                            frame counterpart
                            of the raw dataset. The structure of the dataset folder is illustrated below:
                        <ul>
                            <li><b>Raw</b>
                                <ul>- annotations.zip</ul>
                                <ul>- images.zip</ul>
                            </li>
                            <li><b>Preprocessed</b>
                                <ul>- annotations.zip</ul>
                                <ul>- images.zip</ul>
                            </li>
                        </ul>

                            </p>
                    </div>
                    <div class=”col-md-6”>
                        <table class=”table table-hover”>
                            <thead>
                                <tr>
                                    <th scope=”col”>File Code</th>
                                    <th scope=”col”>Capturing Condition</th>
                                    <th scope=”col”>Lighting Condition</th>
                                </tr>
                            </thead>
                            <tbody>
                                <tr>
                                    <td>01</td>
                                    <td>Direct Camera</td>
                                    <td>Low</td>
                                </tr>
                                <tr>
```

```
                                    <td>02</td>
                                    <td>Direct Camera</td>
                                    <td>Natural</td>
                                </tr>
                                <tr>
                                    <td>03</td>
                                    <td>Skewed Upwards</td>
                                    <td>Low</td>
                                </tr>
                                <tr>
                                    <td>04</td>
                                    <td>Skewed Upwards</td>
                                    <td>Natural</td>
                                </tr>
                                <tr>
                                    <td>05</td>
                                    <td>Skewed Downwards</td>
                                    <td>Low</td>
                                </tr>
                                <tr>
                                    <td>06</td>
                                    <td>Skewed Downwards</td>
                                    <td>Natural</td>
                                </tr>
                                <tr>
                                    <td>07</td>
                                    <td>Skewed to Left</td>
                                    <td>Low</td>
                                </tr>
                                <tr>
                                    <td>08</td>
                                    <td>Skewed to Left</td>
                                    <td>Natural</td>
                                </tr>
                                <tr>
                                    <td>09</td>
                                    <td>Skewed to Right</td>
                                    <td>Low</td>
                                </tr>
                                <tr>
                                    <td>10</td>
                                    <td>Skewed to Right</td>
                                    <td>Natural</td>
                                </tr>
                            </tbody>
                        </table>
                    </div>
                </div>
                <br>
        </section>
    </div>
</body>

</html>
```

source–code/guide.html

```
<!−− FAQ.HTML −−>

<!DOCTYPE html>
<html lang=”en”>

<head>
    <meta charset=”UTF−8”>
    <meta http−equiv=”X−UA−Compatible” content=”IE=edge”>
    <meta name=”viewport” content=”width=device−width, initial−scale=1.0”>
    <title>Guide</title>
    {% load static %}
    <link rel=”icon” type=”image/png” href=”{% static ’/images/guide.ico’ %}” />

    <!−− Boostrap Dependencies −−>
    <link href=”https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css” rel=”stylesheet”
        integrity=”sha384−EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC” crossorigin=”anonymous”>
    <script src=”https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js”
        integrity=”sha384−MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM”
        crossorigin=”anonymous”></script>
</head>

<body>
    <!−− Navbar −−>
    <nav class=”navbar navbar−expand−lg navbar−light” style=”background−color: maroon;”>
        <div class=”container−fluid”>
            <a class=”navbar−brand” href=”{% url ’home’%}” style=”color: white”>Monixor</a>
            <button class=”navbar−toggler” type=”button” data−bs−toggle=”collapse” data−bs−target=”#navbarScroll”
                aria−controls=”navbarScroll” aria−expanded=”false” aria−label=”Toggle navigation”>
                <span class=”navbar−toggler−icon”></span>
            </button>
            <div class=”collapse navbar−collapse” id=”navbarScroll”>
                <ul class=”navbar−nav ms−auto my−2 my−lg−0 navbar−nav−scroll” style=”−−bs−scroll−height: 100px;”>
                    <li class=”nav−item”>
                        <a class=”nav−link active” aria−current=”page” href=”{% url ’capture’ %}”
                            style=”color: white”>Capture</a>
                    </li>
                    <li class=”nav−item”>
                        <a class=”nav−link active” aria−current=”page” href=”{% url ’upload’ %}”
                            style=”color: white”>Upload</a>
                    </li>
                    <li class=”nav−item”>
                        <a class=”nav−link” href=”{% url ’dataset’ %}” style=”color: white”>Dataset</a>
                    </li>
                    <li class=”nav−item”>
                        <a class=”nav−link” href=”{% url ’guide’ %}” style=”color: white”><b>Guide</b></a>
                    </li>
                </ul>
            </div>
        </div>
    </nav>
    <!−−Section: FAQ−−>
    <div class=”container” style=”padding−top: 20px;”>
        <section>
            <h3 class=”text−center mb−4 pb−2 fw−bold”>FAQ</h3>
            <p class=”text−center mb−5”>
                This section aims to provide tips on how to navigate the system.
            </p>

            <div class=”row”>
                <div class=”col−md−6 col−lg−4 mb−4”>
                    <h6 class=”mb−3 ”><i class=”far fa−paper−plane”></i>What’s the difference between Capture and
                        Upload?</h6>
                    <p>
                        Capture mode is ideal when patient monitor is readily available and user wishes to take a
                        picture or video of it.
                        Upload is when a pre−taken input file is available.
                    </p>
                </div>

                <div class=”col−md−6 col−lg−4 mb−4”>
                    <h6 class=”mb−3 ”><i class=”fas fa−pen−alt”></i> I can’t access my device camera as of the
                        moment.</h6>
                    <p>
                        Please make sure your camera has at least 720x480 resolution. Alternatively, you can click the
                        <em>Upload</em> button to send a pre−captured input file taken from another
                        device.
                    </p>
                </div>

                <div class=”col−md−6 col−lg−4 mb−4”>
                    <h6 class=”mb−3 ”><i class=”fas fa−user”></i> Results are not showing when I click <em>View
                            Results</em>
                    </h6>
                    <p>
                        Please try to wait a few seconds before clicking <em>View Results</em> button (or simply reload
                        the page) as the system may have
                        encountered a lag/buffer problem.
                    </p>
                </div>

                <div class=”col−md−6 col−lg−4 mb−4”>
                    <h6 class=”mb−3 ”><i class=”fas fa−rocket”></i> Upload process takes a long time.
                    </h6>
                    <p>
                        This may naturally occur in inputs of big size. The upload is currently limited to 500 MB input.
                        If problem persists, please retry uploading or trimming your videos.
                    </p>
                </div>

                <div class=”col−md−6 col−lg−4 mb−4”>
                    <h6 class=”mb−3 ”><i class=”fas fa−home”></i>Why can’t I access the front camera of my
                        mobile device when recording?
                    </h6>
                    <p>Back camera is intentionally accessed for use to achieve an ideally higher quality.</p>
                </div>

                <div class=”col−md−6 col−lg−4 mb−4”>
                    <h6 class=”mb−3 ”><i class=”fas fa−book−open”></i>Recording does not work on my end.</h6>
                    <p>
                        The system is compatible with typical browsers (Chrome, Firefox, Opera, Safari). If the problem
                        remains, please try switching to Google Chrome.
                    </p>
                </div>
            </div>
        </section>
        <!−−Section: FAQ−−>
    </div>
</body>

</html>
```

source–code/requirements.txt

```
# REQUIRED DEPENDENCIES

asgiref==3.6.0
backports.zoneinfo==0.2.1
Django==4.1.3
django−cors−headers==3.13.0
django−crispy−forms==1.14.0
gunicorn==20.0.4
numpy==1.24.1
opencv−contrib−python−headless==4.7.0.68
opencv−python−headless==4.7.0.68
Pillow==9.4.0
psycopg2==2.9.5
sqlparse==0.4.3
tzdata==2022.7
whitenoise==6.3.0
```

source–code/models.py

```
#MODELS.PY

from django.db import models
from .validators import file_size

class Monitor(models.Model):
    monitor_input = models.FileField(upload_to=”input/%y/%m/%d/”, validators=[file_size], null=True, blank=True)

    def __str__(self):
        return ’’

class Images(models.Model):
    monitor_images = models.FileField(upload_to=”image/%y”, null=True, blank=True)
    stamp = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return ’’

class Preprocessed(models.Model):
    preprocessed = models.FileField(upload_to=”image/%y”, null=True, blank=True)
    stamp = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return ’’
```

source–code/views.py

```
#VIEWS.PY

from tabnanny import check
from django.shortcuts import render, redirect
from django.http import HttpResponse, FileResponse
from django.core.files.base import File, ContentFile
from .models import ∗
from .forms import ∗
import os
import cv2
import numpy as np
from django.conf import settings
from PIL import Image, ImageEnhance
from django.contrib import messages
from zipfile import ZipFile
from wsgiref.util import FileWrapper
from django.views.decorators.csrf import csrf_exempt
import csv
import time
from django.http import HttpResponse
from django.shortcuts import render
from .models import ∗
from django.core.mail import EmailMessage
from django.views.decorators import gzip
from django.http import StreamingHttpResponse
from threading import Thread

def home(request):
    try:
        Images.objects.all().delete()
    except Images.DoesNotExist:
        pass

    return render(request, ’home.html’)

@csrf_exempt
def capturePic(request):
    try:
        Images.objects.all().delete()
    except Images.DoesNotExist:
        pass

    if request.method == ’POST’:
        f = open(’./file.jpg’, ’wb’)
        f.write(request.body)
        filePath = os.path.realpath(f.name)
        f.close()

        img = cv2.imread(filePath)
        final = preprocess(img)
        ret, buf = cv2.imencode(’.jpg’, final)
        content = ContentFile(buf.tobytes())
        img_model = Images()
        img_model.monitor_images.save(’outputFrame.jpg’, content)

    return render(request, ’capture.html’)

class WebcamStream:
    def __init__(self, stream_id):
        self.stream_id = stream_id  # default is 0 for main camera
        self.vcap = cv2.VideoCapture(self.stream_id)
        if self.vcap.isOpened() is False:
            exit(0)
        fps_input_stream = int(self.vcap.get(5))  # hardware fps

        self.grabbed, self.frame = self.vcap.read()
        if self.grabbed is False:
            print(’[Exiting] No more frames to read’)
            exit(0)
        self.stopped = False
        # thread instantiation
        self.t = Thread(target=self.update, args=())
        self.t.daemon = True  # daemon threads run in background

    def start(self):
        self.stopped = False
        self.t.start()

    def update(self):
        while True:
            if self.stopped is True:
                break
            self.grabbed, self.frame = self.vcap.read()
            if self.grabbed is False:
                self.stopped = True
                break
        self.vcap.release()

    def read(self):
        return self.frame

    def encode(self):
        img_model = Images()
        ret, buf = cv2.imencode(’.jpg’, self.frame)
        content = ContentFile(buf.tobytes())
        img_model.monitor_images.save(’output’ + ”_” + ”.jpg”, content)

    def stop(self):
        self.stopped = True

@csrf_exempt
def captureVid(request):
    try:
        Images.objects.all().delete()
    except Images.DoesNotExist:
        pass

    try:
        Preprocessed.objects.all().delete()
    except Preprocessed.DoesNotExist:
        pass

    if request.method == ’POST’:
        vidInput = request.FILES[”video”].file.name
        index, nameCounter = 1
        webcam_stream = WebcamStream(
            stream_id=vidInput)  # 0 id for main camera
        webcam_stream.start()
        vidcap = cv2.VideoCapture(vidInput)
        fps = vidcap.get(cv2.CAP_PROP_FPS)

        if ((fps >= 50 and fps <= 80) or fps == 1000):
            fps = 60
        else:
            fps = 30
        success, img = vidcap.read()

        while (success):
            if (index > fps and index % fps == 0):
                ret, buf = cv2.imencode(’.jpg’, img)
                content = ContentFile(buf.tobytes())

                img_model = Images()
                img_model.monitor_images.save(
                    ’output’ + ”_” + str(nameCounter) + ”.jpg”, content)
                nameCounter += 1
            index += 1
            success, img = vidcap.read()

        vidcap.release()
        num_frames_processed = 0
        img_model = Images()

        while True:
            if webcam_stream.stopped is True:
                break
            else:
                frame = webcam_stream.read()
                webcam_stream.encode()
            # adding a delay for simulating video processing time
            delay = 0.5  # delay value in seconds
            time.sleep(delay)
            num_frames_processed += 1

        webcam_stream.stop()

    return redirect(’/results’)

@csrf_exempt
def index(request):
    try:
        Preprocessed.objects.all().delete()
    except Preprocessed.DoesNotExist:
        pass
    if request.method == ”POST”:
        form = MonitorForm(request.POST, request.FILES)

        if form.is_valid() and ’monitor_input’ in request.FILES:
            form.save()
            monitorInput = Monitor.objects.latest(’id’)
            filePath = monitorInput.monitor_input.path
            extension = os.path.splitext(
                str(request.FILES[’monitor_input’]))[1]
            fileName = os.path.splitext(
                str(request.FILES[’monitor_input’]))[0]

            if (extension == ’.png’ or extension == ’.jpg’ or extension == ’.jpeg’ or
                extension == ’.JPG’ or extension == ’.PNG’ or extension == ’.JPEG’):
                img = cv2.imread(filePath)
                orig_img = img.copy()
                final = preprocess(orig_img)
                ret, buf = cv2.imencode(’.jpg’, final)
                content = ContentFile(buf.tobytes())
                img_model = Preprocessed()
                img_model.preprocessed.save(fileName + ’.jpg’, content)
            else:
                sec = 0
                frameRate = 2  # //it will capture image every 2 seconds
                count = 1
                success, img = getFrame(sec, filePath)
                orig_img = img.copy()

                while success:
                    count = count + 1
                    sec = sec + frameRate
                    sec = round(sec, 2)
                    final = preprocess(img)
                    ret, buf = cv2.imencode(’.jpg’, final)
                    content = ContentFile(buf.tobytes())
                    img_model = Preprocessed()
                    img_model.preprocessed.save(
                        fileName + ”_” + str(count−1) + ”.jpg”, content)
                    success, img = getFrame(sec, filePath)
            outputImages = Preprocessed.objects.all()

            context = {’form’: form,
                       ’filePath’: filePath,
                       ’monitorInput’: monitorInput,
                       ’outputImages’: outputImages}
            messages.success(request, ”File succcessfully uploaded.”)
        else:
            context = {’form’: form}
            messages.error(
                request, ”No file chosen or size exceeds limit.”)

        return render(request, ’index.html’, context)

    form = MonitorForm()
    context = {’form’: form}
    return render(request, ’index.html’, context)

def preprocess(img):
    orig_img = img.copy()
    gamma, output = GCME(img)
    img = cv2.bilateralFilter(output, 11, 125, 100)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    canny = cv2.Canny(gray, 40, 120)
    canny = cv2.dilate(canny, cv2.getStructuringElement(
        cv2.MORPH_ELLIPSE, (3, 3)))
    con = np.zeros_like(img)
    contours, hierarchy = cv2.findContours(
        canny, cv2.RETR_LIST, cv2.CHAIN_APPROX_NONE)
    page = sorted(contours, key=cv2.contourArea, reverse=True)[:5]
    con = cv2.drawContours(con, contours, −1, (0, 255, 255), 3)
    con = np.zeros_like(img)
    maxArea = 0
    biggest = []

    for c in page:
        area = cv2.contourArea(c)
        if area > 100000:
            epsilon = 0.02 ∗ cv2.arcLength(c, True)
            corners = cv2.approxPolyDP(c, epsilon, True)
            if area > maxArea and len(corners) == 4:
                biggest = corners
                maxArea = area

    if len(biggest) != 0:
        cv2.drawContours(con, c, −1, (0, 255, 255), 3)
        cv2.drawContours(con, biggest, −1, (0, 255, 0), 10)
        biggest = sorted(np.concatenate(biggest).tolist())

        for index, c in enumerate(biggest):
            character = chr(65 + index)
            cv2.putText(con, character, tuple(
                c), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0), 1, cv2.LINE_AA)

        biggest = order_points(biggest)
        destination_corners = find_dest(biggest)
        M = cv2.getPerspectiveTransform(np.float32(
            biggest), np.float32(destination_corners))
        final = cv2.warpPerspective(
            orig_img, M, (destination_corners[2][0], destination_corners[2][1]), flags=cv2.INTER_LINEAR)
    else:
        final = orig_img

    return final

def getFrame(sec, file_name):
    vidcap = cv2.VideoCapture(file_name)
    vidcap.set(cv2.CAP_PROP_POS_MSEC, sec*1000)
    hasFrames, image = vidcap.read()

    return hasFrames, image

def GCME(image, mask=None, normalize=False):
    if np.ndim(image) == 3 and image.shape[-1] == 3:  # color image
        hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
        img = hsv[:, :, 2]
        color_flag = True
    elif np.ndim(image) == 2:  # gray image
        img = image
        color_flag = False
    else:
        return 1, None

    if normalize:  # max-min normalization
        img = img.astype(np.float)
        img = (255 * (img - np.min(img[:])) / (np.max(img[:]
                                                          ) - np.min(img[:]) + 0.1)).astype(np.float)

    img = (img + 0.5) / 256

    img_log = np.log(img)
    if mask is not None:
        mask[mask < 255] = 0
        img_log[mask == 0] = np.NaN
    gamma = -1 / np.nanmean(img_log[:])

    output = np.power(img, gamma)

    if mask is not None:
        output = (output * 256 - 0.5) * mask / 255.0
    else:
        output = (output * 256 - 0.5)
    output = output.round().astype(np.uint8)
    if color_flag:
        hsv[:, :, 2] = output
        output = cv2.cvtColor(hsv, cv2.COLOR_HSV2BGR)

    return gamma, output

def order_points(pts):
    rect = np.zeros((4, 2), dtype='float32')
    pts = np.array(pts)
    s = pts.sum(axis=1)
    rect[0] = pts[np.argmin(s)]
    rect[2] = pts[np.argmax(s)]
    diff = np.diff(pts, axis=1)
    rect[1] = pts[np.argmin(diff)]
    rect[3] = pts[np.argmax(diff)]
    return rect.astype('int').tolist()

def find_dest(pts):
    (tl, tr, br, bl) = pts
    widthA = np.sqrt(((br[0] - bl[0]) ** 2) + ((br[1] - bl[1]) ** 2))
    widthB = np.sqrt(((tr[0] - tl[0]) ** 2) + ((tr[1] - tl[1]) ** 2))
    maxWidth = max(int(widthA), int(widthB))

    heightA = np.sqrt(((tr[0] - br[0]) ** 2) + ((tr[1] - br[1]) ** 2))
    heightB = np.sqrt(((tl[0] - bl[0]) ** 2) + ((tl[1] - bl[1]) ** 2))
    maxHeight = max(int(heightA), int(heightB))

    destination_corners = [[0, 0], [maxWidth, 0],
                           [maxWidth, maxHeight], [0, maxHeight]]

    return order_points(destination_corners)

def guide(request):
    return render(request, 'guide.html')

def capture(request):
    return render(request, 'capture.html')

def dataset(request):
    return render(request, 'dataset.html')

def resultPic(request):
    monitorImages = Images.objects.all()

    context = {
        'monitorImages': monitorImages
    }

    return render(request, 'resultPic.html', context)

def results(request):
    monitorImages = Images.objects.all()

    context = {
        'monitorImages': monitorImages
    }

    return render(request, 'results.html', context)

def preprocessing(request):
    count = 1

    monitorImages = Images.objects.all()
    for monitorImage in monitorImages:
        img = cv2.imread(monitorImage.monitor_images.path)
        final = preprocess(img)
        ret, buf = cv2.imencode('.jpg', final)
        content = ContentFile(buf.tobytes())
        img_model = Preprocessed()
        img_model.preprocessed.save(
            'output' + "_" + str(count) + ".jpg", content)
        count += 1

    return redirect('/processed')

def processed(request):
    preprocessedImages = Preprocessed.objects.all()

    context = {
        'preprocessedImages': preprocessedImages
    }

    return render(request, 'processed.html', context)

def downloadZipResults(request):
    monitor_images = Images.objects.all()

    with ZipFile('outputframes.zip', 'w') as export_zip:
        for monitor_image in monitor_images:
            img_path = monitor_image.monitor_images.path
            print(img_path)
            export_zip.write(img_path, img_path.split("\\")[-1])

    wrapper = FileWrapper(open('outputframes.zip', 'rb'))
    content_type = 'application/zip'
    content_disposition = 'attachment; filename=Frames.zip'

    response = HttpResponse(wrapper, content_type=content_type)
    response['Content-Disposition'] = content_disposition

    return response

def downloadZipProcessed(request):
    monitor_images = Preprocessed.objects.all()

    with ZipFile('outputframes.zip', 'w') as export_zip:
        for monitor_image in monitor_images:
            img_path = monitor_image.preprocessed.path
            export_zip.write(img_path, img_path.split("\\")[-1])

    wrapper = FileWrapper(open('outputframes.zip', 'rb'))
    content_type = 'application/zip'
    content_disposition = 'attachment; filename=Frames.zip'

    response = HttpResponse(wrapper, content_type=content_type)
    response['Content-Disposition'] = content_disposition

    return response

def downloadCSV(request):
    monitor_images = Images.objects.all()
    stamp = ['Time Stamp']
    img_path = ['Image File']

    response = HttpResponse(
        content_type='text/csv',
        headers={'Content-Disposition': 'attachment; filename="Data.csv"'},
    )

    writer = csv.writer(response)
    for monitor_image in monitor_images:
        stamp.append(monitor_image.stamp)
        img_path.append(monitor_image.monitor_images.name)
    for value in range(len(stamp)):
        writer.writerow([stamp[value], img_path[value]])

    return response

def downloadCSVProc(request):
    monitor_images = Preprocessed.objects.all()
    stamp = [’Time Stamp’]
    img_path = [’Image File’]

    response = HttpResponse(
        content_type=’text/csv’,
        headers={’Content−Disposition’: ’attachment; filename=”Data.csv”’},
    )

    writer = csv.writer(response)
    for monitor_image in monitor_images:
        stamp.append(monitor_image.stamp)
        img_path.append(monitor_image.preprocessed.name)
    for value in range(len(stamp)):
        writer.writerow([stamp[value], img_path[value]])

    return response

def delete(image):
    os.remove(image)
```

source–code/urls.py

```
#URLS.PY

from django.urls import path
from django.urls import path
from . import views

urlpatterns = [
    path(’’, views.home, name=’home’),
    path(’guide/’, views.guide, name=’guide’),
    path(’upload/’, views.index, name=”upload”),
    path(’capture/’, views.capture, name=”capture”),
    path(’captureVid/’, views.captureVid, name=”captureVid”),
    path(’capturePic/’, views.capturePic, name=”capturePic”),
    path(’dataset/’, views.dataset, name=”dataset”),
    path(’results/’, views.results, name=”results”),
    path(’resultPic/’, views.resultPic, name=”resultPic”),
    path(’preprocessing/’, views.preprocessing, name=”preprocessing”),
    path(’processed/’, views.processed, name=”processed”),
    path(’downloadZipResults/’, views.downloadZipResults,
        name=”downloadZipResults”),
    path(’downloadZipProcessed/’, views.downloadZipProcessed,
        name=”downloadZipProcessed”),
    path(’downloadCSV/’, views.downloadCSV, name=”downloadCSV”),
    path(’downloadCSVProc/’, views.downloadCSVProc, name=”downloadCSVProc”),
]
```

source–code/capture.js

```
// VIDEO CAPTURE ACCESS GRANT

let start = document.getElementById(”btnStart”);
let stop = document.getElementById(”btnStop”);
let vidPreview = document.getElementById(”vid1”);
let vidSave = document.getElementById(”vid2”);
let recordingStatus = document.getElementById(”status”);
let canvasImg = document.getElementById(”canvasimg”);
let downloadImage = document.getElementById(”downloadImage”);
let downloadImageContainer = document.getElementById(”downloadImageContainer”);
let videoURL = ””;
let chunks = [];

let constraintObj = {
  audio: false,
  video: {
    width: {
      min: 720,
      ideal: 1280,
      max: 3840,
    },
    frameRate: { min: 10, ideal: 60, max: 80 },
    facingMode: { ideal: ”environment” },
    height: {
      min: 480, //HD
      ideal: 720, //FHD
      max: 2160, //4k
    },
  },
};

start.addEventListener(”click”, (ev) => {
  //handle older browsers that might implement getUserMedia in some way
  if (navigator.mediaDevices === undefined) {
    navigator.mediaDevices = {};
    navigator.mediaDevices.getUserMedia = function (constraintObj) {
      let getUserMedia =
        navigator.webkitGetUserMedia || navigator.mozGetUserMedia;
      if (!getUserMedia) {
        return Promise.reject(
          new Error(”getUserMedia is not implemented in this browser”)
        );
      }
      return new Promise(function (resolve, reject) {
        getUserMedia.call(navigator, constraintObj, resolve, reject);
      });
    };
  } else {
    navigator.mediaDevices
      .enumerateDevices()
      .then((devices) => {
        devices.forEach((device) => {
          console.log(device.kind.toUpperCase(), device.label);
        });
      })
      .catch((err) => {
        console.log(err.name, err.message);
      });
  }

  navigator.mediaDevices
    .getUserMedia(constraintObj)
    .then(function (mediaStreamObj) {
      //connect the media stream to the first video element
      let video = document.querySelector(”video”);
      if (”srcObject” in video) {
        video.srcObject = mediaStreamObj;
      } else {
        //old version
        video.src = window.URL.createObjectURL(mediaStreamObj);
      }

      video.onloadedmetadata = function (ev) {
        //show in the video element what is being captured by the webcam
        video.play();
      };

      let mediaRecorder = new MediaRecorder(mediaStreamObj);

      mediaRecorder.start();
      recordingStatus.innerHTML =
        ”<span style=’color: red’>” + ”Status: Currently Recording” + ”</span>”;
      console.log(mediaRecorder.state);

      stop.addEventListener(”click”, (ev) => {
        mediaRecorder.stop();
        console.log(mediaRecorder.state);
      });

      mediaRecorder.ondataavailable = function (ev) {
        if (ev.data.size > 0) {
          chunks.push(ev.data);
        } else {
          console.log(”NO DATA”);
        }
      };

      mediaRecorder.onstop = (ev) => {
        let blob = new Blob(chunks, {
          type: ”video/mp4”,
        });
        chunks = [];
        videoURL = URL.createObjectURL(blob);
        vidSave.src = videoURL;
        recordingSize = parseFloat(blob.size / 1000000).toFixed(2);
        recordingStatus.innerHTML =
          ”<span style=’color: green’>” +
          ”Status: Stopped Recording (” +
          recordingSize +
          ” MB)”;
        (”</span>”);
        let loadingBtnVid = document.getElementById(”loadingBtnVid”);
        loadingBtnVid.style.display = ”block”;

        var fd = new FormData();
        var file = new File([blob], ”vidd.mp4”);
        fd.append(”video”, file);

        console.log( file );
        var xhr = new XMLHttpRequest();
        xhr.open(”POST”, ”/captureVid/”, true);
        xhr.onload = function (e) {
          console.log(”Sent”);
          loadingBtnVid.style.display = ”none”;

          let x = document.getElementById(”proceed”);

          x.style.display = ”inline”;
        };
        xhr.send(fd);
      };
    })
    .catch(function (err) {
      console.log(err.name, err.message);
    });
});

function getCookie(name) {
  var cookieValue = null;
  if (document.cookie && document.cookie !== ””) {
    var cookies = document.cookie.split(”;”);
    for (var i = 0; i < cookies.length; i++) {
      var cookie = cookies[i].trim();
      if (cookie.substring(0, name.length + 1) === name + ”=”) {
        cookieValue = decodeURIComponent(cookie.substring(name.length + 1));
        break;
      }
    }
  }
  return cookieValue;
}

function sendPicData(data) {
  let csrftoken = getCookie(”csrftoken”);
  let response = fetch(”/capturePic/”, {
    method: ”post”,
    body: data,
    headers: { ”X-CSRFToken”: csrftoken },
  }).then((data) => {
    let loadingBtnPic = document.getElementById(”loadingBtnPic”);
    loadingBtnPic.style.display = ”none”;

    downloadImage.style.color = ”inherit”;
    downloadImage.style.textDecoration = ”none”;
    downloadImage.style.display = ”inline”;
    downloadImageContainer.style.display = ”inline”;
    proceedPic.style.display = ”inline”;
  });
}

const width = 1280; // We will scale the photo width to this
let height = 0; // This will be computed based on the input stream

let streaming = false;

let video = null;
let canvas = null;
let photo = null;
let startbutton = null;
let startPic = document.getElementById(”initiate”);
let proceedPic = document.getElementById(”proceedPic”);

startPic.addEventListener(”click”, (ev) => {
  video = document.getElementById(”video”);
  canvas = document.getElementById(”canvas”);
  photo = document.getElementById(”photo”);
  startbutton = document.getElementById(”startbutton”);

  navigator.mediaDevices
    .getUserMedia(constraintObj)
    .then((stream) => {
      video.srcObject = stream;
      recordingStatus.innerHTML =
        ”<span style=’color: red’>” + ”Status: Currently Active” + ”</span>”;
      video.play();
    })
    .catch((err) => {
      console.error(‘An error occurred: ${err}‘);
    });

  video.addEventListener(
    ”canplay”,
    (ev) => {
      if (!streaming) {
        height = video.videoHeight / (video.videoWidth / width);

        if (isNaN(height)) {
          height = width / (4 / 3);
        }

        video.setAttribute(”width”, width);
        video.setAttribute(”height”, height);
        canvas.setAttribute(”width”, width);
        canvas.setAttribute(”height”, height);
        streaming = true;
      }
    },
    false
  );

  startbutton.addEventListener(
    ”click”,
    (ev) => {
      takepicture();
      recordingStatus.innerHTML =
        ”<span style=’color: green’>” + ”Status: Photo Taken!” + ”</span>”;
      ev.preventDefault();
    },
    false
  );

  clearphoto();
});

function takepicture() {
  canvasImg.style.display = ”none”;
  const context = canvas.getContext(”2d”);
  if (width && height) {
    canvas.width = width;
    canvas.height = height;
    context.drawImage(video, 0, 0, width, height);

    const data = canvas.toDataURL(”images/png”);
    photo.setAttribute(”src”, data);

    canvas.toBlob((blob) => {
      let loadingBtnPic = document.getElementById(”loadingBtnPic”);
      loadingBtnPic.style.display = ”block”;
      imageURL = URL.createObjectURL(blob);
      downloadImage.setAttribute(”href”, data);
      sendPicData(blob);
    });
  } else {
    canvasImg.style.display = ”none”;
  }
}
```

source-code/object_detection.py

```
# -*- coding: utf-8 -*-
”””[Diversified] Train_Object_Detection_model_TF2.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1CyHKmW2VMhHu00ViJOoAJfS69NT5iUoG

## **SP ROADMAP**

Author: Jan Federico Coscolluela IV

* Collect the dataset of images and label them to get their xml files.

* Install the TensorFlow Object Detection API.

* Generate the TFRecord files required for training. (need generate_tfrecord.py script and csv files for this)

* Edit the model pipeline config file and download the pre-trained model checkpoint.

* Train and evaluate the model.

* Apply OCR to detected objects.

# **Initialization**

## **1) Import Libraries**
”””

import os
import glob
import xml.etree.ElementTree as ET
import pandas as pd
import tensorflow as tf
import cv2
```

```
print(tf.__version__)

”””## **2) Mount drive and link your folder**”””

from google.colab import drive
drive.mount(’/content/gdrive’, force_remount=True)
!ln −s /content/gdrive/My\ Drive/ /mydrive

”””## **3) Clone the tensorflow models git repository & Install
        TensorFlow Object Detection API**
”””

!git clone −−q https://github.com/tensorflow/models.git
!protoc object_detection/protos/*.proto −−python_out=.
!cp object_detection/packages/tf2/setup.py .
!python −m pip install .

”””## **4) Test the model builder**
”””

# testing the model builder
!python object_detection/builders/model_builder_tf2_test.py

”””# **Workspace Setup**

## **5) Unzip the *images.zip* and *annotations.zip* files into
        the *data* folder**
”””

!unzip /mydrive/Monixor/Detection/diversified/images.zip −d .
!unzip /mydrive/Monixor/Detection/diversified/annotations.zip
        −d .

”””## **6) Create test_labels & train_labels**

Divide annotations into test_labels (20%) and train_labels(80%).
”””

!mkdir test_labels train_labels
!ls annotations/* | sort −R | head −100 | xargs −I{} mv {}
        test_labels/
!ls annotations/* | xargs −I{} mv {} train_labels/

”””## **7) Create the CSV files and the ”label_map.pbtxt” file
        **

Run xml_to_csv script below to create ***test_labels.csv*** and
        ***train_labels.csv***

This also creates the ***label_map.pbtxt*** file using the
        classes mentioned in the xml files.
”””

def xml_to_csv(path):
  classes_names = []
  xml_list = []

  for xml_file in glob.glob(path + ’/*.xml’):
    tree = ET.parse(xml_file)
    root = tree.getroot()
    for member in root.findall(’object’):
      classes_names.append(member[0].text)
      value = (root.find(’filename’).text ,
               int(root.find(’size’)[0].text),
               int(root.find(’size’)[1].text),
               member[0].text,
               int(member[4][0].text),
               int(member[4][1].text),
               int(member[4][2].text),
               int(member[4][3].text))
      xml_list.append(value)
  column_name = [’filename’, ’width’, ’height’, ’class’, ’xmin’,
        ’ymin’, ’xmax’, ’ymax’]
  xml_df = pd.DataFrame(xml_list, columns=column_name)
  classes_names = list(set(classes_names))
  classes_names.sort()
  return xml_df, classes_names

for label_path in [’train_labels’, ’test_labels’]:
  image_path = os.path.join(os.getcwd(), label_path)
  xml_df, classes = xml_to_csv(label_path)
  xml_df.to_csv(f’{label_path}.csv’, index=None)
  print(f’Successfully converted {label_path} xml to csv.’)

label_map_path = os.path.join(”label_map.pbtxt”)
pbtxt_content = ””

for i, class_name in enumerate(classes):
    pbtxt_content = (
        pbtxt_content
        + ”item {{\n    id: {0}\n    name: ’{1}’\n}}\n\n”.
      format(i + 1, class_name)
    )
pbtxt_content = pbtxt_content.strip()
with open(label_map_path, ”w”) as f:
    f.write(pbtxt_content)
    print(’Successfully created label_map.pbtxt’)

”””## **8) Create train.record & test.record files**
Run the *generate_tfrecord.py* script to create *train.record*
        and *test.record* files
”””

!python /mydrive/Monixor/Detection/diversified/
        generate_tfrecord.py train_labels.csv label_map.pbtxt
        images/ train.record
!python /mydrive/Monixor/Detection/diversified/
        generate_tfrecord.py test_labels.csv label_map.pbtxt
        images/ test.record

”””## **9) Download pre−trained model checkpoint**
Download **ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu−8.tar.
        gz** into the ***data*** folder & unzip it.
A list of detection checkpoints for tensorflow 2.x can be found
         [here](https://github.com/tensorflow/models/blob/master
        /research/object_detection/g3doc/tf2_detection_zoo.md).
”””

!wget http://download.tensorflow.org/models/object_detection/
        tf2/20200711/ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu
        −8.tar.gz
!tar −xzvf ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu−8.tar.gz

”””## **10) Get the model pipeline config file, make changes to
         it and put it inside the *data* folder**
Edit the config file from ***/content/models/research/
        object_detection/configs/tf2*** in colab and copy the
        edited config file to the ***/mydrive/customTF2/data***
         folder.
You can also find the pipeline config file inside the model
        checkpoint folder we just downloaded in the previous step.
**You need to make the following changes:**
*   change ***num_classes*** to number of your classes.
*   change ***test.record*** path, ***train.record*** path &
        ***labelmap*** path to the paths where you have created
        these files (paths should be relative to your current
        working directory while training).
* change ***fine_tune_checkpoint*** to the path of the directory
         where the downloaded checkpoint from step 12 is.
* change ***fine_tune_checkpoint_type*** with value **
        classification** or **detection** depending on the type..
* change ***batch_size*** to any multiple of 8 depending upon
        the capability of your GPU.
(eg:− 24,128,...,512) .Mine is set to 64.
* change ***num_steps*** to number of steps you want the
        detector to train.
”””

!cp /content/models/research/object_detection/configs/tf2/
        ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu−8.config /
        mydrive/Monixor/Detection/mAP/data

”””## **11) Load Tensorboard**”””

load tensorboard
%tensorboard −−logdir ’/content/gdrive/MyDrive/Monixor/
        Detection/diversified/training’

”””# **Model Training**

## Navigate to the ***object_detection*** folder in colab vm
”””

”””## 12) Training using model_main_tf2.py

Here **{PIPELINE_CONFIG_PATH}** points to the pipeline
        config and **{MODEL_DIR}** points to the directory in
        which training checkpoints and events will be written.

For best results, you should stop the training when the loss is
         less than 0.1 if possible, else train the model until the
         loss does not show any significant change for a while.
         The ideal loss should be below 0.05 (Try to get the loss
         as low as possible without overfitting the model.)
”””

!python model_main_tf2.py −−pipeline_config_path=/mydrive/
        Monixor/Detection/diversified/data/
        ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu−8.config −−
        model_dir=/mydrive/Monixor/Detection/diversified/
        training −−alsologtostderr

”””## 13) Export inference graph
”””

!python exporter_main_v2.py −−trained_checkpoint_dir=/
        mydrive/Monixor/Detection/diversified/training −−
        pipeline_config_path=/content/gdrive/MyDrive/Monixor/
        Detection/diversified/data/
        ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu−8.config −−
    output_directory /mydrive/Monixor/Detection/diversified/data/inference_graph

"""# **Object Detection & OCR**

## 14) Test Object Detection
"""

!wget https://freefontsdownload.net/download/160187/arial.zip
!unzip arial.zip -d .

!sed -i "s/font = ImageFont.truetype('arial.ttf', 50)/font = ImageFont.truetype('arial.ttf', 50)/" visualization_utils.py

import tensorflow as tf
import time
import numpy as np
import warnings
warnings.filterwarnings('ignore')
from PIL import Image
from google.colab.patches import cv2_imshow
from object_detection.utils import label_map_util
from object_detection.utils import visualization_utils as viz_utils
import matplotlib.pyplot as plt

filename = "test"

IMAGE_SIZE = (10, 8) # Output display size as you want
PATH_TO_SAVED_MODEL="/mydrive/Monixor/Detection/diversified/data/inference_graph/saved_model"
print('Loading model...', end='')

# Load saved model and build the detection function
detect_fn=tf.saved_model.load(PATH_TO_SAVED_MODEL)
print('Done!')

category_index=label_map_util.create_category_index_from_labelmap("<path/to/label_map.pbtxt>",use_display_name=True)

def load_image_into_numpy_array(path):

    return np.array(Image.open(path))

image_path = "<path/to/image>"

image_np = load_image_into_numpy_array(image_path)
input_tensor = tf.convert_to_tensor(image_np)
input_tensor = input_tensor[tf.newaxis, ...]
detections = detect_fn(input_tensor)

num_detections = int(detections.pop('num_detections'))
detections = {key: value[0, :num_detections].numpy()
              for key, value in detections.items()}

detections['detection_classes'] = detections['detection_classes'].astype(np.int64)

image_np_with_detections = image_np.copy()

viz_utils.visualize_boxes_and_labels_on_image_array(
        image_np_with_detections,
        detections['detection_boxes'],
        detections['detection_classes'],
        detections['detection_scores'],
        category_index,
        use_normalized_coordinates=True,
        max_boxes_to_draw=200,
        line_thickness=3,
        min_score_thresh=0.4, # Adjust this value to set the minimum probability boxes to be classified as True
        agnostic_mode=False)
# %matplotlib inline
plt.figure(figsize=IMAGE_SIZE, dpi=200)
plt.axis("off")
plt.imshow(image_np_with_detections)
plt.show()

"""## 16) Test Optical Character Recognition"""

pip install easyocr

import torch
import torch.nn as nn
import torch.nn.functional as F
import torchvision
import torchvision.transforms as transforms
import easyocr
import cv2 #opencv
from matplotlib import pyplot as plt
import numpy as np
from google.colab.patches import cv2_imshow

def ocr_detection(full_path, imgWidth, imgHeight, min_score):
  IMAGE_SIZE = (10, 8) # Output display size as you want
  PATH_TO_SAVED_MODEL="<path/to/saved/detection_model>"

  detect_fn=tf.saved_model.load(PATH_TO_SAVED_MODEL)

  category_index=label_map_util.create_category_index_from_labelmap("<path/to/label_map.pbtxt>",use_display_name=True)

  def load_image_into_numpy_array(path):

      return np.array(Image.open(path))

  image_np = load_image_into_numpy_array(full_path)

  input_tensor = tf.convert_to_tensor(image_np)
  input_tensor = input_tensor[tf.newaxis, ...]
  detections = detect_fn(input_tensor)

  image_np_with_detections = image_np.copy()
  image = image_np_with_detections

  num_detections = int(detections.pop('num_detections'))
  detections = {key: value[0, :num_detections].numpy()
                for key, value in detections.items()}

  detections['detection_classes'] = detections['detection_classes'].astype(np.int64)

  scores = list(filter(lambda x:x> min_score, detections['detection_scores']))
  boxes = detections['detection_boxes'][:len(scores)]
  classes = detections['detection_classes'][:len(scores)]

  for idx, box in enumerate(boxes):
    roi = box*[imgHeight, imgWidth, imgHeight, imgWidth]
    region = image[int(roi[0]):int(roi[2]), int(roi[1]):int(roi[3])]
    reader = easyocr.Reader(['en'], verbose=False)
    result = reader.readtext(region, detail=0, min_size=20, paragraph=True)

    if detections['detection_classes'][idx] == 1:
      print("bloodpressure: ", result)
    elif detections['detection_classes'][idx] == 2:
      print("heartrate: ", result)
    elif detections['detection_classes'][idx] == 3:
      print("map: ", result)
    elif detections['detection_classes'][idx] == 4:
      print("oxygensaturation: ", result)
    elif detections['detection_classes'][idx] == 5:
      print("pulserate: ", result)
    elif detections['detection_classes'][idx] == 6:
      print("respiratoryrate: ", result)
    else:
      print("temperature: ", result)

  viz_utils.visualize_boxes_and_labels_on_image_array(
        image_np_with_detections,
        detections['detection_boxes'],
        detections['detection_classes'],
        detections['detection_scores'],
        category_index,
        use_normalized_coordinates=True,
        max_boxes_to_draw=200,
        line_thickness=3,
        skip_labels=True, #removes lables
        min_score_thresh=min_score, # Adjust this value to set the minimum probability boxes to be classified as True
        agnostic_mode=False)
  plt.figure(figsize=IMAGE_SIZE, dpi=200)
  plt.axis("off")
  plt.imshow(image_np_with_detections)
  plt.show()

full_path = "<path/to/test_image>"
img = cv2.imread(full_path)
img_width = img.shape[1]
img_height = img.shape[0]
ocr_detection(full_path, img_width, img_height, 0.4)
```

source–code/extract_frame.py

```
# Frame Extraction Script

import os
import cv2
import glob
from pathlib import Path
import time

```

```
def getFrame(sec,file_name,count, short_name):
    vidcap = cv2.VideoCapture(file_name)
    vidcap.set(cv2.CAP_PROP_POS_MSEC,sec*1000)
    hasFrames,image = vidcap.read()
    if hasFrames:
        cv2.imwrite(r"<path>"+short_name+"_"+str(count)+".jpg", image)     # save frame as JPG file
        print(short_name+str(count))
    return hasFrames

# apply getFrame to all videos in a folder
for filename in glob.iglob(f'{video_folder_path}/*'):
    sec = 0
    frameRate = 2 #//it will capture image every 2 seconds
    count=1
    if filename.endswith("<video_file_extension>"):
      nameNoExtension = Path(filename).stem
      success = getFrame(sec,filename,count, nameNoExtension)
      while success:
        count = count + 1
        sec = sec + frameRate
        sec = round(sec, 2)
        success = getFrame(sec,filename,count, nameNoExtension)
```

XI. Acknowledgment

I would like to express my sincere gratitude to my adviser, Sir Marasigan, for his invaluable guidance and support throughout the duration of this project. His constructive feedbacks and insights were instrumental in shaping the direction and progress of my work.

I am also deeply thankful to Doc Aljibe and Doc Marcelo for their mentorship and for providing me with the opportunity to engage in a meaningful project with practical applications in healthcare. Their expertise in the medical field and dedication to fostering a stimulating learning environment have been pivotal in shaping my understanding and passion for the subject matter.

Furthermore, I would like to extend my heartfelt appreciation to my mom for being my go-to support system since college day 1. Her belief in my abilities has been a constant source of motivation, and for that I am grateful. To my family, whom I hold in the highest regard, this accomplishment is equally yours as it is mine.