Improved Automated Analysis of Coronary Doppler Echocardiograms to Predict Early Coronary Microvascular Disease

Thesis

Presented in Partial Fulfillment of the Requirements for Honors Research Distinction in Computer Science and Engineering at The Ohio State University

By

Jamie Bossenbroek

Undergraduate Program in Computer Science and Engineering

The Ohio State University

2022

Thesis Committee

Dr. William Ray, Advisor

Dr. Xia Ning, Co-Advisor

Abstract

Coronary microvascular disease (CMD) is a heart condition that frequently precedes the development of more serious heart diseases. Although it can be assessed through Transthoracic Doppler echocardiography (TTDE) by observing changes in coronary blood flow patterns, manual analysis of TTDE is time consuming and subject to bias. In a previous study, a program was created to automatically analyze coronary blood flow patterns by parsing TTDE videos into a single continuous image, binarizing and separating the image into distinct cardiac cycles, and extracting characteristic data values from each cycle. The program significantly reduced variability and time to complete analysis, but obstacles such as interfering noise and varying video sizes left room to increase the program's accuracy. The goal of this study was to improving the program's ability to handle challenging video variations and to remove unnecessary manual intervention to further reduce analysis time. To confirm this improved analysis, several videos were analyzed using both the original MATLAB program and updated Python program. Comparison of specific examples showed the new program was better able to identify and remove difficult noise objects, and more consistently and fully captured the Doppler region. This improved analysis has the potential to provide more insight into the early diagnosis of unhealthy coronary flow by offering a quick, easy, and accurate method of analysis.

Dedication

Dedicated to the Students at the Ohio State University.

Acknowledgments

Thank you to Dr. William Ray, Dr. William Bartlett, Dr. Aaron Trask, and the many other members of Trask Lab for inviting me to join this project and for encouraging my growth and participation over the last several semesters.

Thank you to Dr. Xia Ning for taking interest in my project and agreeing to serve as my co-advisor. I'm incredibly appreciative for your ongoing support and flexibility.

Vita

May 2022	B.S. Computer Science and Engineering, The Ohio State University
2019 – Present	Undergraduate Teaching Assistant, OSU Engineering Education Department
Summer 2020	Technology Intern, Nationwide Mutual Insurance
Summer 2021	Technology Intern, Capital One Bank

Publications

Bossenbroek J., Ueyama Y., McCallinhart P., Bartlett C., Ray W., Trask A., Improvement of Automated Analysis of Coronary Doppler Echocardiograms. Scientific Reports.

Fields of Study

Major Field: Computer Science and Engineering

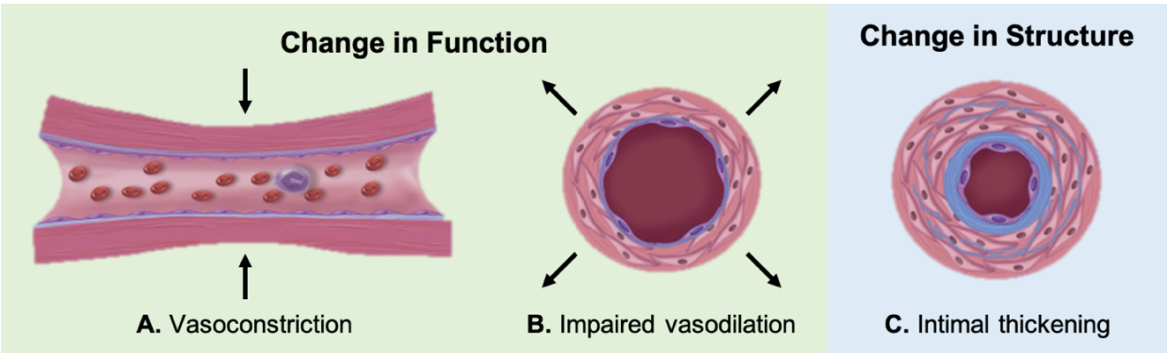
Table of Contents

List of Tables

List of Figures

Chapter 1. Introduction

Coronary microvascular disease (CMD) is a heart condition affecting the smaller blood vessels that branch off from the main coronary arteries. Impairments in the coronary microcirculation disrupt the healthy regulation of myocardial blood flow and nutrient exchange[1,2]. CMD is a nonobstructive coronary artery disease, meaning that although there is no physical blockage in the arteries, oxygenated blood is unable to move through smaller blood vessels at an adequate rate to maintain physiological demand[3]. Instead, functional and structural changes to the microvasculature, such as those shown in Figure 1 below, prevent healthy levels of blood flow to the heart[4]. This condition has been shown to be strongly associated with diabetes, and when paired with myocardial ischemia – or reduced blood flow – it is referred to as nonobstructive coronary artery disease (INOCA).



A. Vasoconstriction, or narrowing of the blood vessels, limits blood flow. **B.** Impaired dilation of blood vessels lowers blood pressure. **C.** Increased thickness of vessel walls results from smooth muscle cells and restricts blood flow.

Figure adapted by author from [4]

Figure 1: Examples of Abnormal Structure and Function or Coronary Microcirculation

CMD shares many of the same risk factors as atherosclerosis, or blockage in the arteries, and it often overlaps or precedes the development of obstructive coronary artery disease[2]. For example, previous studies by our lab observed inward hypertrophic remodeling and reduced vessel stiffness contributing to myocardial ischemia before the occurrence of occlusive atherosclerosis[7]. Most importantly, CMD is one of the earliest signs of heart disease which can lead to myocardial infarction, heart failure, and/or stroke[1,2]. The functional and structural deficits associated with CMD are indicators that can be observed before the appearance of symptoms such as circulating biomarkers or atherosclerosis[2]. With early and accurate identification of CMD, more serious and life-threatening cardiac conditions can be treated and prevented before they become deleterious.

While indirect methods to diagnose CMD are available, they are fraught with subjectivity. Positron emission tomography (PET) and magnetic resonance imaging (MRI) offer value in identifying impairments in cardiac perfusions, but currently include no direct measures to diagnose CMD[4]. Transthoracic Doppler echocardiography (TTDE) is an affordable and non-invasive method used to assess cardiovascular function through direct measurements of coronary blood flow (CBF), with potential to assess CMD. CBF is measured from one of the main coronary arteries under both baseline and stress (hyperemic) conditions, and this yields uniquely characteristic flow patterns in which diastole predominates and which can be analyzed to indicate impaired CBF[5]. For example, coronary flow velocity reserve (CFVR) is indicative of the amount of additional blood flow that the microvasculature can carry under stress; CFVR is lower in cases of coronary artery

disease, even in otherwise asymptomatic subjects or in patients with INOCA[6]. However, manual analysis of TTDE can be time consuming, difficult to learn, and subject to both intra-rater and inter-rater bias[8]. To resolve these issues, our groups began developing a MATLAB program to automatically extract data values from coronary flow patterns of TTDE video files[9].

For each cardiac cycle in the TTDE flow pattern, the original MATLAB program automatically extracted several key parameters including the peak velocity and velocity time integral, which are metrics commonly used to quantify coronary health. CFVR was then calculated as the average peak hyperemic velocity divided by the average peak baseline velocity. When analyzing 98 baseline files and 117 hyperemic files both manually and with the automatic program, linear regression analysis showed significantly reduced variability when using automatic analysis, and the time to analyze videos was reduced from 1500 minutes to 50 minutes. However, agreement between manual and automatic parameter output ranged from less than 1% difference to over 55% difference for certain variables[9]. This parameter variability suggested that with continued testing and program adjustments, automatic analysis of TTDEs could become more accurate and capable of processing challenging videos.

Extensive testing identified several potential areas in which improved analysis was possible, including the removal of interfering noise, the identification of fainter cardiac cycles, and the verification of peak selection in the ECG region. The original program was also limited to a single video height and width in pixels, which excluded the analysis of many Doppler videos. In this study, we present the results of an effort to improve the

accuracy of the first-generation program through development of several key areas of analysis. The original program was developed in MATLAB but was recapitulated in Python in order to leverage libraries such as OpenCV for computer vision and Google's TensorFlow for downstream machine learning. As Python is also an industry standard for machine learning development, changing to that environment allows us to leverage innovations in machine learning from both academia and industry much faster going forward.

We hypothesized that modified heuristics could better address the original program's limitations, and that these refinements would produce a comprehensive and accurate method for examiners to classify coronary flow issues through interpretation of CFVR values and other patterns in parameter output. These improvements would allow examiners to take advantage of the speed and consistency offered by automated analysis without sacrificing diagnostic accuracy in assessing coronary diseases.

Chapter 2. Materials and Methods

TTDE video files with approximately 20 distinct heart beats each were acquired from 12-week, 16-week, and 36-week old normal Db/db and type 2 diabetic (T2DM) db/db mice (Jackson Laboratories) at both baseline and hyperemic (high flow) conditions[10]. Doppler readings were measured at 1% isoflurane (baseline) and 3% isoflurane (hyperemia), and all measurements were taken from the left main coronary artery of the mice as previously described[10]. These videos were exported as .avi files from the VevoLab 3.1.1 software and analyzed offline using the original MATLAB program and improved Python program. Mice were housed under a 12-hr light/dark cycle at 22°C and 60% humidity. They were allowed ad libitum access to water and were fed standard laboratory mice chow. This study was conducted in accordance with National Institutes of Health Guidelines and was approved by the Institutional Animal Care and Use Committee at the Abigail Wexner Research Institute at Nationwide Children's Hospital.

2.1 Algorithm Description

The improved program was written in Python, and utilized the following libraries: sys, cv2 (OpenCV), PIL, scipy, skimage, matplotlib, tkinter, pandas, and numpy. Initial data processing began with prompts to select the folder containing the video files to be analyzed, input a name for the output excel file, select the type of analysis as 'Doppler' or 'Combined' (the latter including analysis of color mode videos to measure vessel diameters), and finally to select each video file to be analyzed. The tkinter library was used

to generate all GUIs. The new interface expanded on the functionality of the original program by allowing more than one baseline and/or hyperemic video file to be selected for analysis in each run as well as by accepting videos with any pixel height and width. The peak velocity on each video's Doppler scale was extracted using optical character recognition, and then if color mode analysis was selected the user was prompted to enter the probe angle and minimum/maximum penetration in mm from the B Mode window.

Once all parameters had been entered, the program parsed each video by inspecting the difference between subsequent video frames to identify frames where the scroll bar reset from the right to the left side of the doppler window. This section of the program was expanded from the original version by adding additional checks to confirm that no reset frames were missed or duplicated when parsing. These frames were then concatenated into a single continuous image which was cropped to the region of interest containing the coronary flow pattern and electrocardiogram (ECG) recording. A gaussian filter was applied, the image was dilated with a linear structuring element, and then a global threshold value was calculated using OpenCV and Otsu's method for image binarization. A representative binarized image is depicted in Figure 2 below.

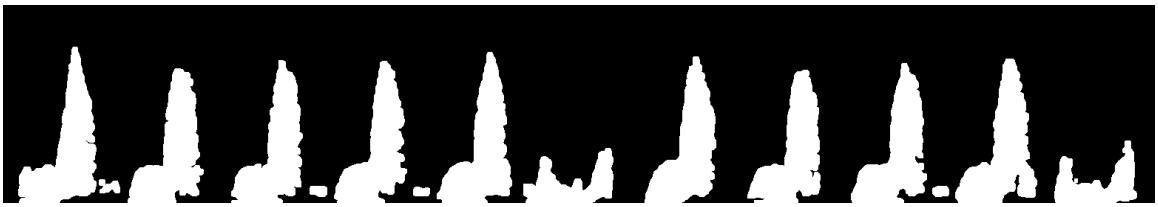


Figure 2: Binarized image of Doppler region

Users had the opportunity to adjust the manually selected threshold value in a 'Scroll Test' window, where the threshold could be incremented or decremented to visually inspect how the level of filtering would affect the amount of data captured in the binarized

doppler image. Increasing the threshold value removed additional noise, while decreasing the threshold expanded the included envelope. Once the threshold value was verified, the program removed any remaining noise objects and filled small holes in the image using OpenCV. The updated program applied more complete noise removal by identifying all contours in the image and removing any objects that were not within ten pixels of the horizontal baseline. This corrected for any remaining noise objects and was especially useful in removing any interfering noise located at the top of the Doppler window. Next, the program split the Doppler region into cardiac cycles by identifying peaks in the corresponding ECG pattern. Peaks were initially identified using scipy, and then an added check compared the distance between each peak to remove extra peaks that were too close together and to fill larger empty gaps with estimated peak locations. Finally, the program extracted the following parameters from each cardiac cycle: peak velocity, diastolic velocity, decay velocity, systolic rise time, diastolic rise time, diastolic decay time, systolic slope, diastolic slope, decay slope, heart rate, and velocity time integral. These were the same parameters extracted in the original MATLAB program, and in the update program the numpy library was used to complete calculations[9]. All parameters were saved to a pandas data-frame and then output to a Microsoft Excel file in .xlsx format.

The program generated an image of the coronary flow pattern with diastolic velocity (indicating the beginning of the diastolic phase), peak velocity (maximum velocity for each cycle), decay velocity (the point at which acceleration switched signs closest to peak diastolic deceleration), peak diastolic deceleration (minimum acceleration), and end of cycle (correlating with the peaks found in the ECG region) indicated with green, yellow,

pink, red, and blue points respectively. An example of this image is depicted in Figure 3 below. The plots and corresponding output parameters generated by both the MATLAB and Python programs were inspected to find discrepancies where algorithmic or heuristic improvements could increase analysis accuracy.

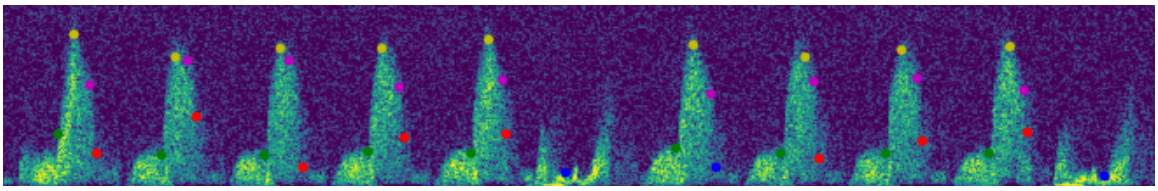


Figure 3: Image generated by Automatic Program

The program also included an option for analysis of coronary diameters from B-mode color videos, which is required to calculate coronary blood flow (CBF)[10]. The algorithm began by masking the first frames of both the color mode and corresponding Doppler videos and identifying the borders of the B mode window and the center lines indicating where vessel measurements were taken from. For each video frame of the color mode file, these values were used to crop to a region around the center location before rotating the image based on the angle of the probe. The corresponding length of each pixel in mm was calibrated from the minimum and maximum probe depths entered by the user at the beginning of video analysis. The program then masked each image and identified any contours; if the contour was large enough and in the correct location to exclude noise or ventricle filling, the diameter of the identified vessel was then calculated by finding the average distance between the left and right vessel walls of the object. The program output the minimum, maximum, mean, median, mode, and standard deviation of all diameters for each analyzed video. An example of a measured vessel is shown in Figure 4.

Figure 4: Example of a Vessel Diameter measured during Colormode Analysis

2.2 Methodology

A collection of 18 Doppler video sets evenly distributed between 12, 16, and 36 week old healthy and diabetic mice were processed with both the original MATLAB program and the improved Python program. All tests were performed on the same computer by a single tester who entered in any prompted values and adjusted the threshold value for binarization as needed to fully capture the Doppler envelope without including noise. Each video set included one baseline and one hyperemic video, and the Python program also analyzed the corresponding color mode videos acquired at baseline and hyperemic conditions. Videos were intentionally selected by the tester through visual inspection of video files in order to demonstrate a wide range of processing difficulty, from videos containing distinct Doppler regions with little noise to videos that the MATLAB program struggled to handle. Some challenging patterns included interfering ‘top noise’ descending from the top of the Doppler image, poor contrast between background noise and the Doppler signal, and inconsistent ECG readings that led to the incorrect separation of cardiac cycles. Testing with the improved program could then demonstrate through specific examples that modified heuristics were better able to handle challenging videos,

while videos with clearer signals that had already been fully captured by the original program would continue to generate similar data values.

2.3 Statistics

The table of parameters for each cardiac cycle generated by the two programs were saved to a Microsoft Excel file, and for each parameter the mean and standard deviation (SD) across all cycles were calculated. With color mode analysis included, CBF could be calculated using the equation as previously described by our lab[10]:

$$CBF \text{ (mL/min)} = ((\pi/4) \times D2 \times VTI \times HR)/1000$$

The percent difference between the MATLAB and Python average values and standard deviations were then calculated for each parameter. The percent difference was a useful statistic to uniformly evaluate the change in values between MATLAB and Python program analysis as opposed to the numerical change which varied based on the maximum velocity of each individual video's scale. An f-test was performed to compare the peak velocity values of the two data sets and to determine if the variances of the sets were equal. Finally, a t-test (assuming equal or unequal variance based on the results of the f-test) with a significance level of $p<0.05$ was performed to compare the average peak velocity values and CBF. The calculated data was then categorized into groups based on the obstacles present in the video for comparison and evaluation of the improved program's effectiveness.

3. Results

Overall, standard deviation decreased from the MATLAB program to the Python program (Table 1). Standard deviation for peak velocity values decreased by an average of 50.0% for baseline flow videos and 32.1% for hyperemic flow videos, and VTI standard deviation decreased by 51.2% and 35.2% for baseline and hyperemic videos respectively. In individual cases where standard deviation noticeably increased for these parameters, factors such as interfering noise (videos labeled as 'Top noise') or incorrect identification of fainter peaks (videos labeled as 'Missing fainter peaks') had influenced the calculated standard deviation for the MATLAB program's output.

When examining the two tailed t-tests performed between the peak velocity values of the MATLAB and Python programs, p-values indicated statistical significance when the Python program made significant improvements to the video's analysis, such as through removal of top noise, extraction of cycles that were missed in the original analysis, or removal of unrepresentative peaks from the final data set. For cases where the original analysis was accurate, the p-values suggested that the two data sets were equal. In Table 2, baseline videos which were accurately captured and analyzed by the MATLAB program had an average p-value of 0.20, which did not indicate significance between the peak velocity values of the two programs. On the other hand, baseline videos that had several cardiac cycles that were not fully captured by the MATLAB program but which were correctly analyzed by the Python program had an average p-value of 0.004, which did indicate significant differences. Videos with top noise saw similarly lower p-values.

Average peak velocity and VTI values tended to increase when using the new algorithm, with the exception of videos where noise at the top of the Doppler region had been captured by the MATLAB program. The change in each individual baseline video's peak velocity from the MATLAB to the Python program is displayed in Figure 10 in Appendix A, with videos affected by top noise indicated with red dots, accurate analysis indicated with green points, videos with fainter peaks indicated with yellow, and inaccurate ECG peak identification shown with blue points. The overall average peak velocity values, excluding top noise videos, are shown by the green line. In this figure, the peak velocity for accurately analyzed videos remained similar from the MATLAB to the Python program, while videos with top noise had a significant decrease in peak velocity values and videos with fainter peaks that were not fully captured by the original program tended to have an increase in peak velocity values when analyzed by the updated program. Overall, when not considering top noise videos, peak velocity values increased by an average of 19.3% ± 13.6% and 10.9% ± 8.3% for baseline and hyperemic videos respectively and VTI values increased by 26.4% ± 25.7% and 8.1% ± 29.3% (Table 1).

Several examples of the specific changes that contributed to overall performance improvement are investigated in the rest of this section. Removal of interfering top noise from the Doppler envelope made it possible for the improved program to capture the correct cardiac velocities. The program also added checks to verify ECG peak values so that cardiac cycles weren't skipped or broken into multiple sections. Finally, the program fully captured fainter cardiac cycles that had been previously overlooked and removed unrepresentative cycles from consideration, both of which were changes that decreased

standard deviation and increased average peak velocity and VTI values. When making comparisons, peak velocity and VTI values were selected as the parameters for analysis because they are most representative of the analyzed doppler region and are the values most commonly utilized in clinical practice.

3.1 Removal of Top Noise

Many of the videos analyzed in this dataset demonstrated the Python program's ability to identify and remove difficult noise objects from the binarized image. To accomplish this, the new algorithm added steps to eliminate any large areas of noise which weren't close to the baseline of the image. This was accomplished using openCV's findCountours function to identify larger artifacts. For example, the representative baseline and hyperemic videos displayed in Figure 5A contained significant top noise which was captured by the MATLAB program. However, when analyzed by the Python program, this noise was removed from consideration in the binarized image and the program could extract accurate values, as shown by the critical points in Figure 5B.

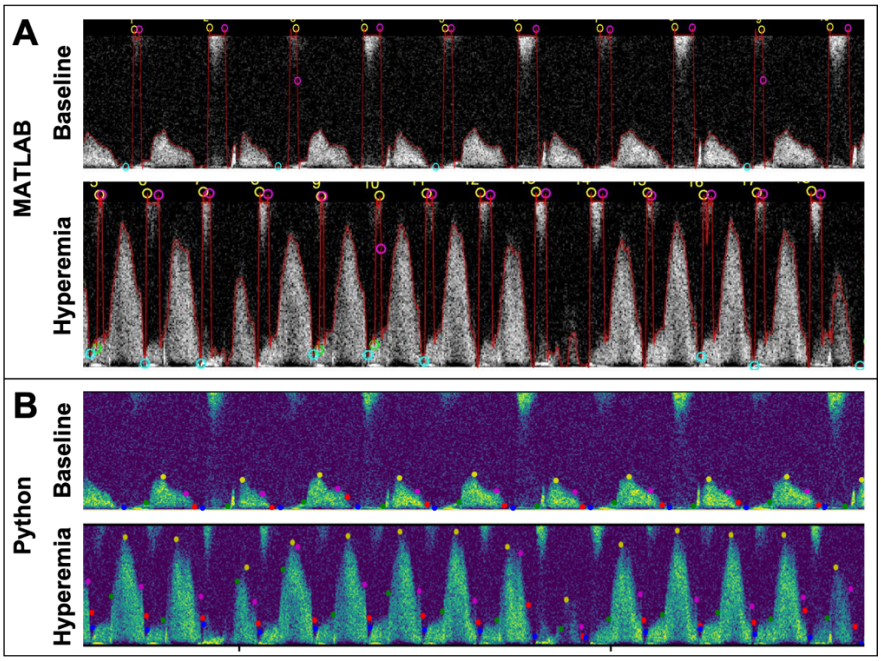


Figure 5: Removal of Top Noise

Removing top noise to more accurately capture the correct velocity values often produced a decrease in average values for the Python program's values as velocities were no longer forced to the top of the Doppler window. On average, baseline videos with top noise had a 95.6% ± 27.5% decrease in average peak velocity values, and hyperemic videos had a 8.5% ± 10.2% decrease (Table 2). This change also contributed to an overall decrease in standard deviation values; standard deviation of peak velocity values decreased by 101.3% and 5.3% for baseline and hyperemic videos respectively. For some individual examples of videos with top noise, standard deviation for peak velocity values increased by up to 88.7% because when processed by the original program, most or all peak velocities were driven up to an identical maximum value. However, contrary to indications of increased variability, removal of top noise is a step that allows the new program to more

appropriately extract data values from videos which could not be optimally analyzed by the MATLAB program.

3.2 Division of ECG Region

Before extracting data values, the Doppler region is broken into distinct cardiac cycles by identifying peaks in the ECG region. In cases of unusual ECG readings however, the MATLAB program was unable to identify the correct number of peaks in this region. For example, the ECG pattern dropping below the baseline frequently led to a skipped peak. This resulted in the program missing several QRS complex peaks and thus leaving some cardiac cycles unanalyzed - as depicted in Figure 6 - or in the program selecting multiple peaks within one cardiac cycle, as shown in Figure 7. In Figure 6, 9 of the total 15 ECG peaks were identified by the original program, while in Figure 7 an additional 7 peaks were identified along with the 15 correct ones. With added verification and corrections in the Python program, all 15 ECG peaks are identified in Figure 6, and only the correct 15 peaks are identified in Figure 7 with no additional peaks. Adding more of the correct ECG peaks to the analysis of this first example decreased standard deviation for peak velocity values by 11.9% and increased the average value by 6.4%, and removing incorrectly added peaks from the second representative example decreased standard deviation of peak velocity values by 31.2% and increased the average value by 34%.

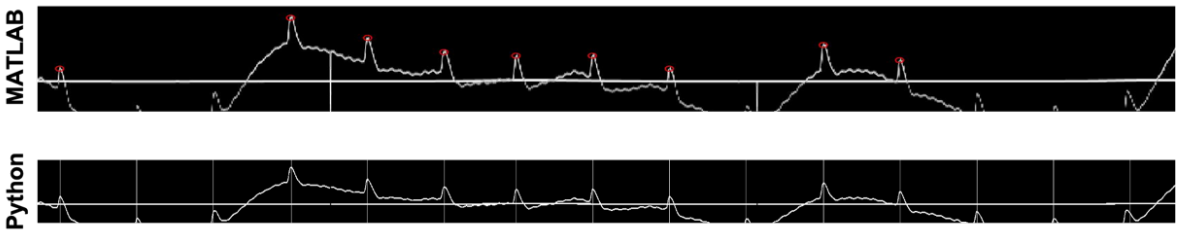


Figure 6: Corrected ECG peak identification to identify missed peaks

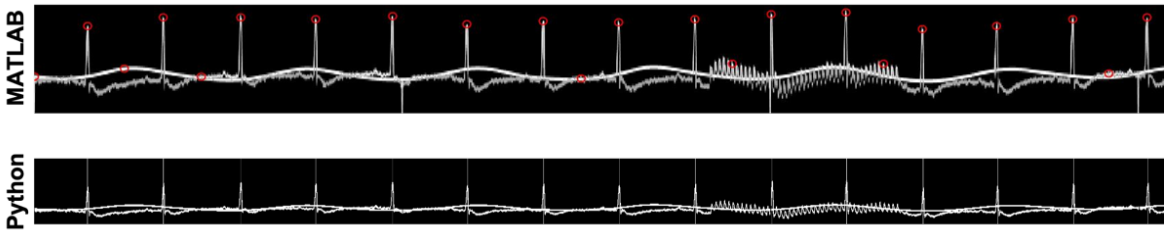


Figure 7: Corrected ECG peak identification to remove incorrect peaks

3.3 Identification of Fainter Peaks

The new program was able to overcome some of the difficulty of identifying fainter peaks in the Doppler region, especially in cases where other cardiac cycles were significantly brighter or there was surrounding noise. By employing a more aggressive method of noise removal in the region above the doppler flow, the Python program was able to accept a lower threshold for binarization in order to capture these fainter peaks without also including surrounding noise in the final binarized image. The original algorithm does not fully capture cycles 2 and 4 when analyzing a representative baseline video, as shown in Figure 8A, but these peaks are fully captured and analyzed by the Python program in Figure 8B. This adjustment was incorporated into the program without compromising its ability to exclude unrepresentative flow cycles that result from the coronary artery moving in and out of view of the flow probe.

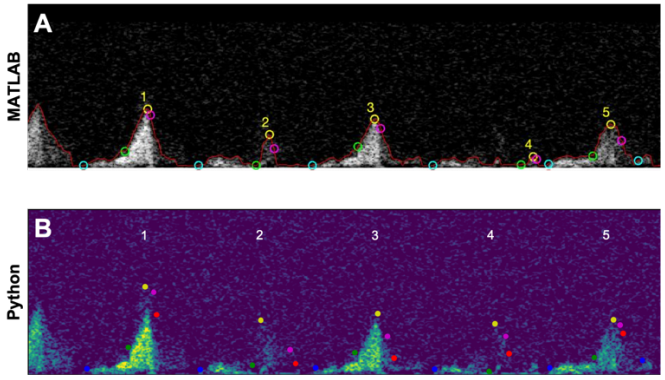


Figure 8: Correction to fully capture fainter cycles in the Doppler region

Correct identification and analysis of these previously overlooked cycles resulted in increased average peak velocity, VTI, and decreased standard deviation. On average, baseline videos in this category increased peak velocity and VTI values by 29.6% ± 7.6% and 40.8% ± 22.6% respectively when incorporating the Python program's corrections, while standard deviation fell by 41.2% and 16.1% respectively (Table 2).

3.4 Removal of Unrepresentative Cycles

The final main improvement made by the Python program to increase accuracy was to remove any unrepresentative cycles from the program's final output. Unrepresentative cycles were identified by comparing a cycle's peak velocity and VTI values to the data set's averages, and if these values were comparatively too low (due to the coronary transiently falling out of the view of the ultrasound during the cardiac cycle) they were removed from the final data set. Removing this information helped to produce more uniform results by not taking into consideration either incorrectly analyzed cardiac cycles or cycles that may have been correctly captured but were not representative of the doppler region's overall trends. For example, in Figure 9, the Python program had correctly

identified the Doppler region and analyzed each cardiac cycle, but cycles 5, 6, 11, and 12 were not representative of the rest of the data set, so they were removed from the final data table and subsequent calculation of average values and standard deviation.

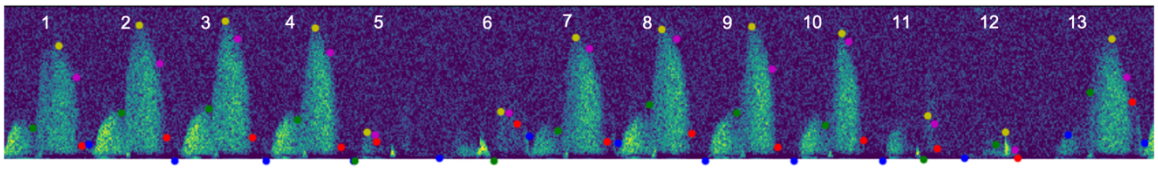


Figure 9: Image showing unrepresentative cycles in the Doppler region

Removing the unrepresentative cardiac cycles from this representative video resulted in a 16.8% increase in average peak velocity and a 17.05% increase in VTI. Standard deviation then decreased by 102.21% and 57.53% for peak velocity and VTI respectively. For the subsequent analysis, videos in which unrepresentative cycles were removed were grouped together with videos which improved identification of fainter peaks because both modifications accomplished the shared purpose of removing inaccurate lower values from analysis. Because of this, both adaptations saw a similar increase in peak velocity and VTI values and decrease in standard deviation values.

4. Discussion

The early identification of CMD has the potential to treat and prevent the development of more serious heart conditions such as atherosclerosis, stroke, and heart failure. TTDE is an effective and non-invasive method which can be used to assess coronary flow by observing coronary flow patterns, and automatic analysis of coronary blood flow was demonstrated in a previous study by this laboratory to reduce the time required for analysis and the bias typical of manually-analyzed TTDE files[9]. Here, we presented improvements to the original program. The updated program took advantage of several Python libraries, and with improved heuristics was able to handle a larger scope of data inputs and accurately analyze more challenging Doppler echocardiogram videos.

This study aimed to refactor the original program, transition to a Python environment for use of OpenCV, Tensorflow, and other libraries, and to add additional checks and improvements identified through testing of the original program in order to increase analysis accuracy and more effectively handle difficult video cases. The new code improves handling of interference from top noise, validates identification of ECG peaks, correctly estimates parameters from fainter peaks, and removes unrepresentative data from the final data set. In addition, the program functionality was expanded by accepting videos of any pixel height and width, allowing multiple baseline and hyperemic videos to be analyzed in one run, and reducing manual intervention by implementing optical character recognition to determine the maximum velocity on the Doppler window's scale.

One major advantage of moving the program to Python was the use of OpenCV for video processing and image analysis. The MATLAB program interpreted each video frame

as a cell array of pixel values and analyzed the images to identify the horizontal baseline position, regions of interest to crop to, and threshold values for binarization. The Python program utilized functions of the OpenCV library to accomplish these steps, as well as for grayscale conversion and for applying a gaussian filter and dilation to the Doppler region before calculating the binarization threshold. The findContours function was also useful in adding modified heuristics to identify top noise and other noise objects that needed to be removed.

The Python program took advantage of several other Python libraries for specific analysis steps; numpy was used for array manipulation and mathematical calculations as parsed images were treated as arrays of pixel values, tkinter was used to create GUIs for user interaction, and matplotlib was used to plot critical values on the images of the coronary flow pattern that were saved from each processed video. As future developments are added, the Python environment can be extended to utilize TensorFlow, scikit-learn, and other libraries for further data analysis and machine learning algorithms.

The data and examples provided specifically demonstrate the program's ability to remove top noise, to improve identification of peak ECG values, to better capture fainter cardiac cycles, and to remove unrepresentative cardiac cycles from analysis. Overall, this resulted in decreased standard deviations from the original to the improved program. This decrease in standard deviation indicates a more consistent analysis of each cardiac cycle and the proper removal of inaccurate cycles. Increased average peak velocities and VTI values in cases except those dealing with top noise interference also demonstrate the

program's improved analysis as the Doppler region was more fully captured during the binarization and noise removal steps.

4.1 Related Studies

A handful of similar programs have been developed to implement automated analysis to reduce processing time and parameter variability when assessing coronary blood flow. Many of these programs rely on partial-automation combined with expert analysis to enhance accuracy without removing manual intervention. For example, one program developed in MATLAB was used to crop video frames to the region of interest containing the Doppler envelope and apply a binarization threshold adjusted by the user, similar to the verification included in this current study's algorithm[12]. The program analyzed the Doppler region in frames containing three heartbeats at a time, with each frame taking between 10 and 40 seconds to analyze, and calculated a subset of the parameters found in in this study; peak diastolic velocity, peak diastolic acceleration, beginning diastolic phase, peak systolic velocity, and peak diastolic deceleration. When analyzing 200 videos from 100 patients, linear regression indicated strong correlation to manual analysis in PSV ($r = 0.986$, $P < 0.0001$, SE = 2.51 cm/s) and PDV ($r = 0.998$, $P < 0.0001$, SE = 1.58 cm/s)[12].

A similar study focused on removing all manual intervention from Doppler aortic flow analysis in order to minimize bias and analysis time[13]. The program tested Doppler strips of several heartbeats at a time and followed a similar procedure of cropping to the region of interest, binarizing the image to capture the Doppler envelope, and extracting

critical values from each cardiac cycle. The program was advantageous in that it didn't rely on QRS complex peaks in the ECG region to divide the Doppler region into cardiac cycles, but instead relied only on the Doppler area to separate cycles. This program displayed and overall strong correspondence in identified VTI and PV values to expert analysis, and saw a 10-fold reduction in time for analysis.

4.2 Limitations

Removing user interaction in favor of more computer automation would help to increase consistency, especially in identification of the correct threshold level for image binarization. However balancing user interactivity with complete automation is necessary for evaluators to adjust for errors and special cases, so allowing for a manual adjustment of the threshold value for image binarization is most effective. The option to adjust the binarization threshold is a critical element that needs to remain in order for a trained expert in coronary flow to assess the suitability of the pattern moving in and out of view of the Doppler during the cardiac cycle. This is a phenomenon that's difficult to automate, but easily and quickly corrected for by manual verification of the binarization threshold.

5. Conclusions

Comparison of the data values and plots generated from the original MATLAB and improved Python programs serve to demonstrate the increased accuracy of the updated algorithm to automatically measure CBF. Specifically, the updated program increased the ability to process a wider range of video sizes, special cases, and inaccurate readings that the original program didn't have checks to handle. The improved program is able to remove top noise and other large noise artifacts, to verify the correct identification of ECG peaks, to better capture fainter peaks in the doppler region, and to remove unrepresentative values from the final set of parameters. The program accepts any video pixel height and width and allows for the analysis of more than one baseline and hyperemic video at a time. Videos that had already been accurately analyzed by the MATLAB program continued to output similar data values, while videos that were corrected to more fully and consistently capture the Doppler region showed decreased standard deviation and increased peak velocity and VTI values. The program has achieved its goal of improving algorithm heuristics in order to better handle special cases, and can be used by examiners as an efficient, fast, and exact way to automatically analyze coronary Doppler echocardiograms.

Bibliography

1 McCallinhart, P., Scandling, B. W. & Trask, A. J. Coronary Remodeling and Biomechanics: Are We Going with the Flow in 2020? *Am J Physiol Heart Circ Physiol*, doi:10.1152/ajpheart.00634.2020 (2020).

2 Labazi, H. & Trask, A. J. Coronary microvascular disease as an early culprit in the pathophysiology of diabetes and metabolic syndrome. *Pharmacol Res* **123**, 114-121, doi:10.1016/j.phrs.2017.07.004 (2017).

3 Herscovici, R. *et al.* Ischemia and No Obstructive Coronary Artery Disease (INOCA): What Is the Risk? *J Am Heart Assoc* **7**, e008868, doi:10.1161/jaha.118.008868 (2018).

4 Taqueti, V. R. & Di Carli, M. F. Coronary Microvascular Disease Pathogenic Mechanisms and Therapeutic Options: JACC State-of-the-Art Review. *J Am Coll Cardiol* **72**, 2625-2641, doi:10.1016/j.jacc.2018.09.042 (2018).

5 Simova, I. Coronary Flow Velocity Reserve Assessment with Transthoracic Doppler Echocardiography. *Eur Cardiol* **10**, 12-18, doi:10.15420/ecr.2015.10.01.12 (2015).

6 Hartley, C. J. *et al.* Coronary flow reserve in mice: effects of age, coronary disease, and vascular loading. *Annu Int Conf IEEE Eng Med Biol Soc* **2010**, 3780-3783, doi:10.1109/iembs.2010.5627571 (2010).

7 Trask, A. J. *et al.* Dynamic micro- and macrovascular remodeling in coronary circulation of obese Ossabaw pigs with metabolic syndrome. *J Appl Physiol (1985)* **113**, 1128-1140, doi:10.1152/japplphysiol.00604.2012 (2012).

8 Finegold, J. A. *et al.* Choosing between velocity-time-integral ratio and peak velocity ratio for calculation of the dimensionless index (or aortic valve area) in serial follow-up of aortic stenosis. *Int J Cardiol* **167**, 1524-1531, doi:10.1016/j.ijcard.2012.04.105 (2013).

9 Sunyecz, I. L., McCallinhart, P. E., Patel, K. U., McDermott, M. R. & Trask, A. J. Defining Coronary Flow Patterns: Comprehensive Automation of Transthoracic Doppler Coronary Blood Flow. *Sci Rep* **8**, 17268, doi:10.1038/s41598-018-35572-4 (2018).

10 Katz, P. S. *et al.* Coronary arterioles in type 2 diabetic (db/db) mice undergo a distinct pattern of remodeling associated with decreased vessel stiffness. *Basic Res Cardiol* **106**, 1123-1134, doi:10.1007/s00395-011-0201-0 (2011).

11 Husarek, K. E. *et al.* The angiotensin receptor blocker losartan reduces coronary arteriole remodeling in type 2 diabetic mice. *Vascul Pharmacol* **76**, 28-36, doi:10.1016/j.vph.2015.06.013 (2016).

12 Magagnin, V., Delfino, L., Cerutti, S., Turiel, M. & Caiani, E. G. Nearly automated analysis of coronary Doppler flow velocity from transthoracic ultrasound images: validation with manual tracings. *Med Biol Eng Comput* **45**, 483-493, doi:10.1007/s11517-007-0178-x (2007).

13 Zolgharni, M. *et al.* Automated aortic Doppler flow tracing for reproducible research and clinical measurements. *IEEE Trans Med Imaging* **33**, 1071-1082, doi:10.1109/TMI.2014.2303782 (2014).

Appendix A. Data Summary

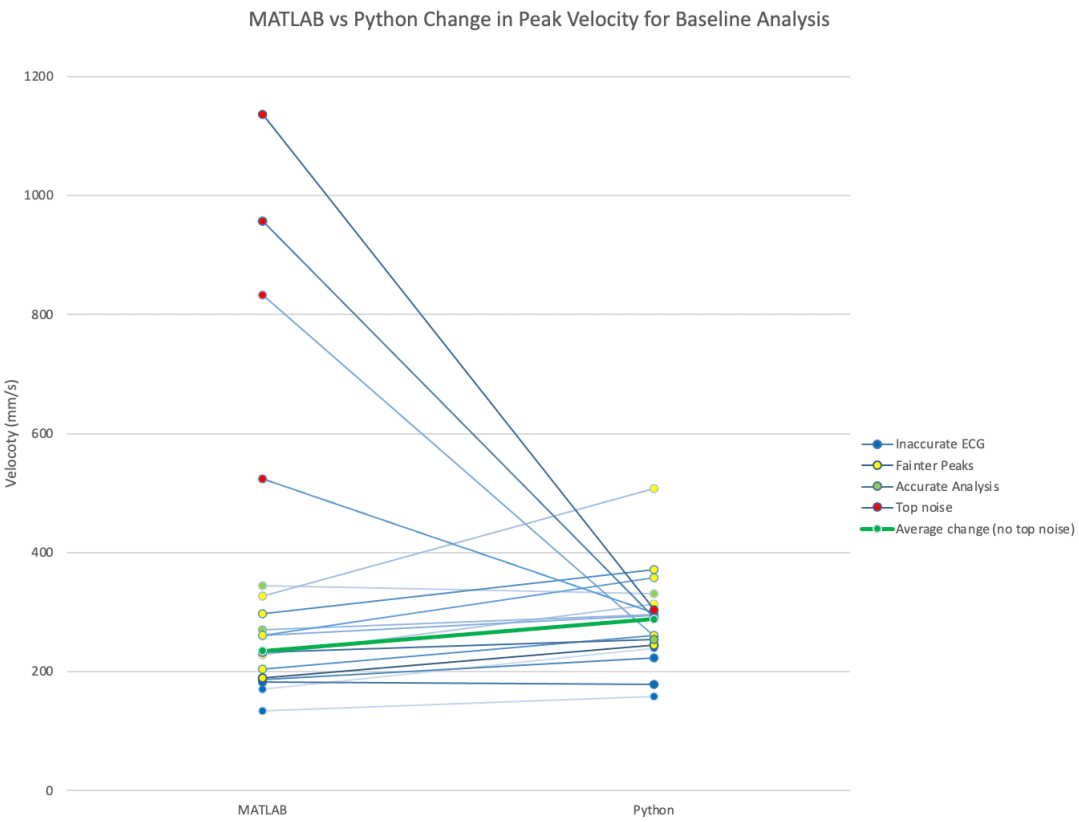


Figure 10: Change in Peak Velocity Values

Appendix B. Tables

Table 1: Coronary blood flow pattern variables assessed by the original MATLAB and the new Python programs at baseline and hyperemia

	BASELINE					HYPEREMIA				
	Average MATLAB Values	**Average Python Values**	**Average % Difference**	**+/- SD**	**Average % SD Difference**	**Average MATLAB Values**	**Average Python Values**	**Average % Difference**	**+/- SD**	**Average % SD Difference**
Systolic Rise Time (ms)	75.58	71.57	0.79	44.55	-54.61	73.94	62.90	-10.64	39.60	-59.22
Diastolic Rise Time (ms)	23.03	26.73	16.28	36.27	-11.65	29.05	29.05	1.25	27.80	-31.23
Diastolic Decay Time 1 (ms)	34.16	41.94	23.63	26.42	-11.43	27.91	31.80	9.36	31.88	-22.92
Diastolic Decay Time 2 (ms)	61.73	35.30	-50.94	35.94	-51.84	64.90	42.16	-40.48	43.29	-50.21
Systolic Slope (mm/s^2)	1042.46	443.38	25.82	173.15	-22.96	2883.07	3686.91	27.85	38.42	7.54
Diastolic Slope (mm/s^2)	24777.52	10698.94	-24.04	68.88	-29.16	29405.50	22284.09	-11.37	45.06	-40.01
Decay Slope 1 (mm/s^2)	-11183.36	-3159.46	-50.62	76.42	-72.15	-12850.25	-9942.69	-13.88	49.86	-68.12
Decay Slope 2 (mm/s^2)	-5509.69	-6718.63	23.72	44.74	16.78	-11247.26	-15374.79	26.34	29.13	9.31
Diastolic Velocity (mm/s)	70.99	71.63	179.36	659.25	2.81	211.53	290.45	31.14	34.73	-15.34
Peak Velocity (mm/s)	**374.33**	**287.91**	**-6.23**	**51.86**	**-50.00**	**798.03**	**847.01**	**5.50**	**12.37**	**-32.10**
Decay Velocity (mm/s)	283.97	187.37	-13.85	56.69	-40.78	573.62	561.72	0.09	25.38	-6.26
Heart Rate (BPM)	323.48	359.14	9.31	17.90	-30.09	320.89	375.22	15.10	16.25	-33.69
VTI (mm)	**24.14**	**22.11**	**4.15**	**49.55**	**-51.19**	**59.04**	**62.24**	**2.99**	**26.41**	**-35.20**

	BASELINE – NO TOP NOISE				HYPEREMIA – NO TOP NOISE			
	Average MATLAB Values	**Average Python Values**	**Average % Difference**	**+/- SD**	**Average MATLAB Values**	**Average Python Values**	**Average % Difference**	**+/- SD**
Systolic Rise Time (ms)	86.71	76.69	-13.14	20.25	83.04	65.92	-22.48	20.55
Diastolic Rise Time (ms)	25.25	26.74	4.46	22.87	30.63	29.11	-5.94	21.97
Diastolic Decay Time 1 (ms)	33.90	41.48	23.75	27.84	28.85	31.47	7.11	28.77
Diastolic Decay Time 2 (ms)	49.65	35.46	-38.11	28.23	58.08	41.66	-32.67	27.80
Systolic Slope (mm/s^2)	201.98	431.48	92.87	129.35	2540.62	3700.73	38.94	23.50
Diastolic Slope (mm/s^2)	10238.15	11007.39	7.71	26.87	21868.16	23225.37	7.45	16.99
Decay Slope 1 (mm/s^2)	-4179.26	-3291.31	-16.58	44.49	-11940.33	-10032.00	-14.98	37.36

Decay Slope 2 (mm/s^2)	-4166.63	-6776.93	42.91	26.96	-11762.49	-15699.49	26.95	18.06
Diastolic Velocity (mm/s)	40.87	73.31	53.85	42.10	222.39	295.15	26.84	17.52
Peak Velocity (mm/s)	**234.84**	**287.84**	**19.30**	**13.60**	**773.08**	**863.84**	**10.91**	**8.29**
Decay Velocity (mm/s)	163.14	189.29	13.46	16.22	525.57	576.42	9.90	16.98
Heart Rate (BPM)	324.32	351.83	6.81	19.70	315.76	370.09	15.14	17.26
VTI (mm)	**16.75**	**22.16**	**26.42**	**25.67**	**57.16**	**64.13**	**8.15**	**29.29**

Table 2: Coronary blood flow peak velocity and VTI as assessed by the original MATLAB and the new Python programs at baseline and hyperemia and under varying circumstances that occur in Doppler videos.

		Baseline		Hyperemia	
		Peak Velocity (mm/s)	**VTI (mm)**	**Peak Velocity (mm/s)**	**VTI (mm)**
Accurate Analysis	**Average MATLAB Values**	282.45	20.75	959.43	62.41
	Average Python Values	293.71	23.42	1000.72	61.89
	p-Value	0.20	0.11	0.26	0.45
	Average % Difference	4.65	11.74	3.26	-2.53
	+/- SD	7.46	32.23	5.05	7.86
	Average % SD Difference	-13.05	-11.64	-5.66	-21.12
ECG Inaccuracies	**Average MATLAB Values**	187.64	16.22	653.11	211.50
	Average Python Values	228.54	19.95	717.62	50.44
	p-Value	0.001	0.10	0.15	0.20
	Average % Difference	20.07	20.80	8.91	-12.55
	+/- SD	9.52	22.72	4.90	10.30
	Average % SD Difference	-25.83	-43.60	-37.55	-58.36

Fainter Peaks/Unrepresentative Cycles	**Average MATLAB Values**	251.33	16.01	718.19	49.50
	Average Python Values	342.65	24.68	886.41	79.59
	p-Value	0.004	0.08	**0.01**	0.01
	Average % Difference	29.63	40.78	20.55	42.79
	+/- SD	7.60	22.58	4.25	29.71
	Average % SD Difference	-41.24	-16.06	-71.06	-27.76
Top Noise	**Average MATLAB Values**	862.55	49.99	829.05	63.03
	Average Python Values	288.15	21.94	757.18	55.79
	p-Value	0.01	0.02	0.44	0.28
	Average % Difference	-95.55	-73.80	-8.54	-10.41
	+/- SD	27.53	25.33	10.16	9.00
	Average % SD Difference	-101.29	-141.09	-5.30	-23.74