ISSN 2278-3091

Volume 9, No.4, July – August 2020

International Journal of Advanced Trends in Computer Science and Engineering

Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse318942020.pdf

https://doi.org/10.30534/ijatcse/2020/318942020

Information Systems Using Object Counting and Background Subtraction from Video in Supermarket

Falih Farhanza[1], Samuel Mahatmaputra Tedjojuwono[2]

[1]Business Information Systems Program, Information Systems Department, Faculty of Computing and Media, Bina Nusantara University, Jakarta, Indonesia 11480

[2] Computer Science Department, Faculty of Computing and Media, Bina Nusantara University, Jakarta, Indonesia 11480, e-mail: samuelmt@binus.ac.id

ABSTRACT

The following work would like to propose an approach of computer vision in detecting and tracking goods on the shelfs. This approach would as well give valuable information of which shelfs are the most dynamic hence indicating traffic. In supermarket, products are taken from shelves by costumers are not detected before they go to the cashier. If that happened, they will have problem to detect how many product taken from shelves to help them maintaining product if all the product are sold. There is some method to help them maintaining product and one of them is using computer vision. Background subtraction can help to see the product that are taken from shelves in supermarket. It's not just it can detect product that are taken from shelves but also it can also count it. It can count how many product that are taken from shelves. This research will be focusing on object counter and object subtraction.

Key words : background subtraction, computer vision, object counting

1. INTRODUCTION

Computer vision is an exciting field that integrates soft-computing algorithms and computer hardware powerful enough to handle the computation required especially if it is to be used in real-time applications [1]. Computer vision is a technology that enables users to recognize object from image or even video using computer. Nowadays, computer can do a lot of things and these examples is not an exception. Image recognition also will be implemented in the program. Image recognition is one of computer vision that focus on recognizing object that are moving or being moved. Image Recognition can detect many things such as logos, object, people and other variable images or even in a video. [2] In this project, Image recognition will be used in video Product Placement means that the placement of the product. Product Placement can be found almost everywhere such as in movies, television shows, grocery stores, and many more. Product Placement also known as product brand placement is a marketing way to promote and advertising a brand name, product, package or even other trademark merchandise. Product Placement nowadays focus on movies and television shows. That way, Product Placement is one way to reduce the cost of production for movies and television shows. [3]

Computer vision by definition means that a machine that help users to recognize object in real world, such as how to extract information inside of a video, how to recognize object inside an image and many more. Computer vision enable machines to learn what kind of object to be observed e and what description to put on the object depends on what algorithm and purpose it is made by the users. [4]

The main goal of computer vision is to enable user to make decision about real physical object and scenes based on images sensed. Background subtraction (BS) is a technique that is used commonly for generating foreground mask (usually use for binary image that contain pixels that belongs to moving object in the scene). [5] There are many types of Morphology Transformation such as Opening, Closing, Top Hat, Black Hat, and Morphology Gradient. [6] It is use for removing small objects in dark foreground (assuming the object is bright). For example, the image on the left is the original image and the image on the right is the result of opening morphology transformation [6]. Image processing is one of computer vision example. Image processing is a processing of an image in order to extract an information that is necessary for further task of navigation, manipulation, and recognition. [7]

Gonzales and Woods stated that an image is a two-dimensional function where x and y are spatial coordinates and f in any parts of f(x,y) is named as gray level of the image. When x,y, and f have limited value, it is called as a digital image [8]. An image has pixel which has values from 0 to 255. These numbers tell about the brightness of the pictures depends on the code algorithm. [10] Binary Image only consist of 2 colors that is black and white because binary image is a digital image that have value of 1 and 0.[8] Grayscale image means that an image color is change to a single value which is gray. Every single value that it holds consist of intensity of the image itself. [10] Consumer Behavior is the behavior of consumers have while they are in a store. Consumer Behavior has many things starting from what are they doing while in a store, their preference, and many more that effect consumers decision [9].

2. RELATED WORK

One of the related work that we found used background subtraction and ACE (Automatic Color Equalization). They also use HSV (Hue-Saturation-Value) to remove shadow. They use ACE algorithm because it helped them to maximize the image dynamic. ACE able to adapt to lighting condition [10]. Another work that we found was using MESA-distance algorithm to count object from image. The MESA distance have 2 properties: Robustness and computability. The MESA distance can help user to detect object even though there is no define of ground truth density [11]. Another paper that we found is using ROI for counting people. They also use Farnebäck algorithm because it already has robust and efficient implementation in it. Robust can help user in image segmentation and restoration of the outliers [6]. Next paper that we found is counting object in machine vision using Otsu threshold and Hough circle transform. They also use Gaussian filter for smoothing. They also use Sobel edge detection after Otsu thresholding so that they can detect the edge of the object and use Hough circle transform to draw circle on the object. This method only works if the camera records the object from above [7]. The next project that we found was to count people in hajj using

3. PROPOSED METHODOLOGY

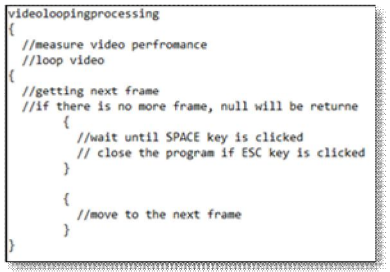
The proposed method to create the program are consist of 7 main modules: (1) Input video, (2) Background subtraction, (3) Looping video frame, (4) Remove noise, (5) Detect contour, (6) Draw line to show the object, (7) Display all window frame

The very first process of the program. In this process, the video will be input manually with code using *VideoCapture*. With the help of *VideoCapture*, we can obtain video frame easily. *VideoCapture* has four functions. The function that we are going to use is a string parameter that is called to load an image. *VideoCapture* also have many ways to obtaining frame from a video such as *QueryFrame* and if this method is use, it will capture a Bgr image frame. This method will return *Mat* class instance. Mat is a class that has 2 data parts: the header of matrix (it contains the size of the matrix, the method for storing and so on) and pointer of matrix that contains of pixel values.[12]

The product detection will be based on finding the difference between background frame and next another frame. Our assumption will be that we can take the first frame from the video as background and compare it with the frame that currently playing. We are going to use *QueryFrame* to get the background. After that we can show the backgro
und with *ImShow* method and show it on window screen.

The aim of this process is so that the video can be loop and play. We will use *QueryFrame* that can get single frame from the video and progress to the next frame and if *QueryFrame* returns *null*, it can't progress anymore and we will use this method to create a loop.

The pseudocode for looping video can be describe as picture bellow:

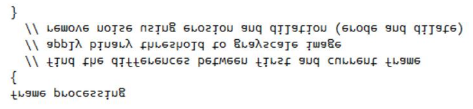


```
videoloopingprocessing
{
  //measure video perfromance
  //loop video
{
  //getting next frame
  //if there is no more frame, null will be returne
        {
          //wait until SPACE key is clicked
          // close the program if ESC key is clicked
        }

        {
          //move to the next frame
        }
}
```

Figure 1: Video looping process

A frame is grabbed from video in each loop iteration. Then it is passed to ProcessFrame to be differentiate with different image like noise removal, drawing, and contour detection.



```
frame processing
{
  \\ find the differences between first and current frame
  \\ apply binary threshold to grayscale image
  \\ remove noise using erosion and dilation (erode and dilate)
}
```

Figure 2: Frame processing pseudocode

The BackgroundFrame will be taken from the first frame in RawFrame. In other words, the BackgroundFrame is subtracted from RawFrame pixel by pixel. Then, the first frame (BackgroundFrame) is compared with the RawFrame (current frame that playing in the video) with *CvInvoke.AbsDiff*. And then the difference image will be converted into GrayscaleFrame with *CvInvoke.CvtColor* code call. We are only using the overall pixel difference and not it's individual blue-green-red color.

GrayscaleFrame is where image only produce white and black (binary) pixels with *CvInvoke.Threshold* call. We will use the change of mark as white. Threshold value allows us to control sensitivity of detection. Below is the image of Threshold effect.

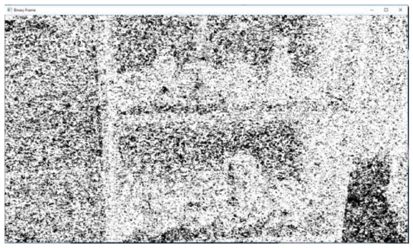


Figure 3: Threshold 1

Figure 4: Threshold 10

Figure 5: Threshold 30

3.1 Erode and Dilate

It is hard to find the perfect threshold for different detection for each different video frame. If the threshold is too low, the pixels will be too many white and if the threshold is too high, the pixels will be too dark. The best way to pick threshold that marks the changes that we need even if we get undesired white spot is to remove the undesired white spot with erosion and followed by dilation. These two can be combined to perform opening morphology transformation. The operation of opening morphology transformation is by probing pixel surrounding with kernel and decide pixel value which is based on the values that is found in the surrounding pixel.

CvInvoke.Erode is mean that a physical process is reduced because of its destructive effect in the surrounding. The idea is when white pixel has black pixel around it, it will be black too. If the more erode are run, then the more white pixel get eaten and become black pixel.

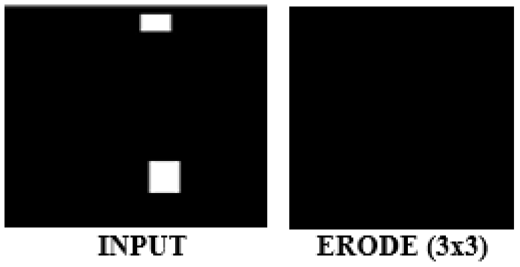


Figure 6: Input and Erode 3x3

If erosion is used to reduce black pixel, then dilation is to make white pixel bigger.

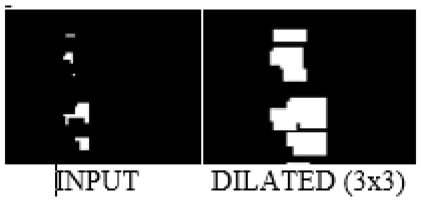


Figure 7: Input and Dilated 3x3

So, erosion will be used to reduce noise in binary frame and dilation will be used so that the white pixel for product is more easily spotted.

```
object detection
{
  using vectorofvectorof point
        {
          //create list of contours
          // select and pick the largest contour
        }
}
```

Figure 8: Contour detection pseudocode

The method to make contour detection is to use binary image from different frame (detection frame) which contour will be detected and another matrix (Mat) instance that detected object will be marked.

CvInvoke.FindContours will be take image and run contour algorithm detection to find line boundaries between zero (black) and one (white) pixels on 8-bit channel image. A contour is *VectorOfPoint*, because there might be many contour inside *VectorOfVectorOfPoint* and we want to pick the biggest contour that is detected. That's why we will use *CvInvoke.ContourArea* method.

```
mark object
{
   // getting rectangle that contain the contour
   // drawing contour and rectangle
}
```

Figure 9: Mark object pseudocode

After we found the position of the product by using contour, we will draw and write a line surrounding the contour. The method that we will use is *CvInvoke.BoundingRectangle*. This method will be used to find rectangle or box that surrounds the entire contour (the biggest) and box or rectangle will be drawn with *CvInvoke.Rectangle* call.

4. RESULT AND DISCUSSION

In this part, we are going to input a video file into the program. Video is being captured from the file. The video being called by *VideoCapture* class. We are going to get a background frame from the video. Background frame are taken from the very first frame of the video.

Figure 10: Dark background

The background frame is taken by using *QueryFrame* call after the *VideoCapture*. Every frame will be grabbed in each iteration of loop. *Stopwatch* is there so that we can measure video processing performance. *WriteFrameInformation* is there for us to put text on top-left corner about frame number and how long it took to process it. *WindowsWithImageProcessing* is there so that rawFrame, backgroundFrame, and other frame are in separate windows. *AbsDiff* is use to compare *RawFrame* with *backgroundFrame* (first frame). The differences will be converted into grayscale using *CvtColor* call. Grayscale image will be change into binary pixel image. Threshold allow us to change and control the detection sensivity. *CvInvoke.Erode* is there for erosion to delete white pixels that has black surrounding it. *CvInvoke.Dilate* is there for dilation to turn black that has white neighbor.

The method that we are using to detect contour is from binary image (*Framedetection*) that the contour will be detected and other Mat instance (*displayFrame*) that the detected object will be marked. *CvInvoke.FindContours* will take the image from frame and implement contour detection algorithm code to find border line between white and black pixels. The contour name is *VectorOfPoint* and because there will be many contours, so we keep it in *VectorOfVectorOfPoint*. We are using *CvInvoke.BoundingRectangle* to find small rectangle that has contour surrounding it. Box will be drawn using *CvInvoke.Rectangle* call. The contour will be spotted using *CvInvoke.Polylines* call.

We are recorded our samples in dark condition, normal room lighting, and bright background. We will see how is the brightness of the background will affect the accuracy of the program.

Figure 11: Dark and bright background

Figure 12: Bright background

This section will explain and elaborate the result of the program that we implemented. Dark background in the program will looks like this. In dark background, it shown that the accuracy is not that high. We used 6 videos to test the program. There are 2 videos that contain 3 objects that were moved from shelf and 4 videos that contain 6 objects.

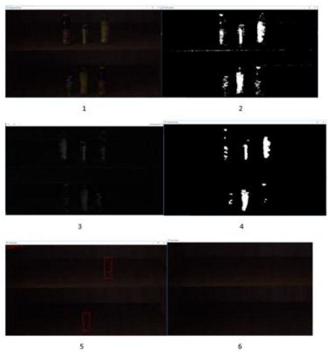


Figure 13: Dark background all 6 windows

From Figure 13, it shows the windows of the program. Image 1 is background frame, Image 2 is binary frame, image 3 is grayscale frame, image 4 is denoised frame, image 5 is final frame, and image 6 is raw frame.

Table 1: Object counter in every video in dark background

	Video 1	Video 2	Video 3	Video 4	Video 5	Video 6
Object counter 1	0	0	0	0	0	0
Object counter 2	1	0	0	0	0	1
Object counter 3	1	0	1	0	0	1
Object counter 4	-	-	2	0	0	1
Object counter 5	-	-	3	1	0	1
Object counter 6	-	-	5	1	0	2

From Table 1, we summarize the accuracy of the program. There are 6 videos that were tested and there are 3 to 6 objects that were tried to be counted.

Room light samples of the program can be seen in Figure 14, it shows every windows that shows in the program. From image 1 it shows background frame, image 2 it shows binary frame, image 3 it shows denoised frame, image 4 it shows final frame, image 5 it shows grayscale frame, and image 6 it shows raw frame.

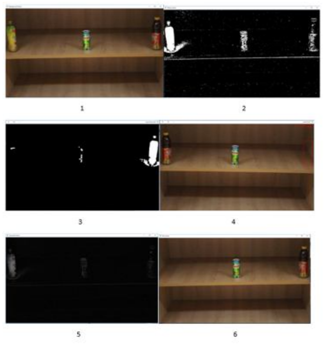


Figure 14: Dark bright background windows program

In this dark bright background, we recorded and use 10 videos to test the program. There is 1 video that only have 3 object in shelf, 3 videos that contains 6 object and 6 videos that contains 8 object.

Table 2: Counter every video in dark bright background

	Video 1	Video 2	Video 3	Video 4	Video 5	Video 6	Video 7	Video 8	Video 9	Video 10
Counter	1	0	12	0	1	1	2	1	11	1
Counter	2	1	15	1	2	3	3	2	10	2
Counter	4	2	16	2	3	5	4	3	13	3
Counter	6	14	10	3	4	-	5	4	5	4
Counter	6	17	14	4	5	-	5	4	4	3
Counter	6	20	16	4	4	-	5	4	3	5
Counter	6	21	10	4	5	-	-	-	-	3
Counter	7	20	12	5	6	-	-	-	-	4

The result of the program testing can be seen in table 2. In table 2, there are total of 10 videos that were tested. There were 3 to 8 objects that were tried to be counted.Whereas for the bright background. The result of the program will look like this

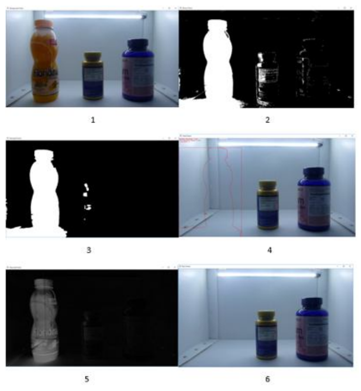


Figure 15: all windows for bright background

Based on Figure 15, image 1 shows background frame, image 2 shows binary frame, image 3 shows denoised frame, image 4 shows final frame, image 5 shows grayscale frame and image 6 shows raw frame. In this bright background, we recorded 8 videos. There is only 1 video that contain 2 objects and 7 videos had 3 objects.

Table 3: Counter every video in bright background

	Video 1	Video 2	Video 3	Video 4	Video 5	Video 6	Video 7	Video 8
Counter	1	2	2	1	1	1	1	1
Counter	2	5	5	2	4	3	4	4
Counter	1	1	1	-	1	1	1	1

The result of the program testing can be seen in table 3. There are total of 8 videos that were tested. There is 1 video that contains of 2 objects and there rest had 3 objects.

We as well tested it against room light with the same variable object

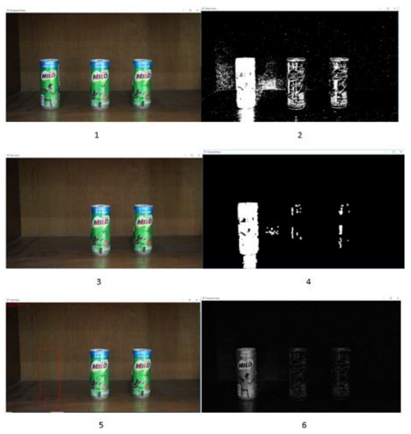


Figure 16: Middle background with the same variable object

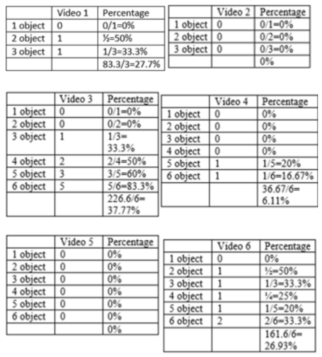
Based on Figure 16, number 1 is background frame, number 2 is binary frame, number 3 is current frame, number 4 is denoised frame, number 5 is final frame, and number 6 is grayscale frame.

Table 4: Counter in middle background with same variable

	Video 1	Video 2	Video 3	Video 4	Video 5
Counter	1	1	2	3	4
Counter	3	2	3	2	3
Counter	4	4	4	4	4

The result of the program testing can be seen in table 4. There are total of 5 videos that were tested. Every video only contains of 3 object and they have the same variable objects.

The evaluation of this test program will be made based on how accurate it is to count the object when the object was taken from a shelf.



	Video 1	Percentage
1 object	0	0/1=0%
2 object	1	½=50%
3 object	1	1/3=33.3%
		83.3/3=27.7%

	Video 2	Percentage
1 object	0	0/1=0%
2 object	0	0/2=0%
3 object	0	0/3=0%
		0%

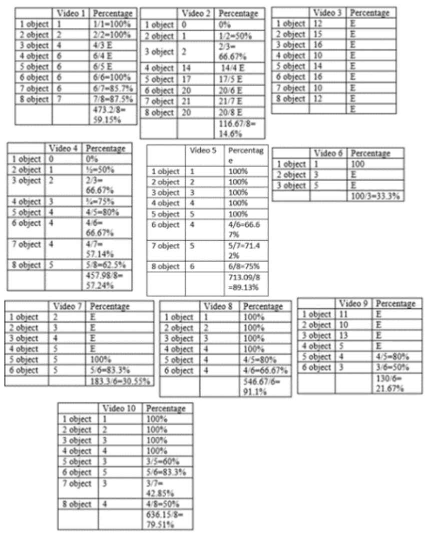
	Video 3	Percentage
1 object	0	0/1=0%
2 object	0	0/2=0%
3 object	1	1/3= 33.3%
4 object	2	2/4=50%
5 object	3	3/5=60%
6 object	5	5/6=83.3%
		226.6/6= 37.77%

	Video 4	Percentage
1 object	0	0%
2 object	0	0%
3 object	0	0%
4 object	0	0%
5 object	1	1/5=20%
6 object	1	1/6=16.67%
		36.67/6= 6.11%

	Video 5	Percentage
1 object	0	0%
2 object	0	0%
3 object	0	0%
4 object	0	0%
5 object	0	0%
6 object	0	0%
		0%

	Video 6	Percentage
1 object	0	0%
2 object	1	½=50%
3 object	1	1/3=33.3%
4 object	1	¼=25%
5 object	1	1/5=20%
6 object	2	2/6=33.3%
		161.6/6= 26.93%

Figure 17: Evaluation of dark background videos test percentage



	Video 1	Percentage
1 object	1	1/1=100%
2 object	2	2/2=100%
3 object	4	4/3 E
4 object	6	6/4 E
5 object	6	6/5 E
6 object	6	6/6=100%
7 object	6	6/7=85.7%
8 object	7	7/8=87.5%
		473.2/8= 59.15%

	Video 2	Percentage
1 object	0	0%
2 object	1	1/2=50%
3 object	2	2/3= 66.67%
4 object	14	14/4 E
5 object	17	17/5 E
6 object	20	20/6 E
7 object	21	21/7 E
8 object	20	20/8 E
		116.67/8= 14.6%

	Video 3	Percentage
1 object	12	E
2 object	15	E
3 object	16	E
4 object	10	E
5 object	14	E
6 object	16	E
7 object	10	E
8 object	12	E
		E

	Video 4	Percentage
1 object	0	0%
2 object	1	½=50%
3 object	2	2/3= 66.67%
4 object	3	¾=75%
5 object	4	4/5=80%
6 object	4	4/6= 66.67%
7 object	4	4/7= 57.14%
8 object	5	5/8=62.5%
		457.98/8= 57.24%

	Video 5	Percentag e
1 object	1	100%
2 object	2	100%
3 object	3	100%
4 object	4	100%
5 object	5	100%
6 object	4	4/6=66.6 7%
7 object	5	5/7=71.4 2%
8 object	6	6/8=75%
		713.09/8 =89.13%

	Video 6	Percentage
1 object	1	100
2 object	3	E
3 object	5	E
		100/3=33.3%

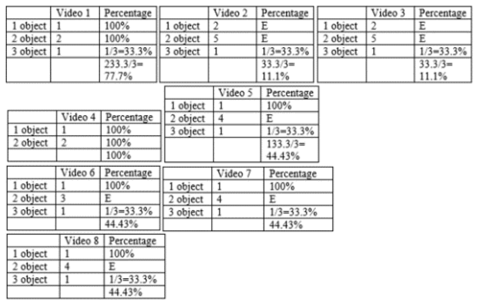
	Video 7	Percentage
1 object	2	E
2 object	3	E
3 object	4	E
4 object	5	E
5 object	5	100%
6 object	5	5/6=83.3%
		183.3/6=30.55%

	Video 8	Percentage
1 object	1	100%
2 object	2	100%
3 object	3	100%
4 object	4	100%
5 object	4	4/5=80%
6 object	4	4/6=66.67%
		546.67/6= 91.1%

	Video 9	Percentage
1 object	11	E
2 object	10	E
3 object	13	E
4 object	5	E
5 object	4	4/5=80%
6 object	3	3/6=50%
		130/6= 21.67%

	Video 10	Percentage
1 object	1	100%
2 object	2	100%
3 object	3	100%
4 object	4	100%
5 object	3	3/5=60%
6 object	5	5/6=83.3%
7 object	3	3/7= 42.85%
8 object	4	4/8=50%
		636.15/8= 79.51%

Figure 18: Evaluation of dark bright background test percentage



	Video 1	Percentage
1 object	1	100%
2 object	2	100%
3 object	1	1/3=33.3%
		233.3/3= 77.7%

	Video 2	Percentage
1 object	2	E
2 object	5	E
3 object	1	1/3=33.3%
		33.3/3= 11.1%

	Video 3	Percentage
1 object	2	E
2 object	5	E
3 object	1	1/3=33.3%
		33.3/3= 11.1%

	Video 4	Percentage
1 object	1	100%
2 object	2	100%
		100%

	Video 5	Percentage
1 object	1	100%
2 object	4	E
3 object	1	1/3=33.3%
		133.3/3= 44.43%

	Video 6	Percentage
1 object	1	100%
2 object	3	E
3 object	1	1/3=33.3%
		44.43%

	Video 7	Percentage
1 object	1	100%
2 object	4	E
3 object	1	1/3=33.3%
		44.43%

	Video 8	Percentage
1 object	1	100%
2 object	4	E
3 object	1	1/3=33.3%
		44.43%

Figure 19: Evaluation of bright background test percentage

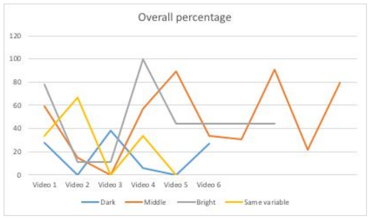


Figure 20: Overall percentage of the program

The overall percentage of the test in every video can be seen in Figure 20. The program has different accuracy for every background and not all of the video has the same accuracy percentage. The average percentage for dark background is 21.985%, the average for dark bright background is around 48.145%, and the average percentage for bright background is around 47.2025%. The overall accuracy percentage for this program is 39.11%. It has a pretty low accuracy. The middle background with the same background was made to test the program accuracy and recognition of object's shapes and sizes. The program accuracy overall for this case is 26.64%. The program has several problem that was also found in other test. There are some problem that occur in the program test that cause the accuracy of the program decrease and some cases that the program cannot do. The first problem that occur when the program was being tested was that the program cannot recognize if the object was being put back in. When the object was placed back to the shelf, the program still count the object that missing from the shelf.

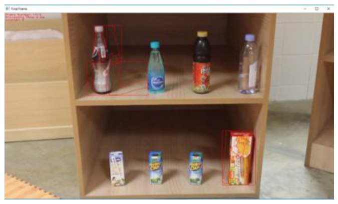


Figure 21: The object cannot be place back

As you can see from Figure 21, the program still count the object that was moved from shelf even though the object has already being place back in the shelf. It doesn't matter if the object being place back differently from their original position, for example if you take mineral water on the right side and tea on the left side and swap their position, the program still count the object that being moved.

It can be fixed by putting the object back precisely at the same position

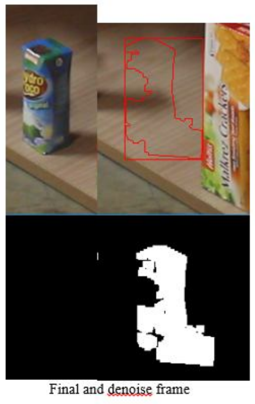


Figure 22: Final frame and denoise frame after and before the object taken

Based on Figure 22, when the object being placed back at the same location, denoise frame did not detect anything because it was place at the same place.

Other problem that occur while we tested the program was shadow. Shadow can makes an object counted as one. When that happens, the counter will count it wrongly.

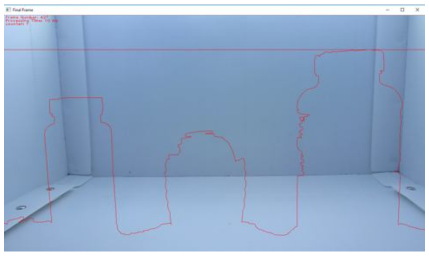


Figure 23: Shadow makes 3 objects counted as 1

Based on Figure 23, 3 objects were counted as one because of shadow merge them together. This cause the accuracy of the program to decrease. Other problem that occur because of shadow was it can separate from the body of an object (see Figure 24).

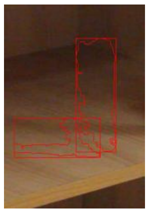


Figure 24: 1 object counted as 2

Based on Figure 24, 2 objects were recorded because of shadow effect when it should be counted as one object. This also cause the program accuracy to decrease.

Other problem that also occur was that we cannot use videos that has object attached to each other especially from behind.

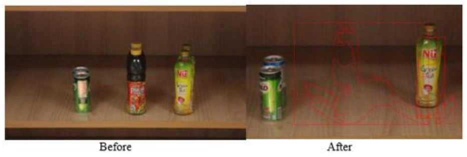


Figure 25: The problem of attached object

From Figure 25, before and after the objects is taken, a problem occur. The problem is that the objects that taken from the same column will be counted as one. When that happens, the object counter cannot count the object accurately. We excluded this type video in our program test. There are so many videos that has problem with noise.

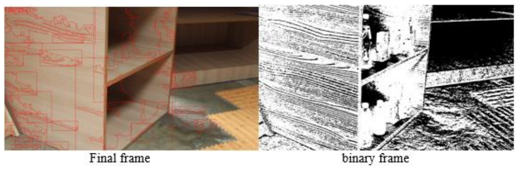


Figure 26: Noise in final frame and binary frame

Based on Figure 26, you can see that noise cause the counter not able to count the object correctly. This research had 2 videos that had error because of noise problem but the rest not had this kind of problem. Another problem that occurred when we tested the program was that, noise can cause 1 object counted as 2 object. Look at a figure below

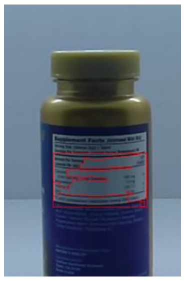


Figure 27: Noise problem cause counted unnecessary part

Based on Figure 27, noise can cause a counter counted unnecessary part of an object even though the object itself is not taken from a shelf. This problem occurred so many times when we tested our program. It causes the accuracy of the program to decrease

5. CONCLUSION

Based on the test result of the program, the conclusion could be drawn are: the average percentage for dark background is 21.985%, the average for dark bright background is around 48.145%, and the average percentage for bright background is around 47.2025%. The program could not work properly at dark environment. The dark bright background is the best way to implement the program. This program overall accuracy is still low at 39.11% There are problems in this program for such as shadow problem, noise problem, object attachment problem, cannot put back in problem

The recommendation that could be given to researchers that wanted to continue this project further in their work are: needs to implement a better noise removal method so that the program can count moving objects without counting unnecessary objects. Implement shadow removal so that 2 objects would not merge into 1 object and 1 object would not be counted as 2 object. Needs to develop other approach that have a better accuracy to count object if this program is cannot be completed.

REFERENCES

1. C. Llorente, E.P. Dadios, **Development and Characterization of a Computer Vision System for Human Body Detection and Tracking under Low-light Condition,** International Journal of Advanced Trends in Computer Science and Engineering, 8(2), pp. 251-254, 2019, https://doi.org/10.30534/ijatcse/2019/24822019
2. **What is The Working of Image Recognition and How It Used?**, [Online]. Available: https://www.marutitech.com/working-image-recognition/. [Accessed 22 March 2018].

3. K. Williams, A. Petrosky, E. Hernandez and R. Page, Jr. **Product placement effectiveness: revisited and renewed**, Journal of Management and Marketing Research, pp. 2-3.
4. L. Shapiro and G. Stockman, **Computer Vision**, 2000.
5. **How to Use Background Subtraction Methods**, [Online]. Available: https://docs.opencv.org/trunk/d1/dc5/tutorial_background_subtraction.html. [Accessed 9 May 2018].
6. **More Morphology Transformation**, [Online]. Available: https://docs.opencv.org/3.0-beta/doc/tutorials/imgproc/opening_closing_hats/opening_closing_hats.html. [Accessed 10 May 2018].
7. S. J. Russel and P. Norvig, in Artificial Intelligence: A Modern Approach, 2010, p. 965.
8. R. C. Gonzalez and R. E. Woods, Digital Image Processing, vol. III, 2008.
9. J. Bray, **Consumer Behaviour Theory: Approaches and Models**.
10. R.-E. Berg, **Real-time people counting system using video came**, 2007.
11. V. Lempitsky and A. Zisserman, **Learning to count objects in images**.
12. J. Baltes, A. Hosseinmemar, J. Jung, S. Sadeghnejad and J. Anderson, **Practical Real-Time System for Object Counting based on Optical Flow**, Canada.
13. M. Baygin, M. Karakose, A. Sarimaden and E. Akin, **An Image Processing based Object Counting Approach for Machine Vision Application**.
14. S. D. Khan, **Automatic Detection and Computer Vision Analysis of Flow Dynamics and Social Groups in Pedestrian Crowds**.
15. R. H. Evangelio, **Background Subtraction for the Detection of Moving and Static Objects in Video Surveillance**, Berlin, 2014.
16. C. Shanthi and S. , **Background Subtraction Techniques: Systematic Evaluation and Comperative Analysis**, 2013.
17. K. Avgerinakis, **Video Processing and Background Subtraction for Change Detection and Activity Recognition**, 2015.
18. **Mat - the basic image container**, [Online]. Available: https://docs.opencv.org/2.4/doc/tutorials/core/mat_the_basic_image_container/mat_the_basic_image_container.html. [Accessed 20 May 2018].