**SIMATS SCHOOL OF ENGINEERING**

**SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES**

**OBJECT SIZE MEASUREMENT**

**A CAPSTONE PROJECT REPORT**

**Submitted by**

**M.NITHYA (192221073)**

**M.AKSHAYA (192221029)**

**Under the Supervision of**

**Dr. P Manjula sai**

**ABSTRACT**

Object size measurement using OpenCV is a fundamental technique in computer vision, enabling precise dimensioning of objects within images or video streams. The process begins with capturing the image, followed by preprocessing steps such as grayscale conversion and noise reduction to enhance image quality. Object detection is then performed using methods like thresholding and contour detection to identify objects of interest. Measurement involves calculating the dimensions of these detected objects, often using bounding boxes or enclosing shapes, and converting pixel values to real-world units based on a known reference object. This approach is widely used in fields such as quality control, automation, and surveillance, offering a robust solution for accurate and efficient object dimensioning.

**CHAPTER 1**

**INTRODUCTION**

* 1. **Introduction**

Object size measurement using OpenCV is a critical process in computer vision allowing for the accurate determination of object dimensions within digital images or video streams. This method is employed in various fields, including manufacturing for quality control, automated sorting systems, and surveillance for monitoring and security purposes.

* 1. **Statement of the problem**

In various industries and applications, accurate measurement of object dimensions is crucial for tasks such as quality control, automated sorting, and surveillance. Traditional manual measurement methods are often time-consuming, error-prone, and impractical for high-speed or large-scale operations.

Therefore, there is a need for an automated, reliable, and efficient system to measure object dimensions from digital images or video streams. The goal is to develop a computer vision solution using OpenCV that can automatically capture, process, and measure the size of objects.

* 1. **Need for the study**

The need for studying and developing automated object size measurement using OpenCV arises from several pressing demands in various industries. In manufacturing and quality control, precise measurements are essential to ensure products meet specific standards and specifications. Traditional manual measurement methods are often inadequate, being time-consuming, prone to human error, and impractical for high-speed production lines. Automated systems can provide consistent and accurate measurements, significantly improving efficiency and reducing the risk of defects.

In the field of logistics and inventory management, automated size measurement can streamline operations by enabling rapid and accurate tracking of goods. This can lead to better space utilization, optimized storage solutions, and more efficient shipping processes.

* 1. **Scope of the study**

The scope of the study on automated object size measurement using OpenCV spans several critical areas to ensure comprehensive development and practical application. Technically, it involves developing robust methods for image acquisition, preprocessing, object detection, and segmentation to enhance image clarity and accuracy. The study also focuses on establishing reliable measurement techniques and ensuring real-time processing capabilities, making the system suitable for high-speed industrial applications. In terms of applications, the technology will be tailored for various industries, including manufacturing for quality control, logistics for inventory management, and surveillance for security enhancement. Additionally, integration with advanced technologies like machine learning, IoT, and cloud computing is a key area, aiming to improve detection accuracy, real-time data collection, and scalable processing

1. **Future decisions**

Future developments in object size measurement using OpenCV can focus on enhancing accuracy, scalability, and usability to meet diverse industrial needs. Advanced calibration techniques and the use of 3D measurement technologies can significantly improve precision. Integrating machine learning algorithms will enable more robust and accurate object detection and classification under varying conditions. Real-time processing optimizations are essential for high-speed applications, while developing user-friendly interfaces, including GUIs and mobile solutions, will make the system more accessible to non-technical users. Incorporating IoT and cloud computing can facilitate seamless integration with industrial systems and remote monitoring, enhancing automation and data management. Additionally, extending the application to include multi-object measurement, quality control, and inventory management will broaden its usability, making it a comprehensive tool for various industries.

**CHAPTER 2**

**LITERATURE REVIEW**

* 1. TITLE: **An Embedded Real-Time Object Detection and Measurement of its size**

AUTHOR: **Nashwan Adnan Othman**

YEAR: **2018**

This study presents an enhanced technique for detecting objects and computing their measurements in real time from video streams. We suggested an object measurement technique for real-time video by utilizing OpenCV libraries and includes the canny edge detection, dilation, and erosion algorithms. The suggested technique comprises of four stages: (1) identifying an object to be measured by using canny edge detection algorithm, (2) using morphological operators includes dilation and erosion algorithm to close gaps between edges, (3) find and sort contours, (4) measuring the dimensions of objects.

* 1. TITLE: **Computer Vision-based Robotic Arm for Object Color, Shape, and Size Detection**

AUTHOR: **Md. Abdullah-Al-Noman**

YEAR**: 2022**

An automatic computer vision-based robotic gripper has been built that can select and arrange objects to complete various tasks. This study utilizes the image processing methodology of the PixyCMU camera sensor to distinguish multiple objects according to their distinct colors (red, yellow, and green). Next, a preprogrammed command is generated in the robotic arm to pick the item employing Arduino Mega and four MG996R servo motors. Finally, the device releases the object according to its color behind the fixed positions of the robotic arm to a specific place. The proposed system can also detect objects' geometrical shapes (circle, triangle, square, rectangle, pentagon, and star) and sizes (large, medium, and small) by utilizing OpenCV image processing libraries in Python language.

**CHAPTER 3**

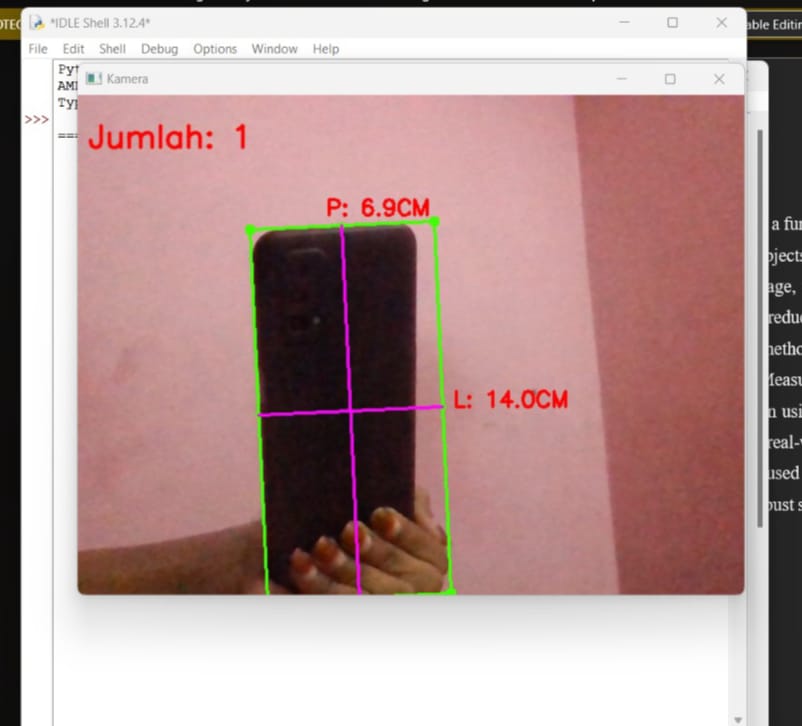
**EXISTING SYSTEM**

The existing systems for object size measurement encompass a mix of manual and automated approaches, each with its strengths and limitations. Manual measurement techniques, such as using callipers, rulers, or micrometres, are still widely used, particularly in smaller operations where precision is essential. However, these methods are time-consuming, and prone to human error, which makes them less suitable for high-speed production environments. On the other hand, automated measurement systems, including laser scanners, offer high accuracy but can be expensive and complex, often requiring specialized training and significant setup time. Non-contact optical systems utilize cameras and image processing to measure objects without physical contact, employing techniques like laser triangulation and photogrammetry. While these systems can process a variety of objects quickly, they also demand considerable calibration and may struggle under challenging lighting conditions or with occluded objects. Additionally, various software solutions exist that apply image processing techniques for automated measurement, but they often lack the real-time processing capabilities needed for fast-paced industrial applications. Overall, while existing systems have advanced significantly, there remains a critical need for more efficient, user-friendly solutions that can streamline object size measurement across different industries.

**PROPOSED SYSTEM**

The proposed system for automated object size measurement using OpenCV aims to address the limitations of existing methods by providing a versatile, efficient, and cost-effective solution. This system will utilize computer vision techniques to capture and preprocess images or video streams, enhancing image quality and isolating objects of interest. Calibration techniques will be used to convert pixel measurements to real-world units, ensuring precision. The system will be optimized for real-time processing, significantly reducing the time required for size determination, making it practical for high-speed industrial environments. Integration with machine learning will enhance robustness, allowing the system to adapt to varying conditions. Additionally, the proposed system will include user-friendly interfaces and mobile applications to make it accessible to non-technical users. By leveraging IoT and cloud computing, the system will enable seamless integration with other industrial systems, facilitating automated data collection and analysis. This comprehensive approach aims to deliver a reliable, scalable, and adaptable solution for diverse industrial applications, significantly improving efficiency and accuracy in object size measurement while reducing the time and labour involved in manual measurement processes.

**CHAPTER 4**

****

**CHAPTER 5**

**CONCLUSION**

In conclusion, the development of an object size measurement system using OpenCV represents a significant advancement in the field of computer vision, addressing the limitations of both manual and existing automated measurement methods. By leveraging advanced image processing techniques, this proposed system aims to enhance accuracy, reduce measurement time, and streamline operations across various industries. The integration of real-time processing capabilities, machine learning, and user-friendly interfaces will not only improve the efficiency of size determination but also make the technology accessible to non-technical users. Moreover, the ability to seamlessly integrate with IoT devices and cloud computing will facilitate better data collection and analysis, driving informed decision-making and operational improvements. As industries continue to embrace automation and seek innovative solutions for quality control and inventory management, this automated measurement system has the potential to significantly enhance productivity and accuracy, ultimately contributing to the advancement of modern manufacturing and logistics processes.

**CHAPTER 6**

**REFERENCE**

[1] Muthu krishnan.R and M.Radha "Edge Detection Techniques for picture Segmentation" International Journal of Computer Science and Information Technology (IJCSIT) Vol3, No 6, Dec 2011.

[2] Geng Xing, Chen ken, Hu Xiaoguang "A superior Canny edge identification calculation for variety picture" IEEE TRANSATION, 2012 978-1-4673-0311-8/12/$31.00 ©2012 IEEE.

[3] Chen.W, Yue.H, Wang.J, Wu.X. A better edge discovery calculation for profundity map in painting. Optics and Lasers in Engineering 2014;55: 69-77.

[4] Moeslund T. Shrewd Edge identification. Denmark: Laboratory of Computer Vision and Media Technology, Aalborg University.

[5] OpenCV, 2016. http://opencv.org/. [Accessed December 23, 2016].

[6] Mengel, P., and Wertheimer, R. (2007). D52.300.1 camera necessities and particulars.

[ 7] Nedev, S., and Ivanova, V.C. (2006). Web camera as an estimating device in the undergrad material science research center. European Journal of Physics 27, 1213.

**CHAPTER 7**

**CODE**

from scipy.spatial import distance as dist

from imutils import perspective

from imutils import contours

import numpy as np

import imutils

import cv2

def midpoint(ptA, ptB):

return ((ptA[0] + ptB[0]) \* 0.5, (ptA[1] + ptB[1]) \* 0.5)

cap = cv2.VideoCapture(0)

while (cap.read()):

ref,frame = cap.read()

frame = cv2.resize(frame, None, fx=1, fy=1, interpolation=cv2.INTER\_AREA)

orig = frame[:1080,0:1920]

gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

blur = cv2.GaussianBlur(gray, (15, 15), 0)

thresh = cv2.adaptiveThreshold(blur,255,cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C,cv2.THRESH\_BINARY\_INV,11,2)

kernel = np.ones((3,3),np.uint8)

closing = cv2.morphologyEx(thresh,cv2.MORPH\_CLOSE,kernel,iterations=3)

result\_img = closing.copy()

contours,hierachy = cv2.findContours(result\_img,cv2.RETR\_EXTERNAL,cv2.CHAIN\_APPROX\_SIMPLE)

hitung\_objek = 0

pixelsPerMetric = None

for cnt in contours:

area = cv2.contourArea(cnt)

if area < 1000 or area > 120000:

continue

orig = frame.copy()

box = cv2.minAreaRect(cnt)

box = cv2.cv.BoxPoints(box) if imutils.is\_cv2() else cv2.boxPoints(box)

box = np.array(box, dtype="int")

box = perspective.order\_points(box)

cv2.drawContours(orig, [box.astype("int")], -1, (0, 255, 64), 2)

for (x, y) in box:

cv2.circle(orig, (int(x), int(y)), 5, (0, 255, 64), -1)

(tl, tr, br, bl) = box

(tltrX, tltrY) = midpoint(tl, tr)

(blbrX, blbrY) = midpoint(bl, br)

(tlblX, tlblY) = midpoint(tl, bl)

(trbrX, trbrY) = midpoint(tr, br)

cv2.circle(orig, (int(tltrX), int(tltrY)), 0, (0, 255, 64), 5)

cv2.circle(orig, (int(blbrX), int(blbrY)), 0, (0, 255, 64), 5)

cv2.circle(orig, (int(tlblX), int(tlblY)), 0, (0, 255, 64), 5)

cv2.circle(orig, (int(trbrX), int(trbrY)), 0, (0, 255, 64), 5)

cv2.line(orig, (int(tltrX), int(tltrY)), (int(blbrX), int(blbrY)),

(255, 0, 255), 2)

cv2.line(orig, (int(tlblX), int(tlblY)), (int(trbrX), int(trbrY)),

(255, 0, 255), 2)

lebar\_pixel = dist.euclidean((tltrX, tltrY), (blbrX, blbrY))

panjang\_pixel = dist.euclidean((tlblX, tlblY), (trbrX, trbrY))

if pixelsPerMetric is None:

pixelsPerMetric = lebar\_pixel

pixelsPerMetric = panjang\_pixel

lebar = lebar\_pixel

panjang = panjang\_pixel

cv2.putText(orig, "L: {:.1f}CM".format(lebar\_pixel/25.5),(int(trbrX + 10), int(trbrY)), cv2.FONT\_HERSHEY\_SIMPLEX,0.7, (0,0,255), 2)

cv2.putText(orig, "P: {:.1f}CM".format(panjang\_pixel/25.5),(int(tltrX - 15), int(tltrY - 10)), cv2.FONT\_HERSHEY\_SIMPLEX,0.7, (0,0,255), 2)

hitung\_objek+=1

cv2.putText(orig, "Jumlah: {}".format(hitung\_objek),(10,50),cv2.FONT\_HERSHEY\_SIMPLEX, 1, (0,0,255),2, cv2.LINE\_AA)

cv2.imshow('Kamera',orig)

key = cv2.waitKey(1)

if key == 27:

break

cap.release()

cv2.destroyAllWindows()