# Steps for setting up Visual Studio and installing opencv:

Step (1) : Installing The Visual Studio 2022.

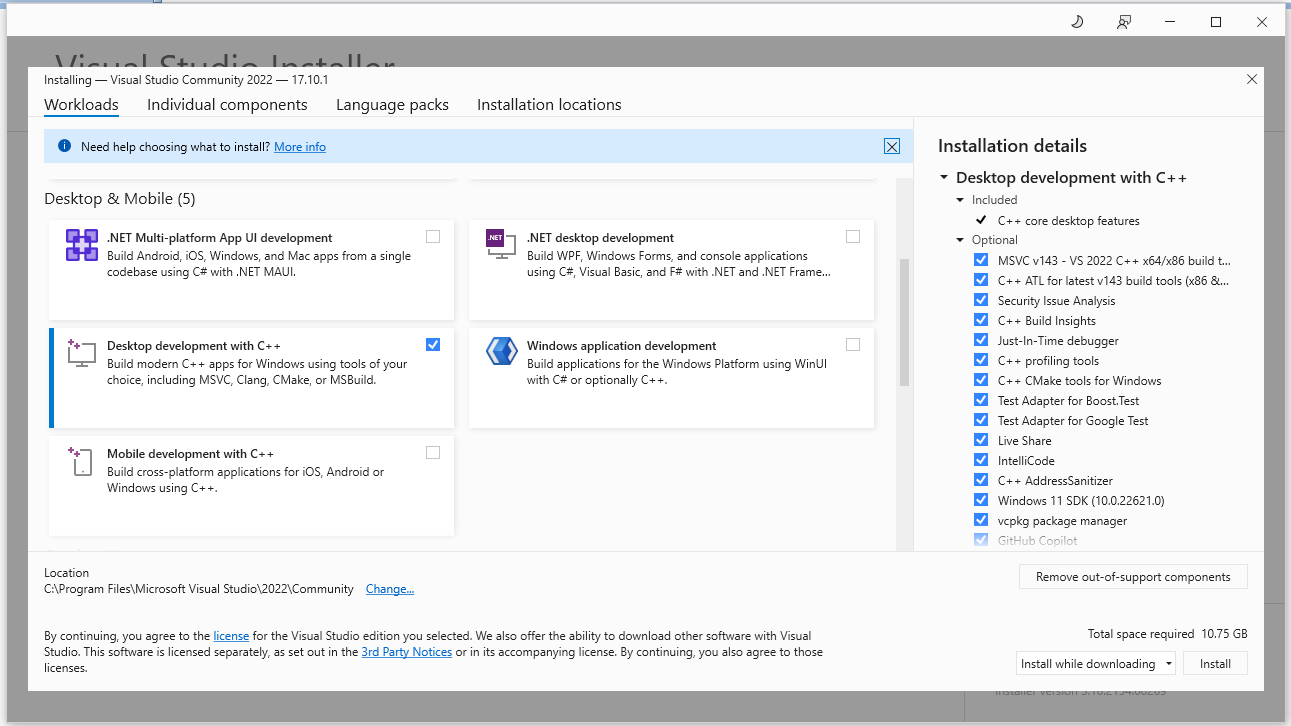
Link:

[Https://Visualstudio.Microsoft.Com/Thank-You-Downloading-Visual-Studio/?Sku=Community&Channel=Release&Version=Vs2022&Source=Vslandingpage&Cid=2030&Passive=False](https://visualstudio.microsoft.com/thank-you-downloading-visual-studio/?sku=Community&channel=Release&version=VS2022&source=VSLandingPage&cid=2030&passive=false)

Necessary: Development Kit For C

(1) Available When The Installer Is Launched.

(2) If Only The Bare Minimum Version Is Installed Which Is Approx Around 1.5 Gb – 2.55 Gb , Then We Have To Again Launch The Installer And Go For Modify Option , Then Further Add The Same Developmet Kit For C.



Important:

Checkout the options in the red box ,and check in only the ones included in the blue marks outside red box.

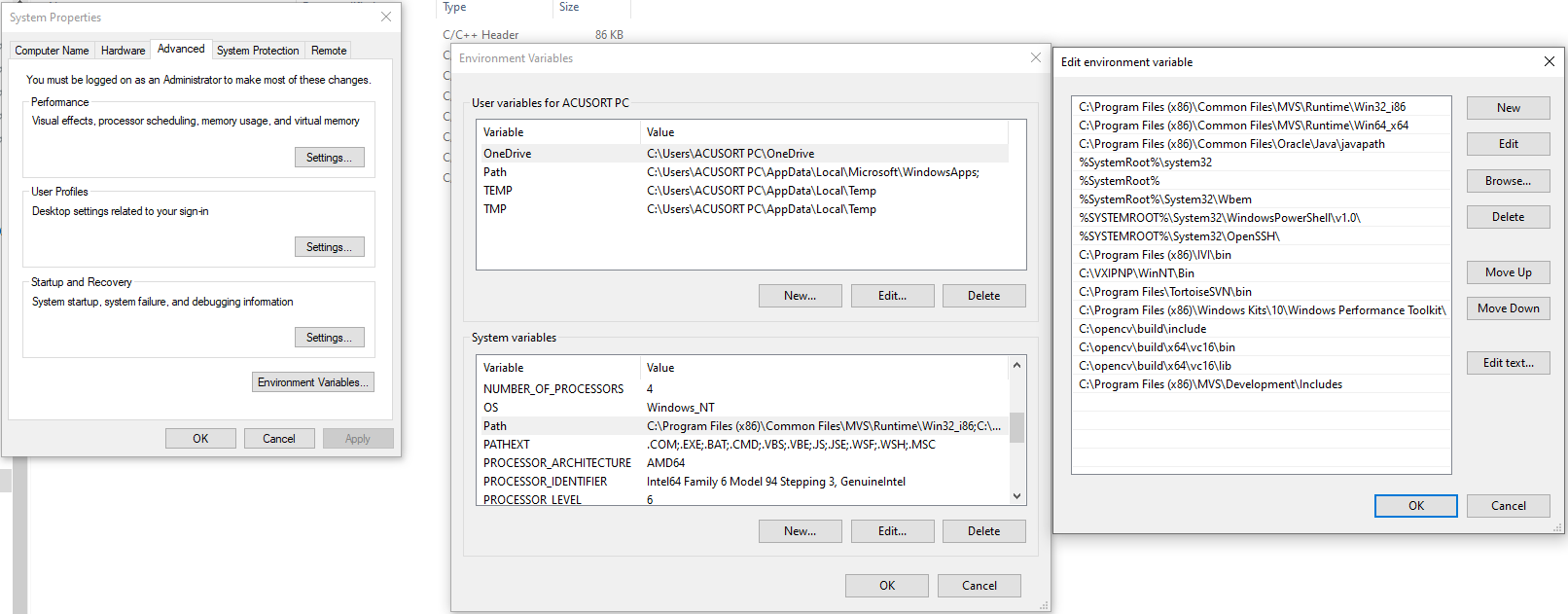
NEXT TIME WHILE INSTALLING ON NEW PC REMEMBER TO TAKE SCREENSHOOT.

While Adding The Development Kit Ensure To Checkout The Latest Release Options.

Step (2) : Installing Opencv

Link:

[Https://Github.Com/Opencv/Opencv/Releases/Download/4.9.0/Opencv-4.9.0-Windows.Exe](https://github.com/opencv/opencv/releases/download/4.9.0/opencv-4.9.0-windows.exe)

After Downloading The opencv .Exe File Execute It, And Then Add The Path Where The opencv Needs To Be Extracted to , mainly This Will Be The “C” Drive Of Your P

Step(2A):Include the following paths in your “PATH” variable in the ENVIRONMENT VARIABLES SETTINGS(in the SYSTEMS VARIABLE).

Step (3): Open An Empty Project In Visual Studio:

Step (4): Setting Up Properties For Opencv In Visual Studio:

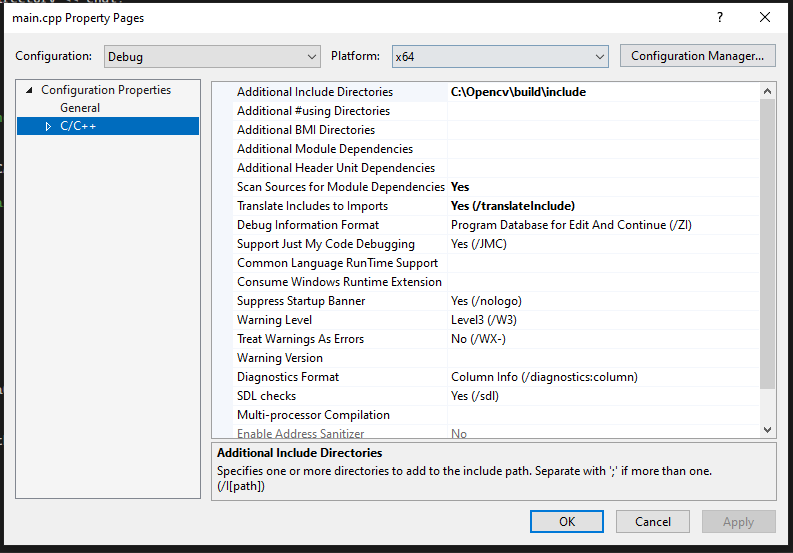
Click On “Source Files” Located In The Right Side Menu Bar, Then Right Click And Press On “Add” Option,

Add A Cpp File And Click Ok.

Write The Code .

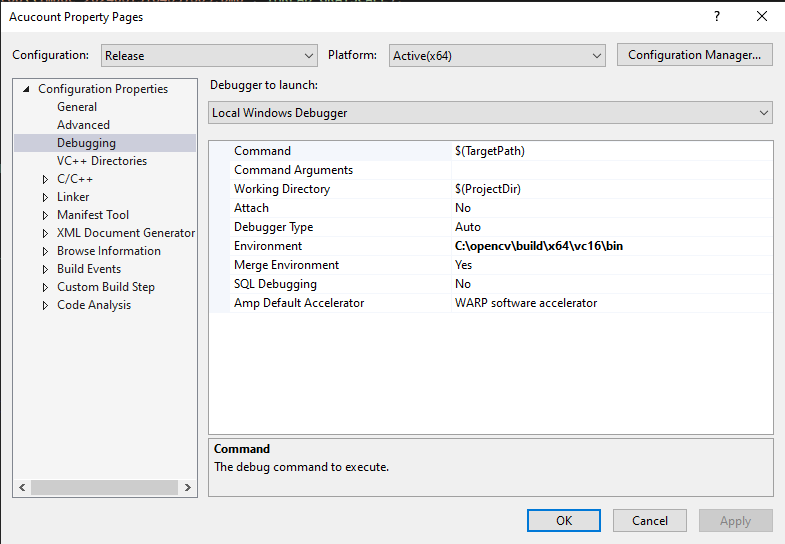
Then Right Click On The Code , Navigate To The Properties Options:

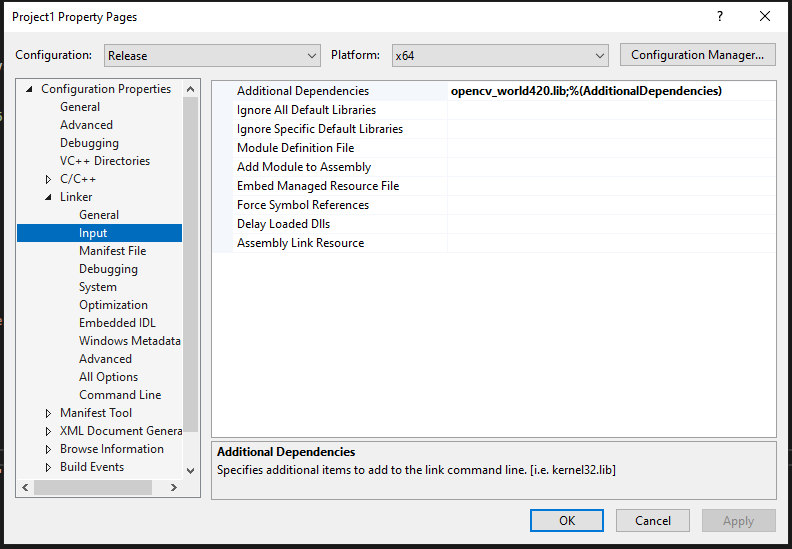
Then In C++ Settings , Add The Path To The Include Folder Of Opencv.

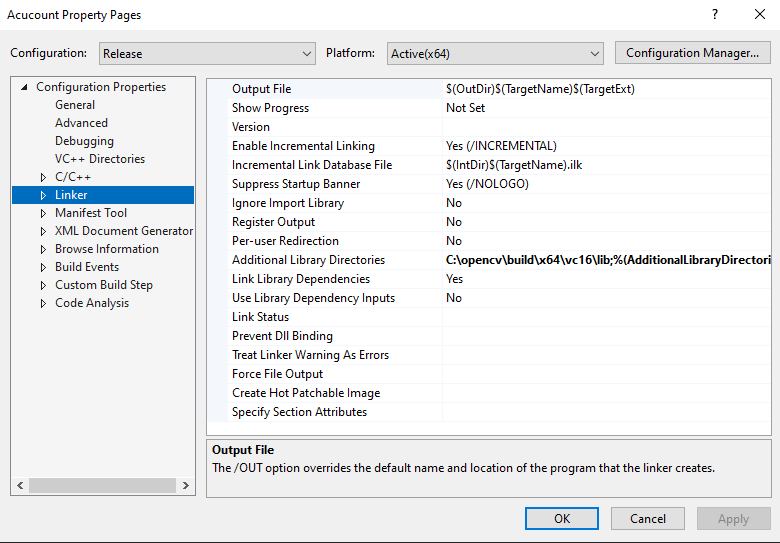


Then Right Click On “Project” In The Top Toolbar,Then Navigate To The Properties:

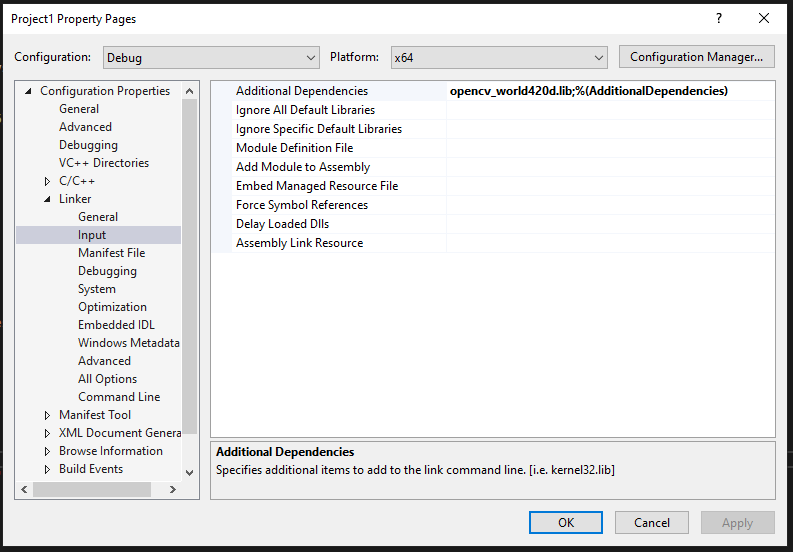
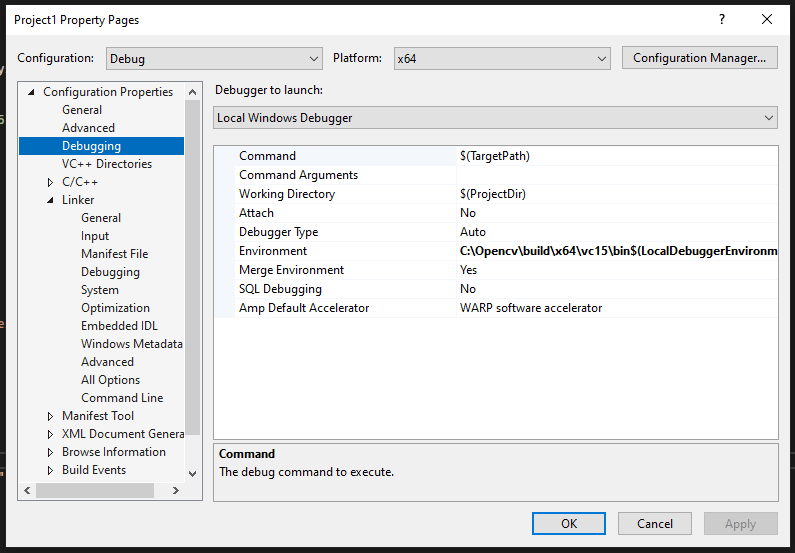
Release Mode:







Debug Mode:

# Setup for including MVS in the Visual Studio:

Step (4) : Setting Up Properties For MVS In Visual Studio:

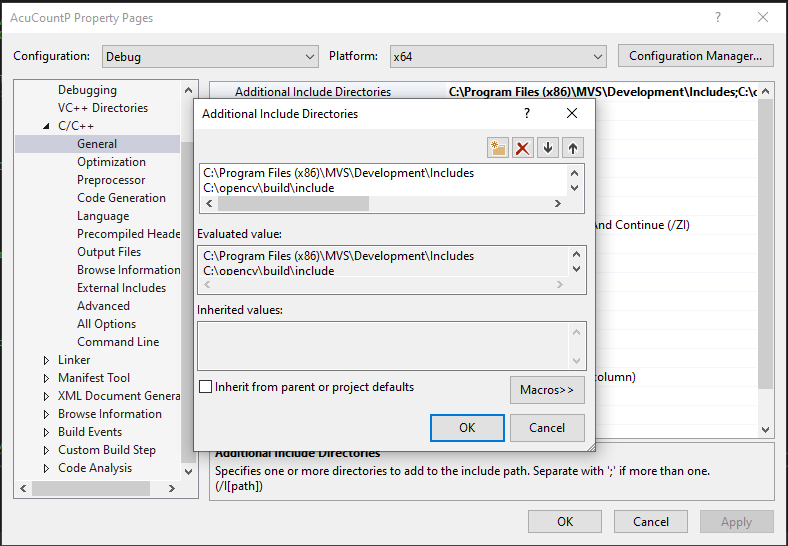
Click On “Source Files” Located In The Right Side Menu Bar, Then Right Click And Press On “Add” Option,

Add A Cpp File And Click Ok.

Write The Code .

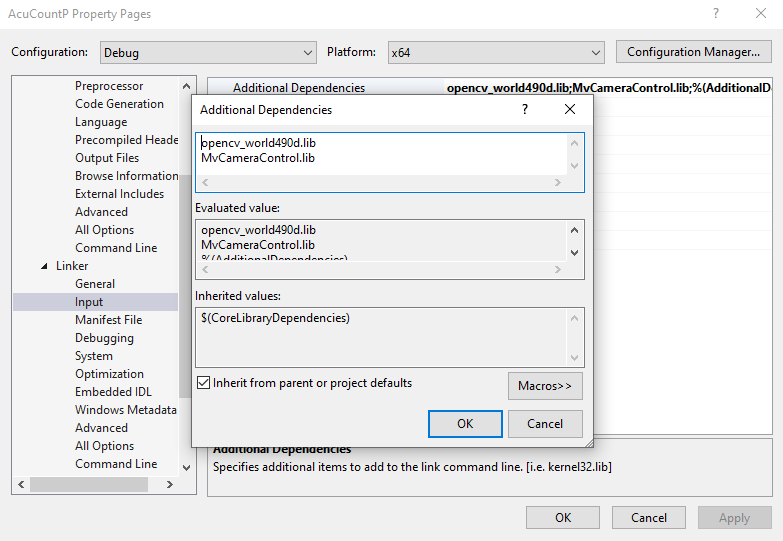
Then Right Click On The Code , Navigate To The Properties Options:

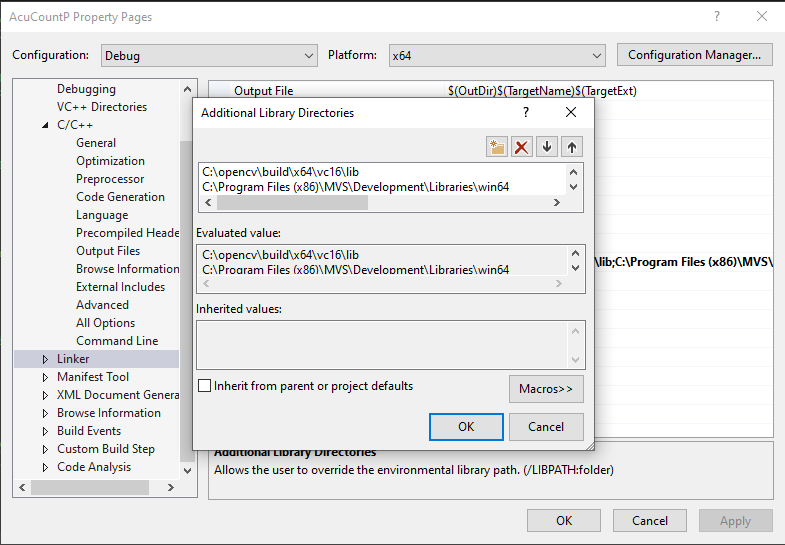
Then In C++ Settings , Add The Path To The Include Folder Of MVS.



Then Right Click On “Project” In The Top Toolbar,Then Navigate To The Properties:

Debug Mode:





VERY IMPORTANT:

**ONCE ALL THE ABOVE IS COMPLETE THEN SAVE THE PROGRAM AND RESTART THE VISUAL STUDIO.**

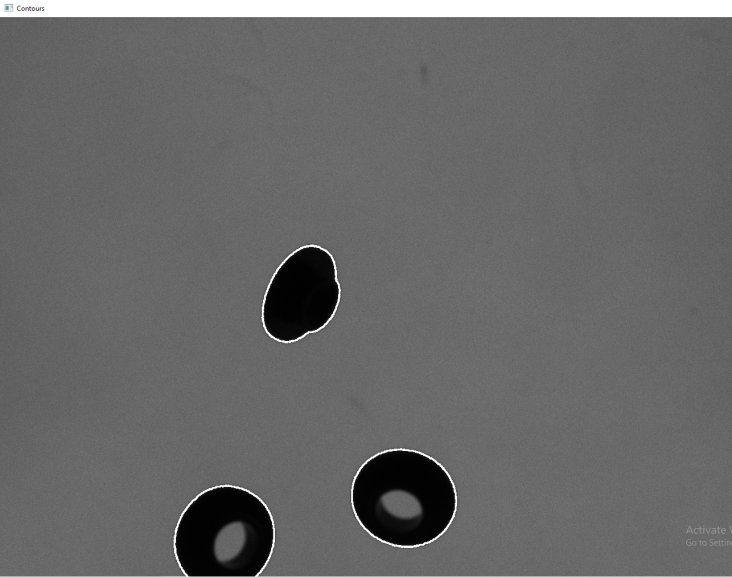
**THEN RUN AGAIN THE APPLICATION.**

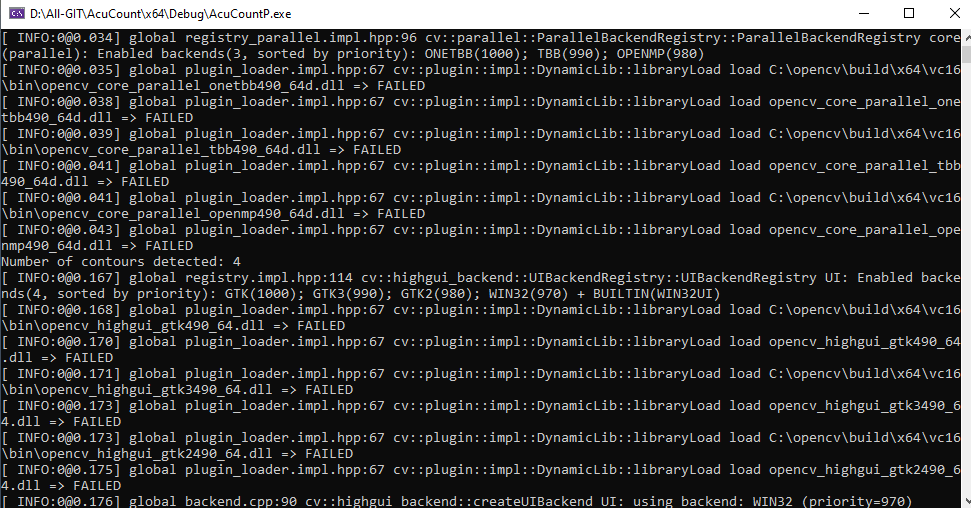
**You should get the following output from the given files below:**

**For OPENCV:**

**BasicCode.cpp**

**OUTPUT:**

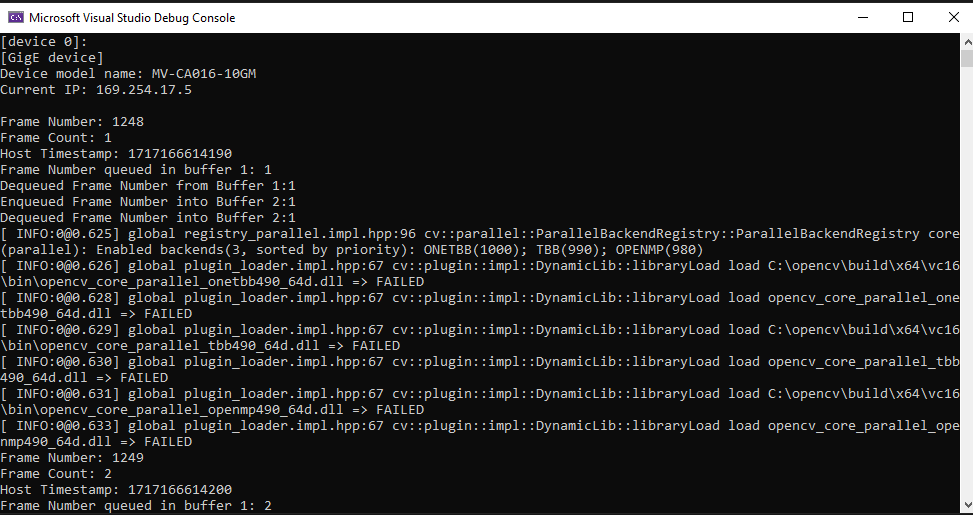
****

****

**The number of contours in the outptut will vary based on the parameters passed in the findcontours function.**

**For MVS:**

**AA20240523\_GOD.cpp**

****

# Process for converting x64 to Win32:

I built OpenCV myself, it turned out not to be difficult. I followed the guides [here](https://perso.uclouvain.be/allan.barrea/opencv/cmake_config.html) and [here](https://docs.opencv.org/master/d3/d52/tutorial_windows_install.html). I'll put my own detailed steps below, it might help someone else.

I am using Visual Studio Enterprise 2019. My choices are shown in parenthesis.

1. Download and install CMake from <https://cmake.org/download> (CMake 3.18.3)
2. Download the OpenCV source code from <https://opencv.org/releases> (OpenCv 4.4.0)
3. Unzip the source code (c:\opencv\src)
4. Run CMake
5. In the CMake UI, specify **Where is the source code** (c:\opencv\src) and **Where to build the binaries** (c:\opencv\build)
6. Check the **Grouped** and **Advanced** checkboxes
7. Click **Configure**. If CMake asks permission to create the build folder, allow it
8. Choose the **Specify the generator for this project** (Visual Studio 16 2019)
9. Choose the **Optional platform for generator** (Win32)
10. For **Optional toolset to use**, I left this blank
11. Choose **Use default native compilers**
12. Click **Finish**. CMake will process for a few minutes.

For me at this point, there is a big list of items in red and in the detail area at the bottom, everything is ok except for some Python errors in red. I don't care about the Python parts, so:

1. In the upper pane, expand the **BUILD** item
2. Uncheck **BUILD\_opencv\_python\_bindings\_generator**
3. Uncheck **BUILD\_opencv\_python\_tests**

Click **Configure** again... wait to see the 'Configuring done' message. All the red disappears. Click **Generate** and wait to see the 'Generating done' message. Look in your build folder... open the solution 'OpenCV.sln' Open it in Visual Studio, build Release or Debug.

If you want to make an x64 build, start over by doing **File->Delete Cache**, and select x64 at the **Optional platform for generator** step.

# findContour() details:

Function Signature:

void findContours(InputOutputArray image, OutputArrayOfArrays contours, OutputArray hierarchy, int mode, int method, Point offset=Point())

Parameters:

image: The input 8-bit single-channel image. It should be a binary image (black and white), where contours are represented by white pixels and the background by black pixels. Typically, you would use a thresholded or edge-detected image as input.

contours: A vector of vector of points. This is the output parameter that stores the contours found in the input image. Each contour is represented as a vector of points.

hierarchy: Optional output vector that contains information about the image topology. This parameter is usually not used for basic contour detection and can be set to noArray() if not needed.

mode: Specifies the contour retrieval mode. It determines how the contours are retrieved from the image. There are several retrieval modes:

RETR\_EXTERNAL: Retrieves only the external contours.

RETR\_LIST: Retrieves all contours without any hierarchical relationships.

RETR\_CCOMP: Retrieves all contours and organizes them into a two-level hierarchy. The first level contains external boundaries of the objects, and the second level contains boundaries of holes within the objects.

RETR\_TREE: Retrieves all contours and reconstructs a full hierarchy of nested contours.

method: Specifies the contour approximation method. It determines how the contour points are approximated.

CHAIN\_APPROX\_NONE: Stores all the contour points. This method saves all the boundary points.

CHAIN\_APPROX\_SIMPLE: Compresses horizontal, vertical, and diagonal segments and leaves only their end points. For example, an up-right rectangular contour is encoded with four points.

CHAIN\_APPROX\_TC89\_L1, CHAIN\_APPROX\_TC89\_KCOS: Applies one of the flavors of the Teh-Chin chain approximation algorithm.

offset: Optional parameter that specifies an offset to be added to each contour point. This parameter is usually not used and can be ignored.

Usage:

Image Preparation: The input image should be preprocessed to extract edges or contours. Common preprocessing steps include thresholding, edge detection (e.g., Canny edge detection), or any other technique to segment the image.

Contour Detection: Call findContours() with the preprocessed image and specify the desired retrieval mode and approximation method.

Contour Processing: Iterate over the detected contours, which are stored in the contours output parameter. Each contour is represented as a vector of points. You can then perform further processing on the contours as needed, such as calculating their areas, centroids, bounding boxes, or drawing them on the original image.

# findContour() differentiation details:

|  |  |  |  |
| --- | --- | --- | --- |
| **Mode** | **Method** | **Specification** | **Utility** |
| RETR\_EXTERNAL | CHAIN\_APPROX\_NONE | Retrieves only the external contours. Stores all contour points without approximation. | Precise representation of external contour points. Suitable for scenarios where exact contours are needed. |
| RETR\_EXTERNAL | CHAIN\_APPROX\_SIMPLE | Retrieves only the external contours. Compresses segments, leaving only their end points. | Minimalistic representation of external contours while preserving overall shape. Less computational cost compared to CHAIN\_APPROX\_NONE. |
| RETR\_EXTERNAL | CHAIN\_APPROX\_TC89\_L1 | Retrieves only the external contours. Applies Teh-Chin chain approximation algorithm with L1 distance. | More accurate contour approximation, especially for contours with sharp angles or edges. |
| RETR\_EXTERNAL | CHAIN\_APPROX\_TC89\_KCOS | Retrieves only the external contours. Applies Teh-Chin chain approximation with Kochanski approximation. | Similar to CHAIN\_APPROX\_TC89\_L1 but may provide better results for certain types of contours. |
| RETR\_LIST | CHAIN\_APPROX\_NONE | Retrieves all contours without hierarchical relationships. Stores all contour points without approximation. | Obtains all contours in the image without hierarchy, suitable when precise contour points are needed for all contours. |
| RETR\_LIST | CHAIN\_APPROX\_SIMPLE | Retrieves all contours without hierarchical relationships. Compresses segments, leaving only their end points. | Minimalistic representation of all contours while minimizing the number of points. Suitable for less computationally expensive contour processing. |
| RETR\_LIST | CHAIN\_APPROX\_TC89\_L1 | Retrieves all contours without hierarchical relationships. Applies Teh-Chin chain approximation with L1 distance. | More accurate contour approximation for all contours without hierarchy. |
| RETR\_LIST | CHAIN\_APPROX\_TC89\_KCOS | Retrieves all contours without hierarchical relationships. Applies Teh-Chin chain approximation with Kochanski approximation. | Similar to CHAIN\_APPROX\_TC89\_L1 but may provide better results for certain types of contours. |
| RETR\_CCOMP | CHAIN\_APPROX\_NONE | Retrieves all contours and organizes them into a two-level hierarchy. Stores all contour points without approximation. | Differentiates between external boundaries and holes within objects with precise contour points for both. |
| RETR\_CCOMP | CHAIN\_APPROX\_SIMPLE | Retrieves all contours and organizes them into a two-level hierarchy. Compresses segments, leaving only their end points. | Minimalistic representation of all contours with hierarchical relationships, suitable for distinguishing between boundaries and holes. |
| RETR\_CCOMP | CHAIN\_APPROX\_TC89\_L1 | Retrieves all contours and organizes them into a two-level hierarchy. Applies Teh-Chin chain approximation with L1 distance. | More accurate contour approximation for both external boundaries and holes within objects. |
| RETR\_CCOMP | CHAIN\_APPROX\_TC89\_KCOS | Retrieves all contours and organizes them into a two-level hierarchy. Applies Teh-Chin chain approximation with Kochanski approximation. | Similar to CHAIN\_APPROX\_TC89\_L1 but may provide better results for certain types of contours within the hierarchical structure. |
| RETR\_TREE | CHAIN\_APPROX\_NONE | Retrieves all contours and reconstructs a full hierarchy of nested contours. Stores all contour points without approximation. | Analyzes hierarchical relationships between contours with precise contour points for each contour. |
| RETR\_TREE | CHAIN\_APPROX\_SIMPLE | Retrieves all contours and reconstructs a full hierarchy of nested contours. Compresses segments, leaving only their end points. | Minimalistic representation of contours with hierarchical relationships, suitable for hierarchical contour analysis. |
| RETR\_TREE | CHAIN\_APPROX\_TC89\_L1 | Retrieves all contours and reconstructs a full hierarchy of nested contours. Applies Teh-Chin chain approximation with L1 distance. | More accurate contour approximation for nested contours within the hierarchical structure. |
| RETR\_TREE | CHAIN\_APPROX\_TC89\_KCOS | Retrieves all contours and reconstructs a full hierarchy of nested contours. Applies Teh-Chin chain approximation with Kochanski approximation. | Similar to CHAIN\_APPROX\_TC89\_L1 but may provide better results for certain types of contours within the hierarchical structure. |