2021

AUGELAB STUDIO USER MANUEL

No-Code Based Computer Vision & AI Solution Development Platform

AugeLab R&D Tech Inc.
1.10.2021

1. Introduction

This user guide has been prepared for you to use AugeLab Studio more effectively and efficiently. In the manual, you can find detailed information about the product's usage areas, content, interface, and how to use the existing functions.
AugeLab Studio is developed by AugeLab Image Processing and Industrial Automation Industry Joint Stock Company.

1.1. AugeLab Studio

AugeLab Studio allows you to develop your own image processing and artificial intelligence systems in the fastest and easiest way without the need for coding knowledge.

Thanks to AugeLab Studio, it aims to accelerate the R&D activities of your project and to commercialize your quick solution by simply integrating it into industrial systems.

With its modules, you can start using only the image processing and artificial intelligence functions you need by dragging and dropping them.

With AugeLab Studio ;

- Automotive
- Consumer products
- Electronics
- Food and drink
- Logistics
- Medical devices
- Medical
- Medicine
- Packaging
- Smart farming
- Solar energy systems

And you can develop applications for other industries.

1.2. System Requirements

Processor: i5 2.5 GHz 4 core

Memory: 8GB

Graphics Card: Internal Graphics Card

Operating System: Windows 7 ,8 or 10

Storage: 1GB available space

Network Place: Internet connection required for activation.

1.3. Product Ingredients

1.3.1. Image Processing Module

Functions you can find in the product;

- Image Filters
- Character, Text, Barcode Reading
- Geometric Image Transformations
- Various Image Transformations
- Drawing On Image Functions
- Color Space Conversions
- Color Maps
- Image Analysis Histograms
- Image Structural Analysis and Shape Descriptors
- Motion Analysis and Object Tracking
- Object Feature Detection
- Object Finding
- Hardware Acceleration Layer

1.3.2. Deep Learning Module

Functions you can find in the full version;

- CNN Architecture Template
- Optimization Modules
- Activation Modules

1.4. Installation and Licensing

In order to run the AugeLab Studio software, the necessary environments must be prepared. To prepare this environment;

- Microsoft Visual Studio
- NVIDIA CUDA Toolkit
- NVIDIA cuDNN

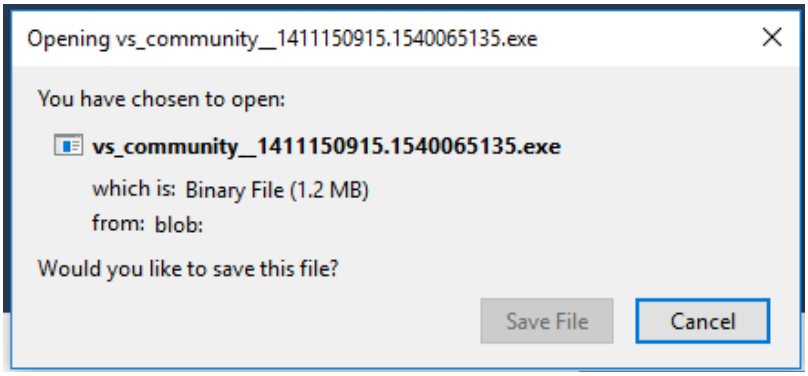
The software must be installed.

1.4.1. Microsoft Visual Studio

Microsoft Visual Studio is a prerequisite for the CUDA toolkit. If you try to install the CUDA Toolkit without first installing Microsoft Visual Studio, you will get the message shown in Figure 1.1.

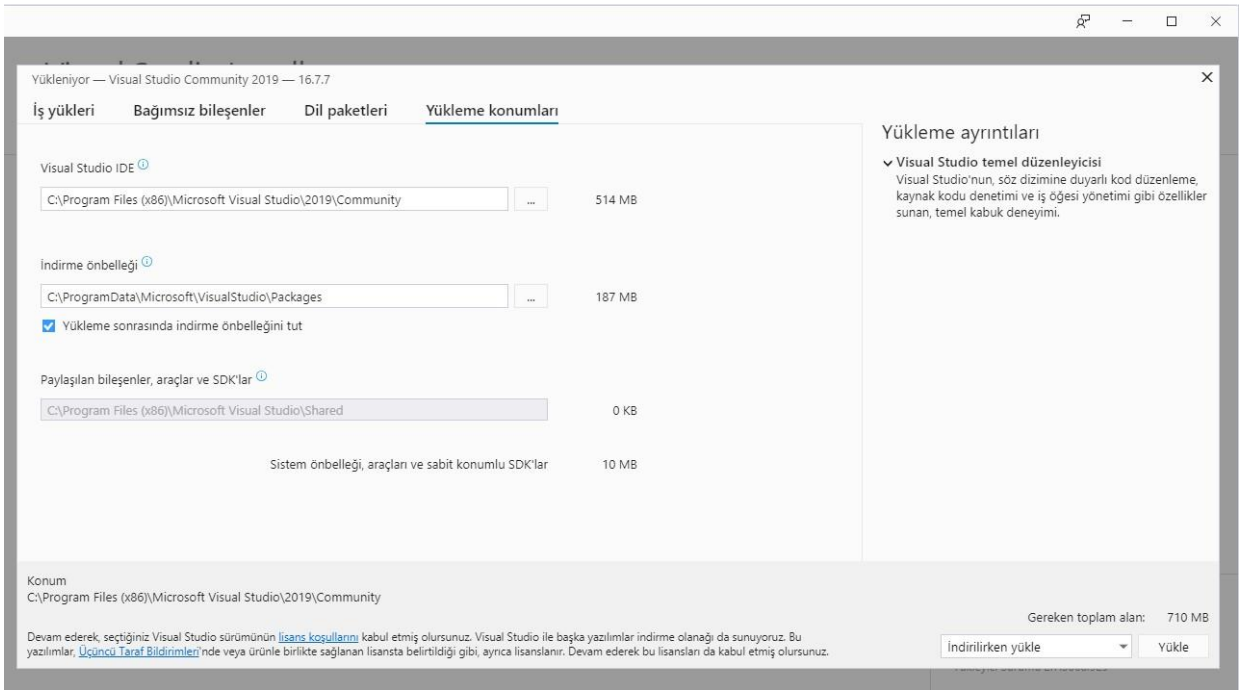
Şekil 1.1. CUDA Toolkit' Message when trying to install without Visual Studio

The latest version of Visual Studio is Visual Studio Community 2019. By joining Visual Studio Dev Essentials, you can download any version of Visual studio for free.
After downloading Visual Studio Community, when starting its installation, Figure 1.2 shows the executable you received as a download.



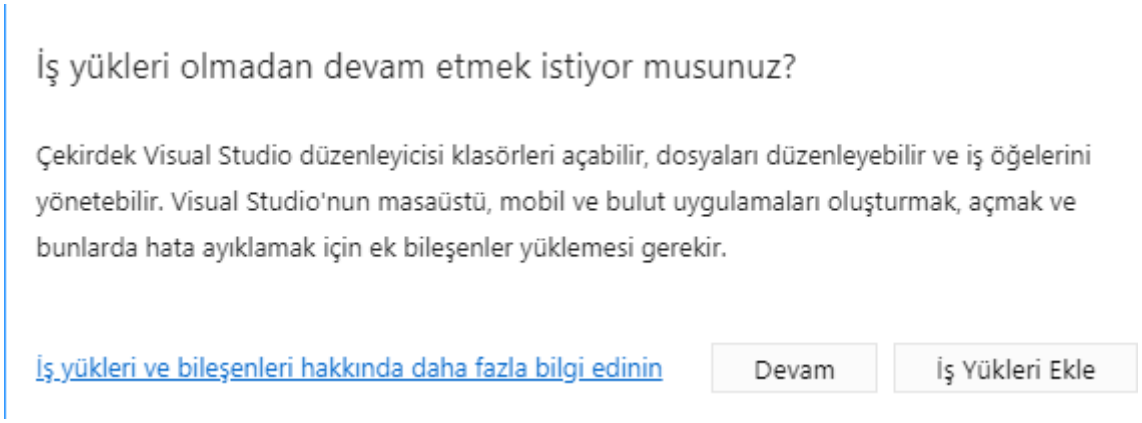
Şekil 1.2. Visual Studio Community executable file

After clicking the 'save file' button in Figure 1.2, the window in Figure 1.3 will appear where you can set the installation options..



Şekil 1.3. Visual Studio Community 2019 for loading window

You can continue by leaving the download options as default. During installation, Visual Studio prompts you to continue without workloads. Since workloads are not required to use AugeLab Studio, you can press the continue button in Figure 1.4.

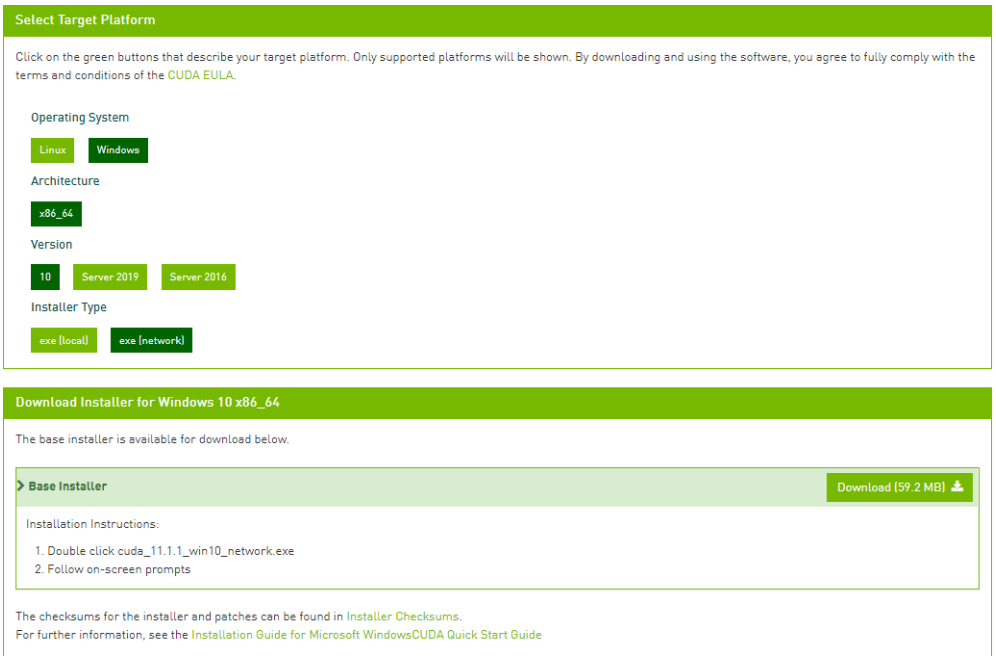


Şekil 1.4. Visual Studio workloads

After completing the Visual Studio installation, you must restart your computer before proceeding to another process.

1.4.2. CUDA Toolkit

The CUDA Toolkit is available for free from the NVIDIA Developer site. The default version of CUDA Toolkit is version 11.1 as shown in Figure 1.5.



Şekil 1.5. Windows için CUDA Toolkit latest version window

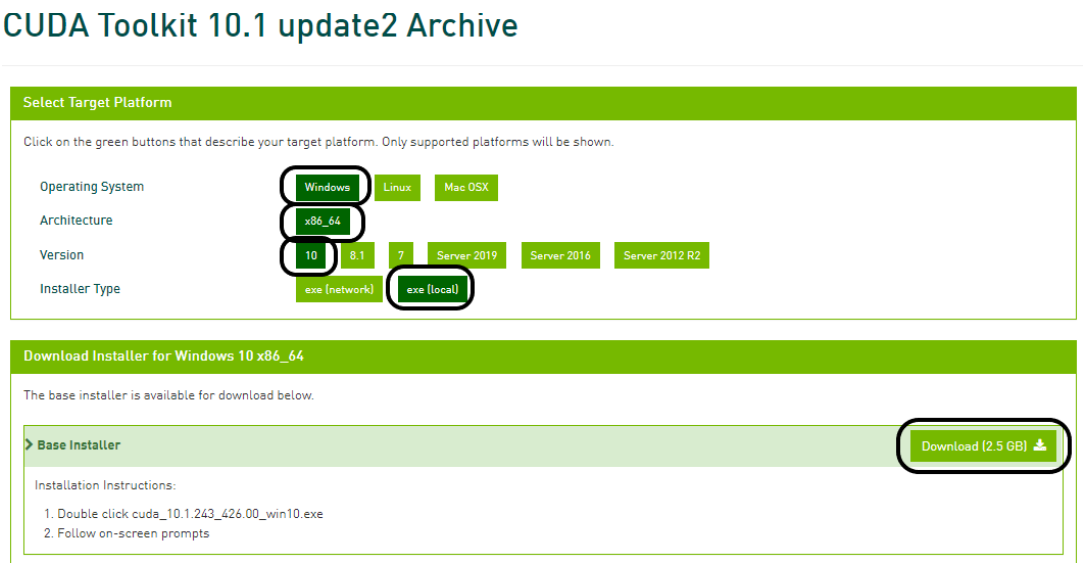
The version required for AugeLab Studio is CUDA Toolkit 10.1. In order to access the old versions, it is necessary to go to the CUDA Releases tab from the "Resources" section shown in Figure 1.6 and located at the bottom of the page.

Şekil 1.6. CUDA Releases tab to access Legacy Releases

It can be downloaded by selecting the CUDA Toolkit version to be installed from the page shown in Figure 1.7.
CUDA Toolkit version 10.1 must be downloaded for AugeLab Studio.

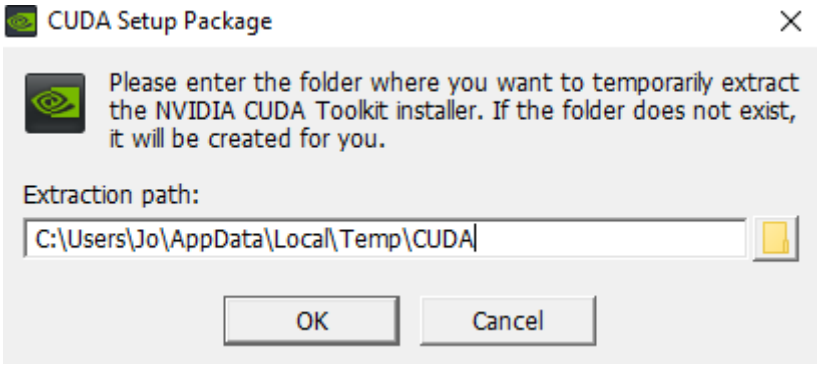
Şekil 1.7. CUDA Toolkit for listed versions

To install CUDA Toolkit 10.1, a screen appears as shown in Figure 1.8. Download options for Windows 10 are shown in Figure 1.8.



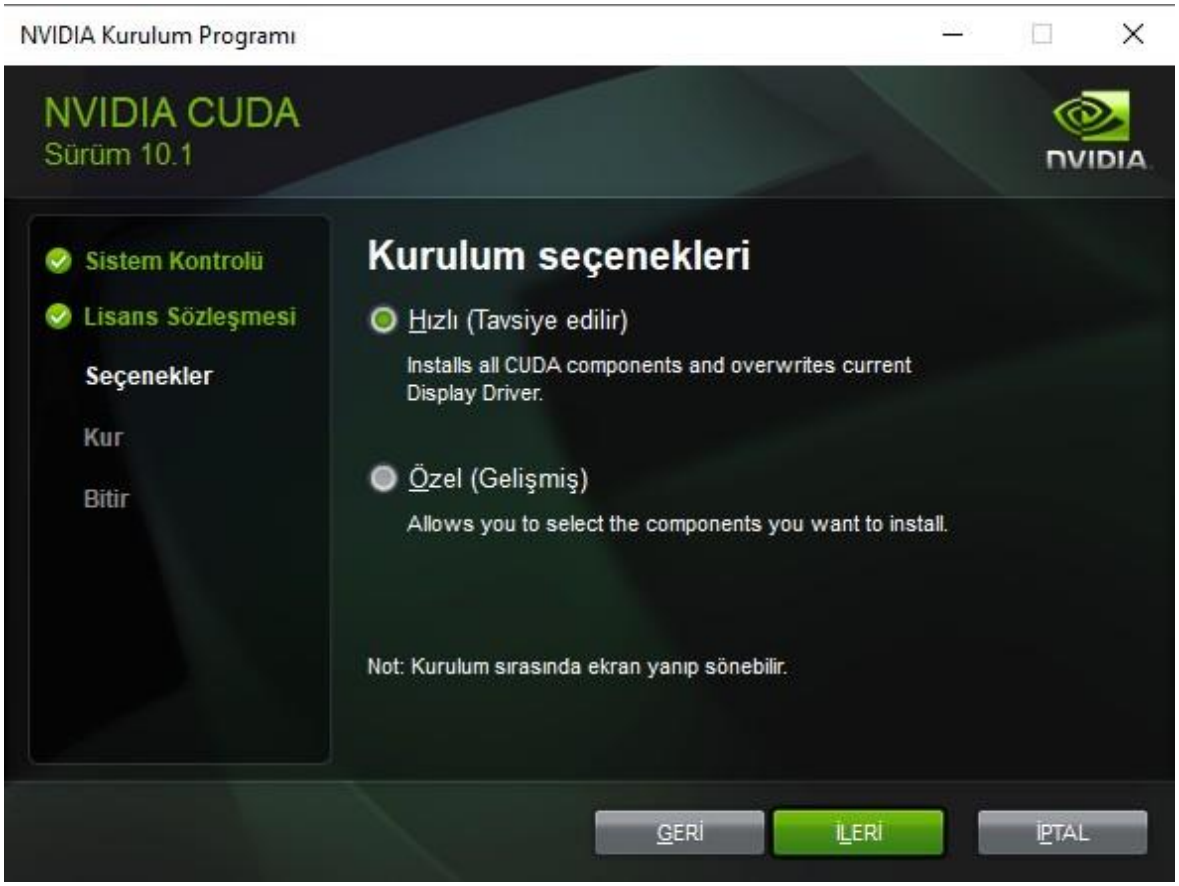
Şekil 1.8. CUDA Toolkit 10.1 Download options for version

When the basic installer with the .exe extension is run after the download is complete, the installation package window shown in Figure 1.9 appears. The location where the CUDA Toolkit installer will be downloaded can be selected on the window. The file location should be left as default.



Şekil 1.9. CUDA Toolkit for installation package window

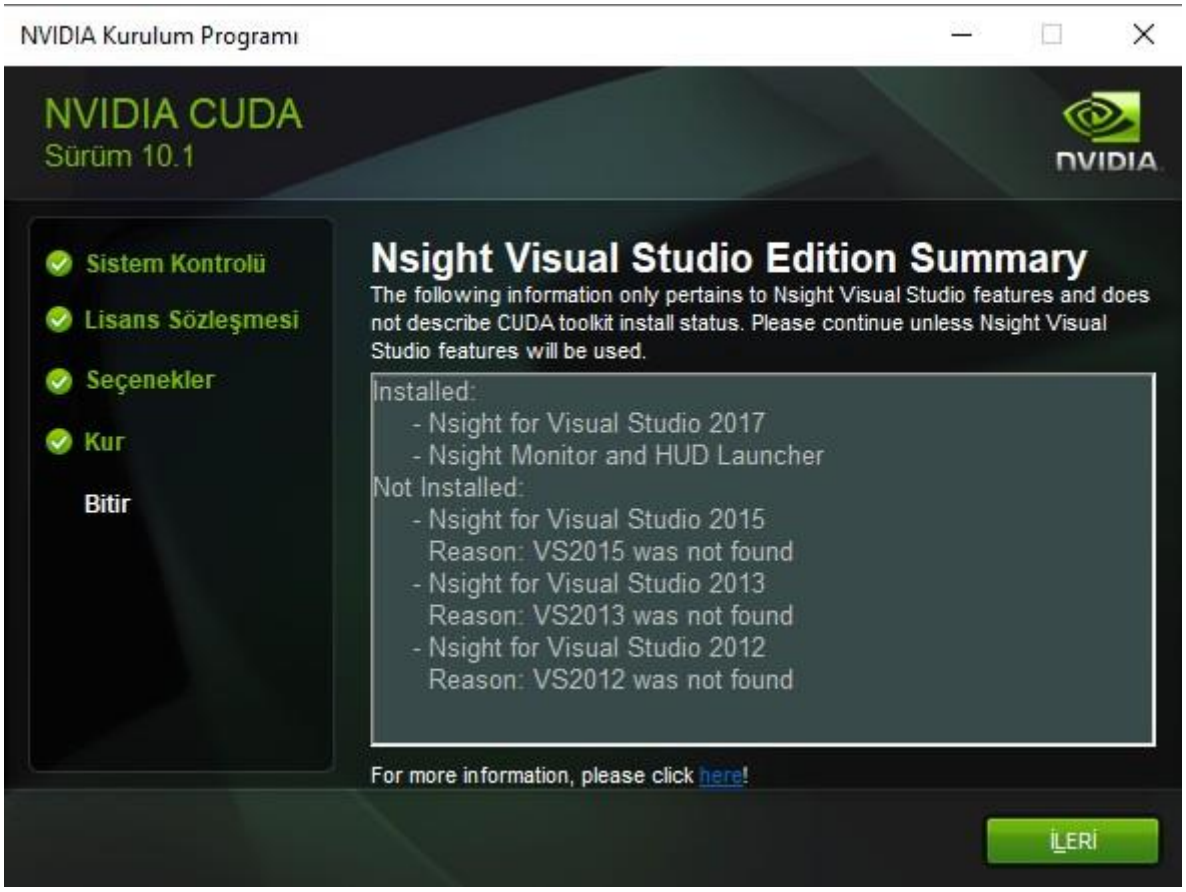
The installation package extracts the CUDA installer to your computer, and when complete, the NVIDIA CUDA Toolkit installation will begin. When the installation starts, you will see the window shown in figure 1.10 and the installation process will continue through this window.



Şekil 1.10. NVIDIA CUDA Toolkit setup window

The installation should be continued by selecting the Quick installation option from the window in Figure 1.10. With the quick installation option, CUDA Toolkit creates the "C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v10.1" directory path by default and installs the files in this directory.

Figure 1.11 shows CUDA installations based on previously installed Visual Studio.



Şekil 1.11. Window showing installations using Visual Studio for CUDA

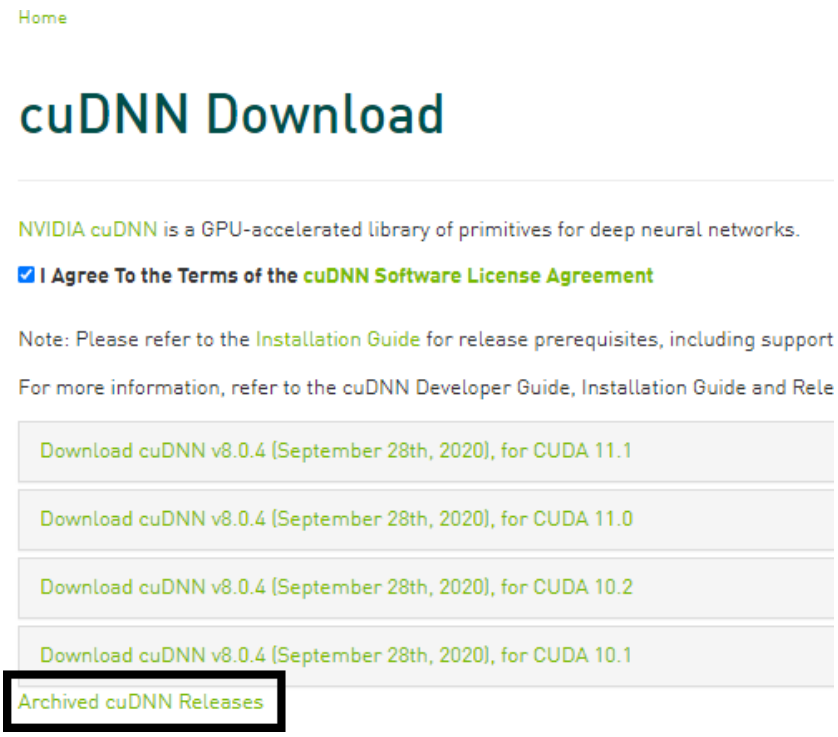
Pressing the next button in the window shown in Figure 1.11 shows the window shown in Figure 1.12 where the NVIDIA CUDA installer is marked as complete.

Şekil 1.12. Final installation window for CUDA installer

1.4.3. CuDNN Installation

After downloading and installing CUDA Toolkit 10.1, the next step is to find a compatible cuDNN version. CuDNN can be downloaded for free from the NVIDIA Developer site. You must be a member of NVIDIA Developer to start the download process.

After creating an account and logging in, the download page shown in Figure 1.13 appears. To access older versions, you need to click on "Archived cuDNN Releases".



Şekil 1.13. cuDNN download list

The required cuDNN version for AugeLab Studio is 7.6.5. For CUDA 10.1 shown in Figure 1.14, the compatible cuDNN 7.6.5 version should be selected and downloaded.

Şekil 1.14. CUDA 10.1 for cuDNN 7.6.5 version

After the download is complete, the files in the zip should be extracted to a folder. There are three files in the subdirectories of the cuDNN folder extracted from the zip file to be copied to the CUDA Toolkit directories.

1. cudnn64_7.dll

The cudnn64_7.dll file in the “bin” folder in the subdirectory of the downloaded cuDNN file is located in the “bin” folder in the file path where CUDA Toolkit is installed (if you installed it by default: C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v10.1) (without creating any new folders).

2. cudnn.h
   As in the file above, copy the cudnn.h file in the "include" folder in the subdirectory of the cuDNN folder to the "include" folder in the same file path where CUDA Toolkit is installed (C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v10.1) required.
3. cudnn.lib
   You need to copy the cudnn.lib file in the "lib" folder in the subdirectory of the downloaded cuDNN folder to the "lib" folder in the same file path where CUDA Toolkit is installed (C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v10.1).

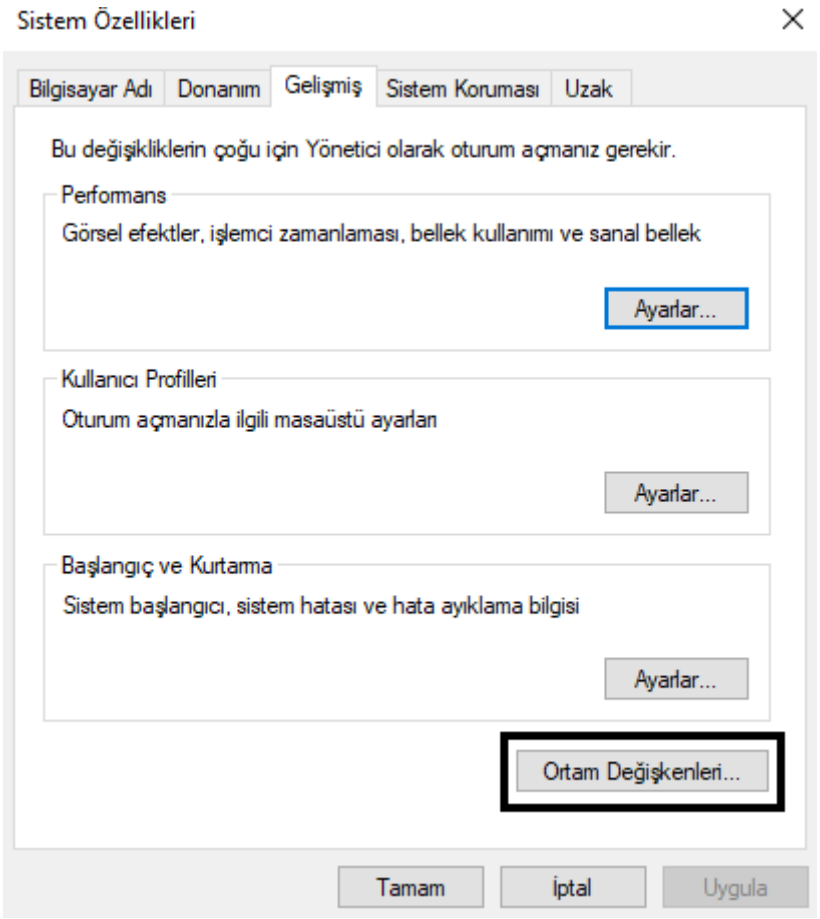
1.4.4. Checking and Manipulating CUDA Environment Variables

After the CUDA Toolkit and cuDNN components are installed, the CUDA environment variables must be added to the system settings.

To access environment variables;

Control Panel ➔ System and security ➔ System ➔ Advanced System Settings

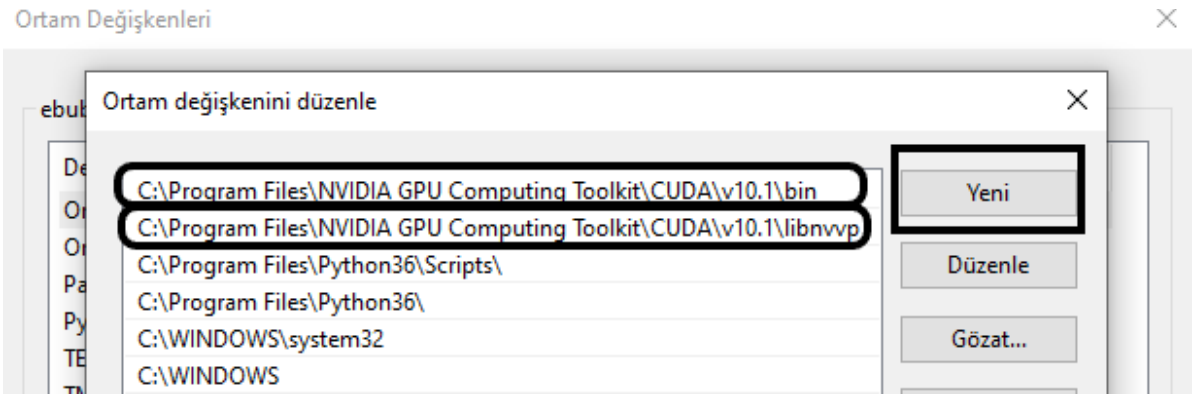
From here, a window called "System Properties", shown in Figure 1.15, opens. “Environment Variables” can be opened from this window.



Şekil 1.15. System Properties window

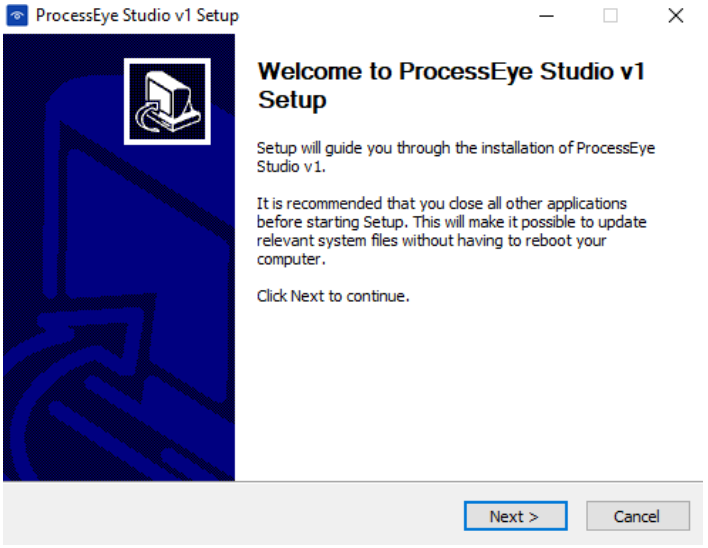
When the Environment Variables window opens, click on “Path” from “system variables” and select the “edit” button. A new window called "Edit environment variable" opens as shown in Figure 1.16.

The CUDA Toolkit that we have installed from the environment variables should contain two paths showing the file path. If these two paths are not available, you need to add the CUDA paths by clicking the "New" button.



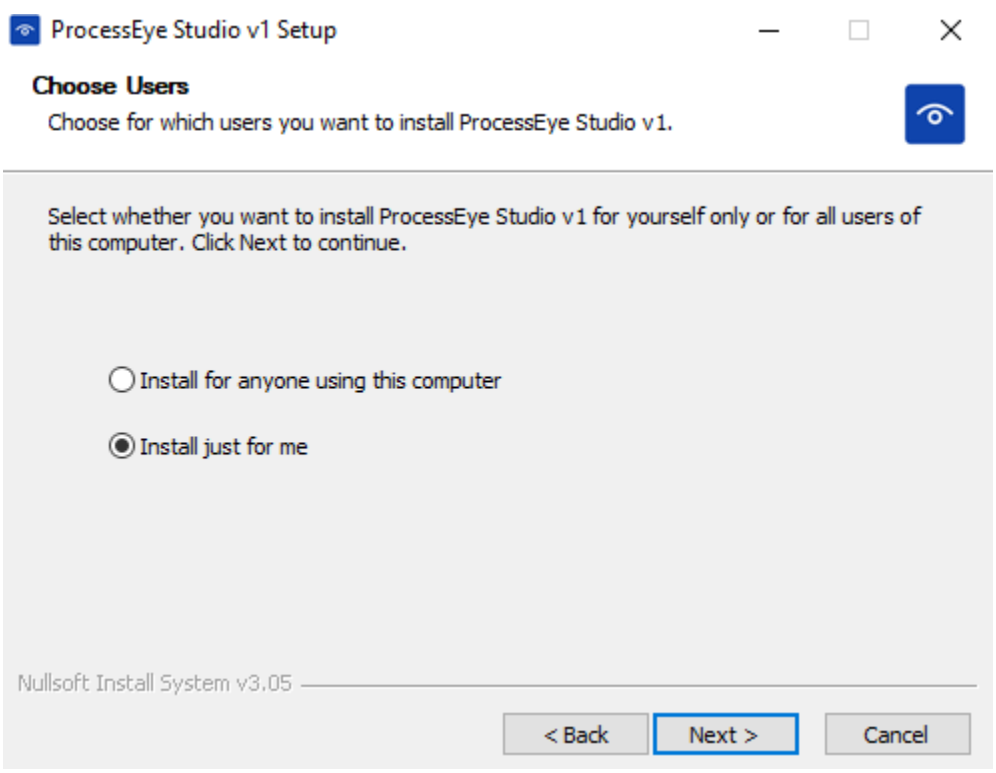
Şekil 1.16. Environment Variable paths that need to be created after setup

1.4.5. AugeLab Studio



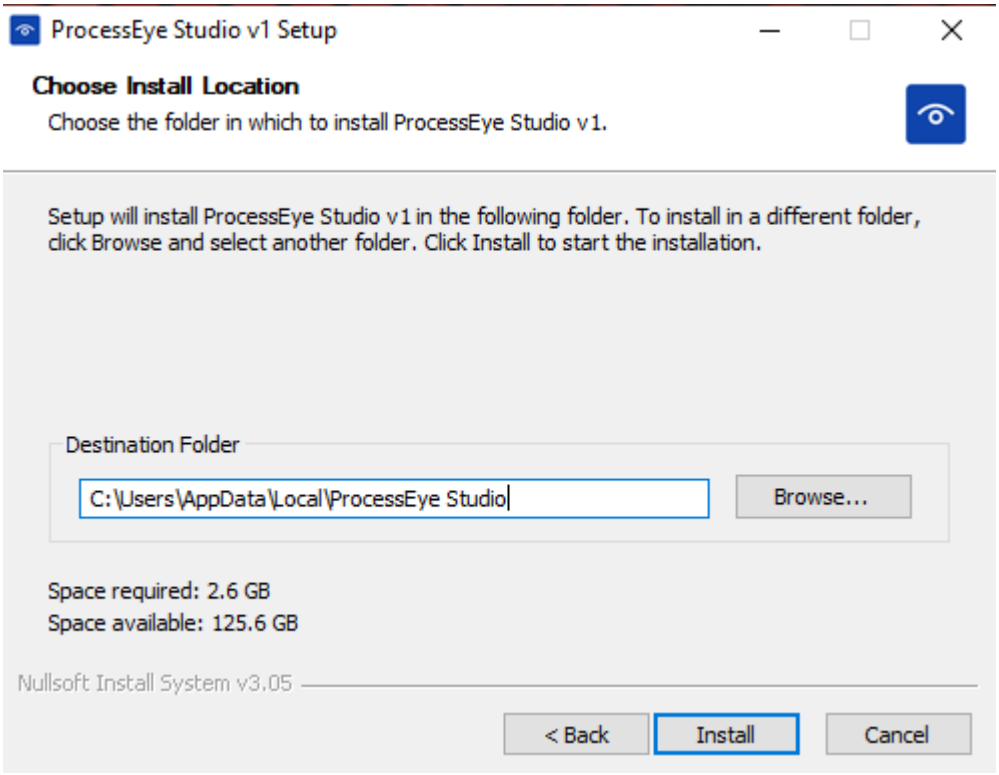
Resim 1.17. Installation first step

After installing the CUDA and cuDNN components required to use the AugeLab Studio software, we can now proceed to the final installation process to use the software. After running the .exe extension setup file for AugeLab Studio, the screen in Picture 1.17 will appear.



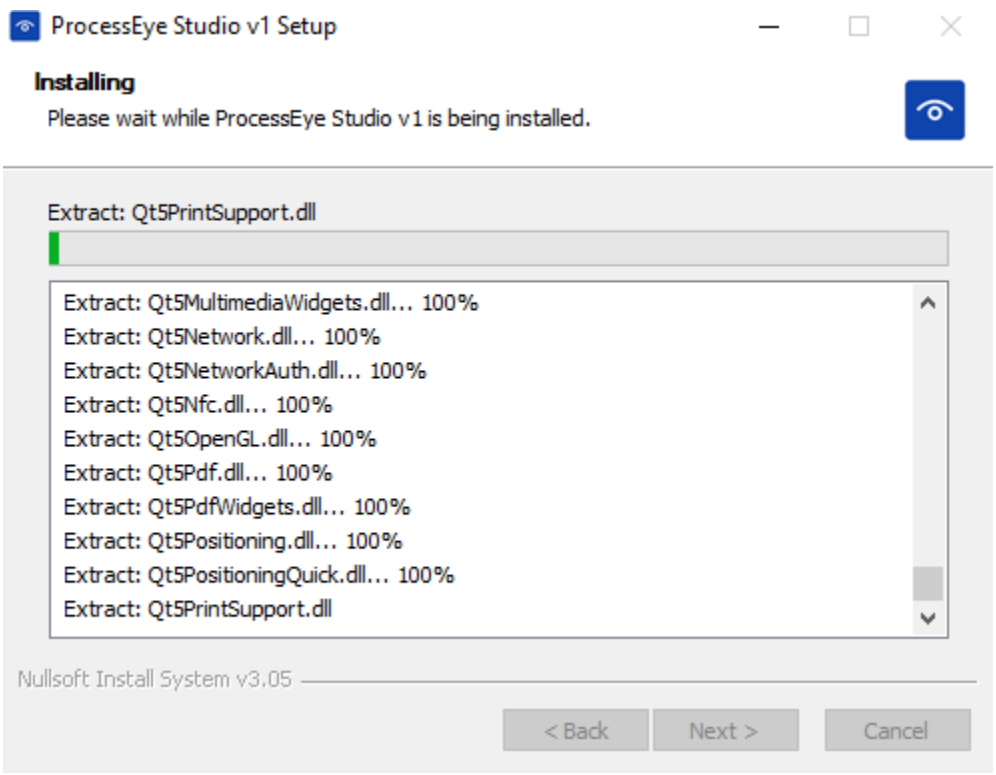
Resim 1.18. User selection

As a second step, on the screen shown in Picture 1.18, whether the AugeLab Studio software will be installed for all users on the computer or for one user is selected. After making the selection, click the Next button for the next step.



Resim 1.19. Destination folder where the files will be installed

The target folder where Auge Lab Studio will be installed should be selected on the screen in Picture 1.19. After selecting the folder, click the Install button to install the files.



Resim 1.20. Installation process of files

Figure 1.20 shows the installation screen of the files in the target folder. After the installation process is completed, click the Finish button to complete the installation.
You can start creating scenarios for your projects by running the AugeLab Studio software.

1.4.6. Licence

Important: Carefully read the rights and obligations set forth in this license agreement. During installation, you will be asked whether you accept these terms. If you do not agree, the software will not be installed on your computer. Installing the program on your computer indicates that you accept the terms of the agreement. This agreement for the AugeLab Studio Software (SOFTWARE) is a legal agreement between AugeLab and the END USER (natural or legal). By installing the SOFTWARE, you accept the terms of this agreement. If you do not accept the agreement, you cannot install and use the SOFTWARE.

I- RIGHTS AND OBLIGATIONS OF THE END USER:

1. The END USER is responsible for meeting the minimum system requirements for the SOFTWARE (defined under the heading 1.3 System Requirements). It accepts, undertakes and declares to use the infrastructure services required for the operation of the SOFTWARE (Example: Internet, camera system or all hardware products belong to the company that will use the software). The END USER declares and accepts that he/she knows that the right to benefit from the provisions of clause III-1.b, 1.c of this contract will cease to exist due to the damages and losses that may occur in the product subject to the contract due to the violation of the above written provision.

2. The END USER accepts, undertakes and declares that he will install the SOFTWARE on a single computer with a single license and use it only in these environments, and will not transfer it to another environment permanently or temporarily by any means or in any way.

3. END USER; The SOFTWARE declares and accepts that if it supports a multi-user system, it can be installed and run on more than one computer in the network, that it must obtain a license for each computer in the network, and that a SOFTWARE license cannot be shared and the SOFTWARE cannot be used on different computers at the same time.

4. If the SOFTWARE is used for multiple users; The END USER has the right to install the maximum program that can be run on other computers as much as the number of clients specified when obtaining the license..

5. The END USER accepts, undertakes and declares that s/he will use the community version for the registration number and duration determined by AugeLab, and not for any reason, for commercial or professional purposes or other profit-making reasons.

6. END USER

a. Will not reproduce the SOFTWARE and documentation other than those permitted by this License Agreement, will not decompile or recompile the SOFTWARE,

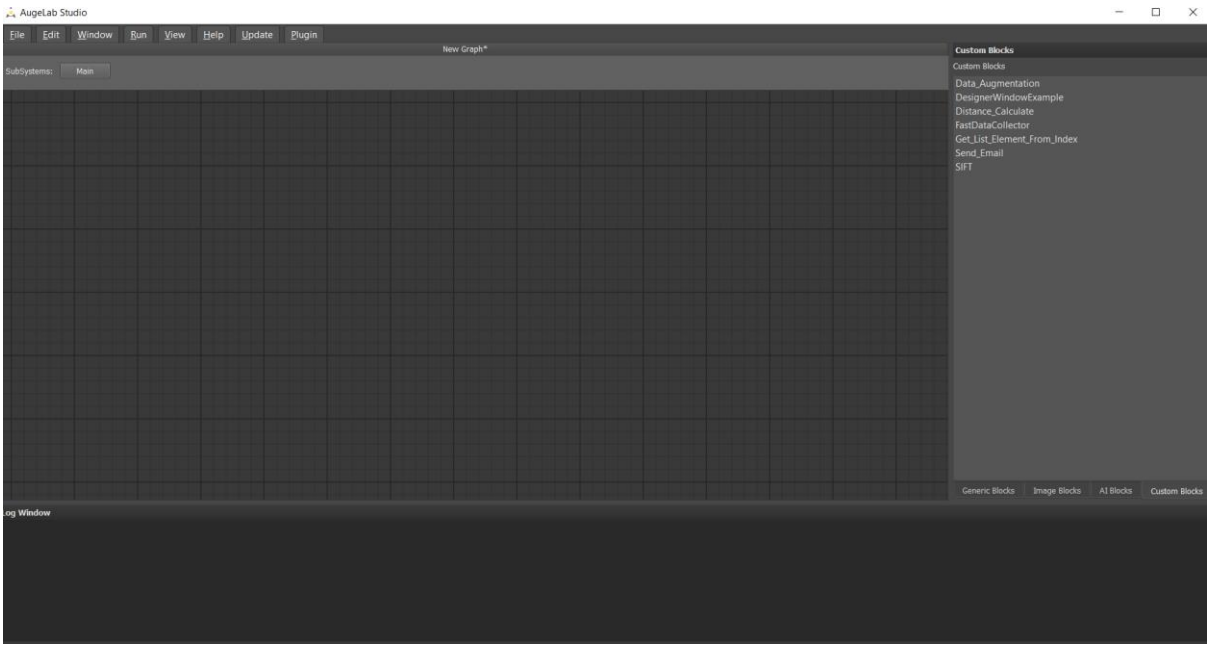
b. shall not distribute, lend, rent, sell, or transmit the Premium version of the SOFTWARE to any other person, without the written consent of AugeLab,

c. It accepts, declares and undertakes that it will not change or adapt the SOFTWARE and its documentation or create inspired works based on the SOFTWARE and documentation.

7. AugeLab Image Processing and Industrial Automation Technologies Industry Inc. reserves all rights on this software. The right holder has the right to make paid features or change the licensing method and price in the next SOFTWARE versions or on this version.

2. User Interface

2.1. Interface Components



2.1.1. Grid Area (1):

It is the area on which the function blocks are placed and interconnected. on this area
You can set up your own flow diagrams and align them easily.
You can zoom-in, zoom-out with the mouse wheel, and navigate in the grid area by pressing and holding the wheel.

2.1.2. Log Window (2):

In case of any warning or error during your transactions
is the window in which the warning/error message is displayed.

2.1.3. Blocks Toolbar (3):

It is the window where the function blocks are located. You can use the function you need on the main screen with the drag and drop method by using the left mouse button.

2.1.4. Generic Blocks (4):

It contains function blocks with input, output and operators to use.

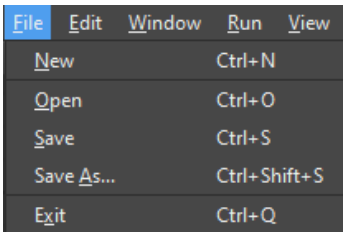
2.1.5. Image Blocks (5):

It contains blocks with image processing functions.

2.1.6. AI Blocks (6):

It includes deep learning function blocks.

2.1.7. File Butonu (7):

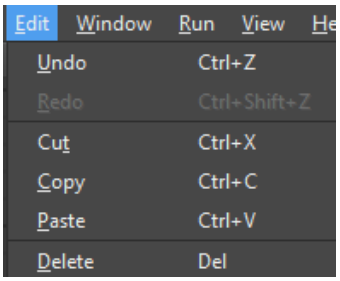


Under the File menu;

New: Creating a new project file
Open: Selecting the saved project file to load
Save: Saving the current project file over the same file
Save as: Saving current project file with different name and location
Exit: Exit

Operations can be performed.

2.1.8. Edit Button (8):



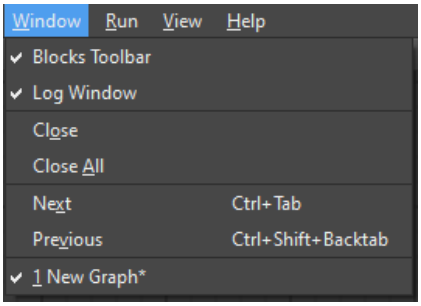
Under the Edit menu;

Undo: Undo the last action
Redo: Back to last commit **Cut:** Cut
Copy: Copy
Paste: Paste
Delete: Delete

Operations can be performed.

2.1.9. Window Button (9):

Under the Window menu;



Blocks Toolbar: You can make the menu with function blocks appear on the work screen.

Log Window: You can make the menu with Warning/Error messages appear on the work screen..

Close: Closes current project tab.

Close All: Closes all project tabs.

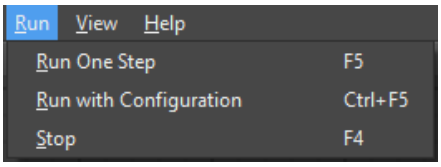
Next: Switches to the next project tab.

Previous: Switches to the previous project tab.

Operations can be performed.

2.1.10. Run Button (10):

Under the Run menu;



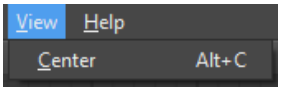
Run One Step: Runs the program for a single loop.

Run With Configuration: Runs the program with the set run parameters.

Stop: Stops the program.

2.1.11. View Button (11):

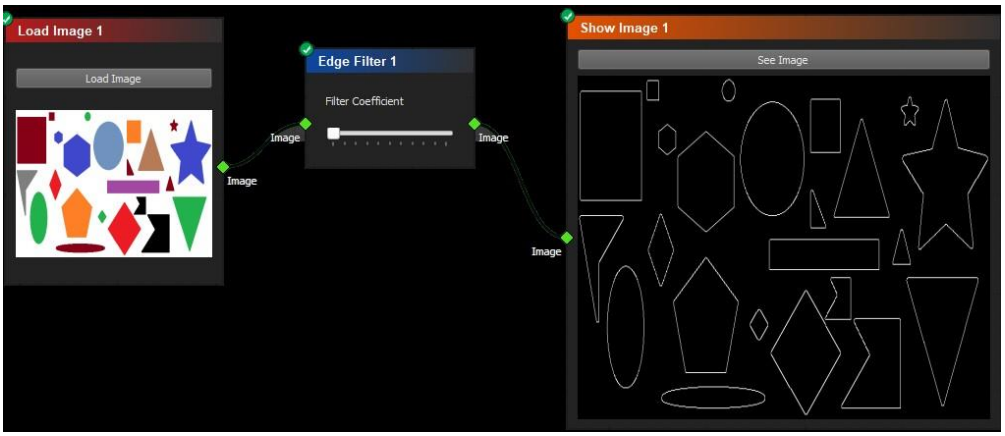
Under the View menu;



Center: Centers the worked blocks in the grid area.

2.2. Scenario Setup

Add the function blocks you need to the grid area to construct your specific image processing scenarios for your project. To establish the connection between the blocks you added and to provide the data/visual flow, connect the blocks as in Picture 2.1.



Resim 2.1. Edge Filter Scenario

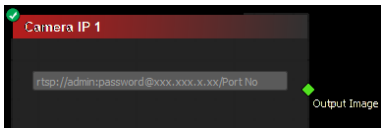
2.3. Keyboard shortcuts

Ctrl + N	Opening a New Work File
Ctrl + O	Opening Existing Work File
Ctrl + S	Saving the File
Ctrl + Shift + S	Saving the File As
Ctrl + Q	Exit
Ctrl + Z	Undo
Ctrl + Shift + Z	Repeat
Ctrl + X	Cut
Ctrl + C	Copy
Ctrl + V	Paste
Del	Delete
Ctrl + Tab	Switching to the Next Project Tab
Ctrl + Shift + Backtab	Switch to Previous Project Tab
F5	One Step Operation
Ctrl + F5	Continuous Operation
F4	Stop
Alt + C	Average to Center
Yön Tuşları	Navigating the Grid Area

3. General Blocks

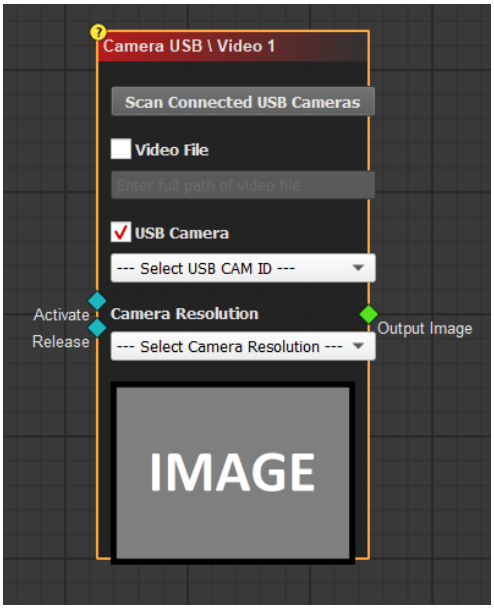
3.1. Inputs

3.1.1. Camera IP



It is used to provide camera connection by entering IP information.

3.1.2. Camera USB



Variables on the function block:

- **Video File:** You can work on the video file by entering the file path.
- **USB Camera:** Allows the camera type to be selected as USB.
- **Camera Resolution:** Allows to change the camera resolution.

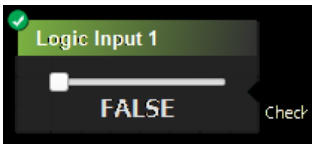
3.1.3. Input

Input block is used for number input. It gives the number entered as output.

3.1.4. Load Image

The Load Image block allows receiving image input from the system for the scenario to be designed. The image to be loaded is selected from the file path with the Load image button. It gives the selected image as output.

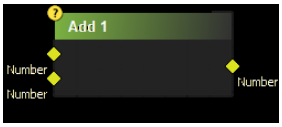
3.1.5. Logic Input



The Logic Input block is used to enter boolean values.

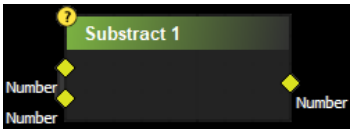
3.2. Simple Operators

3.2.1. Add



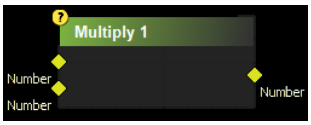
The Add function block is used to add two numeric values. It takes two numeric values as input. Returns the calculated number value as output.

3.2.2. Substract



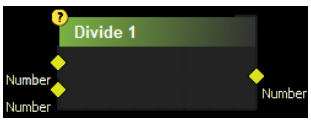
The Subtract function block is used to subtract two numeric values. It takes two numeric values as input. Returns the calculated number value as output.

3.2.3. Multiply



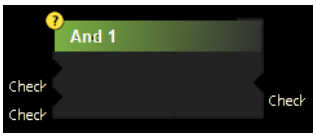
The Multiply function block is used to multiply two numeric values. It takes two numeric values as input. Returns the calculated number value as output.

3.2.4. Divide



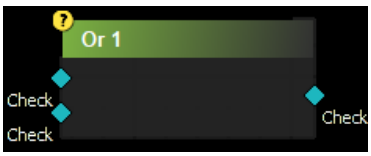
The Divide function block is used to divide two numeric values. It takes two numeric values as input. Returns the calculated number value as output.

3.2.5. And



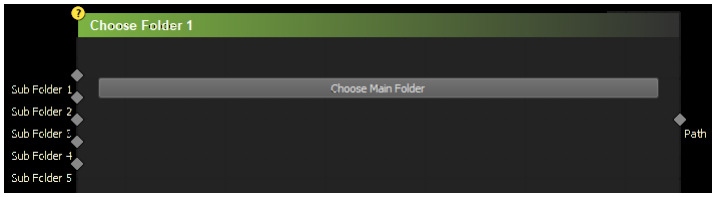
The And function block is used to perform the logical "and" operation. It calculates on the basis of Boolean algebra. It takes two boolean values as input. Returns the calculated boolean value as output.

3.2.6. Or

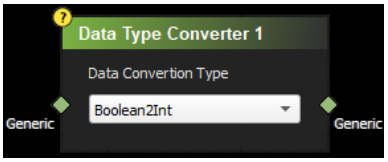


The or function block is used to perform the logical "or" operation. Calculation is made on the basis of Boolean algebra. It takes two boolean values as input. It gives the calculated boolean value as the output.

3.2.7. Choose Folder



3.2.8. Data Type Converter



The Data Type Converter block is used to change between data types

Boolean

⟺Int

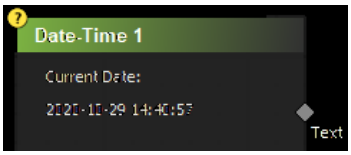
⟺ String

Int ⟺

Float

Data types can be changed between each other. It takes a value of the type to be changed as input and gives an output of the changed type.

3.2.9. Date-Time

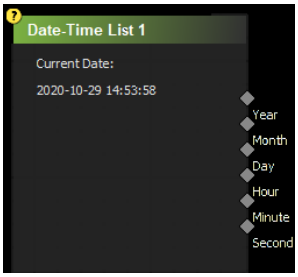


Date- Time block outputs the date and time as text.

3.2.10. Datetime Compare

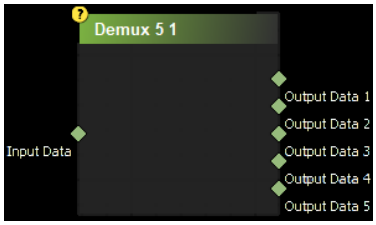
Datetime Compare block contains the year, month, day, hour, date and time given as input. It is used to compare minutes and seconds. It gives boolean value as output.

3.2.11. Date-Time List



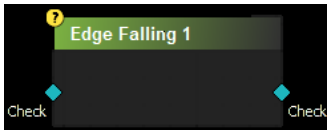
Date-Time List block, date and time; It allows you to use the year, month, day, hour, minute, second parameters as different outputs.

3.2.12. Demux



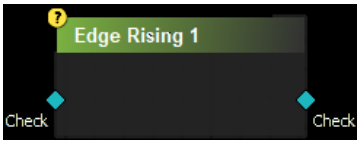
It divides the input value given as a string into parts and gives it as an output.

3.2.13. Edge Falling



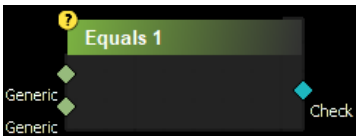
In the case of a signal from the input, the value of 1 is considered to be 0 if there is no signal. The state of falling from 1 to 0 in this way is called Edge Falling.
It outputs TRUE when a falling edge is detected.

3.2.14. Edge Rising



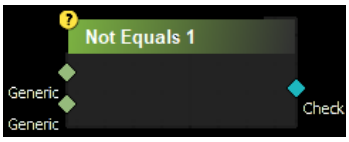
If a transition from 0 to 1 is detected in the signal from the input, it gives a TRUE value from the output.

3.2.15. Equals



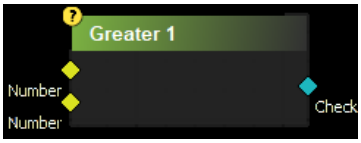
The Equals function block is a relational operator that outputs TRUE (true) provided the two inputs are equal.

3.2.16. Not Equals



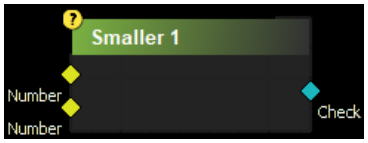
Not Equals function block is the relational operator that outputs FALSE (false) provided the two inputs are equal.

3.2.17. Greater



The Greater function block is the relational operator that outputs TRUE (true) provided the first input is greater than the second input.

3.2.18. Smaller

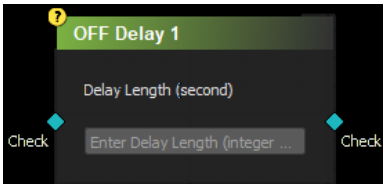


The Smaller function block is the relational operator that outputs TRUE (true) provided the first input is smaller than the second input.

3.2.19. Multi Port Switch

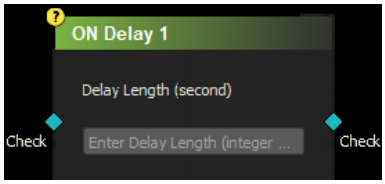
It is used to give the desired input value to the output by directing the different input values given from the inputs with the number value to be entered to the Switch.

3.2.20. Off Delay



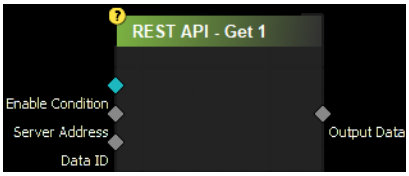
TRUE value as output for the time written in it after the signal from the input is cut off.
gives.

3.2.21. On Delay



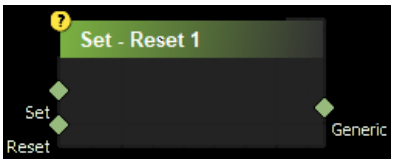
After the signal comes to its input, it gives a TRUE value at the output for the time written in it.

3.2.22. REST API – Get



Used for rest API communication. Server address and data ID are required as input.

3.2.23. Set-Reset

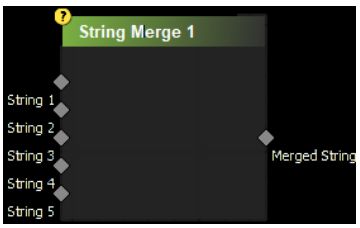


When TRUE value comes from set input, it gives TRUE output value continuously. As soon as TRUE value is given from the reset input, the output value changes to FALSE.

3.2.24. String Input

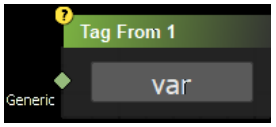
The String Input block is used to assign a string value.

3.2.25. String Merge

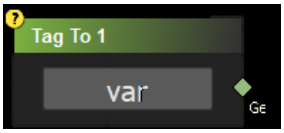


The String Merge block is used to get output by combining the string values it takes as input.

3.2.26. Tag From

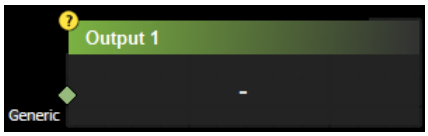


3.2.27. Tag To



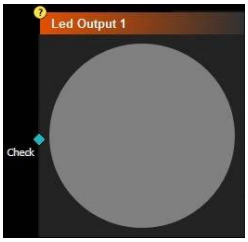
3.3. Outputs

3.3.1. Output



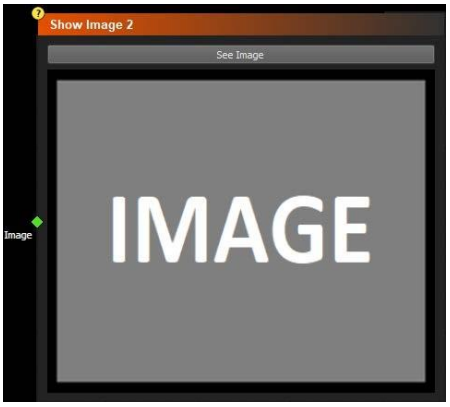
Output block is used to get output values in commonly used string, integer or boolean types.

3.3.2. Led Output



The led output block allows the output value to be displayed with the help of a lamp.

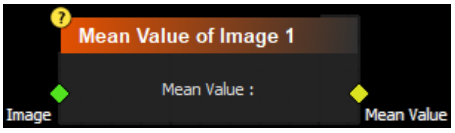
3.3.3. Show Image



The Show Image block is used to visualize the output data to be taken as an image.

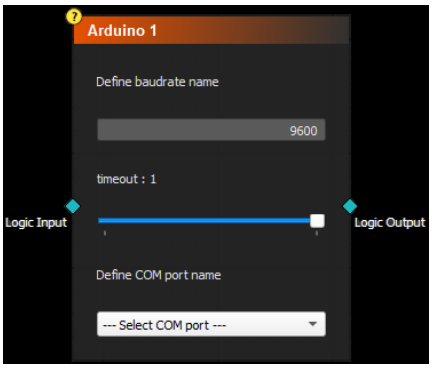
See Image button: It can be used to crop, cross out, resize, save and print the received image.

3.3.4. Mean Value of Image



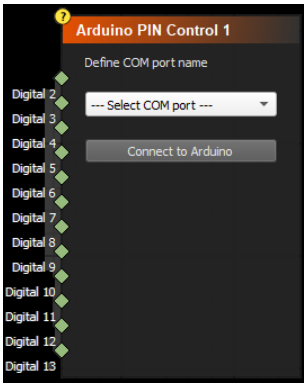
The Mean Value of Image block is used to calculate the average value of the image output. Image should be given as input. The calculated average value can be taken as output.

3.3.5. Arduino



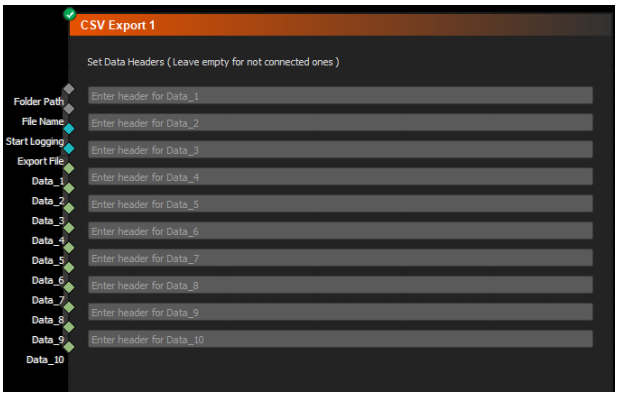
It can be used for data exchange by communicating serially over the connected COM port.

3.3.6. Arduino PIN Control



It can control pins directly via Arduino.

3.3.7. CSV Export



It is used to save data as Excel File.

Folder Path: Path to the folder where the file will be saved

File Name: Name of the file to save

Start Logging: save status

Export File: transfer file

3.3.8. Data to JSON

It is used to save data in JSON format.

3.3.9. GPU Statistics

The GPU Statistics block allows you to see the GPU information you are using and the amount of usage.

3.3.10. Image Write

It is used to save the image.

Folder Path: File path where the image is saved

File Name: Filename for the image to be saved

Write Condition: The necessary condition for recording the image; When TRUE is entered as a value the image will be saved.

Input Image: The image to be saved

3.3.11. Multi Image Write

It is used to save multiple images.

Folder Path: File path where images are saved

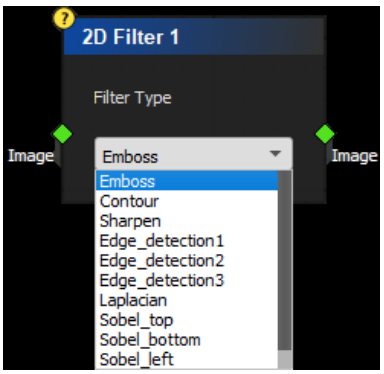
File Name: Filename for images to be saved

Write Condition: The necessary condition for recording the images; When TRUE is entered as a value the image will be saved.

Input Image: Images to be saved

4. Image Processing Function Blocks

4.1. 2D Filter



The 2D Filter function block is used to perform filtering operations on the image input. It gives the filtered image as output. Different filter functions can be used. These;

4.1.1. Emboss

The emboss filter function converts the input image into an embossed image. Basically every pixel is replaced with a shadow or highlight. It performs the filtering process by making gray areas that do not have a flat feature and white pixels (emphasis) or dark pixels (shadows) in contrast areas according to the direction of the relief.

4.1.2. Contour

The Contour filter function creates a curve connecting all continuous points of the same color or density. It can be used for shape analysis or object detection and recognition.

4.1.3. Sharpen

Sharpen filter function gives output by sharpening the image given as input. It can be used to enhance and sharpen the edges on the image.

4.1.4. Edge detection

Edge detection function is used to find sharp lines and edges on the image given as input.

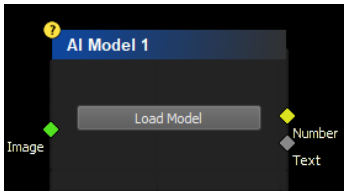
4.1.5. Laplacian

The Laplacian function is an edge detection algorithm. It is used for edge detection by taking the second order derivatives of the pixel values on the image.

4.1.6. Sobel

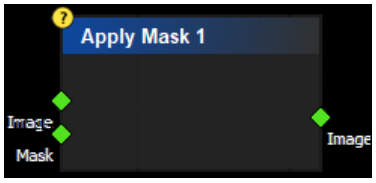
Sobel function is edge detection algorithm. It is a gradient-based method based on the first-order derivative on the image. Gradients can be calculated separately for each axis.

4.2. AI Model



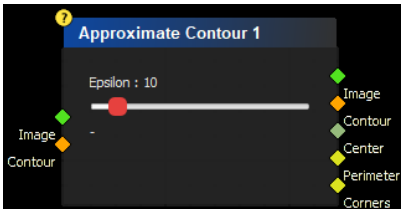
The AI Model block replays the recorded weights of the previously trained deep learning models. It is used to test on images by uploading.

4.3. Apply Mask



The input image is used to add a mask. Masked image as output gives.

4.4. Approximate Contour



The image and the contours in the same image are given as input.

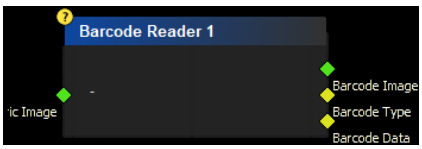
Epsilon value: It is used to determine the curve to be created by the ratio of the drawn contour to the arc length. The higher the Epsilon value, the more the curve between the two points decreases, becoming a straight line.

As output;

- Contoured image
- Pixel values of contour points
- Coordinate of the center point of the contour on the image
- Circumference of the contour
- Number of corner points

It gives.

4.5. Barcode Reader

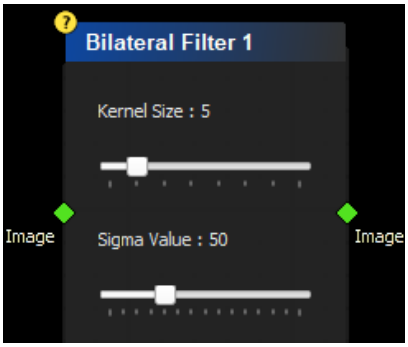


As output by detecting the barcode in the images given as input;

- Barcode marked image
- Type of barcode
- The value of the barcode

It gives.

4.6. Bilateral Filter

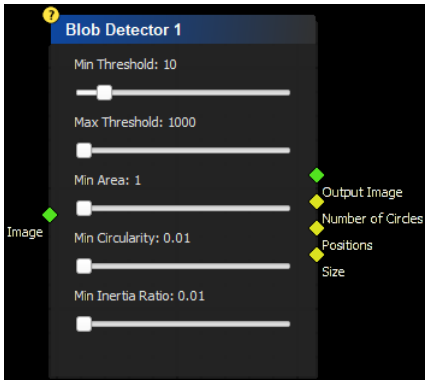


It is used to reduce noise and soften the image in the images given as input. Bilateral filtering can be used to preserve edge points while smoothing the image. Filtered image is given as output.

Kernel size: The size of the kernel to be used when filtering

Sigma Value: The sigma value used when filtering

4.7. Blob Detector



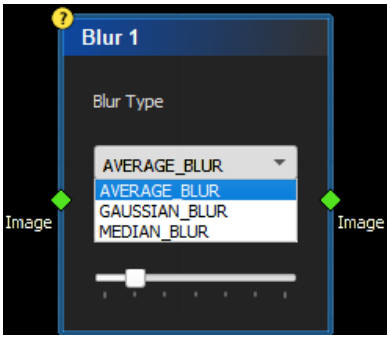
It is a group of pixels that share some common features on the image given as input. The purpose of the blob detector is to identify and detect these regions.

Variables on the function block;

Min threshold: Minimum threshold value

- **Max threshold:** Maximum threshold value
- **Min area:** Smallest field value to select to filter blobs by size
- **Min circularity:** It is the ratio of how close the blobs to find are to a circle. It filters by circularity.
- **Min inertia ratio:** It measures how long a shape is. It takes a value between 0 and 1

4.8. Blur



It is used to blur the image by passing a kernel filter on the image given as input. It can be used to remove noise found in images.

Kernel size: The size of the filter to be hovered over the image.

4.8.1. Average Blur

The kernel takes the average of all the pixels under the filter area and replaces the center value with that average.

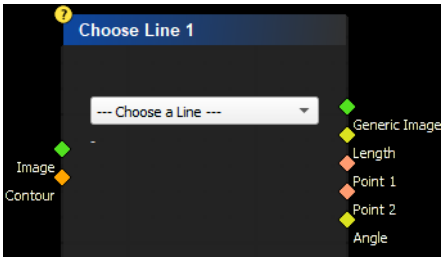
4.8.2. Gausssian Blur

A gaussian kernel is used instead of a box filter with equal filter coefficients.

4.8.3. Median Blur

Calculates the median of all pixels under the core filter area and replaces the center pixel value with this median value.

4.9. Choose Line

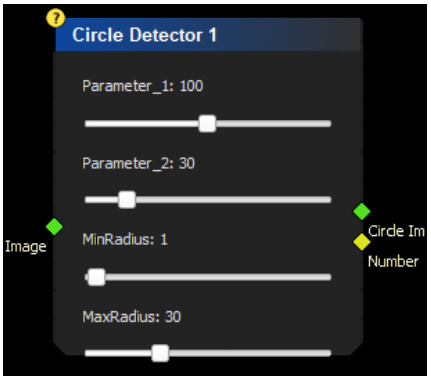


The image and the contours in the same image are given as input. The line line selected from the Choose a Line section;

- Image with contours drawn
- Length
- Starting point
- Endpoint
- The angle it creates

It can be taken as output.

4.10. Circle Detector



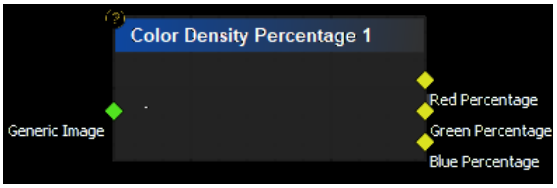
It is used to detect circles or circularities on the image given as input.

Variables on the function block;

- Parameter 1 and Parameter 2 are parameters used to set the circularity ratio.
- The Min Radius and Max Radius parameters are used to determine the range of radius length of the circle.

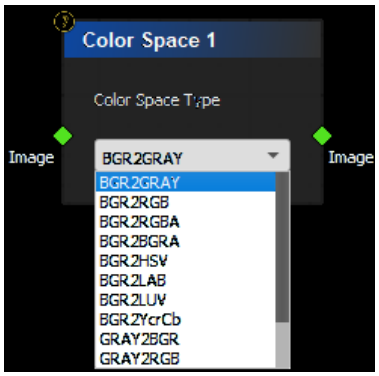
It gives the image in which the circles are detected and drawn and the number of detected circles as output.

4.11. Color Density Percentage



It gives the RGB color percentages on the image given as input as output.

4.12. Color Space



By changing between color spaces, it transforms the image given as input into a different color space and gives it as a new image.

Color Spaces;

BGR ➔ Gray

BGR ➔ RGB

BGR ➔ RGBA

BGR ➔ BGRA

BGR ➔ HSV

BGR ➔ LAB

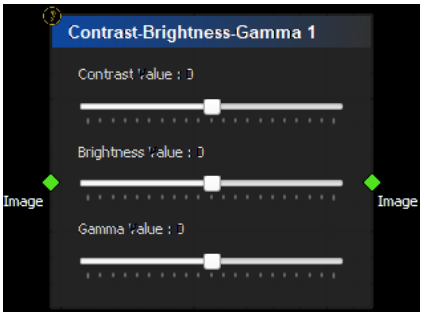
BGR ➔ LUV

BGR ➔ YcrCb

Gray ➔ BGR

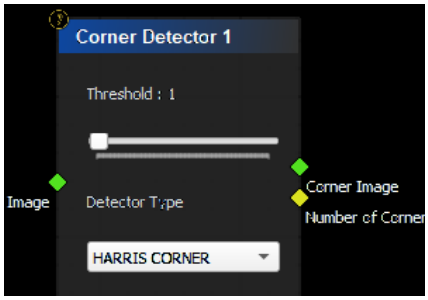
Gray ➔ RGB

4.13. Contrast-Brightness-Gamma



It gives output as a new image by changing the contrast, brightness and gamma values of the image given as input.

4.14. Corner Detector



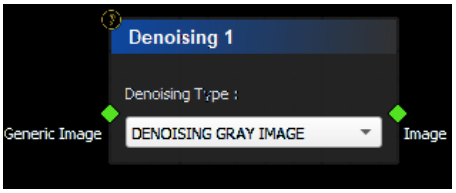
It includes methods to be used to detect corner points on images given as input.

Threshold değeri: Threshold value set to be used when detecting corner points

Detector Type:

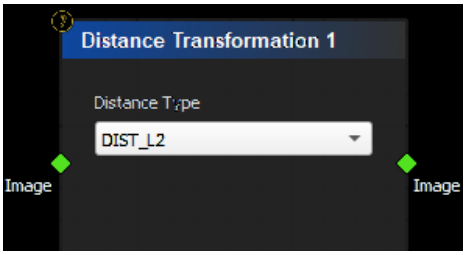
- **Harris Corner:** It is used to detect corners with a mathematical formula by detecting that the corners are in regions with large density differences in all directions in the image.
- **SHI-Thomas Corner:** Corners are detected by looking for a significant change in each direction. By moving a filter over the whole image, the places that make big changes in the image are described as corners.

4.15. Denoising



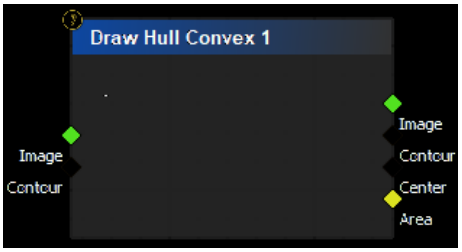
It makes smoothing by blocking the noises on the gray and colored image that it receives as input.

4.16. Distance Transformation



The value of each pixel on the image given as input is changed with its distance from the nearest background pixel, and a new image is output. Distance formula has 3 different variables that change.

4.17. Draw Hull Convex

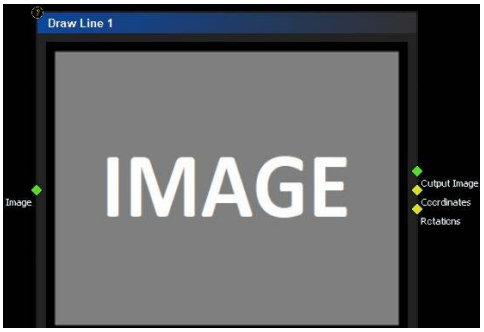


It draws a convex body by following the end points of a shape with the image given as input and the contours in that image.

The image on which the convex bodies are drawn, the contours, the center and area of the convex structure are output.

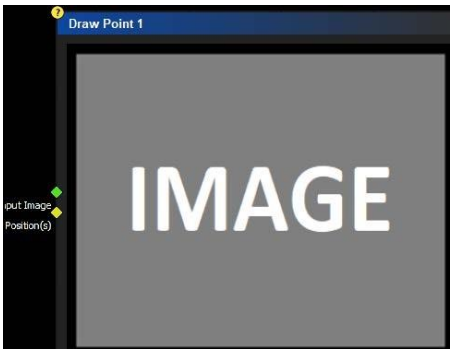
is given.

4.18. Draw Line



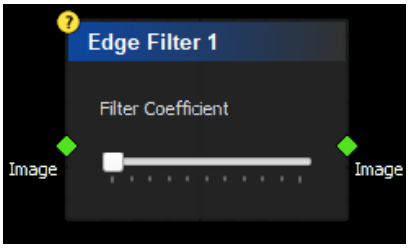
It is used to draw a line on the image given as input. The new image, the coordinates of the lines and the rotation information are output.

4.19. Draw Point



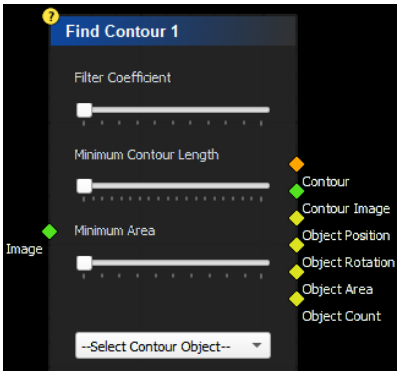
It is used to mark the coordinate points given as input on the image given as input.

4.20. Edge Filter



It is the algorithm used to find the edges on the image given as input. The edges to be found are filtered by changing the threshold value with the Filter coefficient setting.

4.21. Find Contour



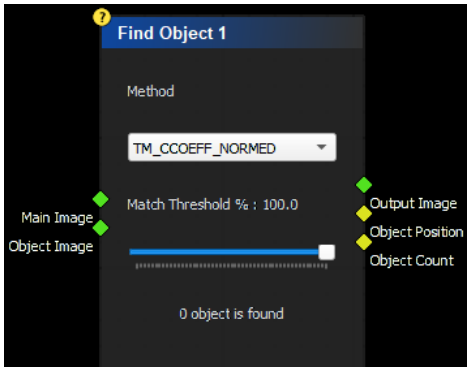
It creates a curve that connects all continuous points of the same color and density on the image given as input.

Variables on the function block;

- **Filter Coefficient:** The filter coefficient is a threshold value for continuous points.
- **Minimum Contour Length:** Smallest counter length
- **Minimum Area:** Contour areas found with the smallest area value created with the contours. Allows filtering.
- **Select Contour Object:** Allows switching between found contour entities. As output;
- Contours
- Image drawn with contours
- Position of the object surrounded by contours
- Rotation of the object surrounded by contours
- The area of the object surrounded by contours
- Number of objects surrounded by contours

is given.

4.22. Find Object

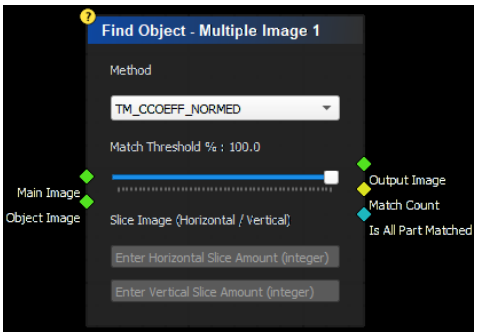


The main image and the image of the object to be included in this image should be given as input.

As output;

- Image where found objects are marked
- Positions of found objects on the image
- Number of found objects
  is given.

4.23. Find Object – Multiple Image

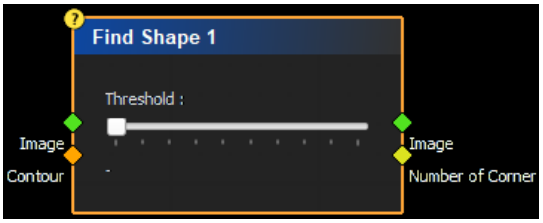


The main image and the image of the object to be included in this image should be given as input. It searches for the desired object in the multi-image by dividing the image into parts on the horizontal and vertical axes.

As output;

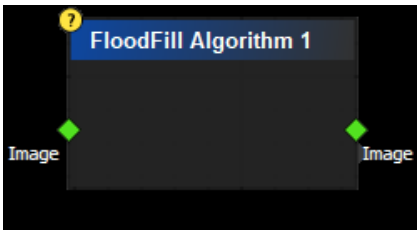
- Image where found objects are marked
- It gives the number of parts that the image is divided into.
- TRUE that the image can be divided exactly by the values given on the horizontal and vertical axis, or It outputs as FALSE.

4.24. Find Shape



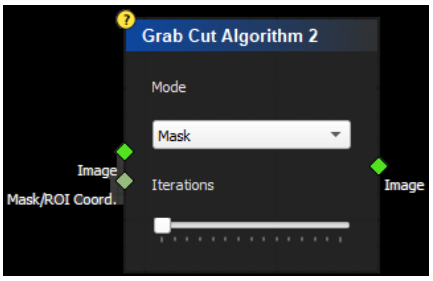
It is used to detect the corner points on the image with the image and its contours given as input. It outputs the number of vertices.

4.25. FloodFill Algorthm



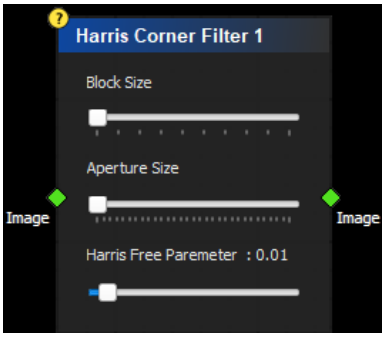
The Flood Fill algorithm is used to fill the connected area with a specific color and set different lower and upper limits of connected pixels. Usually a portion of the image is used for further processing or analysis.

4.26. Grab Cut Algorthm



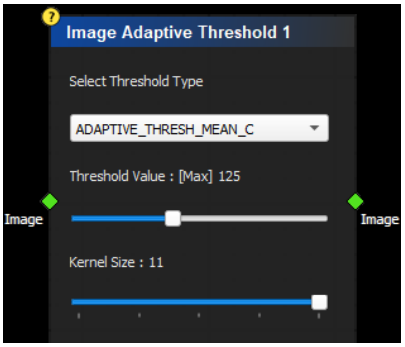
The image given as input and which area on the image will be defined as the foreground should also be given as a mask or ROI. The Grab Cut algorithm is used to separate the foreground on the image from the background.

4.27. Harris Corner Filter



It is used to find the vertices on the image given as input. The Harris Corner filter is used to detect corners with a mathematical formula by detecting that the corners are in regions with large density differences in all directions in the image.

4.28. Image Adaptive Threshold



Since the images have different lighting conditions in different areas, adaptive threshold is used. The adaptive threshold algorithm calculates the threshold value for small regions on the image and obtains different threshold values for different points on the same image.

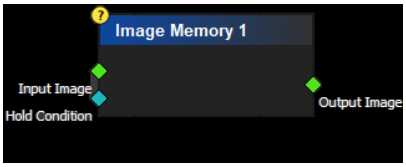
Mean: The threshold value is calculated according to the average of the determined regions.

Gaussian: The threshold value is calculated according to the weighted sum of the determined regions.

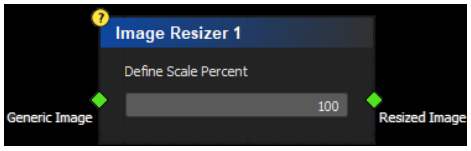
Threshold Value: The highest threshold value is set.

Kernel size: The size of the region on which different threshold values will be calculated on the image

4.29. Image Memory

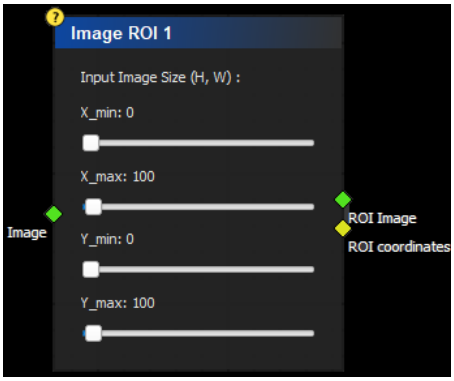


4.30. Image Resizer



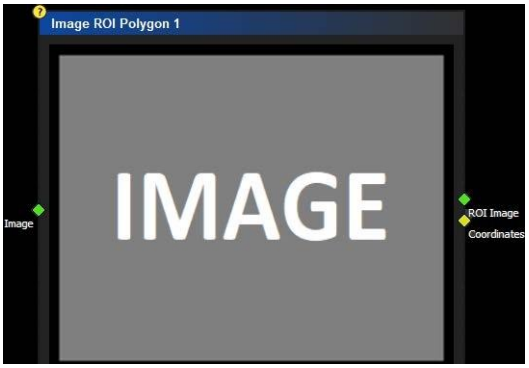
It is used to resize the image given as input. The image is resized according to the scale percentage and output is given.

4.31. Image ROI



An area is determined in the direction of the coordinates entered on the image given as input. The image and coordinates of the area determined as output are given.

4.32. Image ROI Polygon

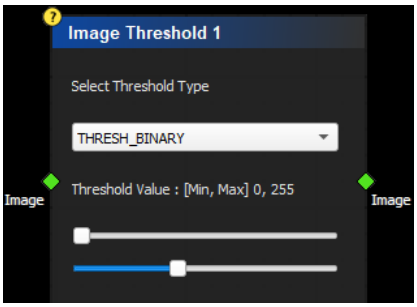


An area to be drawn on the image given as input and the coordinates of that area as output.
is given.

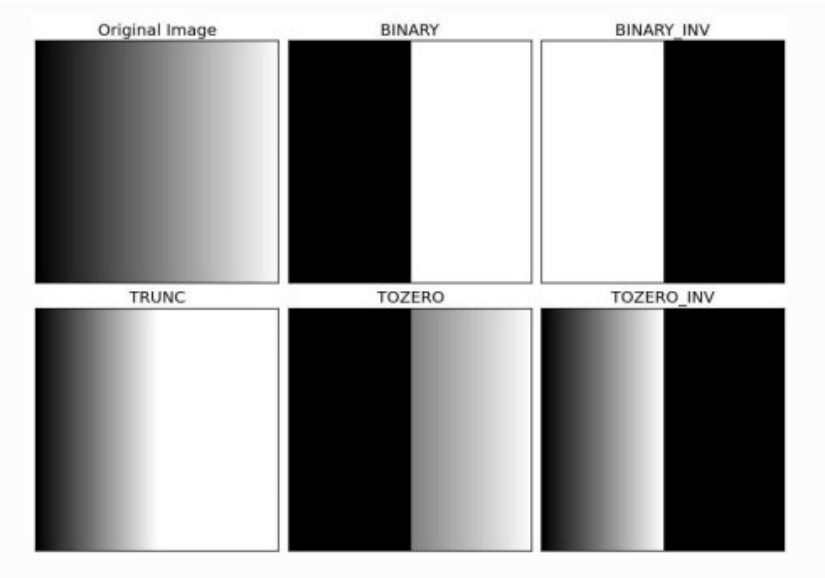
4.33. Image ROI Select

The area selected with the help of a rectangular frame on the image given as input and the coordinates of that area are given as output.

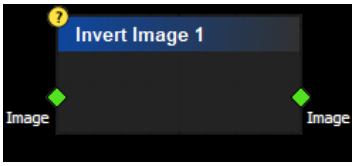
4.34. Image Threshold



If the pixel values in the image given as input are greater than the selected threshold value, it can be assigned a value as white, and if it is small, a value can be assigned as black.

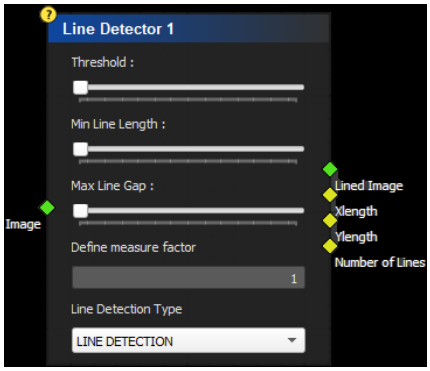


4.35. Invert Image



It gives the output by inverting the pixel values in the image given as input.

4.36. Line Detector

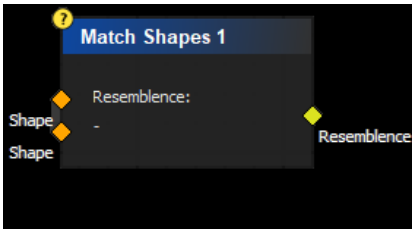


It is used for line detection on the image. The physical properties of the line can be taken with outputs.

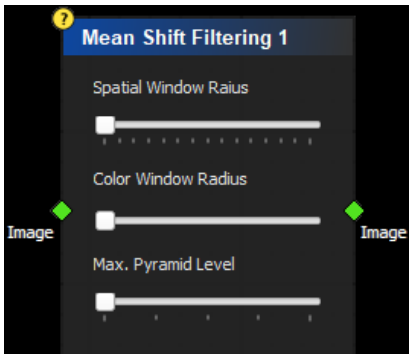
4.37. Load AI Model

It allows you to use the artificial intelligence models you have developed. Indicates which output the current object belongs to by taking the image.

4.38. Match Shapes

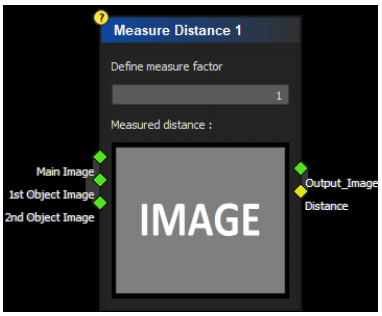


4.39. Mean Shift Filtering



Mean shift filtering is checking each instance of the video in the form of the pixel distribution in that frame. It is a method used to track a specific object within a video.

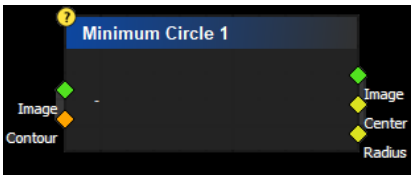
4.40. Measure Distance



In addition to the main image given as input, the image of two objects in the main image is used to find the distance between them.

Define measure factor: The measurement on the image is made with pixels. It is used to obtain the actual measurement value with the coefficient value to be entered here.

4.41. Minimum Circle



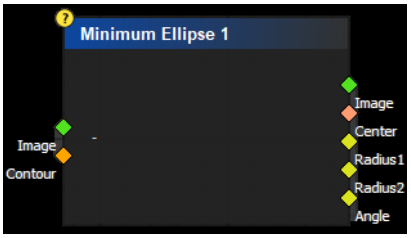
There is a minimum circle surrounding the contours with the image given as input and the contours on the image.

As output;

- New image with circle drawn
- Midpoint of the circle
- Radius of the circle

is given.

4.42. Minimum Ellipse



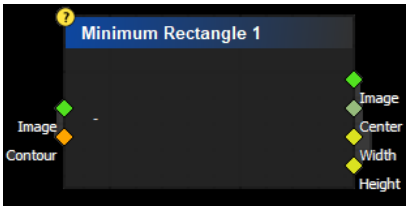
The image given as input and the contours on the image and the minimum ellipse surrounding the contours are available.

As output;

- New image drawn with ellipses
- Midpoint of the ellipse
- Radius of the ellipse
- Angle of ellipse

is given.

4.43. Minimum Rectangle



The image given as input and the contours on the image and the minimum surrounding the contours

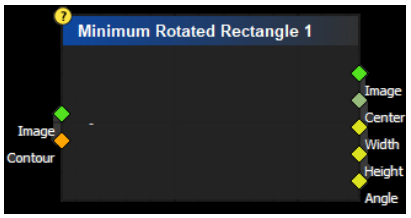
There is a rectangle.

As output;

- New image drawn with rectangle
- Midpoint of the rectangle
- Height and width

is given.

4.44. Minimum Rotated Rectangle



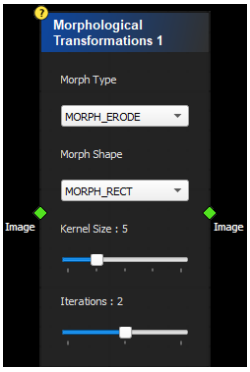
There is a rectangle with a minimum angle surrounding the contours with the image given as input and the contours on the image.

As output;

- New image drawn with rectangle
- Midpoint of the rectangle
- Height, width and angle of rotation

is given.

4.45. Morphological Transformations



Morphological transformations are simple operations based on image shape. There are two entries, the first is the original image and the second is the configuration item.

Always have the foreground white in binary images.

Kernel size: It is the kernel filter size of the morphological operation to be performed.

İteration: How many times the morphological process to be applied will be repeated in a row.

4.45.1. Erode

It erodes the boundaries of the foreground object. A pixel in the original image will be considered 1 only if all pixels below the core are 1 otherwise it will be eroded (Zero is made).

4.45.2. Dilate

It is the opposite of Erode. Increases the white area in the binary image or increases the size of the foreground object. It is used after erode in situations such as noise reduction

4.45.3. Open

Dilate is applied after erode. It is useful in eliminating noise.

4.45.4. Close

It is the opposite of Open. Erode is applied after dilate. Noise in foreground objects used to turn it off.

4.45.5. Gradient

It is the difference between dilated and eroded an image. The result will look like the outline of the object.

4.45.6. Tophat

It is the difference between input image and open.

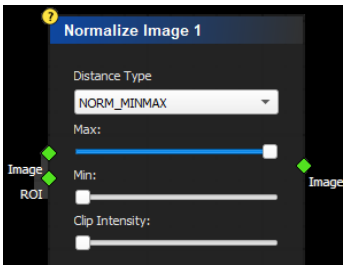
4.45.7. Blackhat

It is the difference between the input image and the close.

4.45.8. Hitmiss

It is used to find patterns in binary images.

4.46. Normalize Image

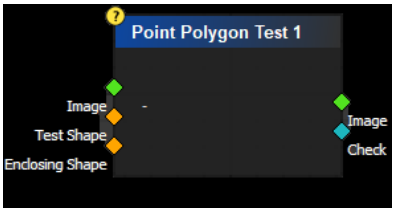


It is an operation that changes the range of pixel density values of the image given as input. It is used to remove the noise of the data in the image. It removes high frequency noise and very low frequency noise from the image.

4.47. Perspective Transform

Bird's eye view of the selected area by selecting 4 reference points on the image given as input
It is used to output the image and the image matrix.

4.48. Point Polygon Test

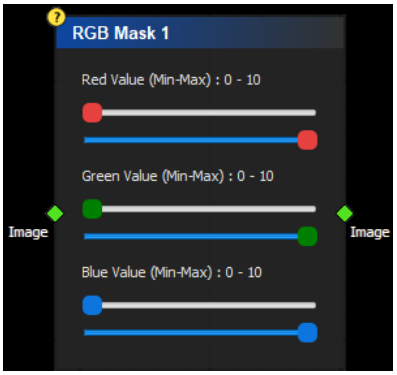


Finds the shortest distance between a point and a contour in the input image. Returns the distance as negative when the point is outside the contour, positive when the point is in the contour, and zero when the point is on the contour.

Input values;

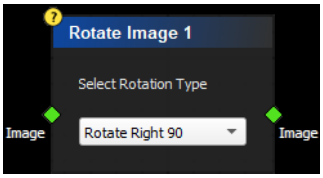
- Image
- Test Shape and Enclosing Shape: Different contour values on the image

4.49. RGB Mask



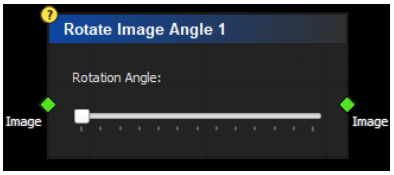
It is used to perform masking using the RGB (Red-Green-Blue) color space on the image given as input.

4.50. Rotate Image



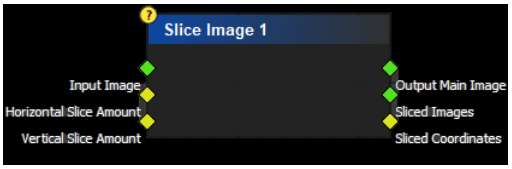
To the image given as input; It is used to rotate right and left 90 degrees, 180 degrees, x and y axis.

4.51. Rotate Image Angle



It is used to rotate the image given as input at the desired angle.

4.52. Slice Image



It divides the image into parts with the given values.

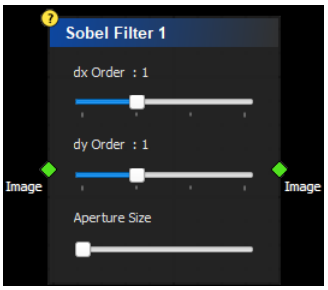
As input;

- **Input Image:** input image
- **Horizontal Slice Amount:** Amount to split image horizontally takes input as numbers
- **Vertical Slice Amount:** Amount to split image vertically takes input as numbers

As output;

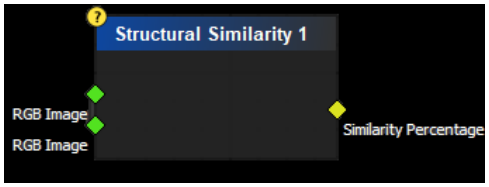
- **Output Main Image:** Main image divided into parts
- **Sliced Images:** All fragmented images
- **Sliced Coordinates:** Coordinates of parts

4.53. Sobel Filter



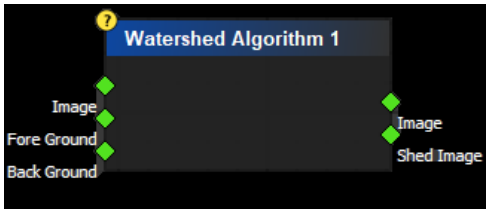
Sobel function is edge detection algorithm. It is a gradient-based method based on the first-order derivative on the image. Gradients can be calculated separately for each axis.

4.54. Structural Similarity



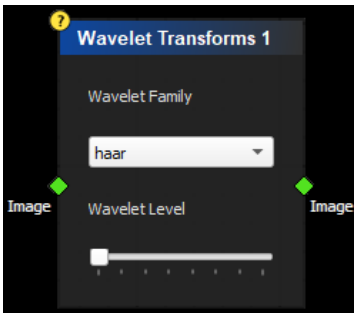
Calculate the percentage of similarity as output by comparing the similarities in the two color images given as input gives.

4.55. Watershed Algorithm



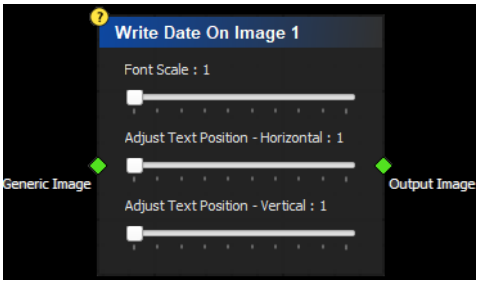
If we consider a grayscale image as a topographic surface, regions of high density represent peaks and peaks, while regions of low density represent valleys. Barriers are created between the foreground and background images and the valleys. Each isolated valley, ie local minimum points, begins to be filled with water. Filling continues until all peaks are submerged. The created obstacles give the result of segmentation.

4.56. Wavelet Transforms



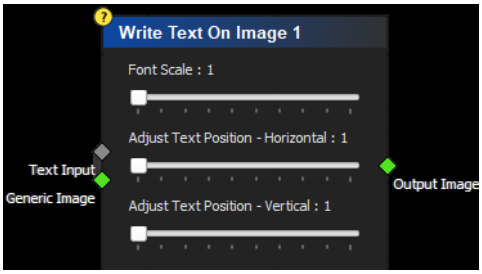
Wavelet Transform uses a set of functions called wavelets, each with a different scale. It works in the signal domain on the input image and gives an output by applying wavelet transform to each signal. Different wavelet transforms are available. Each type of wavelet has a different shape, smoothness and compactness and is useful for a different purpose.

4.57. Write Date On Image



To the image given as input; It is used to add dates in desired font size and coordinates.

4.58. Write Text On Image

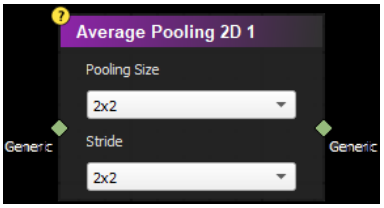


To the image given as input; It is used to add the text to be given as input in the desired font size and coordinates.

5. Derin Öğrenme Model Blokları

5.1. AI Model Elements

5.1.1. Average Pooling 2D



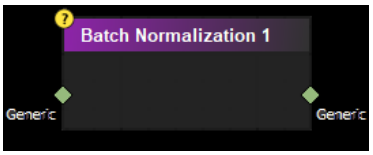
Average pooling layer; it reduces the size of the data, the number of parameters, the amount of computation required and checks for overfitting. It is similar to reducing the size of an image.

This function block generates a tensor smaller than its input. It means that subsequent blocks need less parameters and amount of computation. It determines the average value among the pixels covered by the applied filter and uses it as the pixel value in the newly created matrix.

Pooling size: It is the filter size to be pooled.

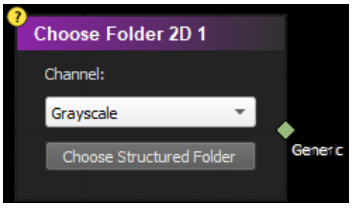
Stride: It is the number of steps (pixels) to be skipped while applying the filter.

5.1.2. Batch Normalization



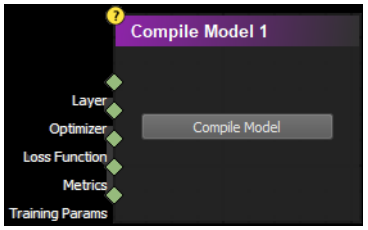
The batch normalization function block is a technique for training very deep neural networks that standardize the inputs to one layer for each mini-group. It significantly reduces the number of train epochs required to stabilize the learning process and train deep networks.

5.1.3. Choose Folder 2D



It is used to select and import the Dataset folder. Images to be imported can be set to gray or color.

5.1.4. Compile Model

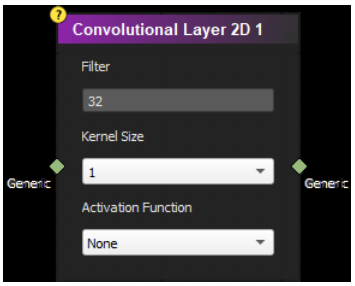


When the deep learning model is ready, it is used to structure the learning process.

Input values;

- **Layer:** Introduction for layers to be used during training
- **Optimizer:** Introduction for the optimizer to use
- **Loss Fonction:** Loss function input to be used by the model to list losses during training and testing
- **Metrics:** The accuracy rate of the measurements will be listed by the model during training and testing metrics function
- **Training Params:** An introduction to give the necessary parameters for the training of the model.

5.1.5. Convolutional Layer 2D



The Convolutional layer block is one of the deep learning layers. It is responsible for perceiving the features of an image. It applies filters to the image to extract low- and high-level features from the image.

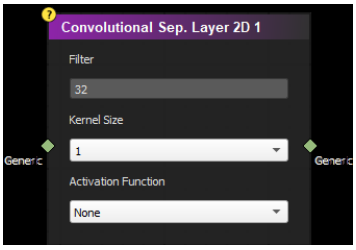
These filters are usually multidimensional. It has depth, width and height. Since the image has 3 channels, the convolutional kernel must also have 3 channels. In other words, when trying to perform convolution with a 5x5 filter, this filter only takes the width and height of the image. In a complete convolution operation, 5x5x3, that is, convolution is performed for each channel at the same time.

Filter: Specifies the number of filters to apply to the image.

Kernel size: Specifies the filter size.

Activation Function: Specifies the activation function to be applied after the convolution operation.

5.1.6. Convolutional Sep. Layer 2D



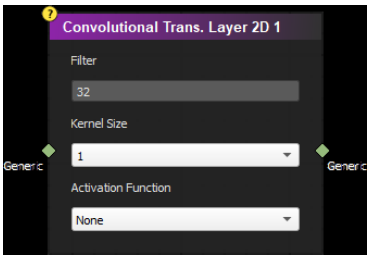
Also called Deptwise Seperable Convolution. A convolution is given to the input image without changing the depth. This process is done using 3 filters in the form of 5x5x1. Then, the images with the convolution process are brought together to create a 5x5x3 image.

Filter: Specifies the number of filters to apply to the image.

Kernel size: Specifies the filter size.

Activation Function: Specifies the activation function to be applied after the convolution operation.

5.1.7. Convolutional Trans. Layer 2D



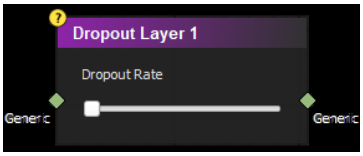
Also called Pointwise Transformed Convolution. It is so named because it uses a 1x1 filter or an all-point filter. This filter has the depth of many channels that the input image has. To obtain a 5x5x1 image, a 1x1x3 filter is repeated over a 5x5x3 image.

Filter: Specifies the number of filters to apply to the image.

Kernel size: Specifies the filter size.

Activation Function: Specifies the activation function to be applied after the convolution operation.

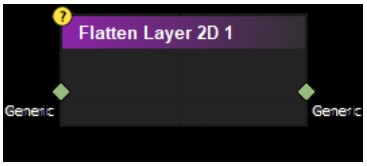
5.1.8. Dropout Layer



Dropout layer to forget some neurons to prevent overfitting during training is used.

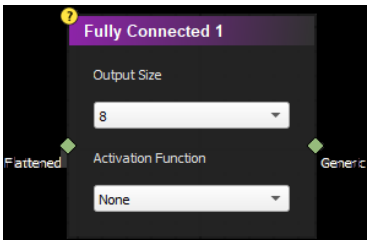
Dropout Rate: Allows to set the percentage of Neurons to Forget.

5.1.9. Flatten Layer 2D



Generally, neural networks receive input data as a one-dimensional array. Data in flatten layer
On the other hand, it is the one-dimensional array of the matrices from the Convolutional and Pooling layers.

5.1.10. Fully Connected

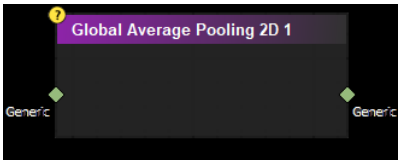


The Fully Connected layer is the last layer of the deep learning neural network. It receives the data from the Flatten Layer and performs the learning process through the neural network. Returns probabilities for label values used in classification.

Output size: Specifies the number of layers to use in the final layer.

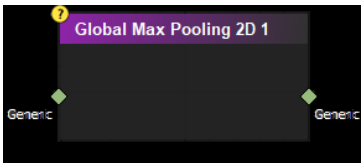
Activation Function: Activation that will perform the classification in the last layer, Fully Connected function is selected.

5.1.11. Global Average Pooling 2D



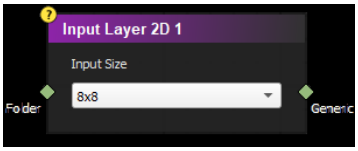
The Global Average Pooling layer is a process that calculates the average output of each feature map in the previous layer. This process significantly reduces data and prepares the model for the final classification layer. No parameter input is needed.

5.1.12. Global Max Pooling 2D



The Global Max Pooling layer is a process that calculates the maximum output of each feature map in the previous layer. This process significantly reduces data and prepares the model for the final classification layer. No parameter input is needed.

5.1.13. Input Layer 2D

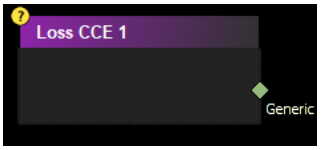


It is the function block where the input data to the deep learning

model is given.

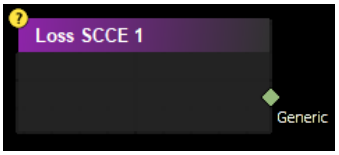
Input Size: Allows to set the size of the input data.

5.1.14. Loss CCE



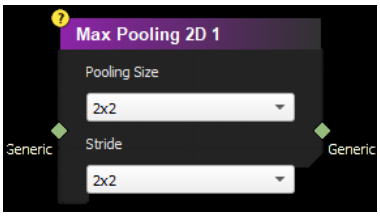
Categorical Cross Entropy Loss includes Softmax activation plus a Cross Entropy Loss. Calculates loss by classifying each image. It is used for multi-class classification. Classes are vector coded. E.g; for a three-class classification: [1,0,0], [0,1,0], [0,0,1]

5.1.15. Loss SCCE



Sparse Categorical Cross Entropy Loss calculates loss by classifying each image. It is used for multi-class classification. The advantage of using sparse entropy loss is that it saves memory and computation time. Because it uses integers instead of vectors for a class. E.g; for a three-class classification: [1], [2], [3]

5.1.16. Max Pooling 2D



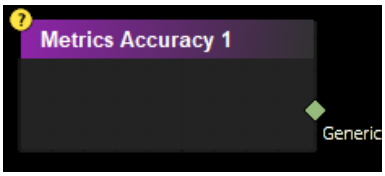
The Max Pooling layer reduces the size of the data, the number of parameters, the amount of computation required, and checks compatibility. It is similar to reducing the size of an image.

This function block generates a tensor smaller than its input. It means that subsequent blocks need less parameters and amount of computation. It determines the maximum value among the pixels covered by the applied filter and uses it as the pixel value in the newly created matrix.

Pooling size: It is the filter size to be pooled.

Stride: It is the number of steps (pixels) to be skipped while applying the filter.

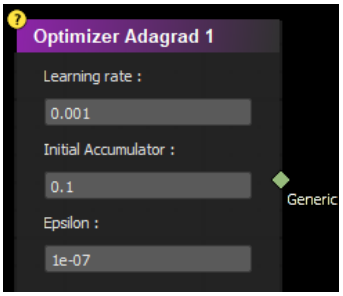
5.1.17. Metrics Accuracy



Accuracy is the metric that is widely used to measure the success of a model but does not appear to be

sufficient on its own. Accuracy value is calculated by the ratio of correctly predicted areas in the model to the total data set.

5.1.18. Optimizer AdaGrad



AdaGrad makes large updates for infrequent parameters, and smaller updates for frequent parameters. It is therefore more suitable for sparse data such as NLP and image recognition.

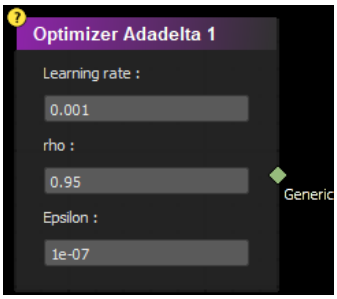
The biggest disadvantage of AdaGrad is that each parameter has its own learning rate and the learning rate is gradually decreasing depending on the characteristics of the algorithm. Therefore, at some point in time, the system stops learning.

Learning rate: It is the learning rate. The rate at which the weights are updated (eg 0.001). Larger values allow faster learning (eg 0.3). Smaller values slow down learning during training (eg 1e-5).

Initial Accumulator: It is the initial value for the accumulator. It should not be negative.

Epsilon: It is a very small number that is used to avoid any division by zero when performing the optimizer operation.

5.1.19. Optimizer Adadelta



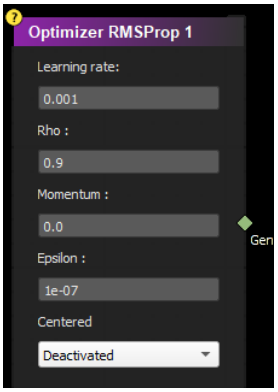
AdaDelta is an extension of AdaGrad. It prevents the rapid decrease in learning rate by solving the disadvantage in AdaGrad.

Learning rate: It is the learning rate. The rate at which the weights are updated (eg 0.001). Larger values allow faster learning (eg 0.3). Smaller values slow down learning during training (eg 1e-5).

rho: Exponential decay rate of first and second order moments

Epsilon: It is a very small number that is used to avoid any division by zero when performing the optimizer operation.

5.1.20. Optimizer RMSprop



RMSprop very simply adjusts the AdaGrad method to reduce the aggressive and monotonously decreasing learning rate.

Learning rate: It is the learning rate. The rate at which the weights are updated (eg 0.001). Larger values allow faster learning (eg 0.3). Smaller values slow down learning during training (eg 1e-5).

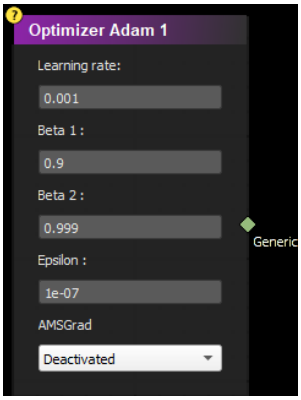
Rho: Reduction factor for past/future gradient

Momentum: momentum factor

Epsilon: A very small number used to prevent any division by zero when performing the optimizer operation.

Centered: Takes a boolean value. If TRUE, it is normalized by estimating the gradient variance.

5.1.21. Optimizer Adam



Adam or adaptive momentum is an algorithm similar to AdaDelta. Unlike AdaDelta, it caches the momentum changes as well as the learning rates of each of the parameters; ie it combines RMSprop and momentum.

Learning rate: It is the learning rate. The rate at which the weights are updated (eg 0.001). Larger values allow faster learning (eg 0.3). Smaller values slow down learning during training (eg 1e-5).

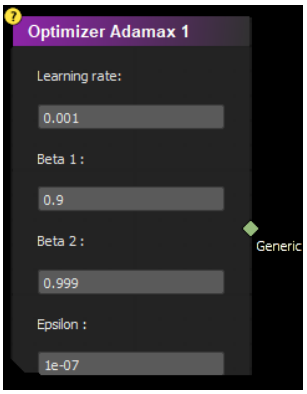
Beta 1: Exponential decay rate for first-moment predictions

Beta 2: Exponential decay rate for second-moment predictions

Epsilon: It is a very small number that is used to avoid any division by zero when performing the optimizer operation.

AMSGrad: Whether or not the AMSGrad variant of this algorithm will be used is selected. Takes a boolean value.

5.1.22. Optimizer Adamax



Adamax is a variant of Adam based on the infinity norm.

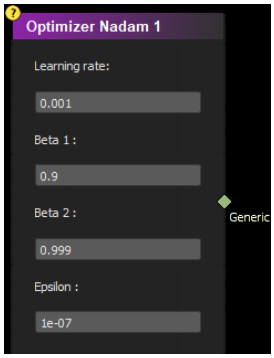
Learning rate: It is the learning rate. The rate at which the weights are updated (eg 0.001). Larger values allow faster learning (eg 0.3). Smaller values slow down learning during training (eg 1e-5).

Beta 1: Exponential decay rate for first-moment predictions

Beta 2: Exponential decay rate for second-moment predictions

Epsilon: It is a very small number that is used to avoid any division by zero when performing the optimizer operation.

5.1.23. Optimizer Nadam



It is a combination of Nadam, Adam and NAG methods.

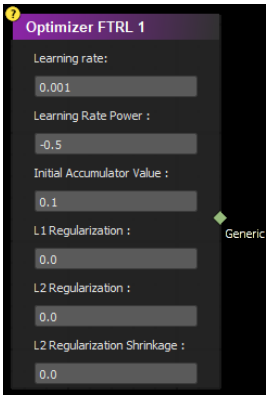
Learning rate: It is the learning rate. The rate at which the weights are updated (eg 0.001). Larger values allow faster learning (eg 0.3). Smaller values slow down learning during training (eg 1e-5).

Beta 1: Exponential decay rate for first-moment predictions

Beta 2: Exponential decay rate for second-moment predictions

Epsilon: It is a very small number that is used to avoid any division by zero when performing the optimizer operation.

5.1.24. Optimizer FTRL



Learning rate: It is the learning rate. The rate at which the weights are updated (0.001). Larger values allow faster learning (0.3). Smaller values slow down learning during training (1e-5).

Learning rate Power:

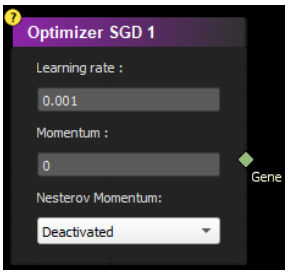
Initial Accumulator Value:

L1 Regularization:

L2 Regularization:

L2 Regularization Shrinkage:

5.1.25. Optimizer SGD



The optimization algorithm generally used by default in deep learning models is Stochastic Gradient Descent (SGD).

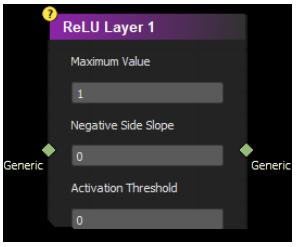
In the SGD algorithm, calculation is made on a training sample instead of the entire training data. In this way, memory problems that may occur are prevented. However, the SGD algorithm works slower than adaptive algorithms.

Learning rate: It is the learning rate. The rate at which the weights are updated (0.001). Larger values allow faster learning (0.3). Smaller values slow down learning during training (1e-5).

Momentum: momentum factor

Nesterov Momentum: Nesterov activates the momentum.

5.1.26. ReLU Layer



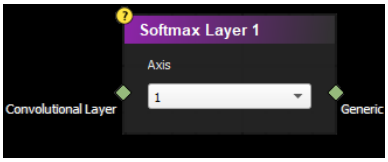
Rectified Linear Unit (ReLU) is an activation function. It takes values in the range from zero to infinity. Taking the value of zero on the negative axis makes the network work faster. It is preferred in multi-layer networks.

Maximum Value: The saturation threshold is the largest value that the function will return.

Negative Side Slope: Slope for values below the threshold

Activation Threshold: It is the threshold value of the activation function at which values below the threshold will be damped or set to zero.

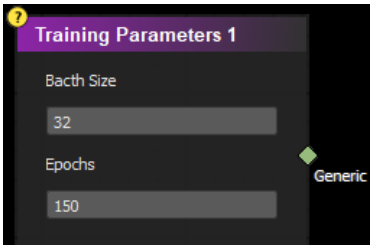
5.1.27. Softmax Layer



Softmax function is preferred as a classifier in the output layer of deep learning models. It determines the probability that the input belongs to a certain class by generating values in the range of 0-1. That is, it performs a probabilistic interpretation.

Axis: Axis to which Softmax normalization is applied

5.1.28. Training Parameters



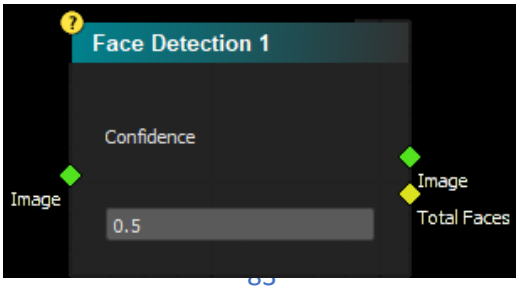
Training Parameters;

Batch size: It is the number of sub-samples given to the network where the parameter update takes place.

Epochs: Cycle count is the number of times all training data is exposed to the network during training.

5.2. AI Applications

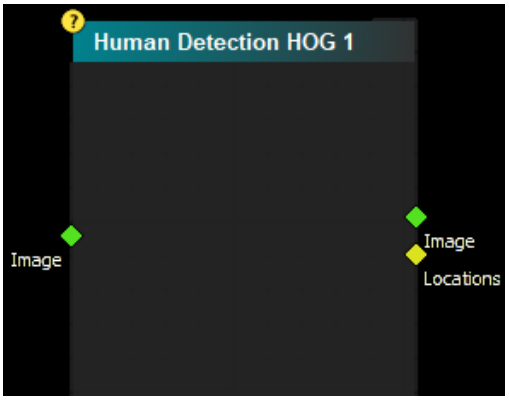
5.2.1. Face Detection



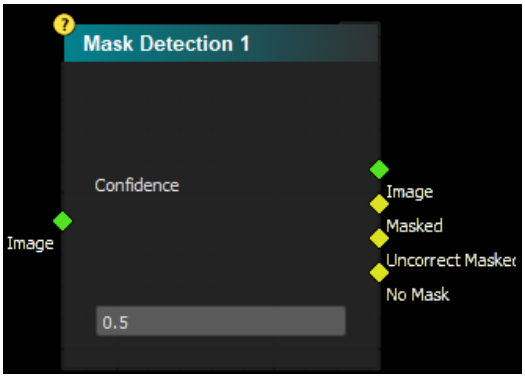
The Face Detection block contains a deep learning module trained with AugeLab Studio to detect the human face on the input images.

It gives the image in which the detected faces are marked as output and the number of detected faces.

5.2.2. Human Detection HOG



5.2.3. Mask Detection



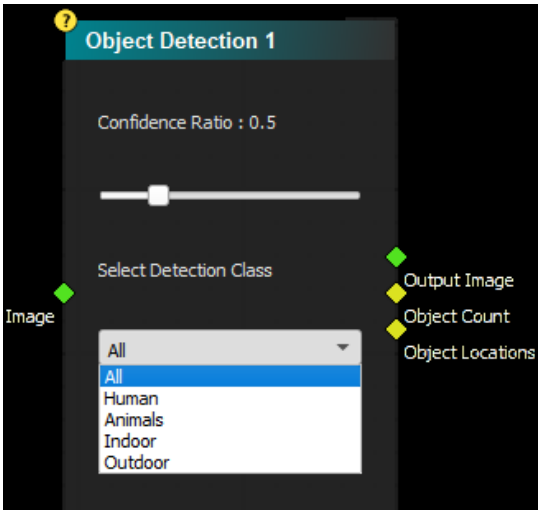
The Mask Detection block contains a deep learning model trained with AugeLab Studio to detect whether people in the input images use medical masks or misuse them.

As output,

- Number of masks used correctly
- Number of masks used incorrectly
- It gives the number of people without masks.
-

5.2.4. Mood Detection

5.2.5. Object Detection

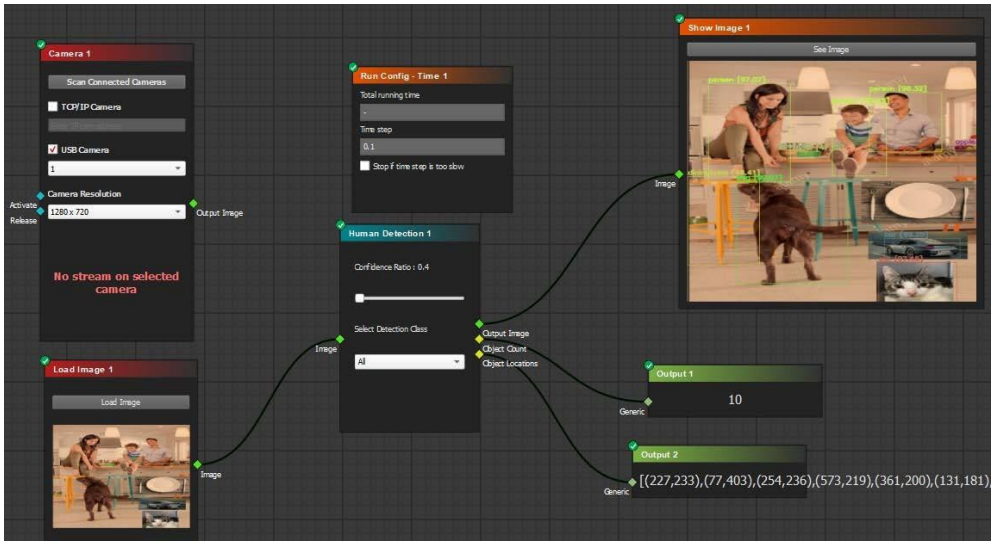


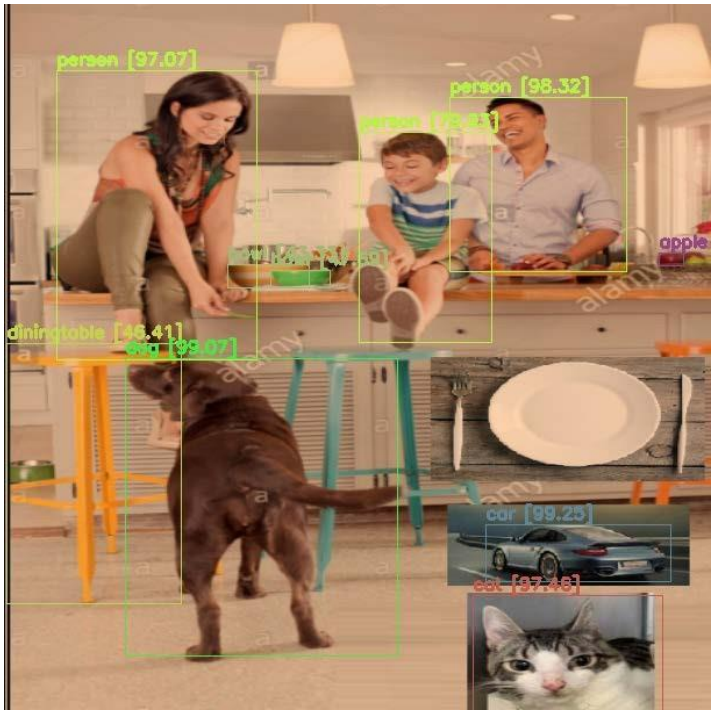
Object Detection is a deep learning model trained to detect and identify objects in an image or camera. There is a trained model block in AugeLab Studio.

Select Detection Class: It allows to detect objects in different categories such as all objects, people, animals, indoor and outdoor.

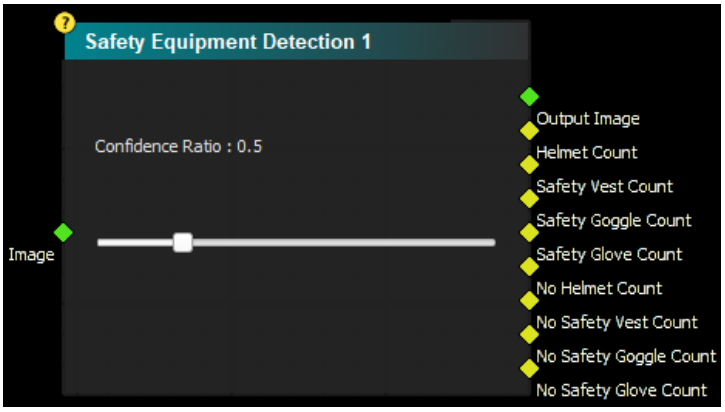
As output;

- The image where the objects are detected and marked and the detected category is written
- Number of detected objects
- It gives the positions of the detected objects on the image





5.2.6. Safety Equipment Detection

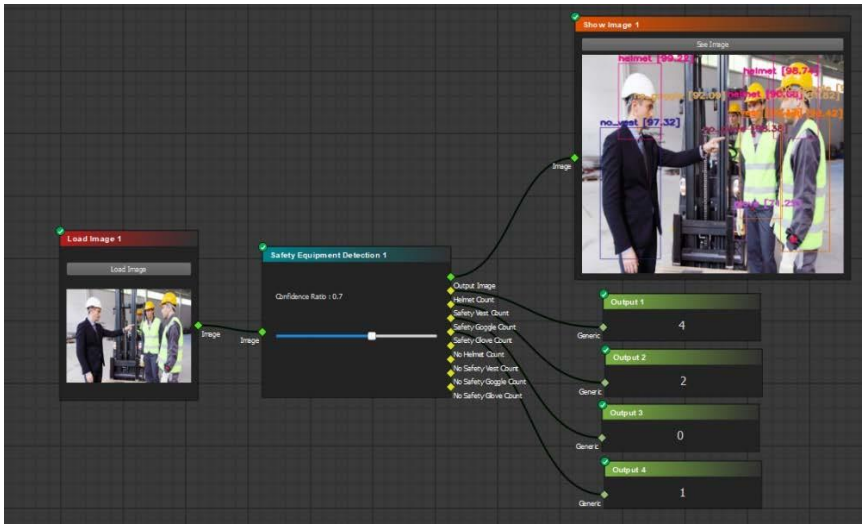


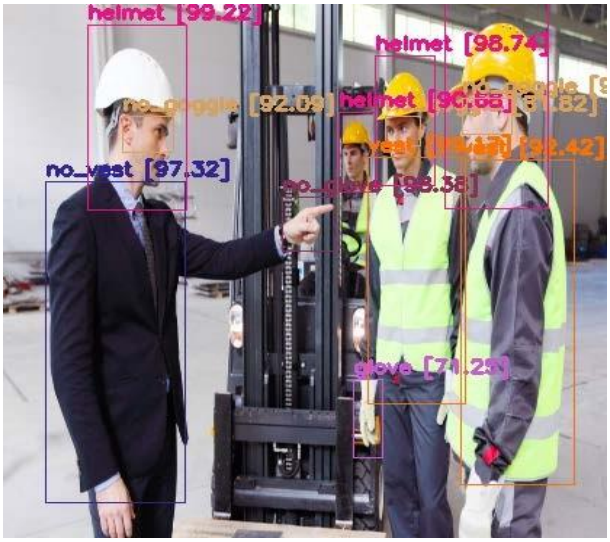
Safety Equipment Detection is a deep learning model trained using AugeLab Studio to detect whether the safety equipment that should be used while working within the scope of occupational health and safety is on working people.

As output;

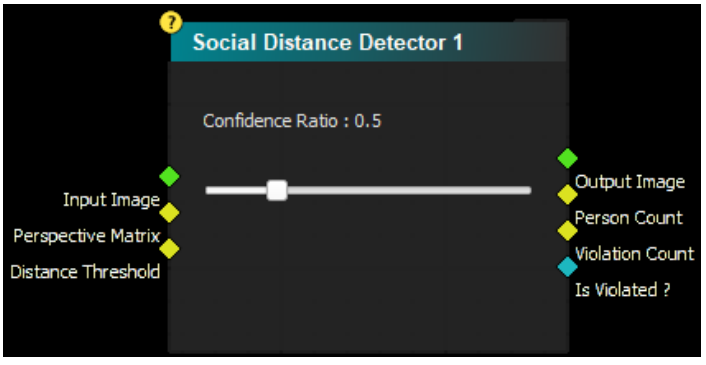
- The image where the detected classes on the image are marked and categorized
- Detected security on the image; number of helmets, vests, goggles and gloves
- Number of people not using the security equipment detected on the image

Is given.





5.2.7. Social Distance Detector



The Social Distance Detector block includes a deep learning model trained with AugeLab Studio to classify people on images as either violating or practicing social distancing.

As input;

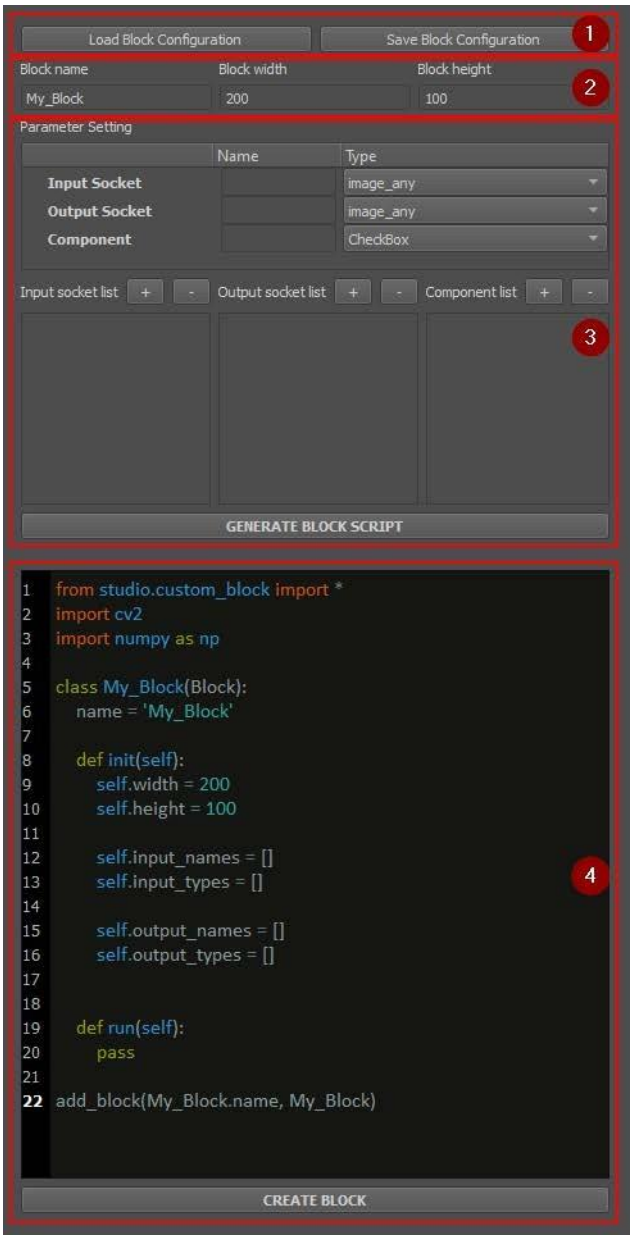
- Camera view to be supervised for social distancing
- Perspective matrix of the camera image
- The threshold value of social distance must be entered.

As output;

- Image where social distancing is controlled
- Number of people detected
- The number of people violating social distance is given.

6. DESIGNER WINDOW

The designer window section in Studio allows users to prepare custom function blocks. The window consists of 4 main parts.

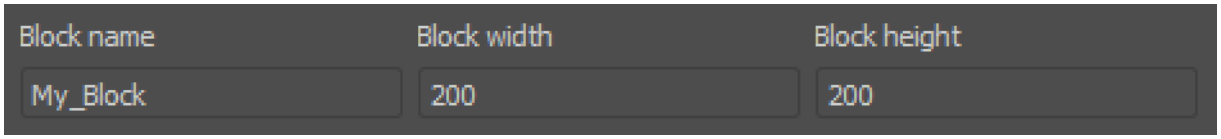


1. Configuration section
2. Function block properties section
3. Adding function block parameters section
4. Function block source code editing section

6.1. Konfigurasyon Bölümü

In the configuration section, there are two buttons to save the information already entered in the designer window and to load the file containing the saved information into the designer window. With the help of these buttons, you can reuse the information you entered later, make changes and save it again.

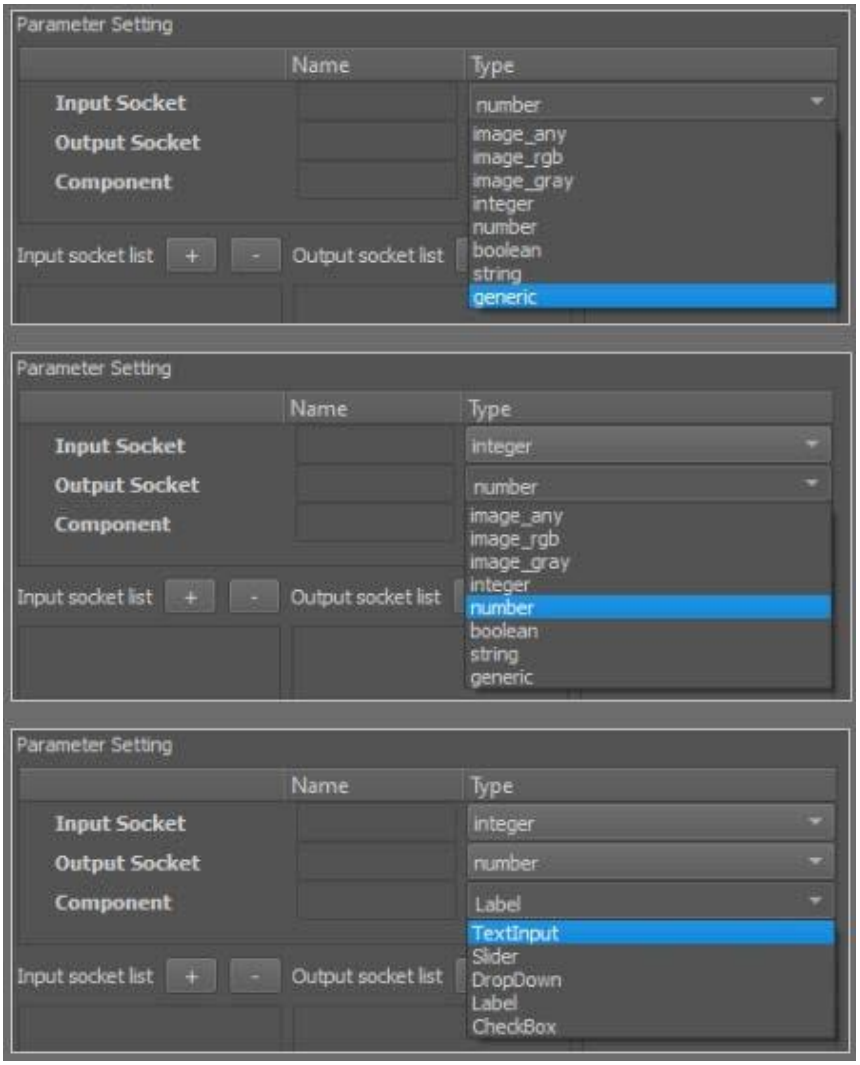
6.2. Fonksiyon Blok Özellikleri Bölümü



The name, width and height of the function block to be created can be adjusted in the block properties section.

6.3. Fonksiyon Blok Parametreleri Ekleme Bölümü

Block parameters are basically divided into three parts as input, output and component each has its own data types..



Types of Input-Output :

1) image_any
   - general photo data, whether single-channel(gray) or three-channel (red,green,blue)
2) image_rgb
   - three channels (red,green,blue) photo data
3) image_gray
   - single channel (gray) photo data
4) integer
   - integer
5) number
   - general number format that can also use decimals
6) boolean

- 0(true) and 1(false)

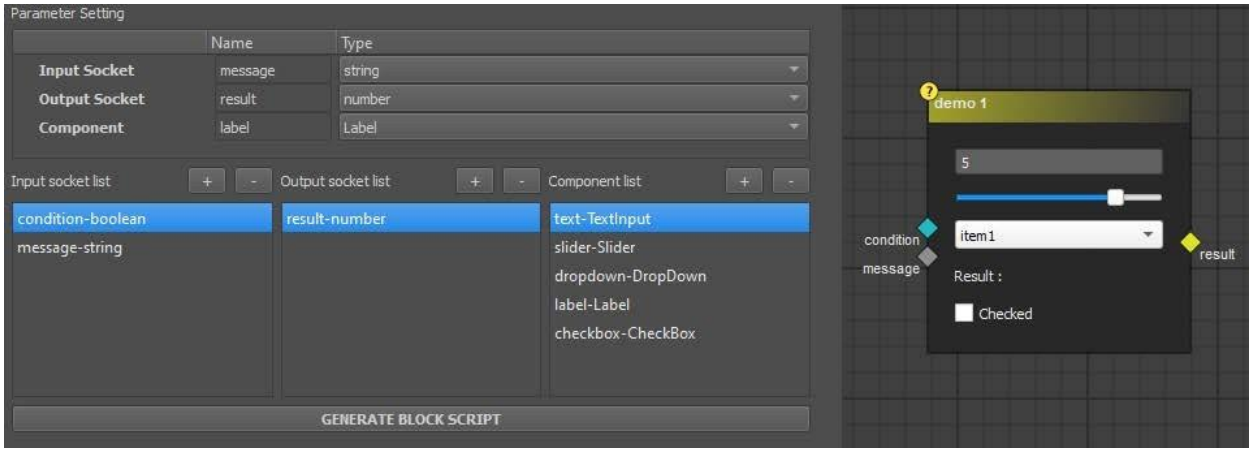
7) string
   - string of characters (Word or set of words)
8) generic
   - general format covering all formats

Component types:

1) TextInput
   - Used as text parameter
2) Slider
   - The min and max value are used as certain variable number parameters
3) DropDown
   - Used for selectable parameters in list form
4) Label
   - Used to display any value on the block
5) CheckBox
   - Used for 0-1 (boolean) parameter.

After the names and types of the components are determined, adding or removing the components to the list can be made with the (+) (-) buttons, and the order of the added components can be changed by dragging them within the list. Blocks are created according to the components in these lists.

After all parameters are determined and added to the relevant lists, the block source code is created according to the selected parameters with the "GENERATE BLOCK SCRIPT" button and written in the source code editing section.

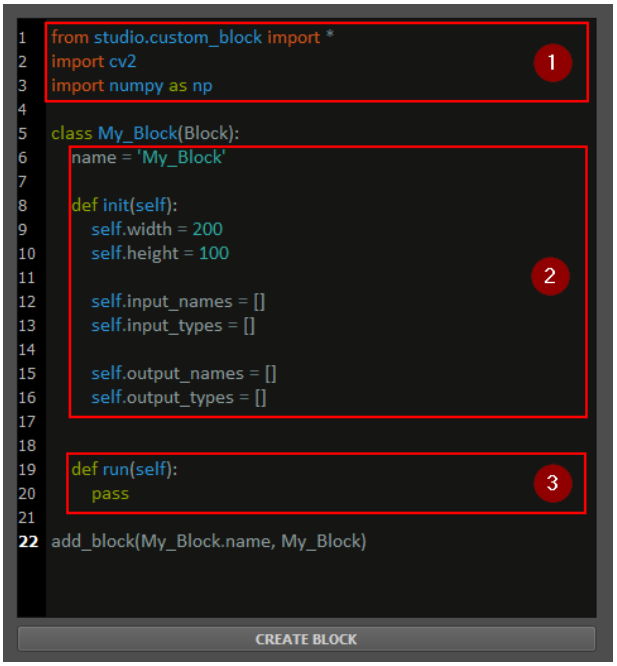


6.4. Blok kaynak kodu düzenleme bölümü

The user can make changes that will constitute the main working logic of the block, such as the properties of the parameters, the operations to be performed on the input data, and the creation of the output data, through the source code created in this section.

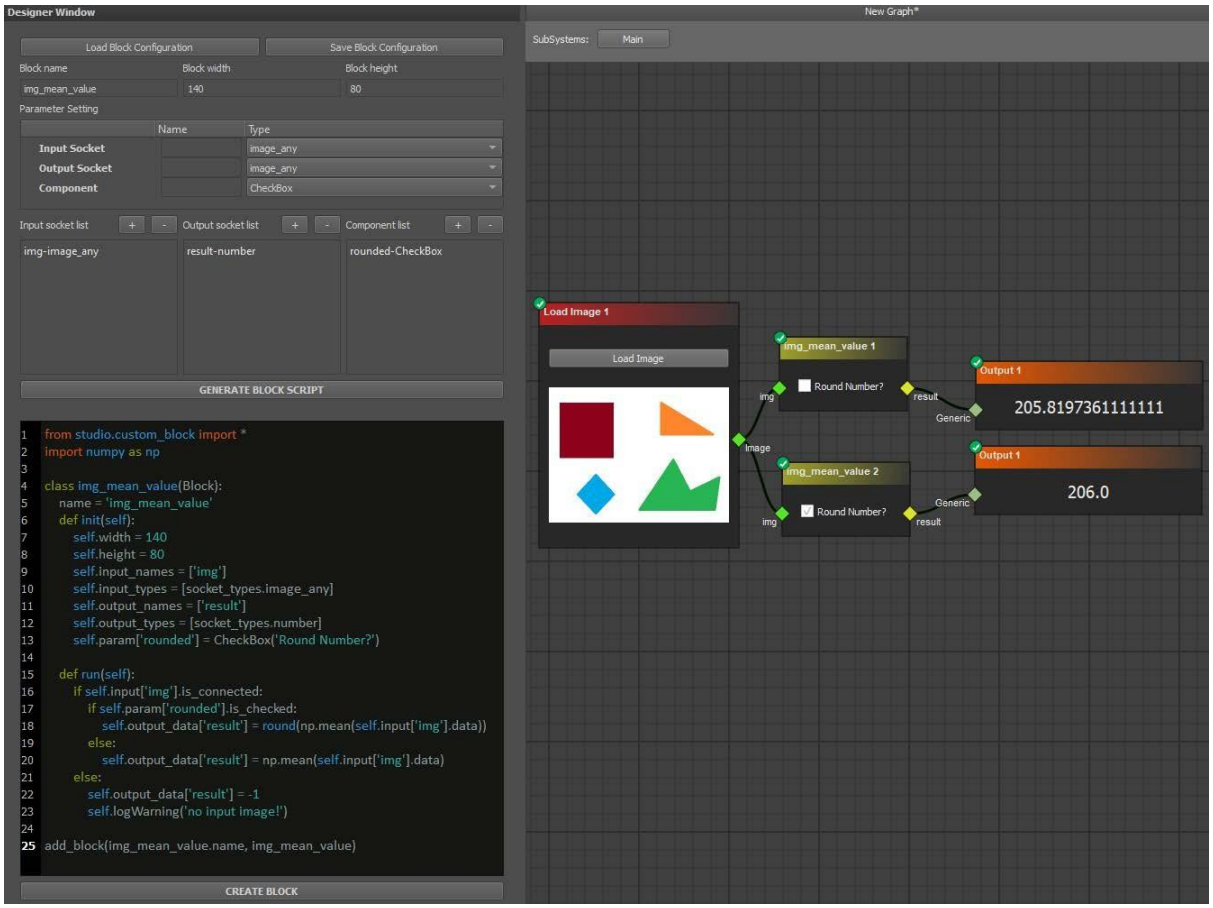
This section consists of 3 main parts.

1. Library installation
   - In this section, the python programming language that should be used in the algorithm part libraries are loaded
   - For the use of extra libraries and their integration into Studio, see the "Import Package Window" section.
2. Initial settings
   - In this section, one-time settings that must be made in the first stage when the block starts to work are made. (example: parameter settings)
3. The main algorithm
   - The algorithm containing the basic working logic of the block is created in this section.



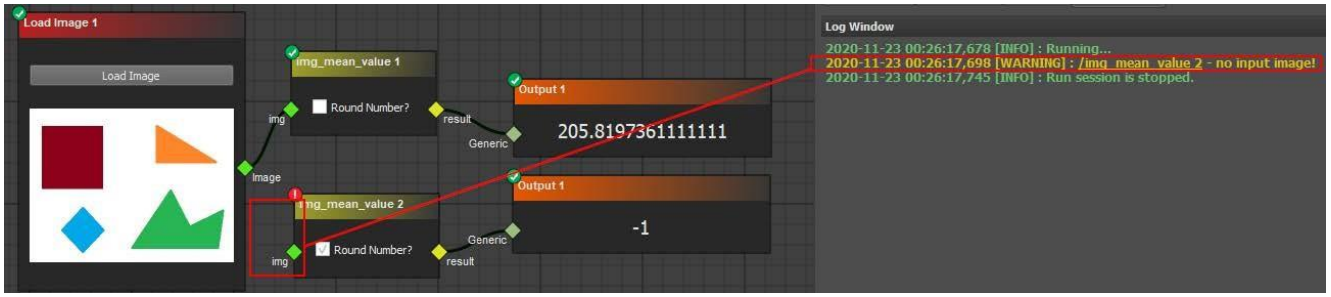
As an example usage below, the average pixel value of the picture given as input is calculated.

6.4.1. Block source code example




```
def run(self):
    if self.input['img'].is_connected:
        if self.param['rounded'].is_checked:
            self.output_data['result'] = round(np.mean(self.input['img'].data))
        else:
            self.output_data['result'] = np.mean(self.input['img'].data)
    else:
        self.output_data['result'] = -1
        self.logWarning('no input image!')
```

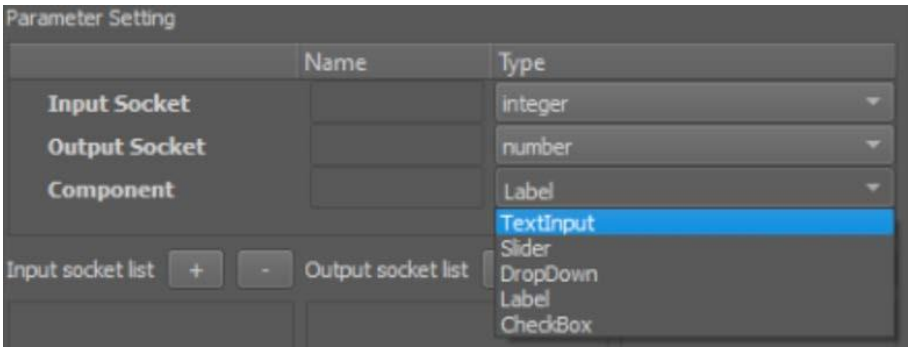
Input data connected to the input (socket) of the block is taken as "self.input['socket_name'].data". Checking whether the socket is connected or not is provided as "self.input['socket_name'].is_connected". With this control, error messages can be printed in the LOG WINDOW part of the Studio. Self.logError, self.logWarning, self.logInfo commands can be used for error messages.



For the creation of output data, it is sufficient to make a definition similar to "self.output_data['socket_name'] = 99". The point to be noted is that the data defined as output_types must be suitable for the relevant socket type. For example, in order to make a definition like "self.output_data['socket_name'] = 99", the relevant socket must be defined as socket_types.number or socket_types.integer.

6.4.2. Component usage

As described in the previous sections, there are 5 types as component types.



The variables required for the use of these components on the source code are as follows.

6.4.2.1. TextInput

Describing

self.param[‘component_ismi’] = TextInput(‘örnek text’)

property

text1 = self.param[‘component_ismi’].text

6.4.2.2. Slider

Describing

self.param[‘component_ismi’] = Slider(-10, 10, 1)

Property

slider1 = self.param[‘component_ismi’].value

self.param[‘component_ismi’].setMinimum(-100)

self.param[‘component_ismi’].setMaximum(100)

self.param[‘component_ismi’].setValue(10)

6.4.2.3. DropDown

Describing

self.param[‘component_ismi’] = DropDown([‘toplama’, ‘çıkarma’, ‘çarpma’])

Property

item1 = self.param[‘component_ismi’].selected_item

id1 = self.param[‘component_ismi’].selected_index

6.4.2.4. Label

Describing

self.param[‘component_ismi’] = Label()

Property

self.param[‘component_ismi’].set_text(‘Sonuç’)

6.4.2.1 CheckBox

Describing

self.param[‘component_ismi’] = CheckBox(‘Sayıyı yuvarla?’)

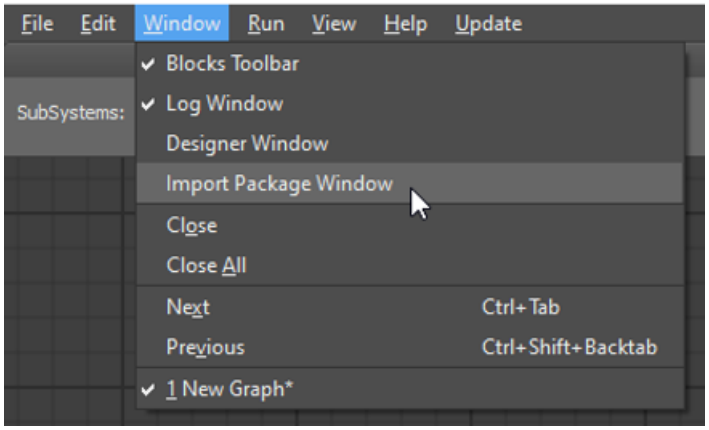
Property

check1 = self.param[‘component_ismi’].is_checked

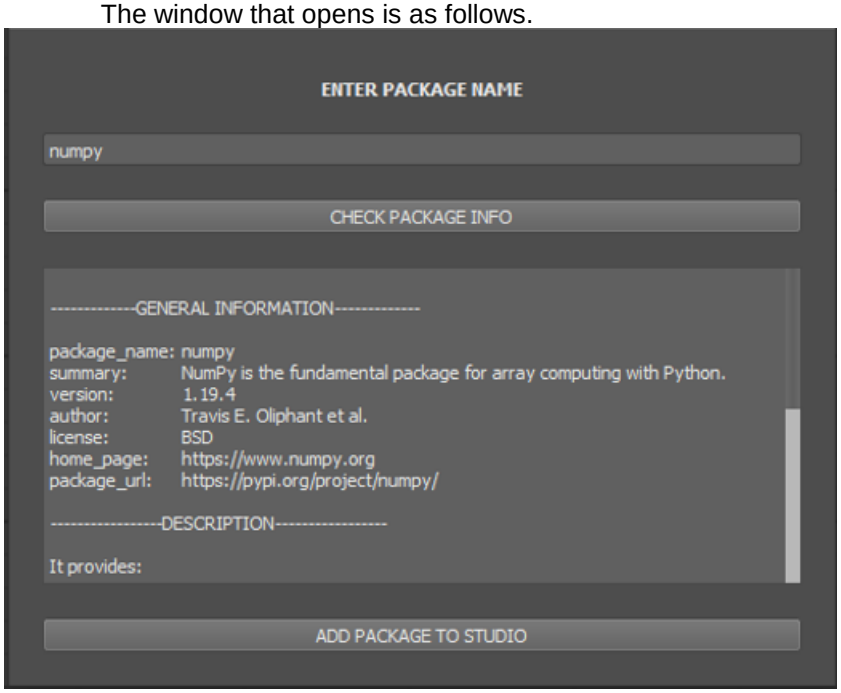
7. IMPORT PACKAGE WINDOW

In this window, it is possible to install python libraries that are not included in Studio, but that are desired to be used for blocks created with DESIGNER WINDOW.

To open the window, go to the main window of Studio from the menu section in the Window section.
Just click "Import Package Window".



The window that opens is as follows.



The name of the library to be loaded is written in the input section and the information about the library is transferred to the window with the "CHECK PACKAGE INFO" button. After checking from here, the library loading process is started with the "ADD PACKAGE TO STUDIO" button. Wait until the text “<library_name> package installation is completed” appears in the description section. After this completion message is received, the desired library can be used by writing “import <library_name>” in the source code section as explained in the sections above.

8. Teknik Destek

If you encounter any problem with the software, you can share your problem under the title of the problem at community.proceye.com or you can contact info@augelab.com.