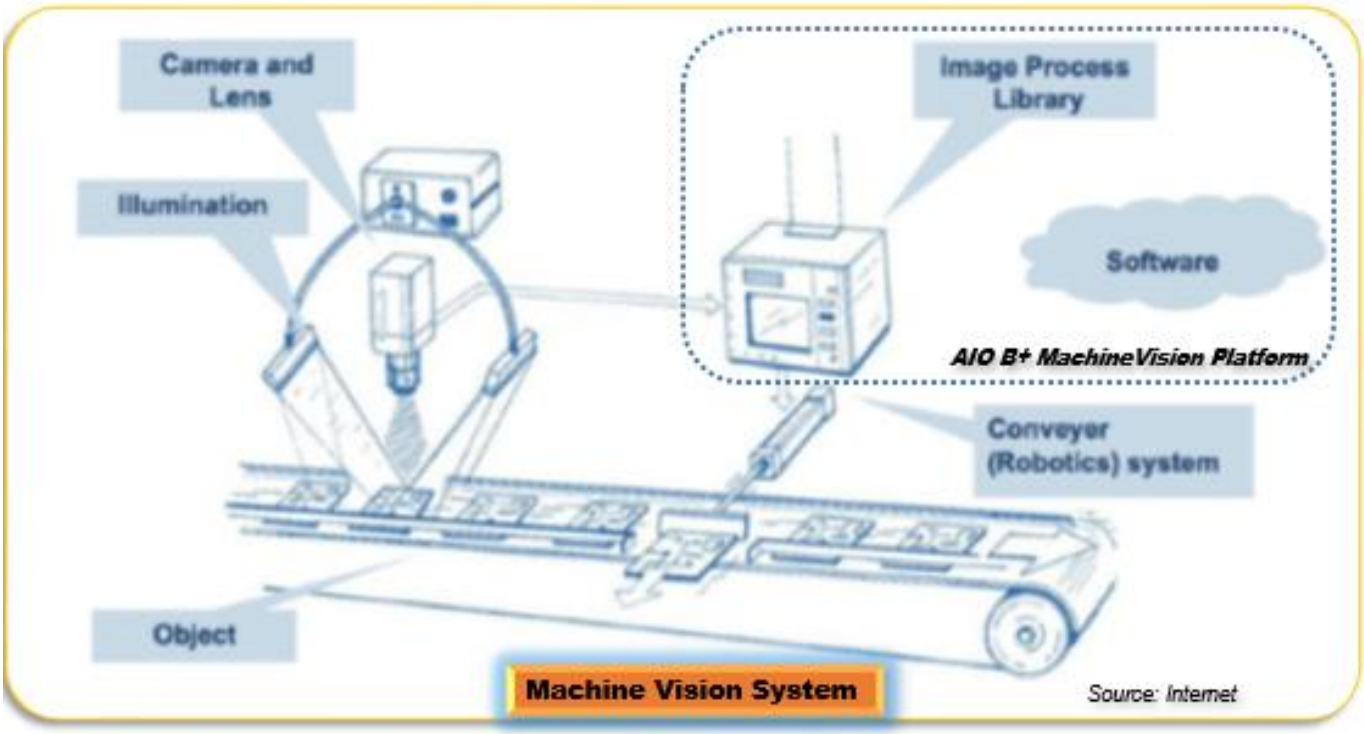
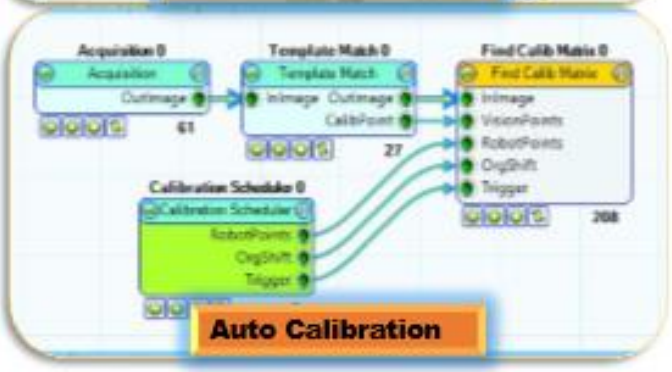
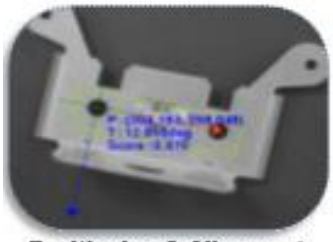
AIO B+ by AIO MATRIX

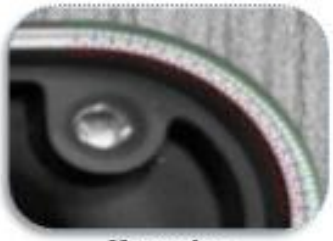
MACHINE VISION PLATFORM FOR FACTORY AUTOMATION





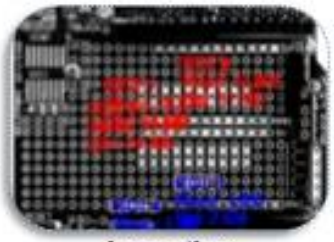


Positioning & Alignment



Measuring

Morphology & Counting



Inspection

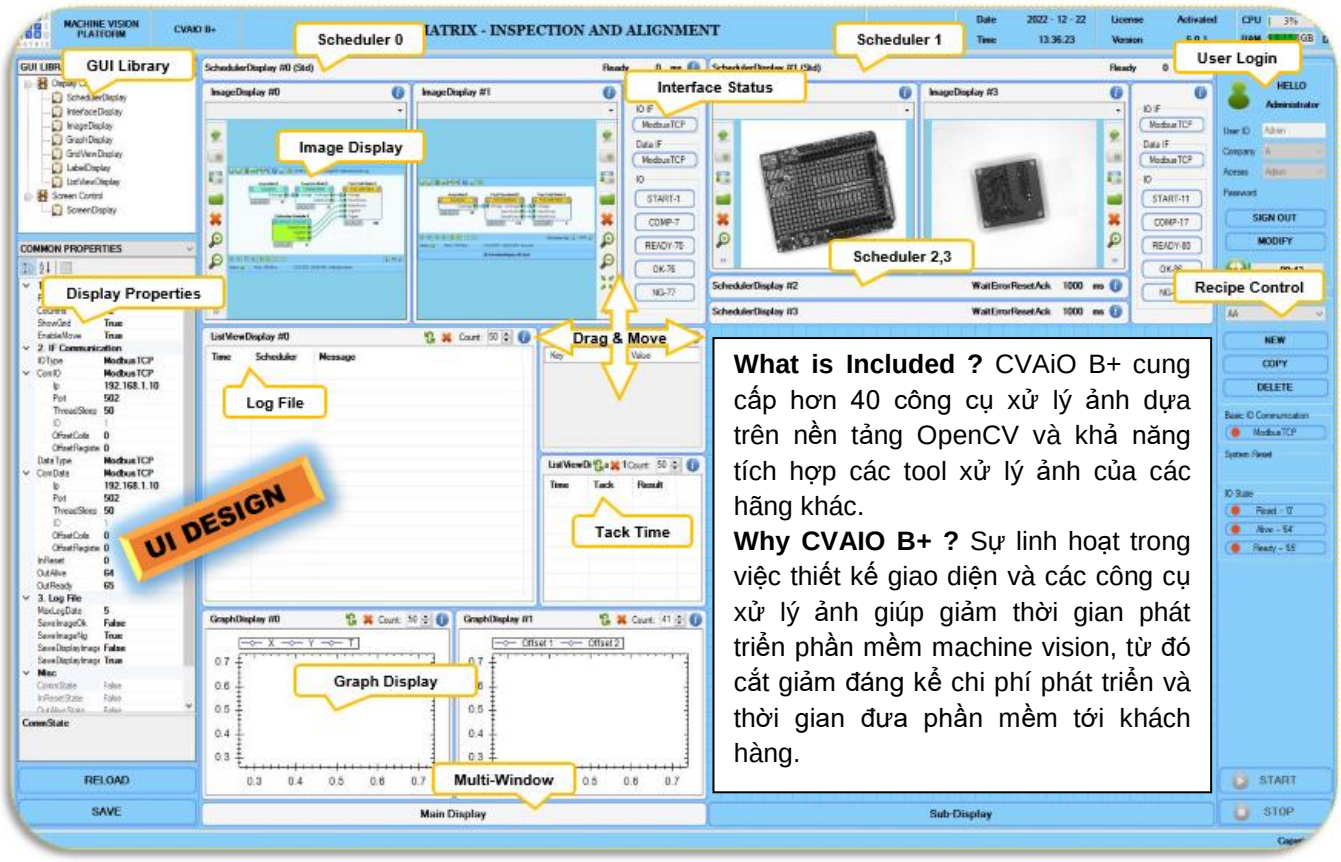
Table of Content

CHAPTER 1: TỔNG QUAN VỀ CVAIO B+

Các nội dung sẽ được đề cập trong chương này:

- Tổng quan về CVAIO B+ Platform
- Các màn hình chính của giao diện hiển thị
- Cấu trúc của phần mềm, các luồng xử lý ảnh, giao tiếp với PLC, Camera, Light Controller
- Chi tiết các bước để thiết lập một chương trình xử lý ảnh với CVAIO B+

1.1 Giới thiệu



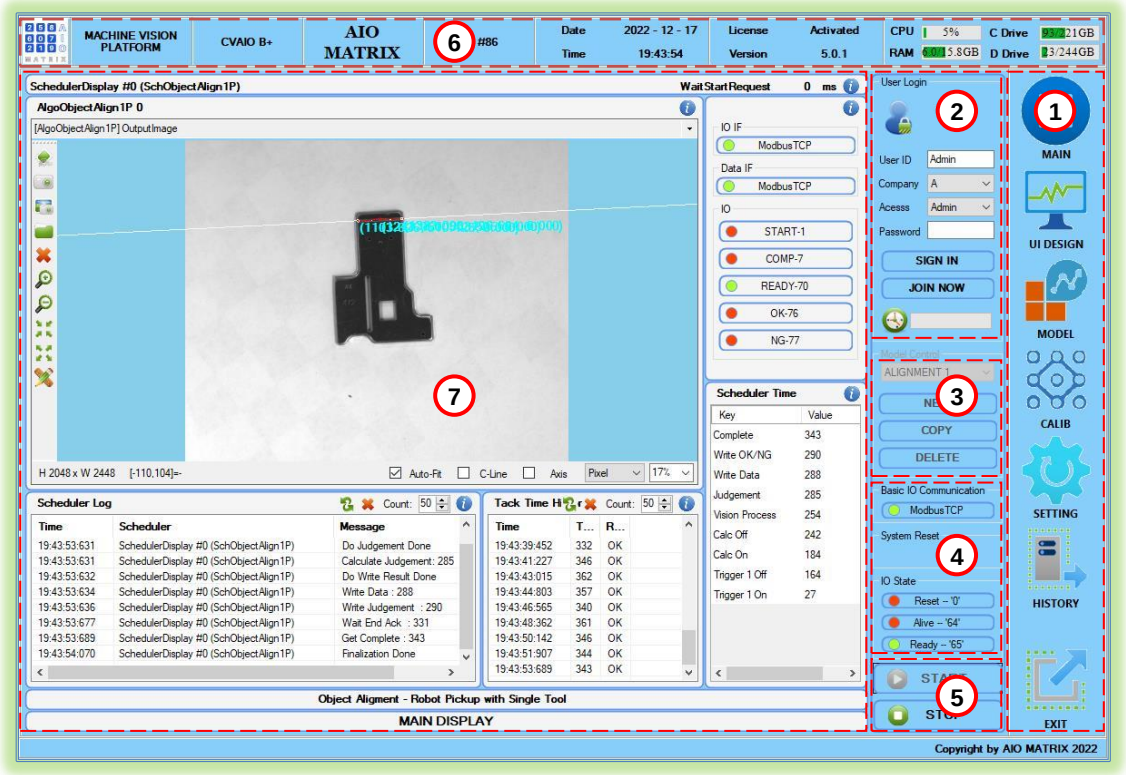
What is CVAIO B+ ? Với mục tiêu giảm thiểu thời gian phát triển một chương trình Machine Vision cũng như cắt giảm tối đa chi phí phát sinh trong quá trình đưa một thuật toán xử lý ảnh vào áp dụng trong nhà máy, AIO MATRIX cung cấp một nền tảng Machine Vision Platform CVAiO B+ dựa trên mã nguồn mở OpenCV. Thông qua thao tác bốc thả (Drag&Drop), người dùng có thể dễ dàng thêm hoặc bớt đi các luồng xử lý ảnh. Cùng với sự linh hoạt của công cụ điều phối (Scheduler), các thuật toán xử lý ảnh có thể được tiến hành lần lượt (Multi-Shot) trước khi kết quả cuối cùng được tổng hợp và phán định bởi công cụ Algorithm. Thêm vào đó là việc tích hợp tính năng căn chỉnh tự động (Auto Calibration) giúp cho việc điều hướng robot tìm gắp vật thể đơn giản và dễ dàng hơn bao giờ hết

1.2 Giao diện người dùng - GUI

Giao diện của chương trình CVAIO B+ bao gồm các phần chính sau:

- **Main:** Hiển thị trạng thái xử lý của toàn bộ chương trình khi chạy tự động
- **UI Design**: Thiết kế màn hình hiển thị Main, điều chỉnh các thuộc tính của từng đối tượng
- **Model:** Thiết kế thuật toán xử lý ảnh, lựa chon IO cho các Scheduler
- **Calibration:** Cài đặt và tiến hành căn chỉnh tự động
- **Setting:** Quan sát trạng thái giao tiếp PLC và các cài đặt khác
- **History:** Hiển thị lịch sử hoạt động của chương trình và kết quả hình ảnh đã được lưu lại

Main GUI – Giao diện chính



1	**Manu Bar**	Lựa chọn chuyển đổi giữa các màn hình hiển thị
2	**User Login**(1)	Quản lý người dùng đăng nhập
3	**Recipe Control**(2)	Quản lý việc tạo mới, thêm, sửa, xóa các recipe
4	**Basic IO Status**(3)	Hiển thị trạng thái của các tín hiệu giao tiếp cơ bản
5	**Program Control**	Quản lý việc chạy và dừng các Scheduler xử lý ảnh
6	**Title Bar**(4)	Hiển thị tiêu đề, ID thiết bị, phiên bản chương trình
7	**Main Display**(5)	Hiển thị trạng thái, kết quả xử lý, trạng thái giao tiếp với PLC/PC

(1) Việc quản lý tài khoản đăng nhập giúp kiểm soát việc thay đổi các thông số cài đặt của chương trình. Trong một số trường hợp, chỉ có các tài khoản admin mới được phép truy cập và thay đổi các parameter. Xem chi tiết tại Section 5.2.

(2) Tất cả các thông số cài đặt được quản lý theo model. Việc quản lý này giúp người dùng dễ dàng phân biệt được model nào đang được sử dụng cũng như thuận tiện trong quá trình thay đổi model (Job change). Dữ liệu thông số của tất cả các model được lưu tại địa chỉ: *C:\CVAiO.Bplus\Recipe*

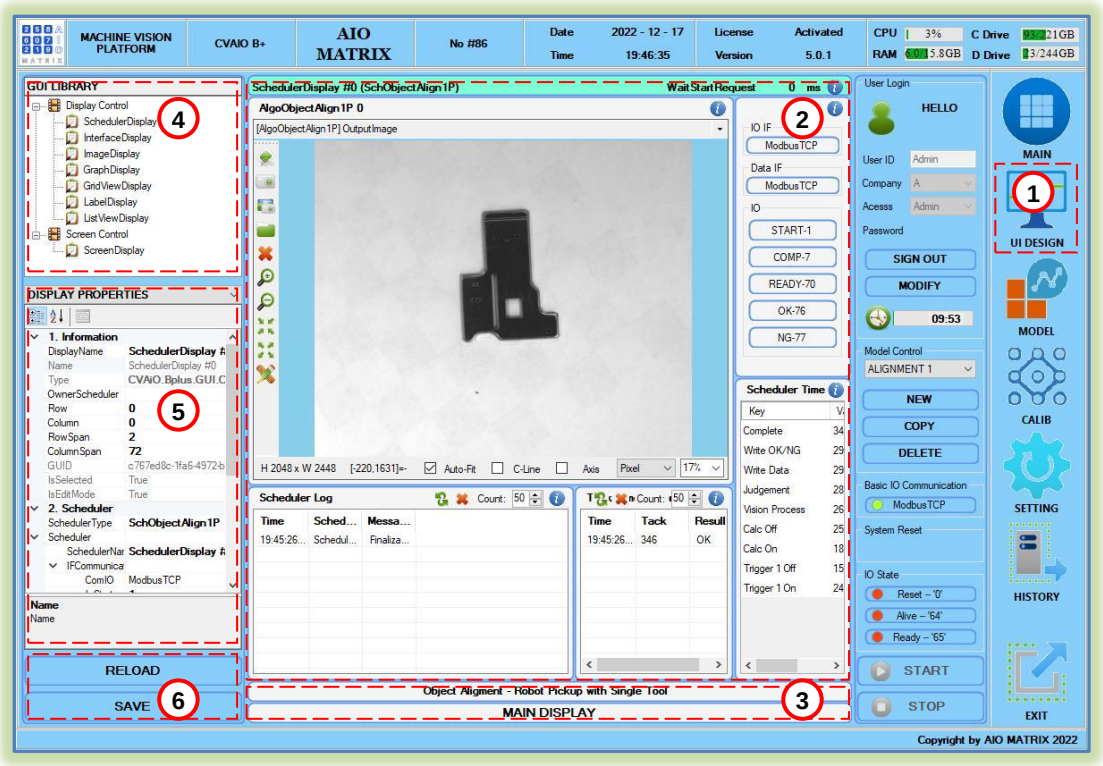
(3) Các tín hiệu IO báo cáo trạng thái của chương trình cho PLC. Bit Ready là tín hiệu duy trì ngay sau khi nút START được bấm. Bit Alive là tín hiệu nháy On, Off theo chu kỳ 3 giây. Sau 3 giây mà không thấy Bit Alive thay đổi trạng thái thì xem như chương trình đang bị lỗi và cần tiến hành khởi động lại chương trình. Xem chi tiết tại Section 1.3

(4) Tên và ID của chương trình có thể thay đổi bằng cách nhấp đúp chuột và thay đổi 1 tên mới hoặc sửa trực tiếp trong system file tại địa chỉ: *C:\CVAiO.Bplus\System.* Kích đúp chuột vào trạng thái "Actived" để kiểm tra thông tin license của dongle key.

(5) Tham khảo thiết kế mẫu của chương trình với 3 Schedulers ở **Case Study** 1

Note: Trong phiên bản dùng thử không có dongle key, chương trình sẽ tự động đóng tất cả các luồng xử lý ảnh và thoát ra sau 30 phút kể từ khi khởi động. Thời gian còn lại của mỗi lần dùng thử được đếm ngược và hiển thị dưới đáy màn hình.

UI Design – Giao diện thiết kế hiển thị



1	**UI Design**[1]	Lựa chọn hiển thị màn hình thiết kế GUI
2	**Main Display**[2]	Danh sách các đối tượng hiển thị
3	**Screen control**	Lựa chọn chuyển đổi giữa các màn hình hiển thị
4	**GUI Library**[3]	Thư viện các đối tượng hiển thị[4]
5	**Display Property**[5]	Hiển thị các đặc tính của đối tượng hiển thị được chọn
6	**Reload/Save-**	Mở lại / lưu các cài đặt hiện tại của model

(1) Cần phải đăng nhập với quyền Engineer hoặc Admin để có thể chuy cập cửa sổ này. Cửa sổ sẽ tự động chuyển về màn hình Main sau 10 phút không có tác động chuột nào.

(2) Ngoại trừ đối tượng ScreenDisplay, tất cả các đối tượng còn lại đều có thể tùy biến vị trí theo yêu cầu của người dùng. Số lượng SchedulerDiplay cũng chính là số lượng luồng xử lý ảnh sẽ được dùng trong chương trình. Xem chi tiết tại Section 2.3

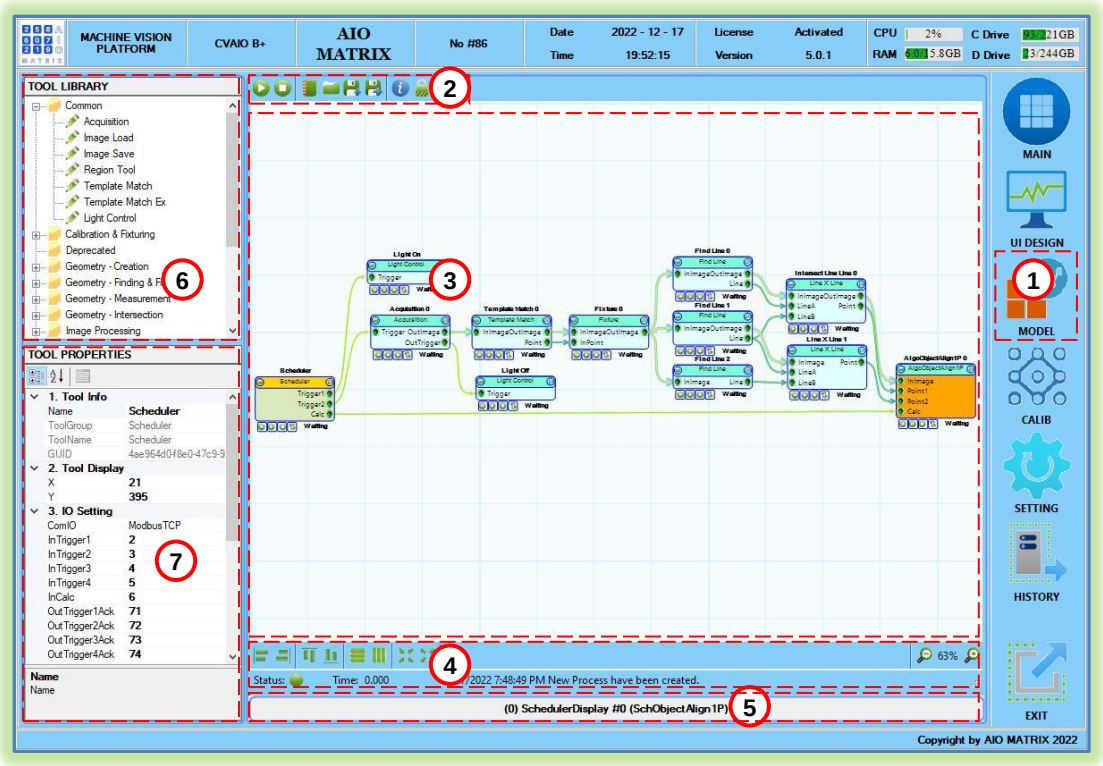
(3) Việc lựa chọn sử dụng đối tượng hiển thị nào tùy thuộc vào yêu cầu của từng bài toán xử lý ảnh. Luồng xử lý ảnh được cấu trúc tách biệt với luồng hiển thị nên số lượng đối tượng hiển thị không làm ảnh hưởng đến thời gian (tack time) xử lý của chương trình.

(4) Cài đặt liên quan đến các đối tượng hiển thị có thể xem chi tiết tại Chapter 2.

(5) Tất cả các thay đổi liên quan đến thuộc tính đều được lưu lại trong log file và không được lưu lại cho đến khi người dùng lựa chọn SAVE toàn bộ model đang được sử dụng.

Note: Để đảm bảo mọi thao tác đã được lưu lại, sau khi lựa chon SAVE nên thử RELOAD và kiểm tra lại dữ liệu 1 lần nữa.

Model – Giao diện thiết kế Vision Process



1	**Model**[1]	Lựa chọn hiển thị màn hình thiết kế Vision Process
2	**Process Control**	Manu các điều khiển chức năng của Vision Process
3	**Process Display**[2]	Hiển thị Scheduler và các công cụ xử lý ảnh
4	**Process Status**	Hiển thị trạng thái của Vision Process
5	**Scheduler Process**[3]	Lựa chọn các process hiện có của chương trình
6	**Tool Library**	Danh sách các công cụ xử lý ảnh
7	**Tool Properties**[4]	Hiển thị các đặc tính của công cụ xử lý ảnh được chọn

(1) Trong lần đầu tiên được lựa chọn, chương trình sẽ mất thời gian để load tất cả các thư viện cần thiết. Đặc biệt là trong trường hợp có sử dụng thư viện của Cognex Vision Pro

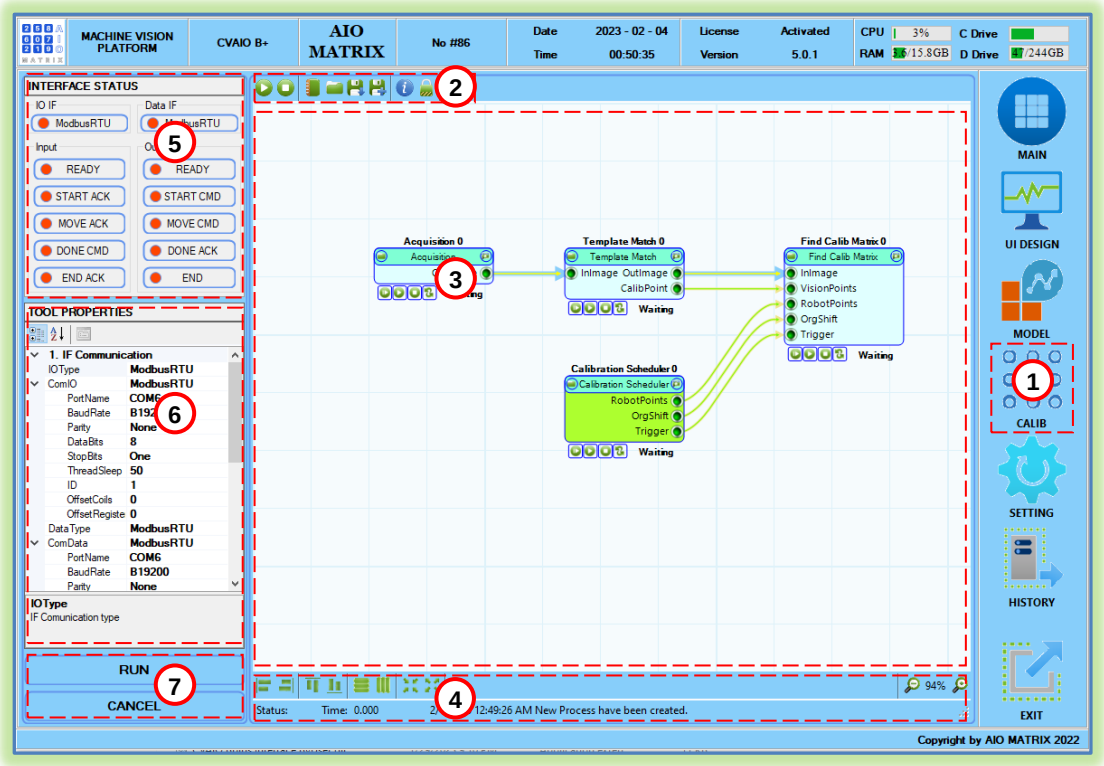
(2) Chỉ có các Input và Output cùng loại mới có thể kết nối được với nhau. Thông tin chi tiết về các loại đầu vào, đầu ra tham khảo Section 3.1. Công cụ Scheduler và Algorithm là bắt buộc để đảm bảo các quá trình xử lý có thể thực hiện được.

(3) Tất cả các process được hiển thị tại đây. Trong trường hợp Scheduler chưa được chọn, Vision Process được gán cho giá trị Null

(4) Tất cả sự thay đổi liên quan đến đặc tính của các công cụ đều được lưu lại trong log file.

Note: Sau khi tất cả các tool được thực thi, cần phải ấn Stop để đưa toàn bộ process về trạng thái sẵn sàng

Calibration – Giao diện căn chỉnh tự động



1	**Calibration**[1]	Lựa chọn hiển thị màn hình căn chỉnh tự động
2	**Calibration Control**[2]	Manu các điều khiển quá trình căn chỉnh chạy single run
3	**Calibration Display**	Hiển thị Calibration Scheduler và các công cụ cho căn chỉnh Auto
4	**Calibration Status**	Hiển thị trạng thái của quá trình căn chỉnh
5	**Interface Status**[3]	Lựa chọn các process hiện có của chương trình
6	**Tool Properties**[4]	Hiển thị các đặc tính của công cụ xử lý ảnh được chọn
7	**Calibration Auto**[5]	Chạy hoặc hủy quá trình căn chỉnh tự động

(1) Cần phải đăng nhập với quyền Engineer hoặc Admin để có thể chuy cập cửa sổ này. Đây là màn hình cho chế độ căn chỉnh tự động có liên kết với PLC/PC. Tham khảo thêm chế độ căn chỉnh manual ở Section 4.2

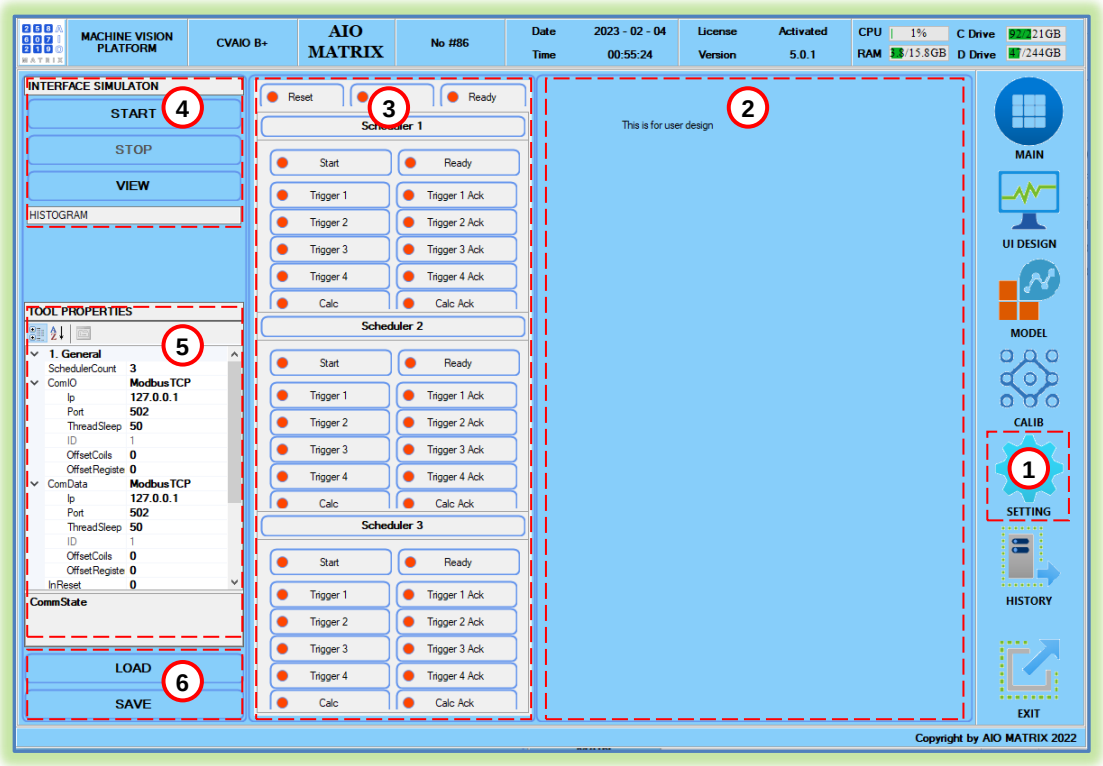
(2) Nút Run chỉ có tác dụng chạy Single Run để kiểm tra cài đặt của từng công cụ xử lý ảnh. Nút New sẽ tạo ra 1 process căn chỉnh mới.

(3) Tham khảo Interface Sequence ở Section 4.3. Interface Type và các thông số cài đặt liên quan đến căn chỉnh độc lập so với các thông tin Interface của chương trình chính.

(4) Tất cả sự thay đổi liên quan đến đặc tính của các công cụ đều được lưu lại trong log file.

(5) Quá trình căn chỉnh sẽ chia làm 2 giai đoạn: Translation và Rotation và sẽ được thực hiện liên tục

Setting – Giao diện cài đặt liên quan

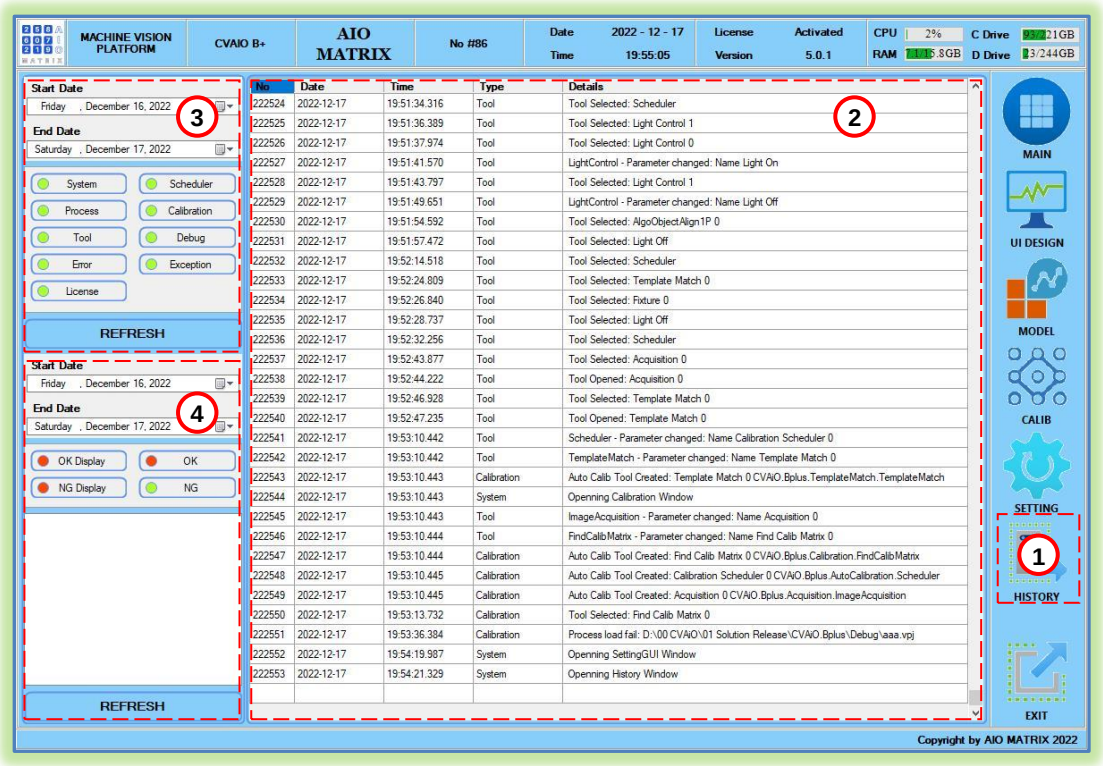


1	**Setting**[1]	Lựa chọn hiển thị màn hình cài đặt Setting
2	**User Design UI**[1]	Màn hình mở để người dùng có thể thiết kế thêm tính năng khác
3	**Interface Status**[2]	Trạng thái On/Off của từng IO trong các Interface tương ứng
4	**Interface Control**	Chạy hoặc dừng quan sát trạng thái của Interface
5	**Interface Properties**	Hiển thị đặc tính của Interface được chọn
6	**Load Save**	Tải hoặc lưu lại thông tin Interface

(1) Người dùng có thể thiết kế thêm phần hiển thị trong project ***"CVAiO.Bplus.SettingGUI"***

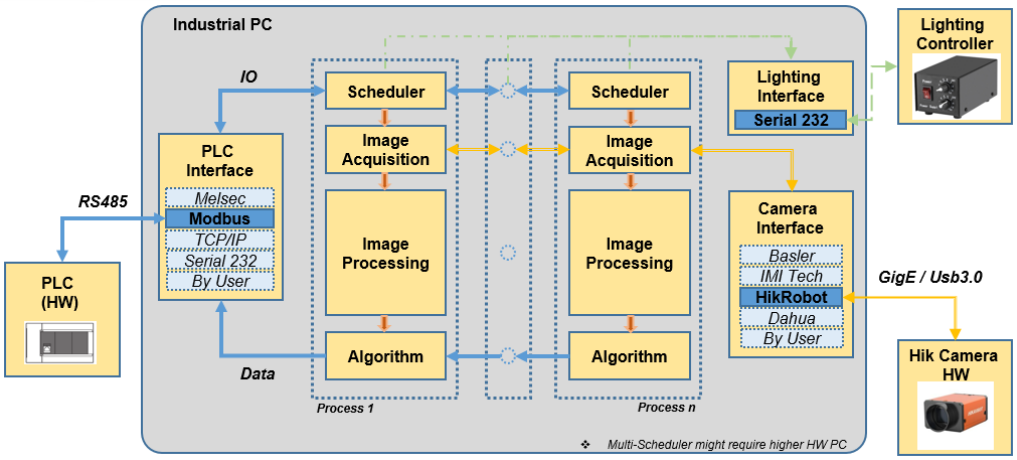
(2) Tất cả các thông tin về interface của chương trình chính sẽ được liên kết và hiển thị tại đây. Trong trường hợp chương trình chính đang ở chế độ Auto thì người dùng sẽ không thể thay đổi các đặc tính của Interface và người dùng có thể quan sát trạng thái làm việc hiện tại của các IO (Monitor Mode)

History – Giao diện lịch sử Log và hình ảnh

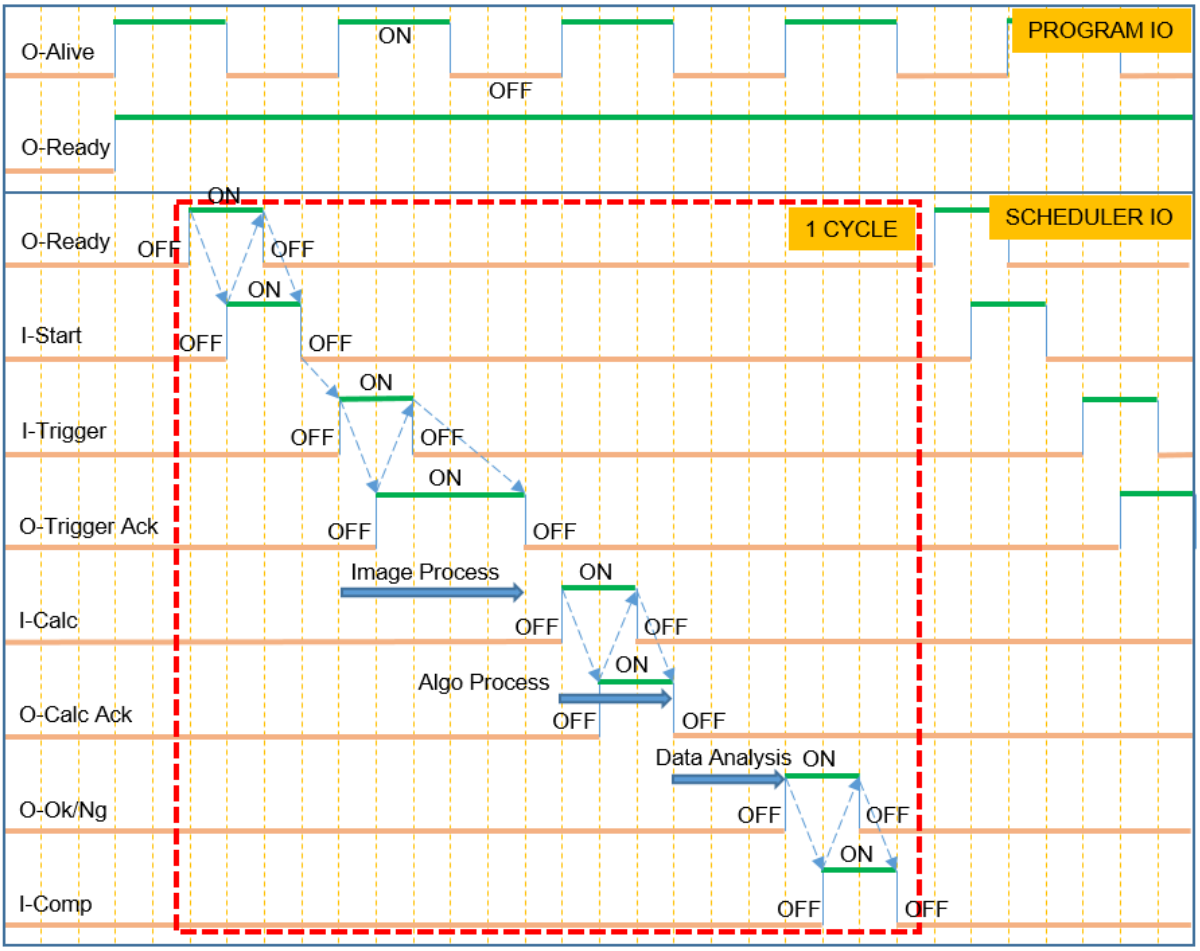


1	**History**	Lựa chọn hiển thị màn hình lịch sử Log
2	**History Display**	Thông tin lịch sử log hoặc hình ảnh
3	**Log History Control**(1)	Lựa chọn hiển thị thông tin lịch sử Log
4	**Image History Control**	Lựa chọn hiển thị lịch sử hình ảnh (Raw Image & Display Image)

1.3 Platform Structure – Cấu trúc chương trình



Chương trình được cấu trúc thành các module riêng biệt bao gồm có module giao tiếp PLC, module giao tiếp với camera, module giao tiếp với bộ điều khiển đèn và các module xử lý ảnh. Chương trình chính được thiết kế để có thể thêm bớt các process vision một cách dễ dàng và các process vision được thêm vào đều có thể liên kết với các module còn lại để thiết lập kết nối với phần cứng. Bản thân bên trong các module giao tiếp với phần cứng, các lớp tương ứng với các phần cứng khác nhau cũng được phân chia rõ ràng để có thể lựa chọn cho phù hợp với từng hệ thống thực tế. Mỗi thao tác bốc thả Scheduler Display, một vision process sẽ được thêm vào cung cấp cho người dùng khả năng thiết kế một luồng xử lý ảnh. Việc điều khiển hoạt động của từng process vision tuân theo IO sequence như mô tả ở hình trên đây.Tham khảo thêm vai trò của các IO ở section 2.3 và section 3.10.

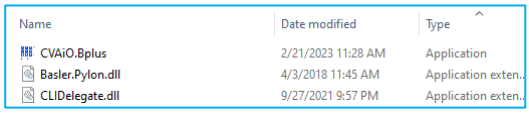


1.4 Setup Guide – Các bước để thiết lập chương trình

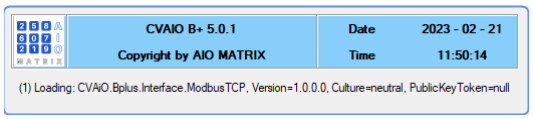
Dưới đây là chi tiết các bước để thiết lập một chương trình xử lý ảnh

❖ Step 1: Khởi động chương trình

- Chương trình: ***CVAiO.Bplus.exe***



- Cửa sổ thiết lập cài đặt chương trình

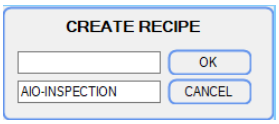


❖ Step 2: Đăng nhập người dùng – User login

- Nhập user & password sau đó bấm log in để đăng nhập tài khoản người quản trị. Phải đăng nhập với quyền admin hoặc engineer mới có thể thiết lập chương trình mới. Chi tiết tham khảo Section 5.2

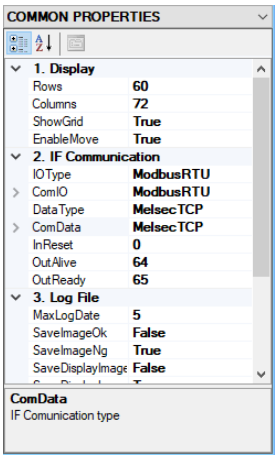
❖ Step 3: Khởi tạo Model - Recipe

- Tạo model mới để quản lý toàn bộ các cài đặt liên quan. Trong trường hợp muốn tạo dựa trên 1 model hiện có, người dùng có thể sử dụng tính năng COPY. Dữ liệu liên quan đến model được lưu tại *C:\CVAiO.Bplus\Recipe*



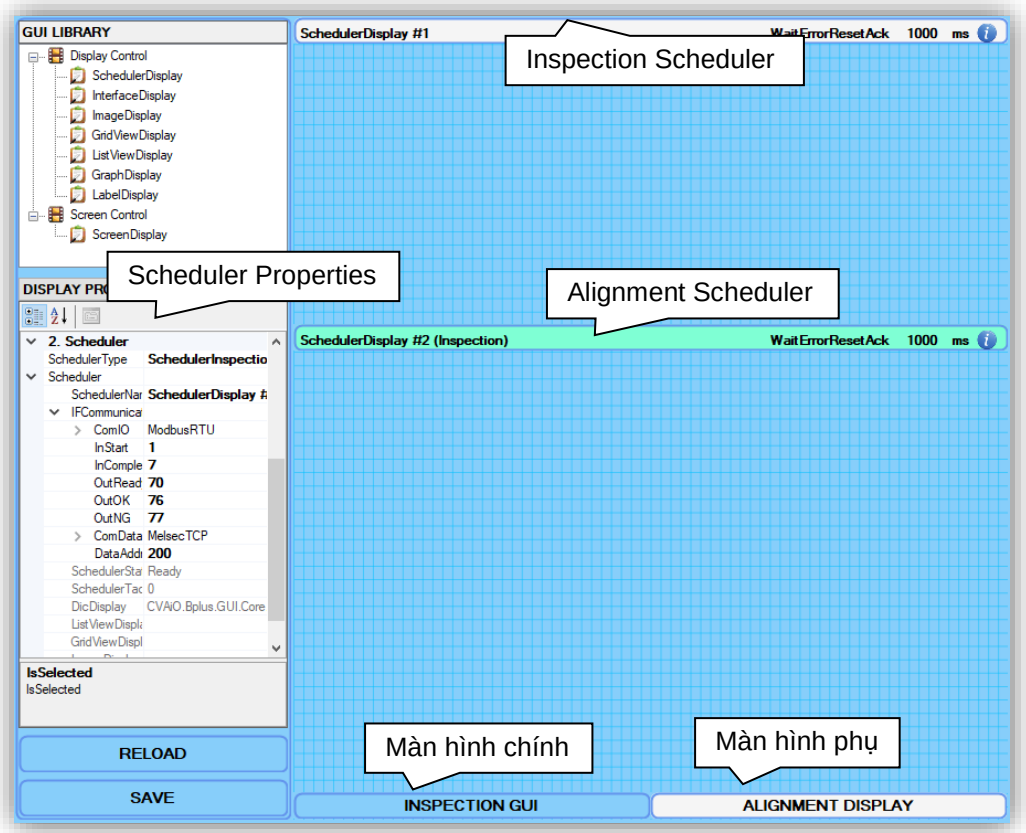
❖ Step 4: Cài đặt giao tiếp với PLC

- Chuyển sang màn hình UI Design để bắt đầu thiết kế GUI
- Chuột trái vào màn hình chính hoặc chuyển lựa chọn properties bên trái thành COMMON PROPERTIES.
- Lựa chọn loại giao tiếp cho dữ liệu IO On/Off tại IOType
- Lựa chọn loại giao tiếp cho dữ liệu data tại DataType
- Tùy theo loại giao tiếp được chọn của IOType và DataType, các lựa chọn tương ứng sẽ mở ra để cài đặt thêm thông tin giao tiếp
- Cài đặt địa chỉ IO cho các tin hiệu Reset, Alive, Ready. Xem chi tiết tại Section 1.3



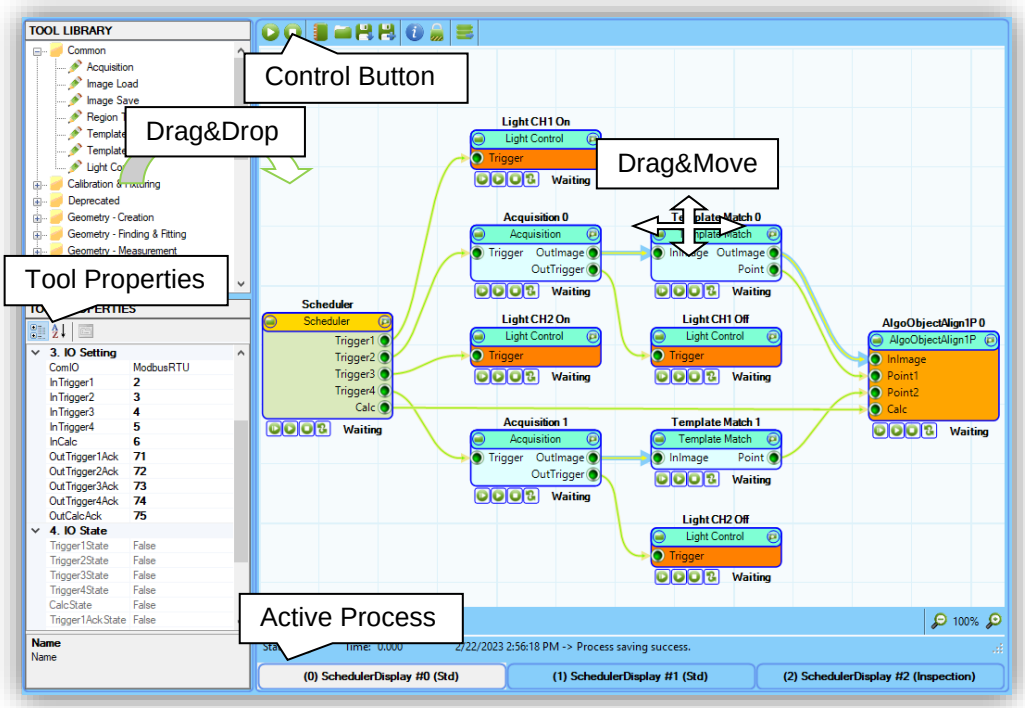
❖ Step 5: Thiết kế giao diện hiển thị 1

- Lựa chọn số màn hình hiện thị bằng cách bốc thả đối tượng ScreenDisplay. Lựa chọn tên hiển thị và chọn *ScreenDetach* là *True* nếu muốn hiển thị ra 1 màn hình riêng biệt.
- Lựa chọn số lượng luồng xử lý ảnh bằng cách bốc thả đối tượng SchedulerDisplay.
- Mỗi Scheduler sẽ có 1 GUID không trùng lặp để sau đó các đối tượng hiển thị khác sẽ lựa chọn để kết nối và lấy dữ liệu hiển thị.
- Chọn vào từng Scheduler sau đó thay đổi các thuộc tính liên quan bao gồm SchedulerType và các IO liên quan đến logic hoạt động của scheduler. Chi tiết tham khảo section 2.3
- Bấm Save để lưu lại các thiết kế hiện tại.



❖ **Step 6: Thiết kế thuật toán xử lý ảnh**

- Chuyển sang màn hình Model để bắt đầu thiết kế process
- Bấm vào từng Scheduler phía dưới để hiển thị từng process



- Chuột phải vào công cụ Scheduler để thêm hoặc bớt số lượng trigger. Tham khảo thêm về Scheduler tại Section 3.10
- Lựa chọn công cụ Algorithm rồi kéo thả vào màn hình. Mỗi process chỉ được sử dụng 1 công cụ Algorithm. Tham khảo thêm về Algorithm tại Section 3.11

- Kéo thả các công cụ xử lý ảnh khác tùy theo yêu cầu của bài toán
- Click đúp vào công cụ Scheduler để bật-tắt các trigger trước khi bấm Start để chạy, Stop để reset.

❖ **Step 7: Thiết kế giao diện hiển thị 2**

- Chuyển sang màn hình UI Design để bắt đầu thiết kế các thành phần còn lại của GUI
- Bốc thả các đối tượng tùy theo yêu cầu của bài toán vision.
- Lựa chọn OwnerScheduler (Scheduler mà từng đối tượng hiển thị sẽ kết nối vào để lấy thông tin) cho từng đối tượng hiển thị.
- Điều chỉnh thuộc tính của các đối tượng hiển thị theo hướng dẫn ở Chapter 2
- Bấm Save để lưu lại các thiết kế hiện tại.

❖ **Step 8: Hoàn thiện và chạy thử với chương trình mô phỏng**

- Trong trường hợp không có PLC để kết nối trực tiếp, người dùng có thể sử dụng bộ mô phỏng PLC với module interface Modbus TCP/IP.
- Quay lại Step 4 và điều chỉnh toàn bộ lựa chọn IOType và DataType thành ModbusTCP (127.0.0.1).
- Khởi động chương trình mô phỏng PLC
- Bấm START trên chương trình Vision để chạy chương trình
- Bấm START bên chương trình mô phỏng để chạy mô phỏng và kết nối với chương trình vision qua Modbus TCP.

CHAPTER 2: GUI LIBRAY – THƯ VIỆN CÁC ĐỐI TƯỢNG HIỂN THỊ

Các nội dung sẽ được đề cập trong chương này:

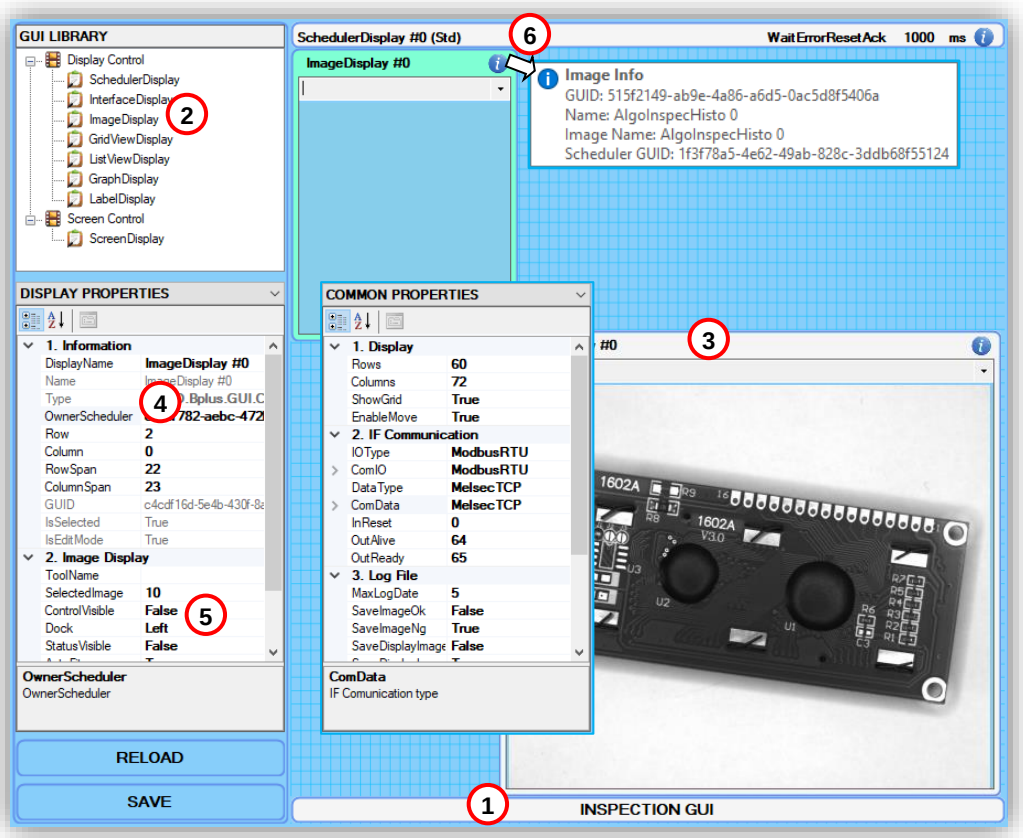
- Tổng quan về ý tưởng thiết kế màn hình hiển thị
- Các chức năng, thuộc tính chung của tất cả đối tượng hiển thị
- Chức năng, thuộc tính và cách sử dụng của từng đối tượng hiển thị
- Case Study 1: Thiết kế màn hình hiển thị cho 1 Task có 3 luồng xử lý ảnh
- Case Study 2: Chương trình hiển thị trên 2 màn hình độc lập

2.1 Overview - Giới thiệu chung

Động lực cho ý tưởng thiết kế màn hình hiển thị đến từ yêu cầu thực tế của các bài toán vision trong hệ thống tự động hóa. Mỗi bài toán khác nhau sẽ yêu cầu thêm hoặc bớt các đối tượng hiển thị. Việc cố định cứng các đối tượng hiển thị như các chương trình hiện nay sẽ làm giảm đi sự linh hoạt trong trải nghiệm của người dùng. Để loại bỏ các trở ngại đó, CVAIO B+ cung cấp sự tùy biến trong giao diện hiển thị từ việc lựa chọn số lượng luồn xử lý cũng như việc tăng giảm màn hình quản lý hiển thị. Bằng các thao tác bốc thả (drag & drop) đơn giản, người dùng có thể tự lựa chọn các đối tượng hiển thị và sắp xếp các đối tượng theo yêu cầu của từng bài toán. Với các thao tác bấm chuột đơn giản, thuộc tính của từng đối tượng hiển thị có thể được thay đổi mà không tốn quá nhiều thời gian.

Hướng dẫn thao tác thiết kế màn hình hiển thị:

- ❖ **Step 1:** Lựa chọn màn hình hiển thị sẽ thiết kế
- ❖ **Step 2:** Lựa chọn đối tượng hiển thị và bốc thả vào vị trí bất kỳ
- ❖ **Step 3:** Điều chỉnh kích thước, vị trí của đối tượng hiển thị
- ❖ **Step 4:** Lựa chọn Owner Scheduler mà đối tượng sẽ kết nối đến
- ❖ **Step 5:** Điều chỉnh thuộc tính của đối tượng
- ❖ **Step 6:** Kiểm tra thông tin và lưu lại kết quả thực hiện



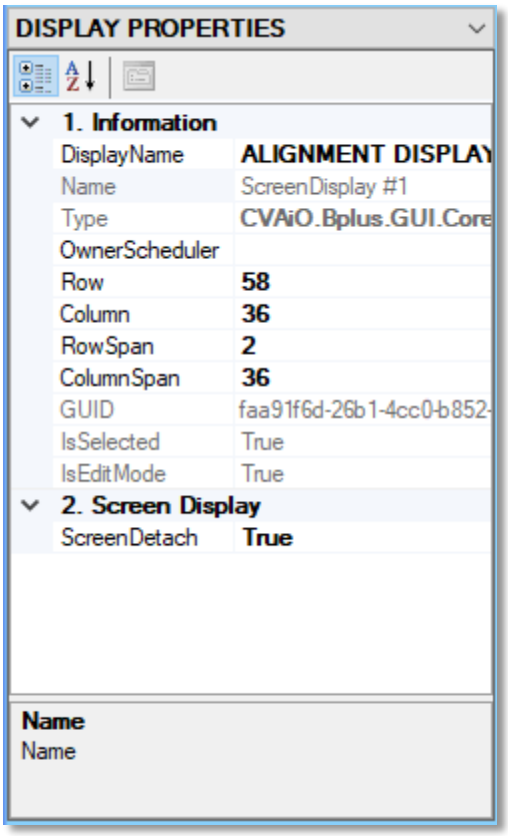
- **Common Properties:**
  - **(1) Rows:** *Số hàng chia của lưới trên màn hình thiết kế GUI*
  - **(2) Colums:** *Số cột chia của lưới trên màn hình thiết kế GUI*
  - **(3) ShowGrid:** *Ẩn hiện lưới trên màn hình thiết kế GUI*
  - **(4) EnableMove:** *Cho phép chỉnh sửa vị trí của các đối tượng hiển thị*

(5) IOType: *Loại giao tiếp với PLC để truyền nhận trạng thái IO*
(6) ComIO: *Thuộc tính cài đặt của giao tiếp với PLC để truyền nhận trạng thái IO*
(7) DataType: *Loại giao tiếp với PLC để truyền nhận Data*
(8) ComData: *Thuộc tính cài đặt của giao tiếp với PLC để truyền nhận dữ liệu data*
(9) InReset: *Tín hiệu IO để reset lại toàn bộ các luồng xử lý ảnh*
(10) OutAlive: *Tín hiệu báo trạng thái đang hoạt động của chương trình (nháy on/off 3-5s)*
(11) OutReady: *Tính hiệu báo trạng thái sẵn sàng của chương trình*
(12) MaxLogDate: *Giới hạn số ngày lưu giữ lịch sử*
(13) SaveImageOk: *Lựa chọn lưu hình ành raw khi kết quả xử lý OK*
(14) SaveImageNg: *Lựa chon lưu hình ảnh raw khi kết quả xử lý NG*
(15) SaveDisplayOk: *Lựa chọn lưu hình ảnh hiển thị trên GUI khi kết quả xử lý OK*
(16) SaveDisplayNg: *Lựa chọn lưu hình ảnh hiển thị trên GUI khi kết quả xử lý NG*

- **Display Properties: (Chung cho tất cả đối tượng hiển thị)**
  (1) DisplayName: *Tên của đối tượng hiển thị sẽ xuất hiện trên màn hình chính*
  (2) Name: *Tên của đối tượng hiện thị dùng để quản lý nội bộ (read only)*
  (3) Type: *Phân loại của đối tượng hiện thị (read only)*
  (4) OwnerScheduler: *Scheduler mà đối tượng hiển thị sẽ được đính vào để lấy dữ liệu*
  (5) Row: *Vị trí tính theo hàng*
  (6) Column: *Vị trí tính theo cột*
  (7) RowSpan: *Độ rộng tính theo số hàng*
  (8) ColumnSpan: *Độ rộng tính theo số cột*
  (9) GUID: *ID định danh của đối tượng (read only)*
  (10) IsSelected: *Trạng thái lựa chọn (read only)*
  (11) IsEditMode: *Trạng thái chỉnh sửa (read only)*

❖ **Note:**
✓ *Các trạng thái read only là các thuộc tính được sử dụng trong nội bộ chương trình, người dùng không thể điều chỉnh các thuộc tính này.*

2.2 Screen Display - Hiển thị đa màn hình

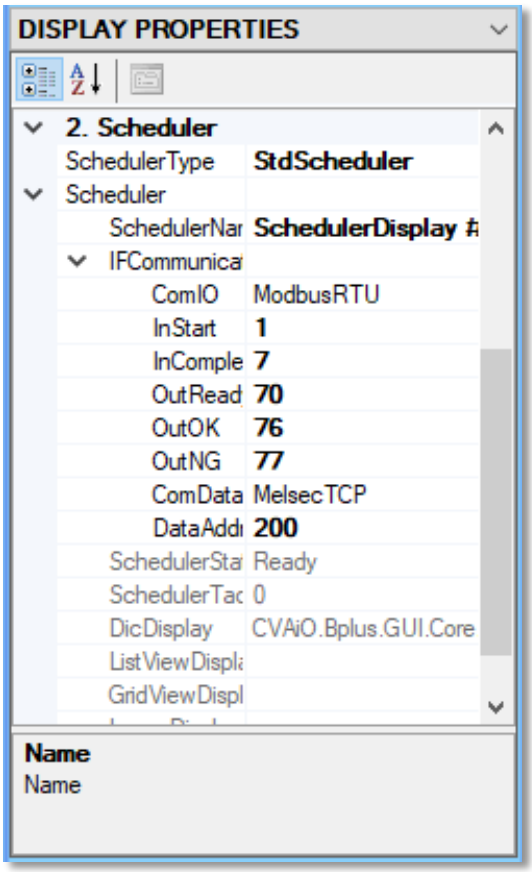


❖ **Mục đích:** Tăng giảm số lượng màn hình hiển thị để đáp ứng trong trường hợp người dùng sử dụng nhiều luồng xử lý ảnh, đặc biệt là trong trường hợp muốn hiển thị trên nhiều pc monitor.

- **Display Properties:**

  (1) ScreenDetach: *Tách màn hình ra hiển thị ở 1 cửa sổ riêng biệt*

2.3 Scheduler Display – Hiển thị luồng xử lý ảnh

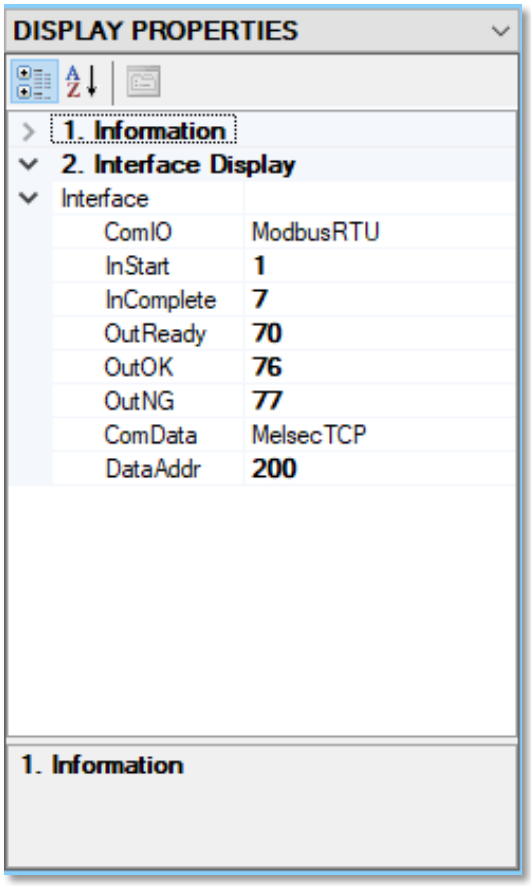


❖ Mục đích: Tăng giảm số lượng luồng xử lý ảnh, lựa chọn loại luồng xử lý (bao gồm các bước trước khi xử lý ảnh, truyền nhận dữ liệu với PLC). Người dùng có thể tự thiết kế 1 logic phù hợp với yêu cầu của từng bài toán. Chi tiết về việc thiết kế tham khảo tại Section 6.3.

Display Properties:

(1) SchedulerType: *Loại luồng xử lý ảnh,*
(2) ComIO: *Loại giao tiếp cho dữ liệu IO On/Off (Melsec TCP, Modbus, Serial Port)*
(3) InStart: *Input báo hiệu bắt đầu quá trình xử lý*
(4) InComplete: *Input báo hiệu kết thúc quá trình xử lý*
(5) OutReady: *Output báo luồng xử lý ảnh đang ở trạng thái sẵn sàng*
(6) OutOk: *Output báo kết quả xử lý và phán định OK*
(7) OutNg: *Output báo kết quả xử lý và phán định NG*
(8) ComData: *Loại giao tiếp cho dữ liệu Data*
(9) DataAddr: *Vị trí bắt đầu ghi dữ liệu Data*

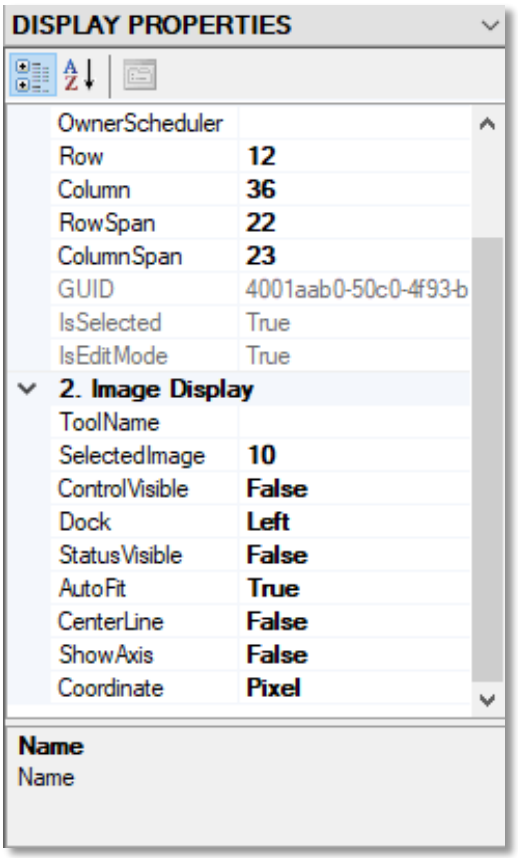
2.4 Interface Display – Hiển thị trạng thái giao tiếp với PLC/PC

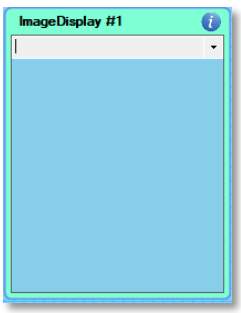


❖ **Mục đích:** Hiển thị trạng thái hiện tại của các IO giao tiếp với PLC của luồng xử lý ảnh được đính vào. Người dùng cần lựa chọn luồng xử lý ảnh tương ứng ở thuộc tính Owner Scherduler.

- **Display Properties:**
  - **(1) ComIO:** *Loại giao tiếp cho dữ liệu IO On/Off (Melsec TCP, Modbus, Serial Port …)*
  - **(2) InStart:** *Input báo hiệu bắt đầu quá trình xử lý*
  - **(3) InComplete:** *Input báo hiệu kết thúc quá trình xử lý*
  - **(4) OutReady:** *Output báo luồng xử lý ảnh đang ở trạng thái sẵn sàng*
  - **(5) OutOk:** *Output báo kết quả xử lý và phán định OK*
  - **(6) OutNg:** *Output báo kết quả xử lý và phán định NG*
  - **(7) ComData:** *Loại giao tiếp cho dữ liệu Data*
  - **(8) DataAddr:** *Vị trí bắt đầu ghi dữ liệu Data*

2.5 Image Display – Hiển thị hình ảnh

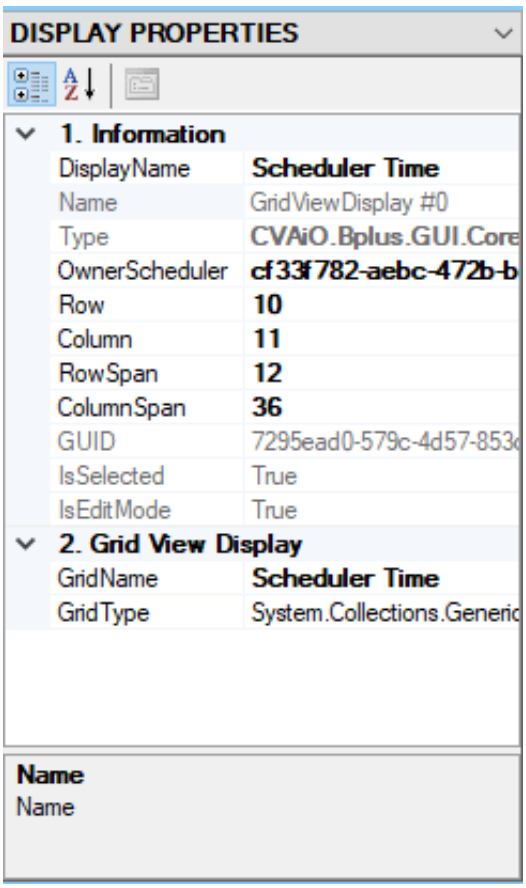


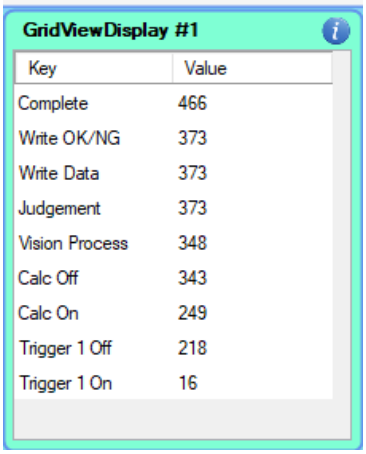


❖ **Mục đích:** Hiển thị hình ảnh của các giai đoạn trong quá trình xử lý. Người dùng cần lựa chọn luồng xử lý ảnh tương ứng ở thuộc tính Owner Scherduler sau đó lựa chọn công cụ xử lý ảnh trong luồng đó ở thuộc tính ToolName.

- **Display Properties:**
  - **(1) ToolName:** *Công cụ xử lý ảnh mà hình ảnh sẽ được lấy*
  - **(2) SelectedImage:** *vị trí trong danh sách ảnh của tool*
  - **(3) ControlVisible:** *Ẩn hiện thanh công cụ điều khiển*
  - **(4) Dock:** *Vị trí của thanh công cụ điều khiển*
  - **(5) StatusVisible:** *Ẩn hiện thanh công cụ trạng thái*
  - **(6) AutoFit:** *Tự động điền đầy của hình ảnh*
  - **(7) CenterLine:** *Ẩn hiện đường tâm*
  - **(8) ShowAxis:** *Ẩn hiện trục tọa độ*
  - **(9) Coordinate:** *Lựa chọn tọa độ hiển thị*

2.6 Grid View Display – Hiển thị dữ liệu dạng lưới

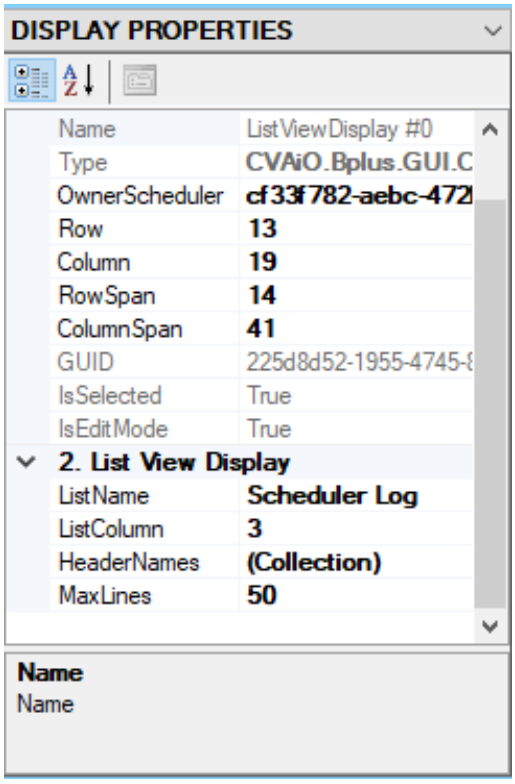


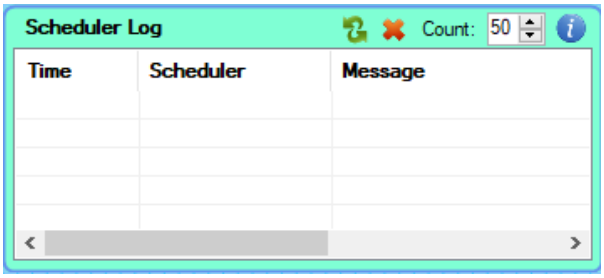


❖ **Mục đích:** Hiển thị các dữ liệu dạng lưới có trong luồng xử lý ảnh tương ứng ở thuộc tính Owner Scherduler. Người dùng có khả năng thêm bớt loại dữ liệu này bằng cách sửa đổi trong mã nguồn của Scheduler tương ứng. Tham khảo Section 6.3

- **Display Properties:**
  - **(1) GridName:** *Tên của dữ liệu dạng lưới trong Scheduler*
  - **(2) GridType:** *Loại dữ liệu dạng bảng*

2.7 List View Display – Hiển thị dữ liệu dạng danh sách

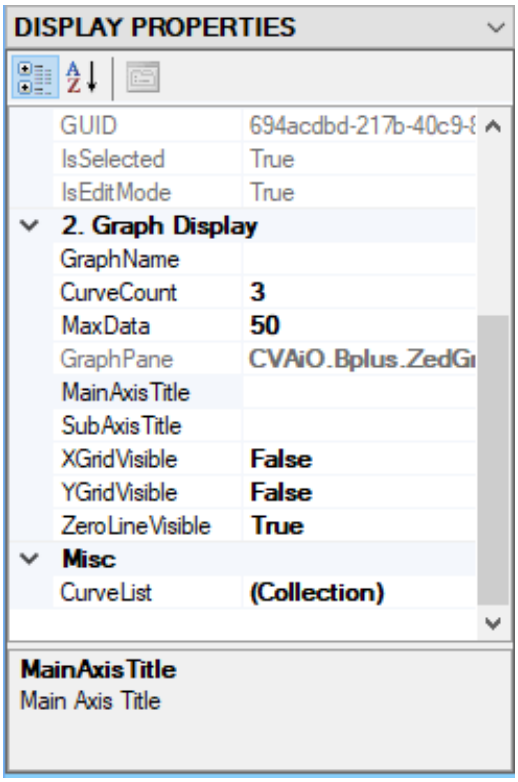


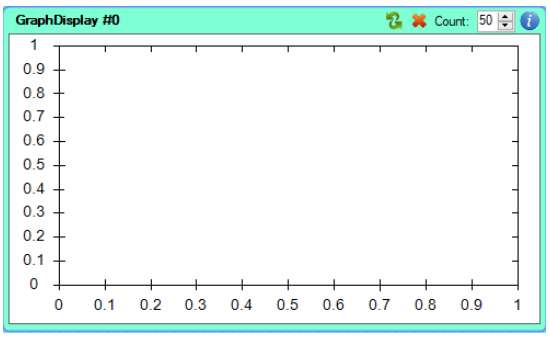


❖ **Mục đích:** Hiển thị các dữ liệu dạng danh sách có trong luồng xử lý ảnh tương ứng ở thuộc tính Owner Scherduler. Người dùng có khả năng thêm bớt loại dữ liệu này bằng cách sửa đổi trong mã nguồn của Scheduler tương ứng. Tham khảo Section 6.3

- **Display Properties:**
  - **(1) ListName:** *Tên của dữ liệu dạng danh sách*
  - **(2) ListColumn:** *Số cột hiển thị*
  - **(3) HeaderNames:** *Tên của các cột hiển thị*
  - **(4) MaxLines:** *Số dòng hiển thị tối đa*

2.8 Graph Display – Hiển thị đồ thị



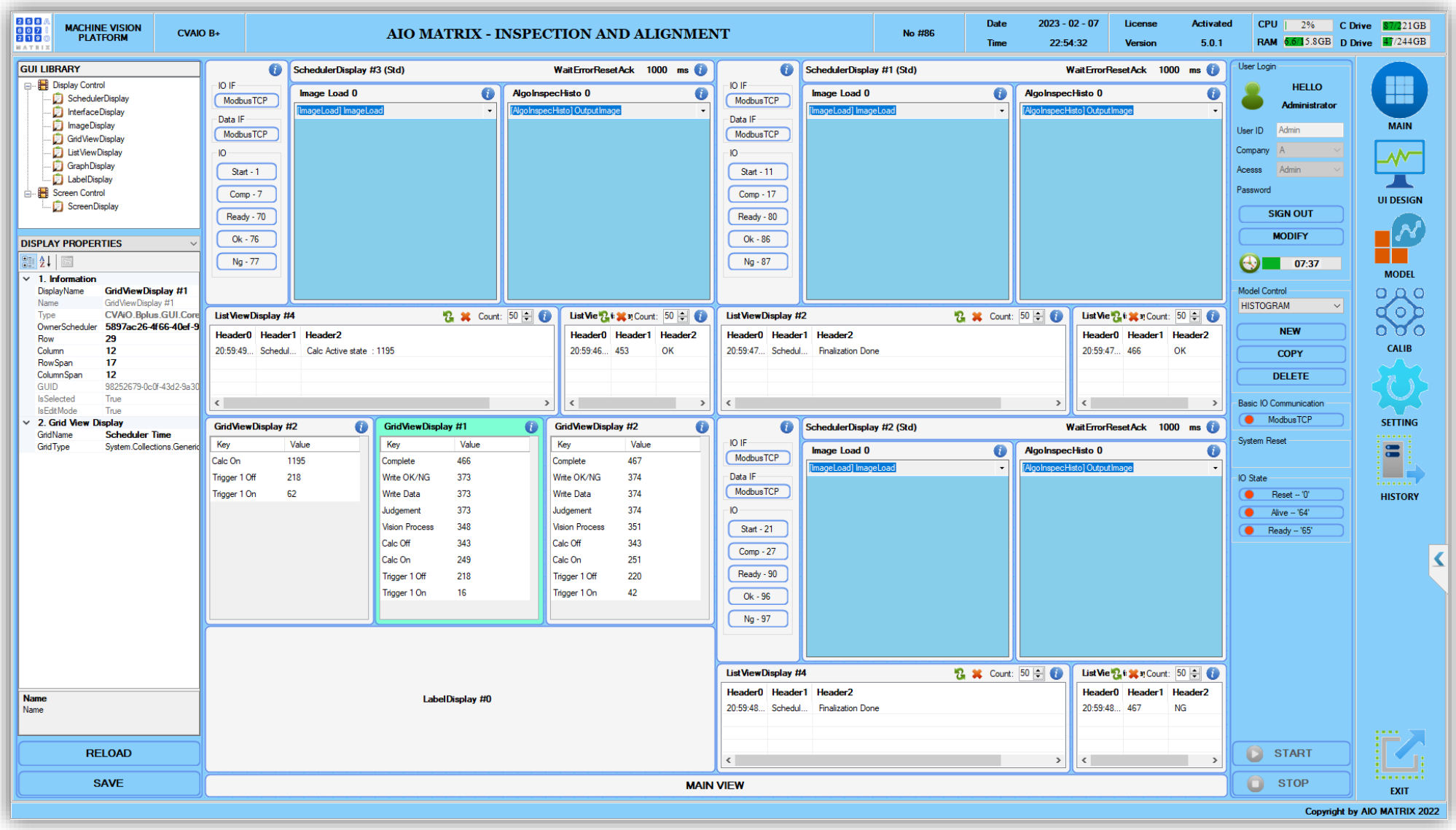


❖ **Mục đích:** Hiển thị các dữ liệu dạng đồ thị có trong luồng xử lý ảnh tương ứng ở thuộc tính Owner Scherduler. Người dùng có khả năng thêm bớt loại dữ liệu này bằng cách sửa đổi trong mã nguồn của Scheduler tương ứng. Tham khảo Section 6.3

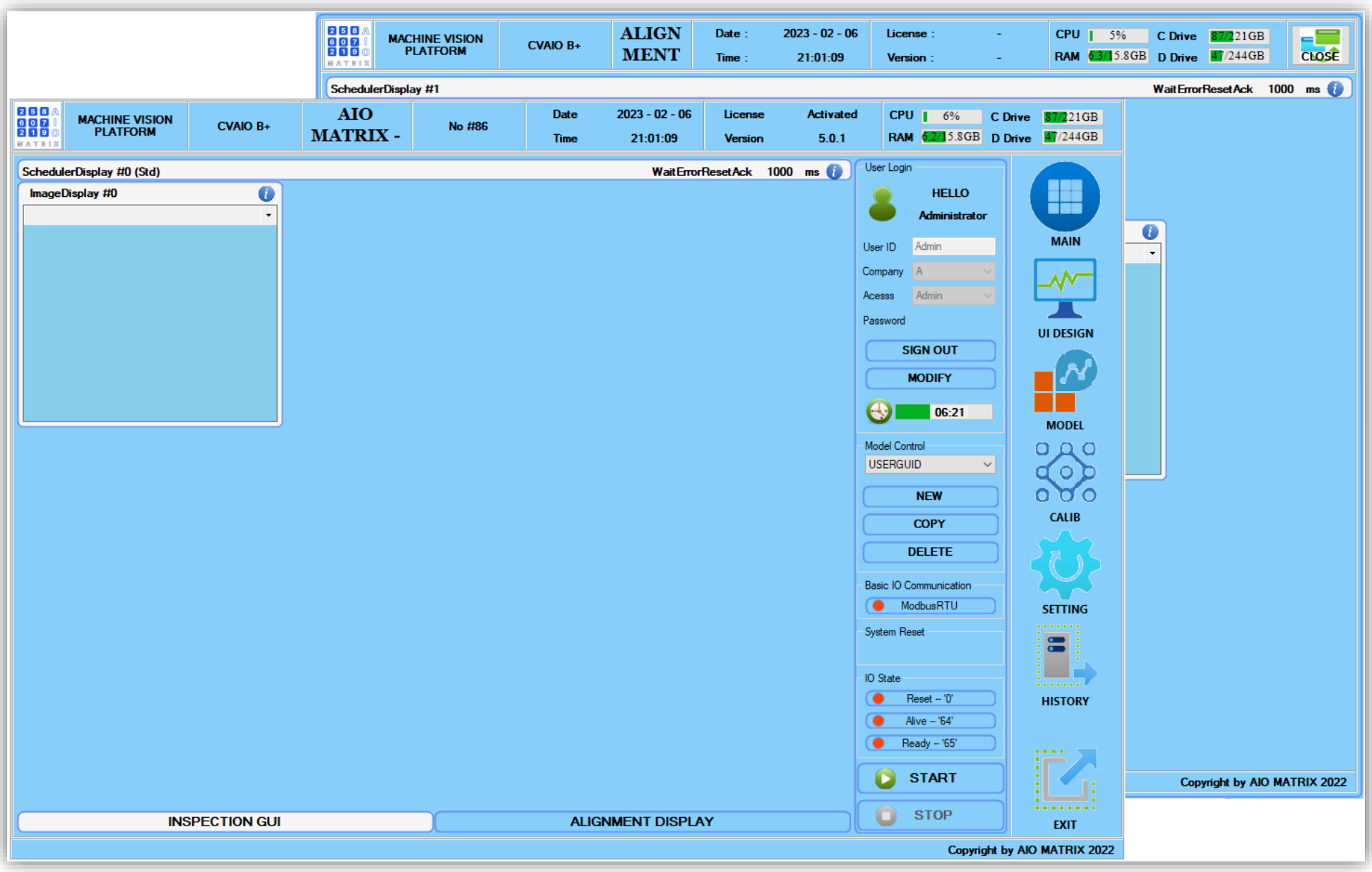
- **Display Properties:**
  - **(1) GraphName:** *Tên của dữ liệu dạng đồ thị*
  - **(2) CurveCount:** *Số lượng dữ liệu*
  - **(3) MaxData:** *Số lượng điểm tối đa của dữ liệu*
  - **(4) MainAxisTitle:** *Tiêu đề của trục chính*
  - **(5) SubAxisTitle:** *Tiêu đề của trục phụ*
  - **(6) XGridVisible:** *Hiển thị lưới của trục X*
  - **(7) YGridVisible:** *Hiển thị lưới của trục Y*
  - **(8) ZeroLineVisible:** *Hiển thị được tọa độ 0*

Case Study 1: UI Design

Giao diện của một chương trình có 3 luồng xử lý ảnh sử dụng truyền thông modbus TCP/IP.



Case Study 2: Multi Screen



CHAPTER 3: TOOL LIBRARY – THƯ VIỆN CÁC CÔNG CỤ XỬ LÝ ẢNH

Các nội dung sẽ được đề cập trong chương này:

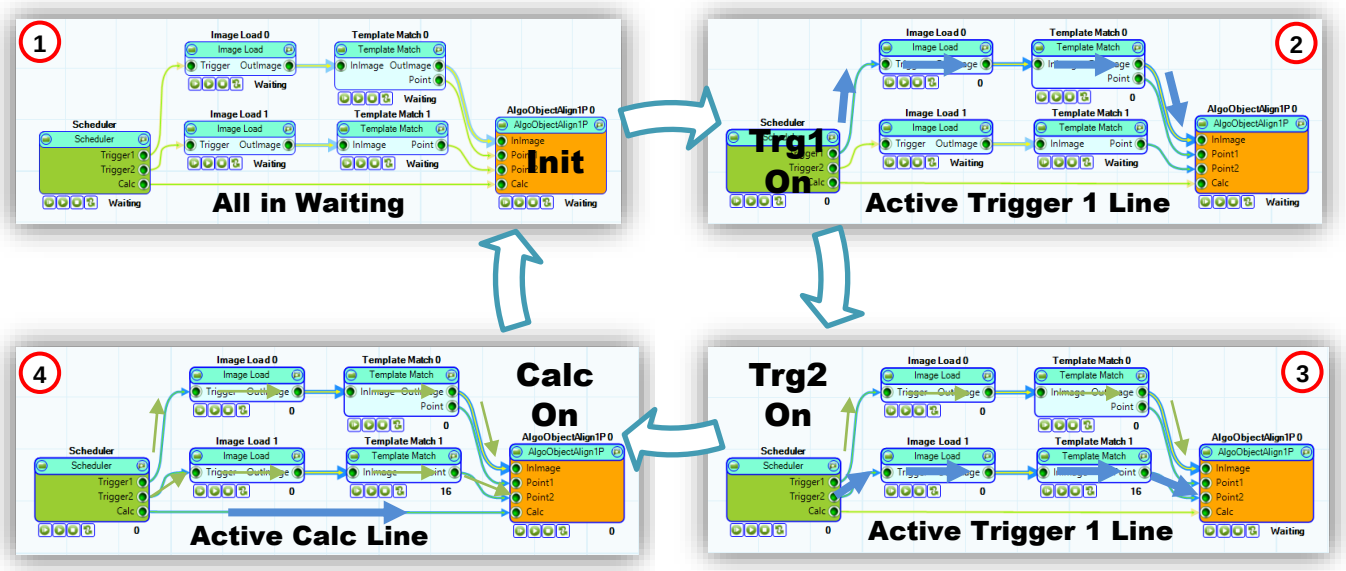
- Tổng quan về các công cụ xử lý ảnh
- Giải thích chi tiết các công cụ xử lý ảnh, đầu vào, đầu ra, thông số cài đặt
- Case Study 1: Kiểm tra bản mạch sử dụng histogram
- Case Study 2: Chương trình hiển thị trên 2 màn hình độc lập

3.1 Overview - Giới thiệu chung

Toàn bộ các công cụ xử lý ảnh sẽ được điều khiển bởi công cụ Scheduler. Đây là công cụ mặc định được tạo ra trong các process xử lý ảnh. Trong công cụ này sẽ có các tín hiệu IO để giao tiếp với PLC thông qua module giao tiếp đã được chon trong phần thiết kế GUI. Mỗi khi nhận được tín hiệu On từ PLC, Scheduler sẽ kích hoạt các Trigger thành active và từ đó tất cả các công cụ xử lý ảnh được nối với Trigger đó sẽ được thực thi theo concept lan truyền dọc theo các đường nối. Có tổng cộng 4 Trigger trong mỗi Scheduler. Trong trường hợp không sử dụng hết cả 4 triggers, người dùng có thể chọn chuột phải và remove. Sau khi tất cả các triggers cần thiết đã được thực thi, PLC sẽ On 1 tín hiệu Calc (nối từ Scheduler đến Algorithm) để báo cho công cụ Algorithm biết toàn bộ các tính toán đã thực hiện xong, tiếp đến công cụ Algorithm sẽ tổng hợp và phát định kết quả trước khi toàn tất toàn bộ các tính toán của 1 bài toán xử lý ảnh.

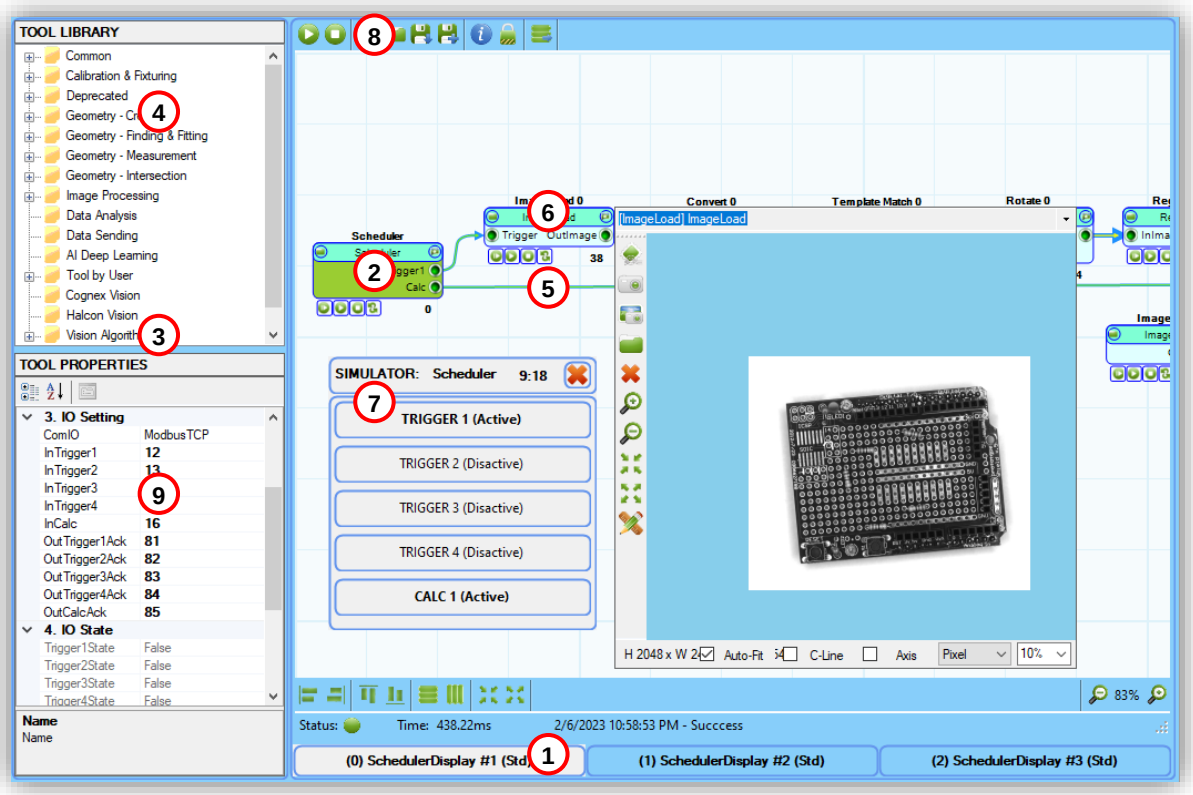
Hình dưới đây mô tả 1 process vision dùng để xác định vị trí của vật thể sử dụng 1 camera và 2 lần chụp riêng biệt sử dụng trigger 1,2.

- **Step 1:** Khởi tạo toàn bộ các params của tất cả các công cụ xử lý ảnh
- **Step 2:** Robot mang camera di chuyển đến vị trí chụp thứ nhất. Kích hoạt trigger 1 để thực hiện việc thu thập ảnh, truyền hình ảnh sang công cụ Template Matching 1 để tìm kiếm vị trí điểm mark. Kết quả tìm được sẽ được truyền sang công cụ thuật toán và chờ xử lý tiếp.
- **Step 3:** Sau khi robot di chuyển sang vị trí chụp tiếp theo. Kích hoạt trigger 2 để 1 lần nữa thực hiện việc thu thập hình ảnh và tìm kiếm vị trí điểm mark bằng công cụ Template Matching 2. Kết quả tìm được sẽ vẫn được truyền sang công cụ thuật toán và chờ xử lý tiếp.
- **Step 4:** Sau khi quá trình thu thập kết quả kết thúc. Kích hoạt tín hiệu Calc để truyền tín hiệu yêu cầu thực hiện tính toán kết quả và đưa ra phát định về cho các luồng xử lý để truyền dữ liệu sang cho PLC.



Hướng dẫn thao tác thiết kế thuật toán xử lý ảnh của từng Scheduler:

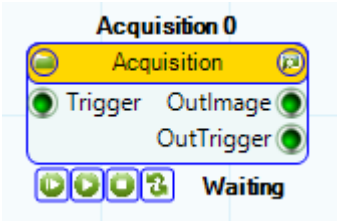
- ❖ **Step 1:** Lựa chọn Vision Process để bắt đầu thiết kế
- ❖ **Step 2:** Điều chỉnh số lượng trigger của Scheduler. Tham khảo Section 3.10
- ❖ **Step 3:** Bốc thả thuật toán Algo-Tool xử lý ảnh tương ứng. Tham khảo Section 3.11
- ❖ **Step 4:** Bốc thả các công cụ xử lý ảnh theo yêu cầu của từng bài toán
- ❖ **Step 5:** Kết nối các đầu ra và đầu vào của các công cụ theo yêu cầu của bài toán
- ❖ **Step 6:** Nháy đúp chuột vào từng công cụ xử lý ảnh để mở cửa sổ chỉnh sửa chi tiết
- ❖ **Step 7:** Nháy đúp chuốt vào Scheduler để mở cửa sổ mô phỏng các tín hiệu IO
- ❖ **Step 8:** Bấm Start để chạy toàn bộ process, Stop để khởi tạo lại toàn bộ các công cụ
- ❖ **Step 9:** Điều chỉnh thuộc tính IO giao tiếp PLC của các Scheduler



❖ **Note:**

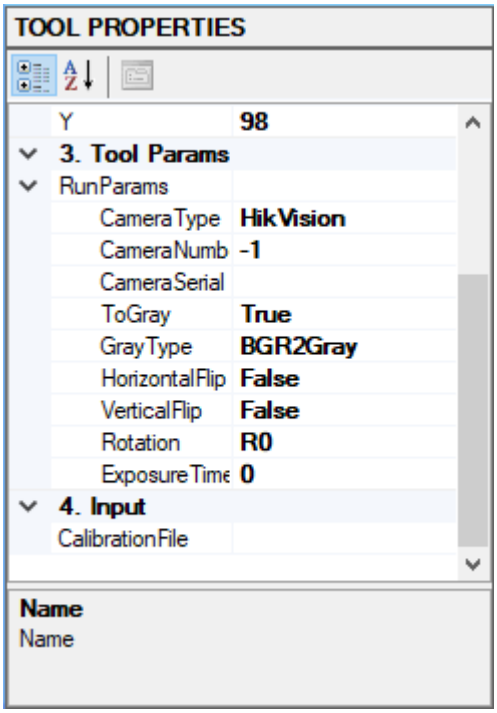
- ✓ *Sau khi kết nối đầu vào và đầu ra của các công cụ, cần chạy lại Step 8 để cập nhật kết quả của toàn bộ process.*
- ✓ *Module giao tiếp PLC được lấy từ các cài đặt khi thiết kết UI Design. Tham khảo tại Section 2.1*

3.2 Common Tool – Acquisition, Image File, Region, Template Match, Light Control



► **Acquisition:** *Cài đặt camera và thu thập hình ảnh* ◄

- **Input:**
  - (1) **Trigger:** *Tín hiệu bắt đầu chụp từ scheduler*
- **Output:**
  - (1) **OutImage:** *Hình ảnh nhận được*
  - (2) **OutTrigger:** *Trigger để tắt đèn nếu cần thiết*

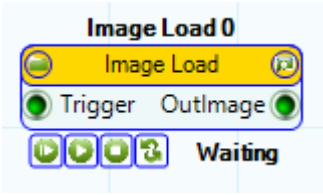


- **Properties:**
  - (1) **CameraType:** *Loại camera cần được kết nối*
  - (2) **CameraNumber:** *Số thứ tự của camera*
  - (3) **CameraSerial:** *Serial number của camera*
  - (4) **ToGray:** *Lựa chọn biến đổi thành ảnh đen trắng*
  - (5) **GrayType:** *Lựa chọn loại ảnh đen trắng*
  - (6) **HorizontalFlip:** *Đối xứng hình qua trục X*
  - (7) **VerticalFlip:** *Đối xứng hình qua trục Y*
  - (8) **Rotation:** *Xoay hình đi 90, 180, 279 độ*
  - (9) **ExposureTime:** *Thời gian phơi sáng của hình*
  - (10) **CalibrationFile:** *Load file căn chỉnh px-mm*

❖ **Note:**

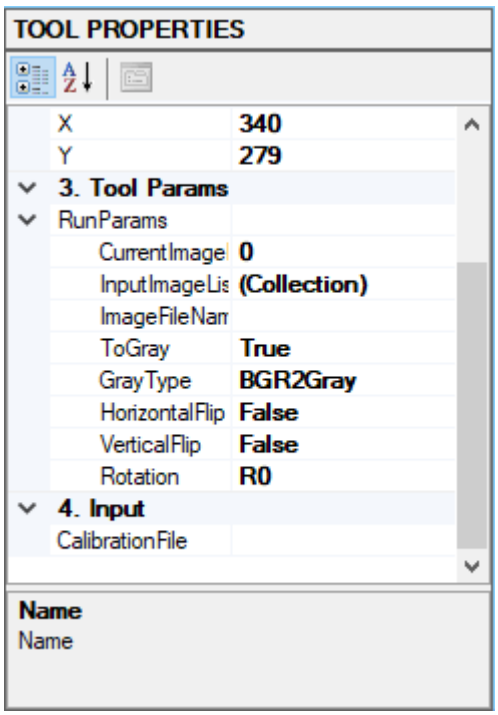
✓ *Sau khi được load vào, file căn chỉnh sẽ được đính cùng hình ảnh OutImage và có thể được sử dụng ở bất kỳ Tool nào.*

✓ *Hiện tại chương trình hỗ trợ kết nối với 4 loại camera thông dụng trên thị trường: HikVision, Basler, IMI, Dahua. Tham khảo section 6.4 nếu HW camera không nằm trong danh sách trên.*



► **Image Load:** *Lấy hình ảnh được lưu trong máy tính* ◄

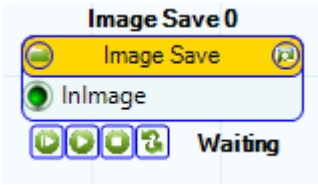
- **Input:**
  - (1) **Trigger:** *Tín hiệu bắt đầu load hình ảnh từ scheduler*
- **Output:**
  - (1) **OutImage:** *Hình ảnh được load lên từ PC*



- **Properties:**
  - (1) **CurrentImageIndex:** *Thứ tự ảnh trong danh sách*
  - (2) **InputImageList:** *Danh sách các ảnh sẽ được load*
  - (3) **ImageFileName:** *Tên của ảnh hiện tại được load*
  - (4) **ToGray:** *Lựa chọn biến đổi thành ảnh đen trắng*
  - (5) **GrayType:** *Lựa chọn loại ảnh đen trắng*
  - (6) **HorizontalFlip:** *Đối xứng hình qua trục X*
  - (7) **VerticalFlip:** *Đối xứng hình qua trục Y*
  - (8) **Rotation:** *Xoay hình đi 90, 180, 279 độ*
  - (9) **CalibrationFile:** *Load file căn chỉnh px-mm*

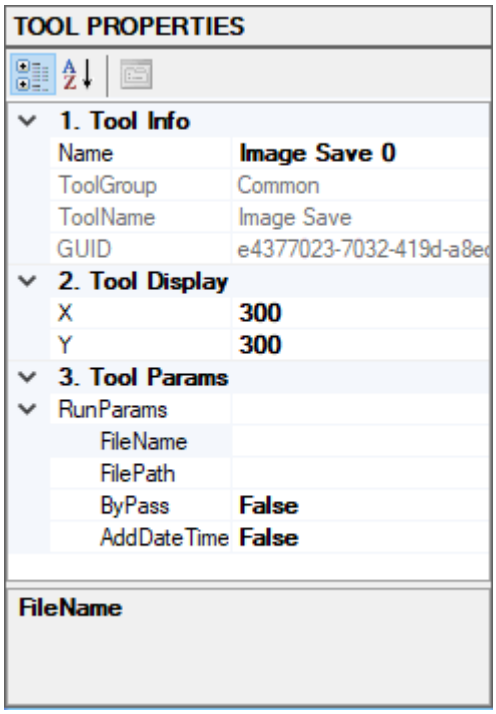
❖ **Note:**

✓ *Sau khi được load vào, file căn chỉnh sẽ được đính cùng hình ảnh OutImage và có thể được sử dụng ở bất kỳ Tool nào.*

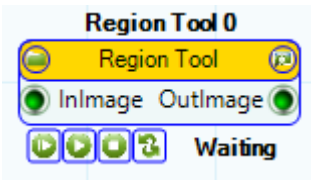


► **Image Save:** *Lưu hình ảnh ra 1 folder được lựa chọn trước* ◄

- **Input:**
  - **(1) InImage:** *Hình sẽ được lưu ra file*
- **Output:**

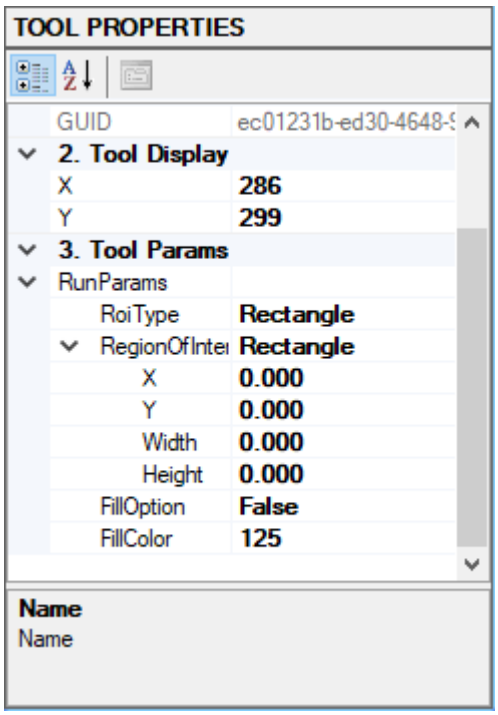


- **Properties:**
  - **(1) FileName:** *Tên file sẽ được lưu ra*
  - **(2) FilePath:** Đường dẫn đến folder lưu file
  - **(3) ByPass:** *Lựa chọn bỏ qua việc lưu file*
  - **(4) AddDateTime:** *Thêm ngày tháng vào tên file*



► **Region Tool:** *Lựa chọn vùng xử lý ảnh, điền đầy các vùng bên ngoài* ◄

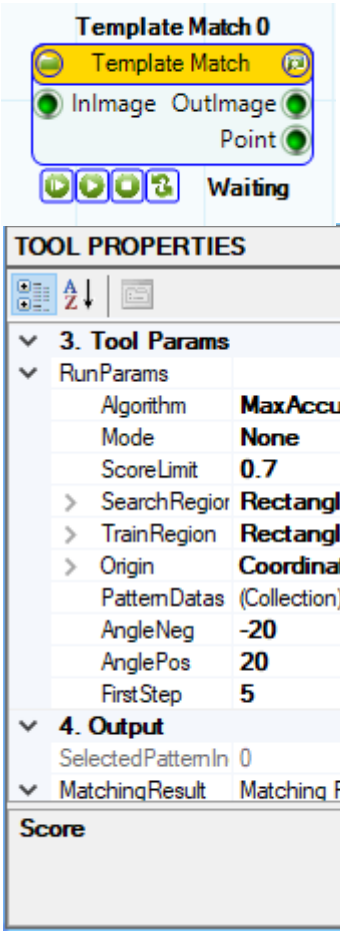
- **Input:**
  - **(1) InImage:** *Ảnh đầu vào*
- **Output:**
  - **(1) OutImage:** *Ảnh đầu ra*



- **Properties:**
  - **(1) RoiType:** *Thứ tự ảnh trong danh sách*
  - **(2) RegionOfInterest:** *Kích thước vùng Roi*
  - **(3) FillOption:** *Điền đầy bên ngoài vùng Roi*
  - **(4) FillColor:** *Màu của vùng sẽ được điền đầy*

❖ **Note:**

✓ *Có 4 loại vùng Roi: Rectangle - hình chữ nhật; Rotated Rectangle – hình chữ nhật xoay, Ellipse – hình Elip;.Polyline – hình đa giác*

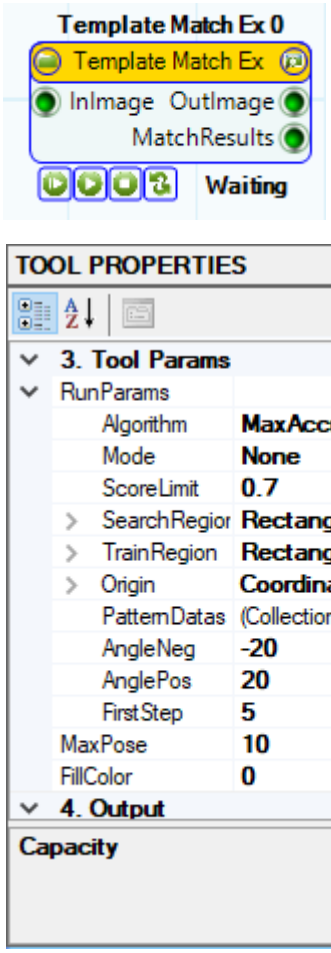


► **Template Match:** *Tìm kiếm vị trí của 1 hình được đăng ký trước (Single)* ◄

- **Input:**
  - (1) **InImage:** *Ảnh đầu vào*
- **Output:**
  - (1) **OutImage:** *Ảnh đầu ra*
  - (2) **Point:** *Vị trí hình giống template đã tìm được*

- **Properties:**
  - (1) **Algorithm:** *Thuật toán tìm kiếm*
  - (2) **Mode:** *Kiểu tìm kiếm nếu sử dụng multi-mark*
  - (3) **ScoreLimit:** *Giới hạn Score của quá trình tìm kiếm*
  - (4) **SearchRegion:** *Vùng tìm kiếm*
  - (5) **TrainRegion:** *Vùng đăng ký Pattern*
  - (6) **Origin:** *Tọa độ gốc của Pattern*
  - (7) **PatternData:** *Các pattern đã được đăng ký*
  - (8) **AngleNeg:** *Giới hạn dưới của góc tìm kiếm*
  - (9) **AnglePos:** *Giới hạn trên của góc tìm kiếm*
  - (10) **FirstStep:** *Bước nhảy của góc tìm kiếm ban đầu*

❖ **Note:**
- ✓ *Các Output khác: Score, SelectedPattern,MatchingResult*
- ✓ *Multi-Mark: Sử dụng trong trường hợp điểm mark bị mờ, làm cho giá trị Score nhỏ hơn ScoreLimit*



►**Template Match Ex:** *Tìm kiếm vị trí của 1 hình được đăng ký trước (Multi)* ◄

- **Input:**
  - (1) **InImage:** *Tín hiệu bắt đầu load hình ảnh từ scheduler*
- **Output:**
  - (1) **OutImage:** *Hình ảnh được load lên từ PC*
  - (2) **MatchResults:** *Danh sách kết quả tìm kiếm*

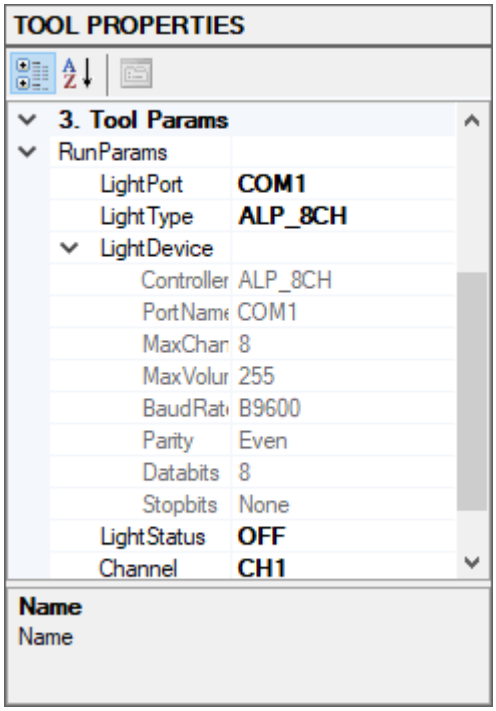
- **Properties:**
  - (1) **Algorithm:** *Thuật toán tìm kiếm*
  - (2) **Mode:** *Kiểu tìm kiếm nếu sử dụng multi-mark*
  - (3) **ScoreLimit:** *Giới hạn Score của quá trình tìm kiếm*
  - (4) **SearchRegion:** *Vùng tìm kiếm*
  - (5) **TrainRegion:** *Vùng đăng ký Pattern*
  - (6) **Origin:** *Tọa độ gốc của Pattern*
  - (7) **PatternData:** *Các pattern đã được đăng ký*
  - (8) **AngleNeg:** *Giới hạn dưới của góc tìm kiếm*
  - (9) **AnglePos:** *Giới hạn trên của góc tìm kiếm*
  - (10) **FirstStep:** *Bước nhảy của góc tìm kiếm ban đầu*
  - (11) **MaxPose:** *Giới hạn số lượng tìm kiếm*
  - (12) **FillColor:** *Mầu điền đầy để tìm kiếm tiếp tục*

❖ **Note:**
- ✓ *Đây là công cụ mở rộng của Template Match. Việc tìm kiếm sẽ được thực hiện giống như Template Match sau đó sẽ tiến hành xóa bỏ hình đã tìm kiếm được bằng cách điền đầy vùng tìm được với Fill Color*

✓

► **Light Control:** *Điều khiển bật tắt đèn thông qua cổng serial port* ◄

- **Input:**
  - **(1) InImage:** *Ảnh đầu vào*
- **Output:**
  -



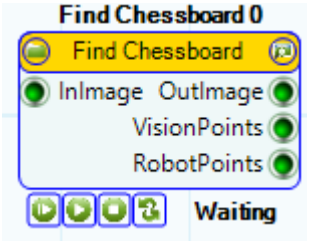
- **Properties:**
  - **(1) LightPort:** *Lựa chọn cổng kết nối với Controller*
  - **(2) LightType:** *Loại controller của đèn*
  - **(3) LighDevice:** *Các thông số cài đặt Interface*
  - **(4) LightStatus:** *Trạng thái sẽ được cài đặt*
  - **(5) Channel:** *Kênh điều khiển tương ứng của đèn*
  - **(6) Volume:** *Mức độ sáng của đèn*

❖ **Note:**

✓ *Volume = 0 tương ứng với việc điều khiển tắt đèn*

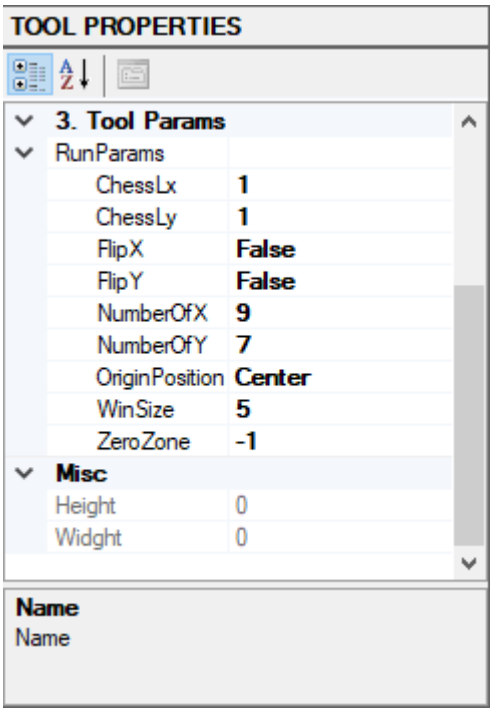
✓ *Thêm bớt các loại đèn có thể tìm thấy trong project:CVAiO.Bplus.LightingDevice*

3.3 Calibration & Fixturing – Find Chessboard, Find Calib Matrix, Fiture



► **Find Chessboard:** *Tìm kiếm vị trí các góc của ô bàn cờ* ◄

- **Input:**
  - **(1) InImage:** *Ảnh đầu vào*
- **Output:**
  - **(1) OutImage:** *Ảnh đầu ra*
  - **(2) VisionPoints:** *Tọa độ các góc tính theo px*
  - **(3) RobotPoints:** *Tọa độ các góc tính theo mm*

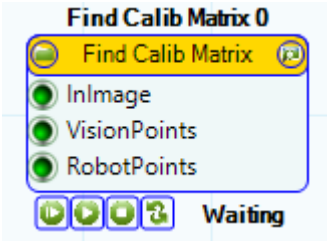


- **Properties:**
  - **(1) ChessLx:** *Độ rộng ô bàn cờ theo trục x*
  - **(2) ChessLy:** *Độ rộng ô bàn cờ theo trục y*
  - **(3) FlipX:** *Lấy đối xứng vị trí theo phương x*
  - **(4) FlipY:** *Lấy đối xứng vị trí theo phương y*
  - **(5) OriginPosition:** *Gốc tọa độ của hệ truc oxy*
  - **(6) WinSize:** *Kích thước vùng tìm kiếm*
  - **(7) ZeroZone:** *Kích thước vùng Dead Region*

❖ **Note:**

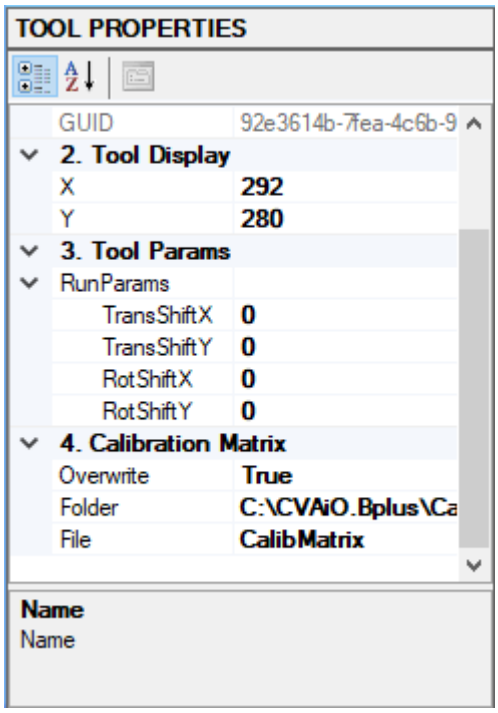
✓ *Đây là công cụ hỗ trợ việc căn chỉnh manual. Chi tiết về cách sử dụng tham khảo Section 4.2*

✓ *Thuật toán trong OpenCV*



► **Find Calib Matrix:** *Tính toán ma trận căn trỉnh dựa trện dữ liệu đầu vào* ◄

- **Input:**
  - (1) **InImage:** *Ảnh đầu vào*
  - (2) **VisionPoints:** *Dữ liệu trong hệ tọa độ px*
  - (3) **RobotPoints:** *Dữ liệu trong hệ tọa độ mm*
- **Output:**

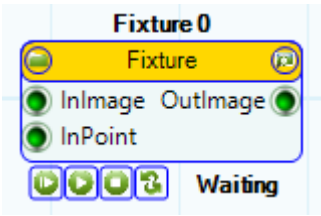


- **Properties:**
  - (1) **TransShiftX:** *Offset giá trị căn chỉnh oxy theo trục X*
  - (2) **TransShiftY:** *Offset giá trị căn chỉnh oxy theo trục Y*
  - (3) **RotShiftX:** *Offset tâm xoay tìm được theo trục X*
  - (4) **RotShiftY:** *Offset tâm xoay tìm được theo trục Y*
  - (5) **Overwrite:** *Lưu đè nếu có file căn chỉnh cùng tên*
  - (6) **Folder:** *Vị trí lưu file căn chỉnh*
  - (7) **File:** *Tên file căn chỉnh*

❖ **Note:**

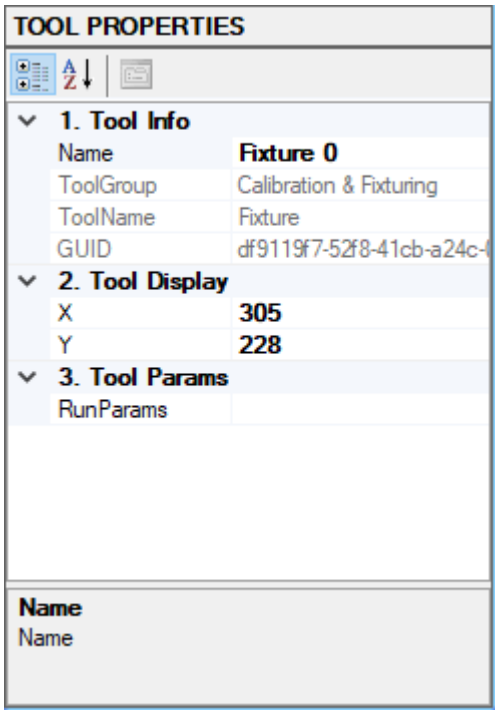
✓ *Công cụ này được sử dụng trong cả 2 trường hợp căn chỉnh manual và Auto*

✓ *Chi tiết về cách sử dụng có thể tham khảo tại Chapter 4*



► **Fixture:** *Định nghĩa 1 hệ tọa độ mới và gắn vào hình ảnh đầu ra* ◄

- **Input:**
  - (1) **InImage:** *Ảnh đầu vào*
  - (2) **InPoint:** *Tọa độ của hệ trục tọa độ mới*
- **Output:**
  - (1) **OutImage:** *Ảnh đầu ra có đính kèm hệ tọa độ Fixture*



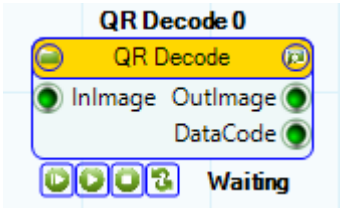
- **Properties:**

❖ **Note:**

✓ *Đây thường là công cụ nối tiếp sau Template Match*

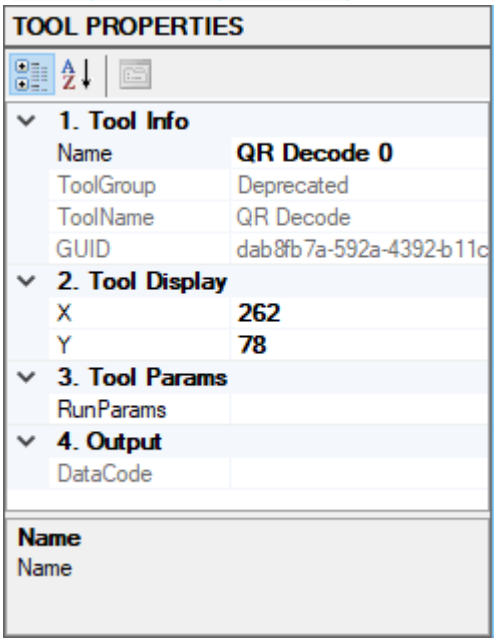
✓ *Hình ảnh sau khi đính kèm hệ tọa độ Fixute thường được sử dụng với công cụ FindLine, FindCircle do các tool caliper sẽ chạy theo hệ tọa độ Fixture giúp giảm độ rộng của các vị trí caliper.*

3.4 Deprecated – QR Decode



► **QR Decode:** *Giải mã QR Code* ◄

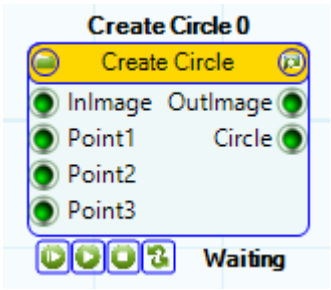
- **Input:**
  - (1) **InImage:** *Ảnh đầu vào*
- **Output:**
  - (1) **OutImage:** *Ảnh đầu ra*
  - (2) **DataCode:** *Kết quả giải mã QR Code*



- **Properties:**

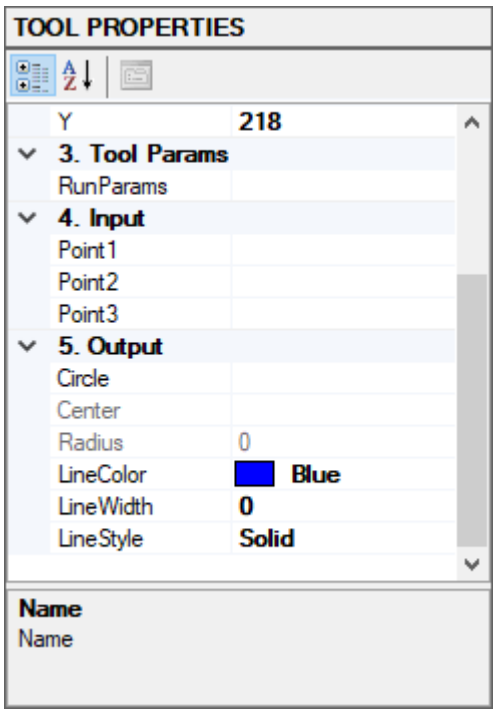
❖ **Note:**

3.5 Geometry Creation – Create Circle, Create Line, Create Rectangle



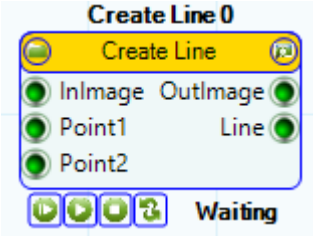
► **Create Circle:** *Tạo đường tròn đi qua 3 điểm đầu vào* ◄

- **Input:**
  - (1) **InImage:** *Ảnh đầu vào*
  - (2) **Points1,2,3:** *Điểm để tạo đường tròn*
- **Output:**
  - (1) **OutImage:** *Ảnh đầu ra*
  - (2) **Circle:** *Đường tròn đi qua 2 điểm đầu vào*



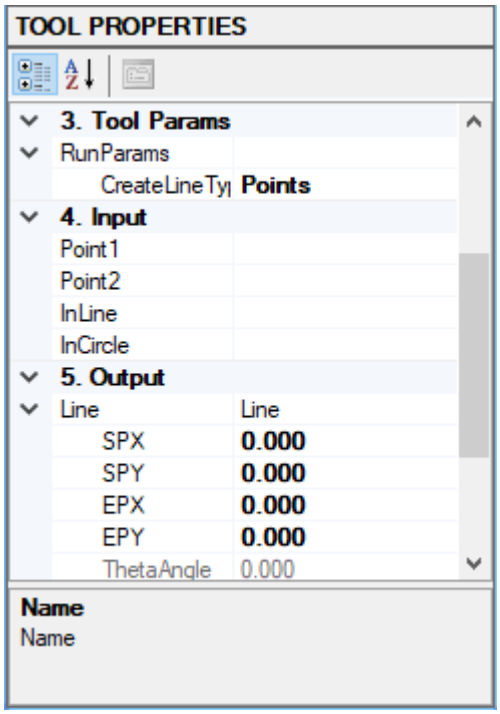
- **Properties:**

❖ **Note:**



► **Create Circle:** *Tạo đường tròn đi qua 3 điểm đầu vào* ◄

- **Input:**
  - (1) **InImage:** *Ảnh đầu vào*
  - (2) **Points1,2:** *Điểm để tạo đường thẳng*
- **Output:**
  - (1) **OutImage:** *Ảnh đầu ra*
  - (2) **Line:** *Đường thẳng đi qua 2 điểm*

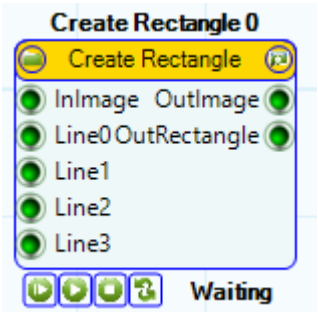


- **Properties:**
  - (1) **CreateLineType:** *Phương pháp tạo đường thẳng*

➢ **Note:**

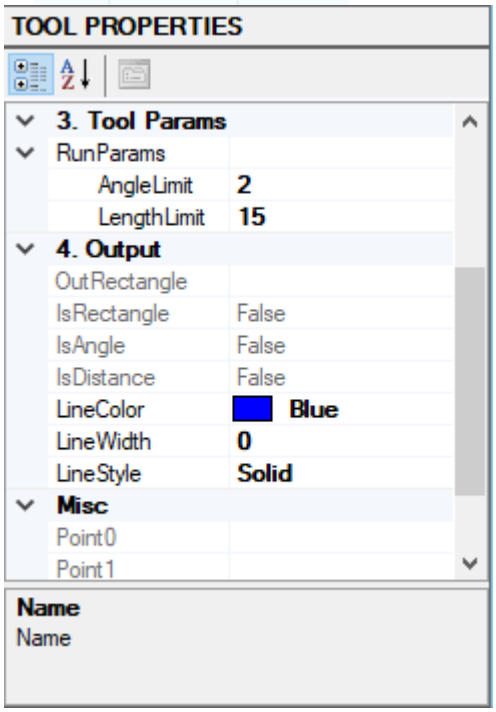
✓ *Other Input: Inline, InCircle*

✓ *Ngoài phương pháp tạo ra đường thẳng qua 2 điểm còn có các phương pháp khác như LineParrale, LinePerpendicurla,CircleTangent sử dụng các đầu vào tương ứng*



► **Create Rectangle:** *Tạo ra hình tứ giác cho công cụ Warping Perspective* ◄

- **Input:**
  - (1) **InImage:** *Ảnh đầu vào*
  - (2) **Line1,2,3,4:** *Các đường thẳng để tạo hình tứ giác*
- **Output:**
  - (1) **OutImage:** *Ảnh đầu ra*
  - (2) **Rectangle:** *Hình tứ giác đầu ra*



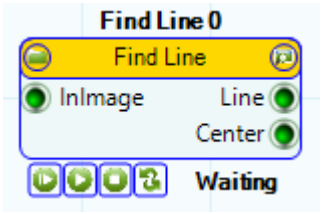
- **Properties:**
  - (1) **AngleLimit:** *Giới hạn góc khi tạo hình tứ giác*
  - (2) **LengthLimit:** *Giới hạn dưới độ dài cạnh tứ giác*

➢ **Note:**

✓ *Đây là công cụ thường được sự dụng để tạo điểm warping. Tìm hiểu thêm thông tin liên quan đến warping tại section 3.9*

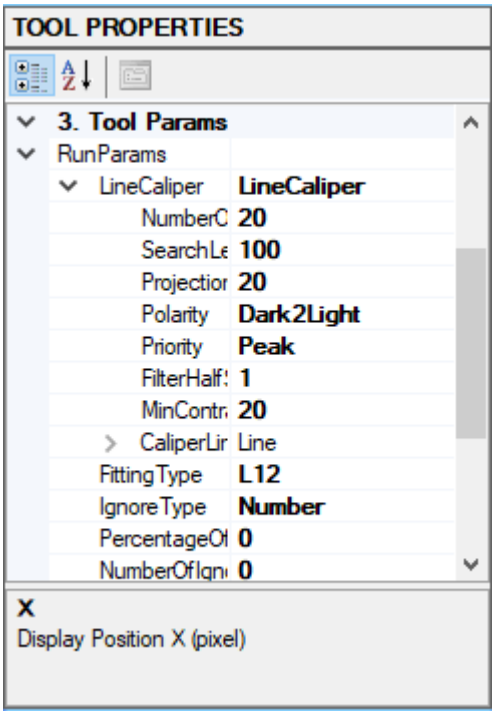
✓ *Lý thuyết về warping Perspective ở OpenCV*

3.6 Geometry Finding & Fitting – Find Line, Find Circle, Find Corner, Find Points



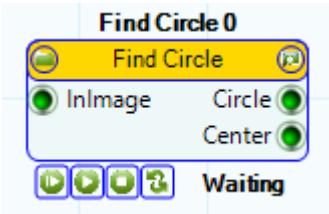
► **Find Line:** *Tìm kiếm đường thẳng* ◄

- **Input:**
  - (1) **InImage:** *Ảnh đầu vào*
- **Output:**
  - (1) **Line:** *Đường thẳng tìm được*
  - (2) **Center:** *Tọa độ trung điểm của đoạn thẳng*



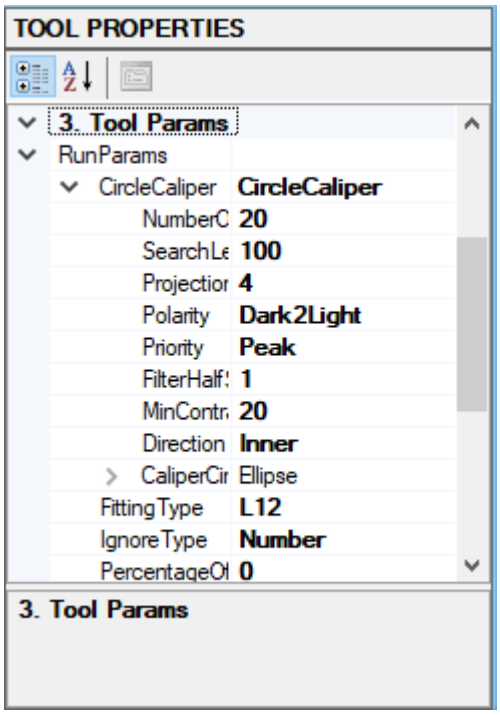
- **Properties:**
  - (1) **LineCaliper:** *Đoạn thẳng chứa các caliper tìm kiếm*
  - (2) **NumberCaliper:** *Số lượng caliper sẽ tìm kiếm*
  - (3) **SearchLength:** *Độ dài của 1 caliper*
  - (4) **ProjectionLength:** *Độ rộng của 1 caliper*
  - (5) **Polarity:** *Loại phân cực khi tìm kiếm*
  - (6) **Priority:** *Mức độ ưu tiên khi tìm kiếm*
  - (7) **FilterHalfPixel:** *Kích thước của ½ pixel lọc*
  - (8) **MinContrast:** *Giới hạn tương phản khi tìm kiếm*
  - (9) **FittingType:** *Phương pháp xấp xỉ đường thẳng*
  - (10) **IgnoreType:** *Phương pháp loại bỏ điểm nhiễu*
  - (11) **PercentageOfIgnore:** *Phần trăm loại bỏ*
  - (12) **NumberOfIgnore:** *Số lượng loại bỏ*
  - (13) **UseNumberOfCalipers:** *Số lượng caliper dùng*

➢ **Note:**
- ✓ *LineCaliper sẽ chạy theo hệ tọa độ của Fixture*



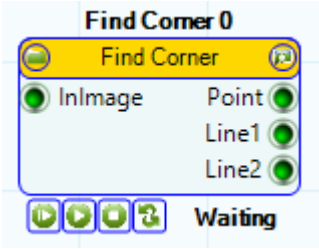
► **Find Circle:** *Tìm kiếm đường tròn* ◄

- **Input:**
  - (1) **InImage:** *Ảnh đầu vào*
- **Output:**
  - (1) **Circle:** *Đường tròn tìm được*
  - (2) **Center:** *Tọa độ trung điểm của đường tròn*



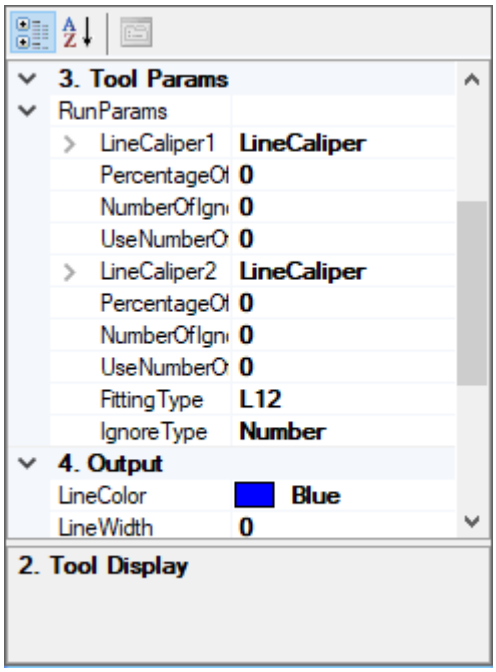
- **Properties:**
  - (1) **CircleCaliper:** *Đường tròn chứa các caliper*
  - (2) **NumberCaliper:** *Số lượng caliper sẽ tìm kiếm*
  - (3) **SearchLength:** *Độ dài của 1 caliper*
  - (4) **ProjectionLength:** *Độ rộng của 1 caliper*
  - (5) **Polarity:** *Loại phân cực khi tìm kiếm*
  - (6) **Priority:** *Mức độ ưu tiên khi tìm kiếm*
  - (7) **FilterHalfPixel:** *Kích thước của ½ pixel lọc*
  - (8) **MinContrast:** *Giới hạn tương phản khi tìm kiếm*
  - (9) **Direction:** *Hướng tìm kiếm*
  - (10) **FittingType:** *Phương pháp xấp xỉ đường tròn*
  - (11) **IgnoreType:** *Phương pháp loại bỏ điểm nhiễu*
  - (12) **PercentageOfIgnore:** *Phần trăm loại bỏ*
  - (13) **NumberOfIgnore:** *Số lượng loại bỏ*
  - (14) **UseNumberOfCalipers:** *Số lượng caliper dùng*

➢ **Note:**
- ✓ *CircleCaliper sẽ chạy theo hệ tọa độ của Fixture*



► **Find Corner:** *Tìm kiếm góc giao nhau của 2 đường thẳng* ◄

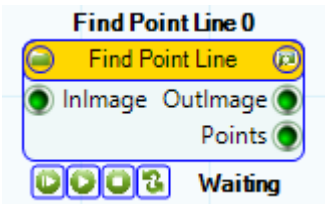
- **Input:**
  - **(1) InImage:** *Ảnh đầu vào*
- **Output:**
  - **(1) Point:** *Điểm giao nhau của 2 đường thẳng*
  - **(2) Line1,2:** *Đường thẳng tạo lên góc cần tìm*



- **Properties:** *(Tham khảo công cụ Find Line)*

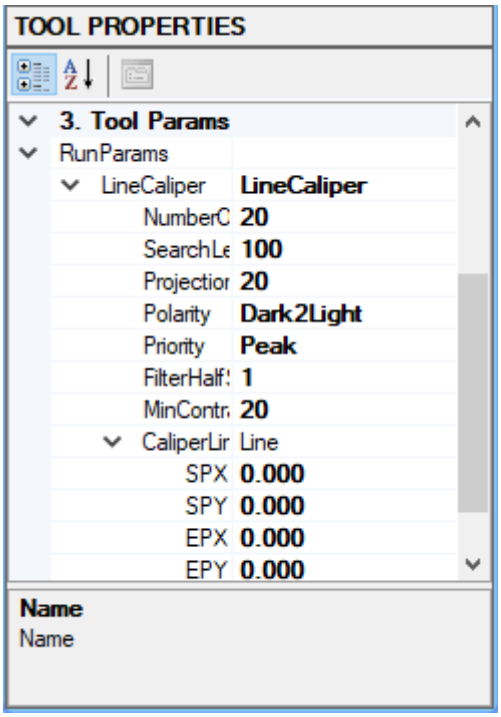
➢ **Note:**

✓ *Find Corner là sự kết hợp của 2 công cụ Find Line*



► **Find Point Line:** *Tìm kiếm danh sách các điểm thông qua CaliperLine* ◄

- **Input:**
  - **(1) InImage:** *Ảnh đầu vào*
- **Output:**
  - **(1) OutImage:** *Ảnh đầu ra*
  - **(2) Points:** *Danh sách các điểm tìm được của mỗi Caliper*

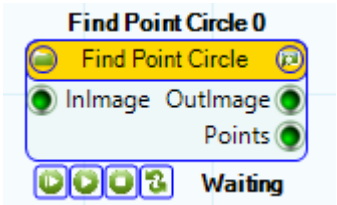


- **Properties:**
  - **(1) LineCaliper:** *Đường thẳng chứa các caliper*
  - **(2) NumberCaliper:** *Số lượng caliper sẽ tìm kiếm*
  - **(3) SearchLength:** *Độ dài của 1 caliper*
  - **(4) ProjectionLength:** *Độ rộng của 1 caliper*
  - **(5) Polarity:** *Loại phân cực khi tìm kiếm*
  - **(6) Priority:** *Mức độ ưu tiên khi tìm kiếm*
  - **(7) FilterHalfPixel:** *Kích thước của ½ pixel lọc*
  - **(8) MinContrast:** *Giới hạn tương phản khi tìm kiếm*

➢ **Note:**

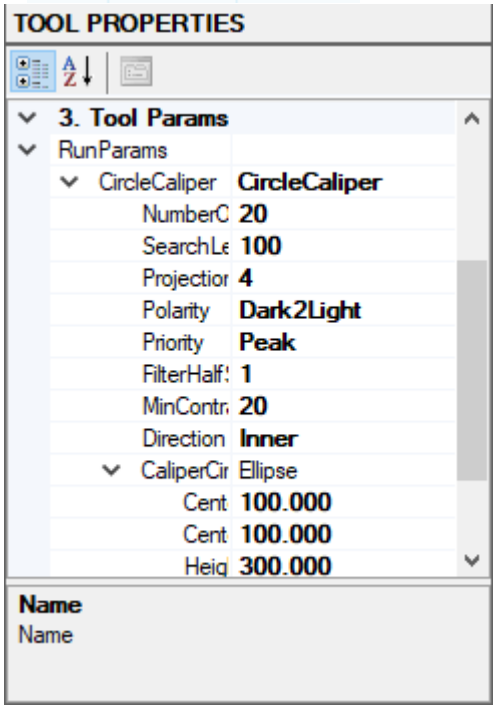
✓ *LineCaliper sẽ chạy theo hệ tọa độ của Fixture*

✓ *Find Line là công cụ mở rộng của Find Point Line sau khi thực hiện xấp sỉ các điểm tìm được bằng 1 đường thẳng*



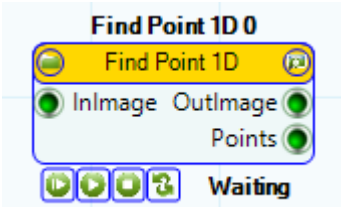
► **Find Point Circle:** *Tìm kiếm dành sách các điểm thông qua CaliperCircle* ◄

- **Input:**
  - (1) **InImage:** *Ảnh đầu vào*
- **Output:**
  - (1) **OutImage:** *Ảnh đầu ra*
  - (2) **Points:** *Danh sách các điểm tìm được của mỗi Caliper*



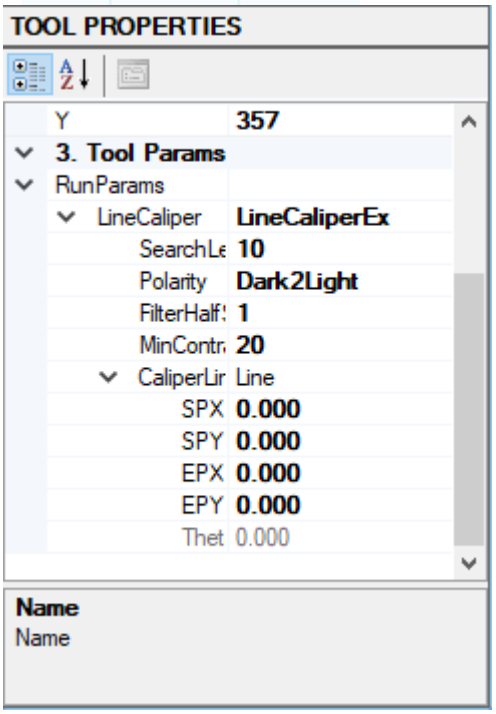
- **Properties:**
  - (1) **CircleCaliper:** *Đường tròn chứa các caliper*
  - (2) **NumberCaliper:** *Số lượng caliper sẽ tìm kiếm*
  - (3) **SearchLength:** *Độ dài của 1 caliper*
  - (4) **ProjectionLength:** *Độ rộng của 1 caliper*
  - (5) **Polarity:** *Loại phân cực khi tìm kiếm*
  - (6) **Priority:** *Mức độ ưu tiên khi tìm kiếm*
  - (7) **FilterHalfPixel:** *Kích thước của ½ pixel lọc*
  - (8) **MinContrast:** *Giới hạn tương phản khi tìm kiếm*

➢ **Note:**
- ✓ *CircleCaliper sẽ chạy theo hệ tọa độ của Fixture*
- ✓ *Find Circle là công cụ mở rộng của Find Point Circle sau khi thực hiện xấp sỉ các điểm tìm được bằng 1 đường tròn*



► **Find Point 1D:** *Tìm kiếm các điểm nằm trên 1 đường thẳng* ◄

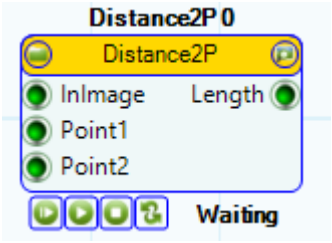
- **Input:**
  - (1) **InImage:** *Ảnh đầu vào*
- **Output:**
  - (1) **OutImage:** *Ảnh đầu ra*
  - (2) **Points:** *Danh sách các điểm tìm được dọc theo đường caliper*



- **Properties:**
  - (1) **LineCaliper:** *Đường thẳng chứa các caliper*
  - (2) **SearchLength:** *Độ dài của 1 caliper*
  - (3) **Polarity:** *Loại phân cực khi tìm kiếm*
  - (4) **FilterHalfPixel:** *Kích thước của ½ pixel*
  - (5) **MinContrast:** *Giới hạn tương phản khi tìm kiếm*

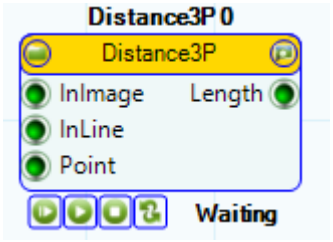
➢ **Note:**
- ✓ *LineCaliper sẽ chạy theo hệ tọa độ của Fixture*

3.7 Geometry Measurement – Distance Points, Distance Line, Distance Circle



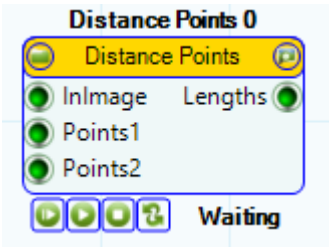
► **Distance2P:** *Tính khoảng cách giữa 2 điểm* ◄

- **Input:**
  - (1) **InImage:** *Ảnh đầu vào*
  - (2) **Point1,2:** *Điểm đầu vào*
- **Output:**
  - (1) **Length:** *Khoảng cách giữa 2 điểm*



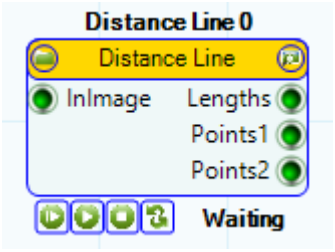
► **Distance3P:** *Tính toán khoảng cách từ 1 điểm đến 1 đường thẳng* ◄

- **Input:**
  - (1) **InImage:** *Ảnh đầu vào*
  - (2) **InLine:** *Đường thẳng đầu vào*
  - (3) **Point:** *Điểm đầu vào*
- **Output:**
  - (1) **Length:** *Khoảng cách từ điểm đến đường thẳng caliper*



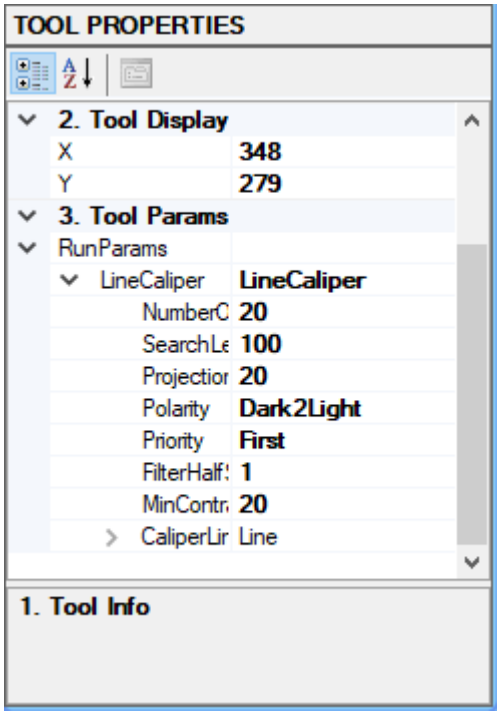
► **Distance Points:** *Tính toán khoảng cách giữa 2 danh sách các điểm* ◄

- **Input:**
  - (1) **InImage:** *Ảnh đầu vào*
  - (2) **Points1,2:** *Danh sách các điểm đầu vào*
- **Output:**
  - (1) **Lengths:** *Các khoảng cách tính toán được*



► **Distance Line:** *Khoảng cách giữa các điểm Black↔White theo đường thẳng* ◄

- **Input:**
  - (1) **InImage:** *Ảnh đầu vào*
- **Output:**
  - (1) **Lengths:** *Danh sách các khoảng cách đầu ra*
  - (2) **Points1,2:** *Danh sách các điểm tìm được của Caliper*

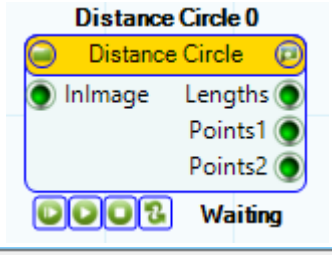


- **Properties:**
  - (1) **CircleCaliper:** *Đường tròn chứa các caliper*
  - (2) **NumberCaliper:** *Số lượng caliper sẽ tìm kiếm*
  - (3) **SearchLength:** *Độ dài của 1 caliper*
  - (4) **ProjectionLength:** *Độ rộng của 1 caliper*
  - (5) **Polarity:** *Loại phân cực khi tìm kiếm*
  - (6) **Priority:** *Mức độ ưu tiên khi tìm kiếm*
  - (7) **FilterHalfPixel:** *Kích thước của ½ pixel lọc*
  - (8) **MinContrast:** *Giới hạn tương phản khi tìm kiếm*

➢ **Note:**

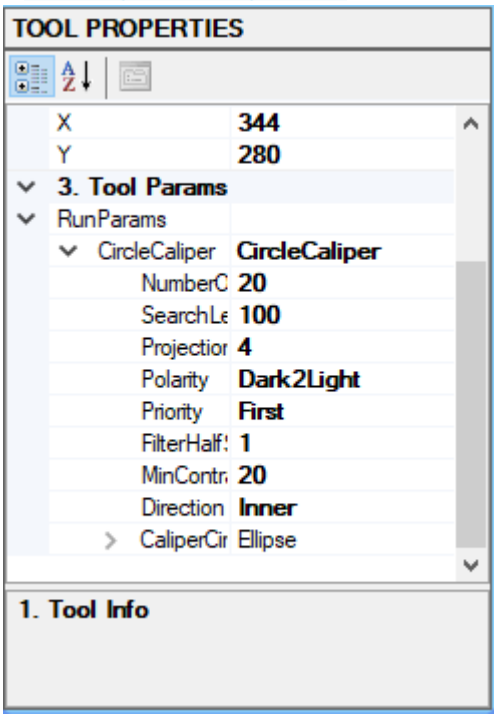
✓ *CircleCaliper sẽ chạy theo hệ tọa độ của Fixture*

✓ *Find Circle là công cụ mở rộng của Find Point Circle sau khi thực hiện xấp sỉ các điểm tim được bằng 1 đường tròn*



► **Distance Circle:** *Khoảng cách giữa các điểm Black ↔White theo đường tròn* ◄

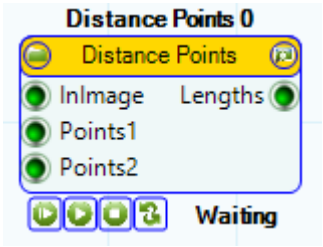
- **Input:**
  - (1) **InImage:** *Ảnh đầu vào*
- **Output:**
  - (1) **OutImage:** *Danh sách các khoảng cách đầu ra*
  - (2) **Points1,2:** *Danh sách các điểm tìm được của Caliper*



- **Properties:**
  - (1) **CircleCaliper:** *Đường tròn chứa các caliper*
  - (2) **NumberCaliper:** *Số lượng caliper sẽ tìm kiếm*
  - (3) **SearchLength:** *Độ dài của 1 caliper*
  - (4) **ProjectionLength:** *Độ rộng của 1 caliper*
  - (5) **Polarity:** *Loại phân cực khi tìm kiếm*
  - (6) **FilterHalfPixel:** *Kích thước của ½ pixel*
  - (7) **MinContrast:** *Giới hạn tương phản khi tìm kiếm*
  - (8) **Direction:** *Hướng bắt đầu tìm kiếm*

➢ **Note:**

✓ *CircleCaliper sẽ chạy theo hệ tọa độ của Fixture*



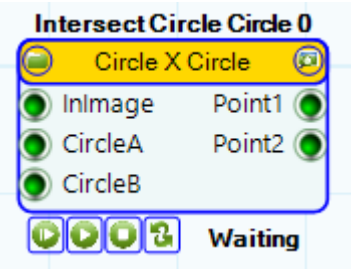
► **Distance Line:** *Khoảng cách giữa các điểm đầu vào* ◄

- **Input:**
  - (1) **InImage:** *Ảnh đầu vào*
  - (2) **Points1,2:** *Danh sách các điểm đầu vào*
- **Output:**
  - (1) **Lengths:** *Danh sách các khoảng cách đầu ra*

➢ **Note:**

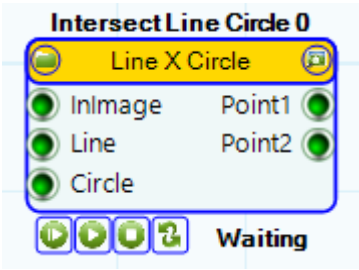
✓ *Points 1,2 thường là kết quả của công cụ tìm kiếm các điểm Find Point Line hoặc Find Point Circle*

3.8 Geometry Intersection – Circle x Circle, Line x Circle, Line x Line



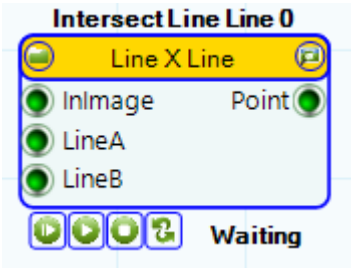
► **Intersect Circle Circle:** *Tìm điểm cắt nhau của 2 đường tròn* ◄

- **Input:**
  - (1) **InImage:** *Ảnh đầu vào*
  - (2) **Circle A,B:** *Các đường tròn đầu vào*
- **Output:**
  - (1) **Point1,2:** *Điểm cắt nhau của 2 đường tròn*



► **Intersect Line Circle:** *Tìm điểm cắt nhau của 2 đường tròn* ◄

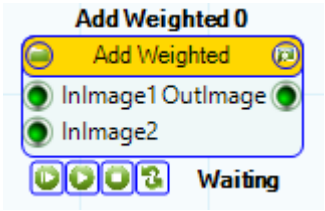
- **Input:**
  - (1) **InImage:** *Ảnh đầu vào*
  - (2) **Line:** *Đường thẳng đầu vào*
  - (3) **Circle:** *Đường tròn đầu vào*
- **Output:**
  - (1) **Point1,2:** *Điểm cắt nhau của 2 đường nếu có*



► **Intersect Line Circle:** *Tìm điểm cắt nhau của 2 đường tròn* ◄

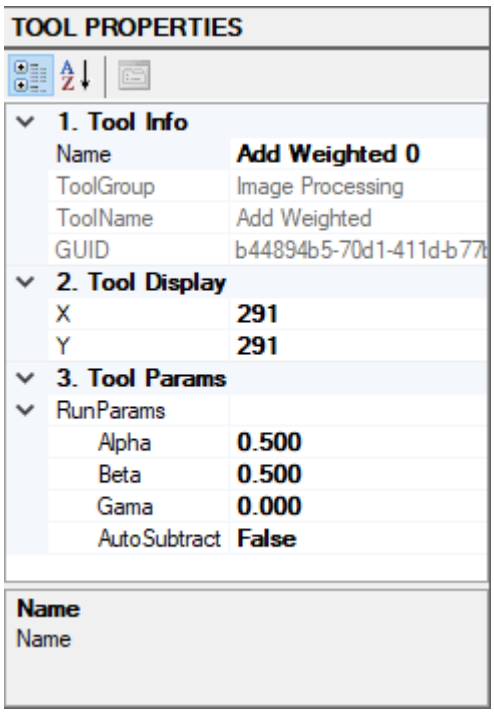
- **Input:**
  - (1) **InImage:** *Ảnh đầu vào*
  - (2) **LineA,B:** *Đường thẳng đầu vào*
- **Output:**
  - (1) **Point:** *Điểm cắt nhau của 2 đường nếu có*

3.9 Image Processing – Add Weighted, Binarization, Blob, Histogram, Morphology, etc.



► **AddWeighted:** *Pha trộn 2 hình theo công thức O = (α)Img$_1$+(β)Img$_2$+γ* ◄

- **Input:**
  - **(1) InImage1,2:** *Ảnh đầu vào*
- **Output:**
  - **(1) OutImage:** *Ảnh đầu ra sau khi pha trộn*

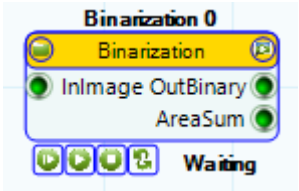


- **Properties:**
  - **(1) Alpha:** *Hệ số α*
  - **(2) Beta:** *Hệ số β*
  - **(3) Gama:** *Hệ số γ*
  - **(4) AutoSubtract:** *Tự động tính toán và thực hiện pha trộn*

➢ **Note:**

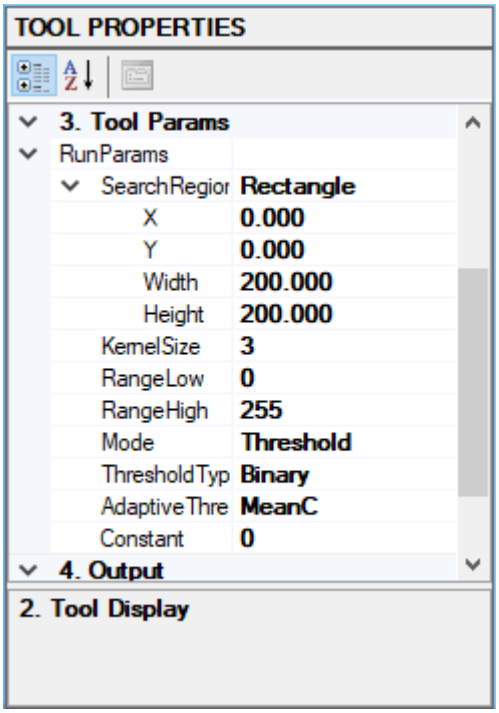
✓ *Lý thuyết về pha trộn 2 hình ảnh đầu vào trên OpenCV*

✓ *Có thể sử dụng công cụ này như 1 phương pháp để kiểm tra (Inspection) bằng cách lấy ảnh nhận được trừ đi ảnh mẫu*

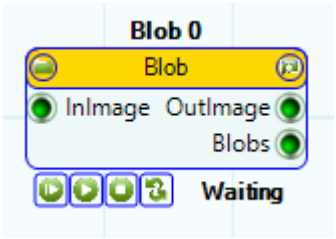


► **Binarization:** *Chuyển đổi ảnh từ Gray Scale thành ảnh đen trắng (Binary)* ◄

- **Input:**
  - **(1) InImage1:** *Ảnh đầu vào*
- **Output:**
  - **(1) OutBinary:** *Ảnh đầu ra đã nhị phân hóa*
  - **(2) AreaSum:** *Tổng điện tích có pixel khác 0*

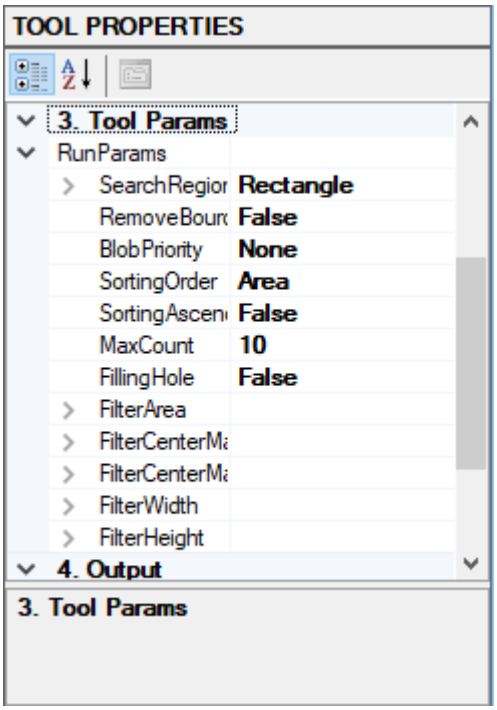


- **Properties:**
  - **(1) Search Region:** *Vùng sẽ thực hiện biến đổi nhị phân*
  - **(2) KernelSize:** *Kích thước Kernel dùng cho Adaptive threshold*
  - **(3) RangeLow:** *Giới hạn dưới*
  - **(4) RangeHigh:** *Giới hạn trên*
  - **(5) Mode:** *Kiểu chuyển đổi Binary*
  - **(6) ThresholdType:** *Loại chuyển đổi Binary*
  - **(7) AdaptiveThresholdType:** *Loại chuyển đổi Binay khi sử dụng AdaptiveMode*
  - **(8) Constant:** *Constant subtracted from the mean or weighted mean*



► **Blob:** *Tìm kiếm và lọc các blob theo các điều kiện cài đặt của các params* ◄

- **Input:**
  - **(1) InImage:** *Ảnh đầu vào*
- **Output:**
  - **(1) OutImage:** *Ảnh đầu ra*

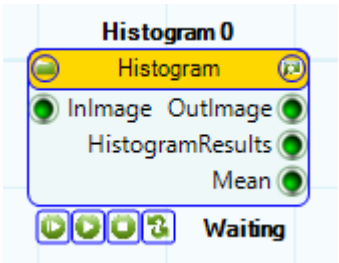


- **Properties:**
  - **(1) SearchRegion:** *Hệ số α*
  - **(2) RemoveBoundary:** *Hệ số β*
  - **(3) BlobPriority:** *Hệ số γ*
  - **(4) Sorting Order:** *Tự động tính toán và thực hiện pha trộn*
  - **(5) Sorting :** *Tự động tính toán và thực hiện pha trộn*
  - **(6)**

➢ **Note:**

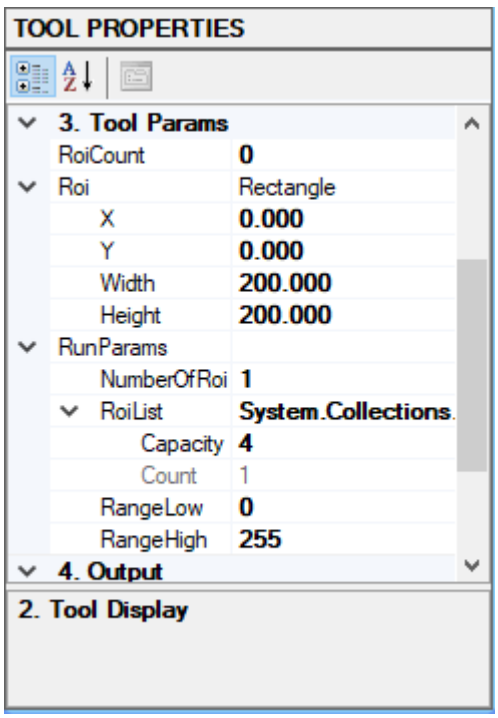
✓ *Lý thuyết về pha trộn 2 hình ảnh đầu vào trên OpenCV*

✓ *Có thể sử dụng công cụ này như 1 phương pháp để kiểm tra (Inspection) bằng cách lấy ảnh nhận được trừ đi ảnh mẫu*

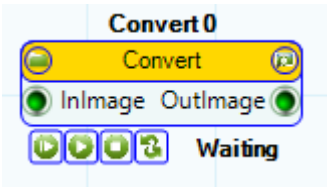


► **Histogram:** *Chuyển đổi ảnh từ Gray Scale thành ảnh đen trắng (Binary)* ◄

- **Input:**
  - **(1) InImage1:** *Ảnh đầu vào*
- **Output:**
  - **(1) OutBinary:** *Ảnh đầu ra đã nhị phân hóa*
  - **(2) AreaSum:** *Tổng điện tích có pixel khác 0*

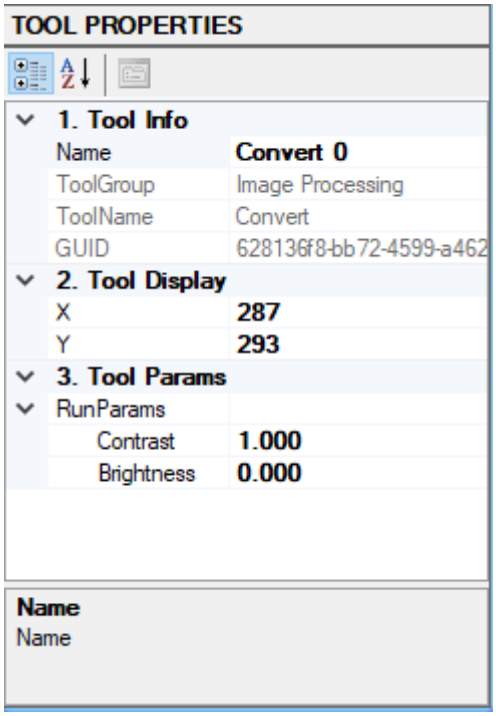


- **Properties:**
  - **(1) SearchRegion:** *Vùng sẽ thực hiện biến đổi nhị phân*
  - **(2) KernelSize:** *Kích thước Kernel dùng cho Adaptive threshold*
  - **(3) RangeLow:** *Giới hạn dưới*
  - **(4) RangeHigh:** *Giới hạn trên*
  - **(5) Mode:** *Kiểu chuyển đổi Binary*
  - **(6) ThresholdType:** *Loại chuyển đổi Binary*
  - **(7) AdaptiveThresholdType:** *Loại chuyển đổi Binay khi sử dụng AdaptiveMode*
  - **(8) Constant:** *Constant subtracted from the mean or weighted mean*

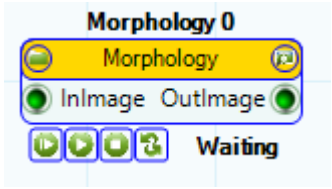


► **Convert:** *Điều chỉnh brightness và contrast* ◄

- **Input:**
  - **(1) InImage:** *Ảnh đầu vào*
- **Output:**
  - **(1) OutImage:** *Ảnh đầu ra sau khi điều chỉnh độ tương phản và độ sáng*

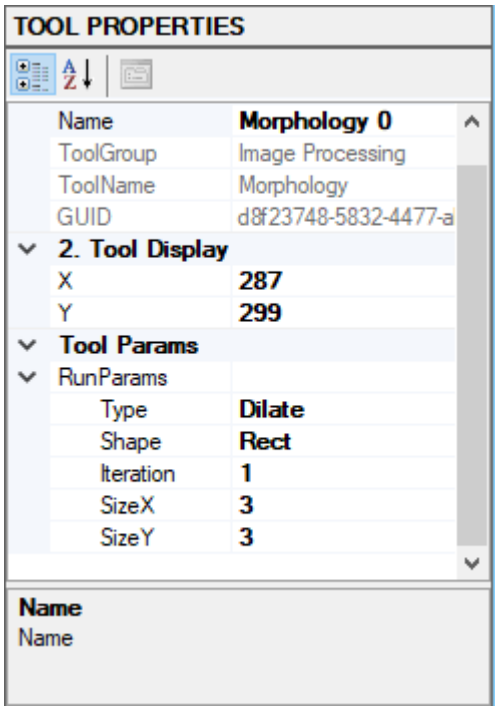


- **Properties:**
  - **(1) Constrast:** *Độ tương phản*
  - **(2) Brightness:** *Độ sáng*



► **Morphology:** *Biến đổi hình thái học, các phép toán liên quan tới cấu trúc hình học. Giãn nở (dialation) phép co (erosion), phép mở (opening), phép đóng (closing).*◄

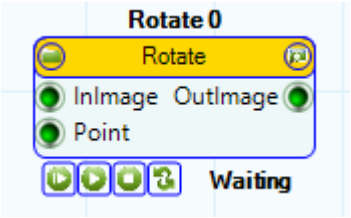
- **Input:**
  - **(1) InImage1:** *Ảnh đầu vào*



- **Output:**
  - **(1) OutBinary:** *Ảnh đầu ra đã nhị phân hóa*
  - **(2) AreaSum:** *Tổng điện tích có pixel khác 0*
- **Properties:**
  - **(1) Type:** *Loại biến đổi hình thái học*
  - **(2) Shape:** *Hình dạng của cửa sổ lọc*
  - **(3) Iteration:** *Số vòng lặp biến đổi*
  - **(4) SizeX:** *Kích thước cửa sổ lọc theo trục X*
  - **(5) SizeY:** *Kích thước cửa sổ lọc theo trục Y*

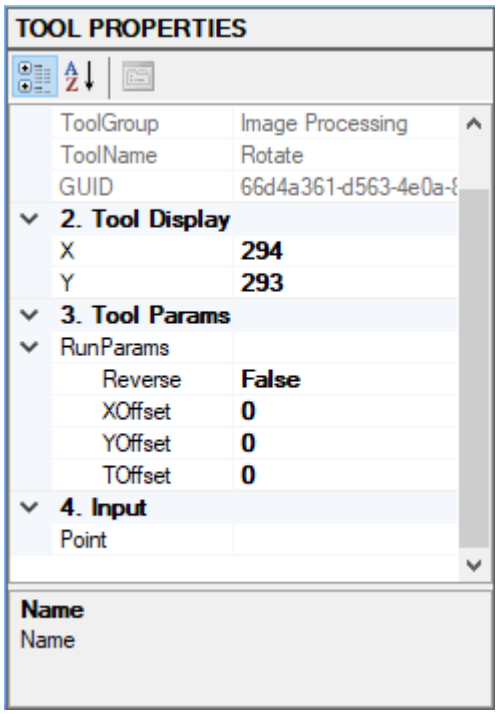
➢ **Note:**

✓ *Lý thuyết về biến đổi hình thái học trên OpenCV*

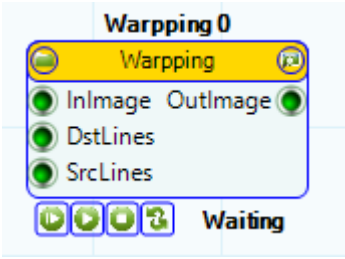


► **Rotation:** *Xoay hình ảnh quanh 1 điểm* ◄

- **Input:**
  - (1) **InImage:** *Ảnh đầu vào*
- **Output:**
  - (1) **OutImage:** *Ảnh đầu ra sau khi xoay*

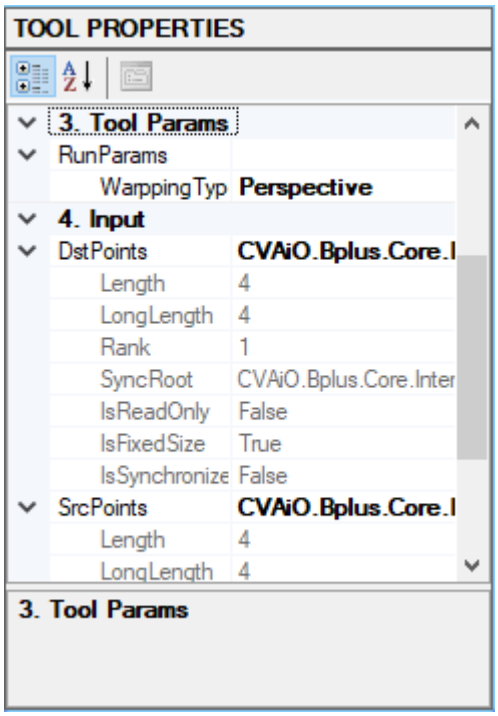


- **Properties:**
  - (1) **Reverse:** *Đảo chiều góc xoay*
  - (2) **XOffset:** *Offset theo chiều X*
  - (3) **YOffset:** *Offset theo chiều Y*
  - (4) **TOffset:** *Offset theo góc Theta*



► **Warpping:** *Biến đổi hình ảnh theo thuật toán warping perspective.*◄

- **Input:**
  - (1) **InImage:** *Ảnh đầu vào*
  - (2) **DstLines:** *Hình chữ nhật đích của quá trình warping*
  - (3) **SrcLines:** *Hình chữ nhật nguồn của quá trình warping*
- **Output:**
  - (1) **OutBinary:** *Ảnh đầu ra sau khi đã warping*

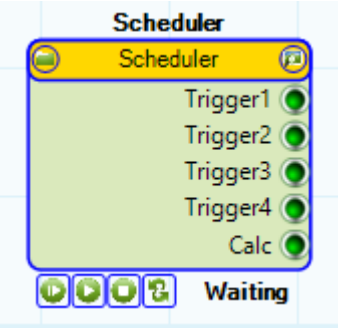


- **Properties:**
  - (1) **WarpingType:** *Loại thuật toán warping*

➢ **Note:**

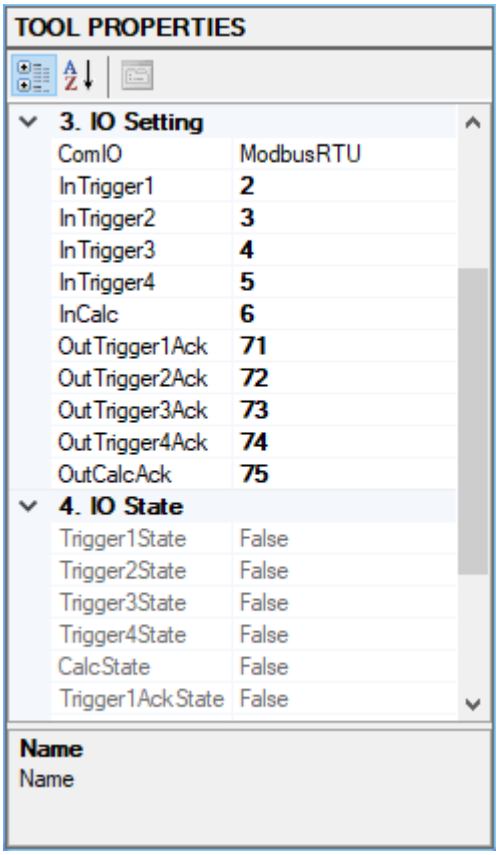
✓ *Lý thuyết về warping trên OpenCV*

3.10 Scheduler – Công cụ điều phối các nhánh xử lý



► **Scheduler:** *Điều khiển hoạt động của cả process xử lý ảnh* ◄

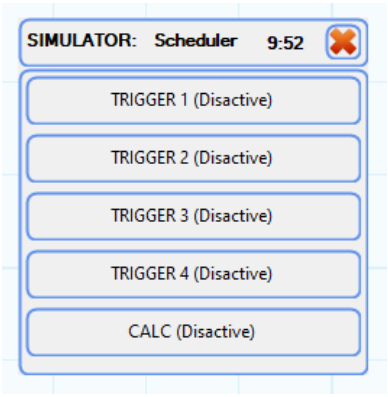
- **Input:**
- **Output:**
  - **(1) Trigger1,2,3,4:** *Tín hiệu điều khiển các process vision*
  - **(2) Calc:** *Tín hiệu điều khiển cho các công cụ Algorithm*



- **Properties:**
  - **(1) ComIO:** *Loại giao tiếp với PLC để truyền nhận IO*
  - **(2) InTrigger1,2,3,4:** *Vị trí tín hiệu trigger*
  - **(3) OutTriggerAck1,2,3,4:** *Vị trí tín hiệu trigger ack*
  - **(4) InCal:** *Vị trí tín hiệu Calculation*
  - **(5) InCalAck:** *Vị trí tín hiệu Calculation Ack*

➢ **Note:**

✓ *Loại ComIO được lấy từ cài đặt trong phần interface của UI Design*

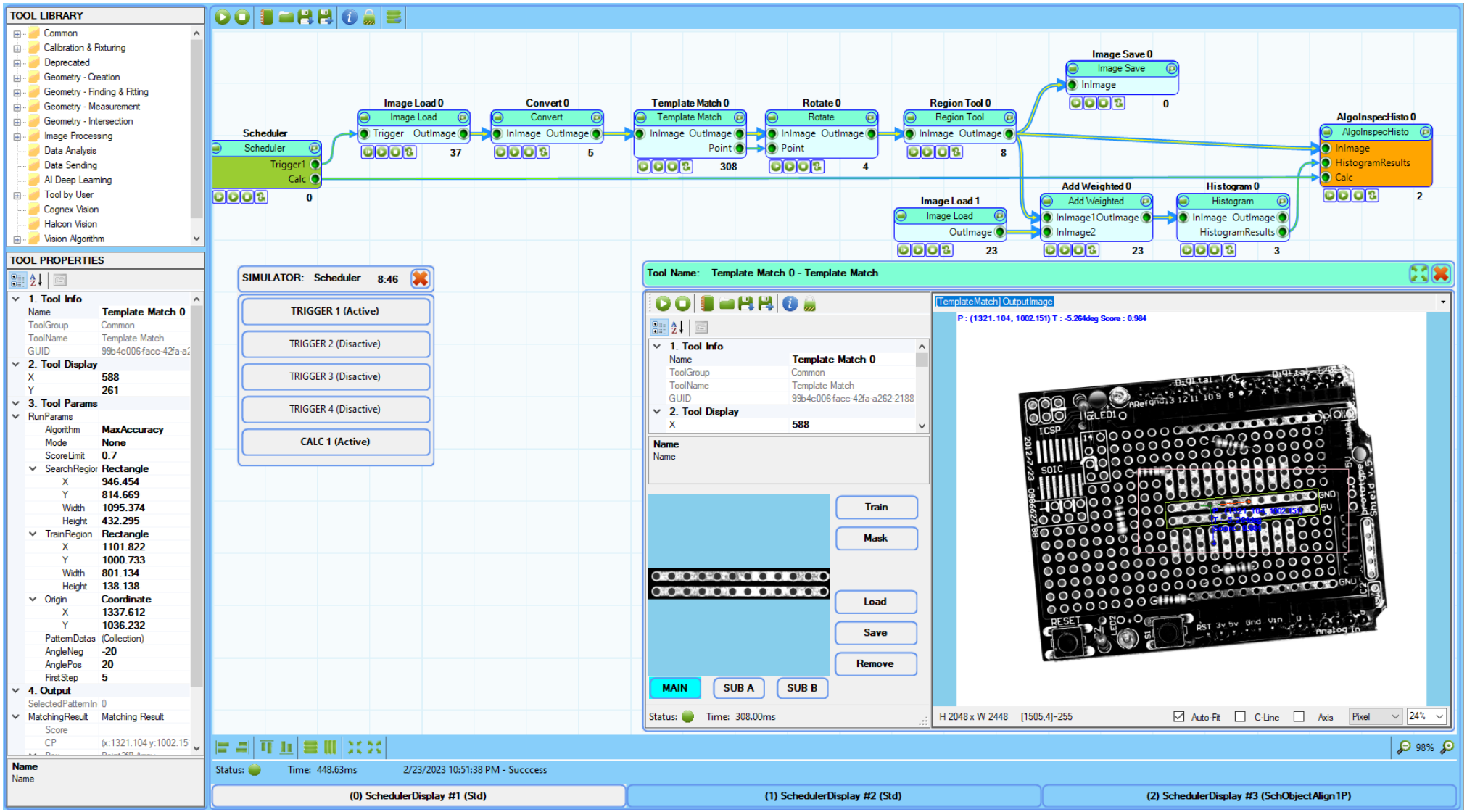


- **Scheduler Simulator:** Để thuật tiện trong việc thiết kế và điều chỉnh các thông số của process vision, simulator cho phép người dùng mô phỏng việc bật tắt các tín hiệu Trigger và Calc bằng cách lựa chọn Active/Disactive.

3.11 Vision Algorithm – Thuật toán xử lý ảnh

Case Study 3: Circuit Inspection

Kiểm tra thiếu linh kiện trên bản mạch sau khi lắp giáp bằng cách sử dụng công cụ Histogram



CHAPTER 4: CALIBRATION – CĂN CHỈNH

Các nội dung sẽ được đề cập trong chương này:

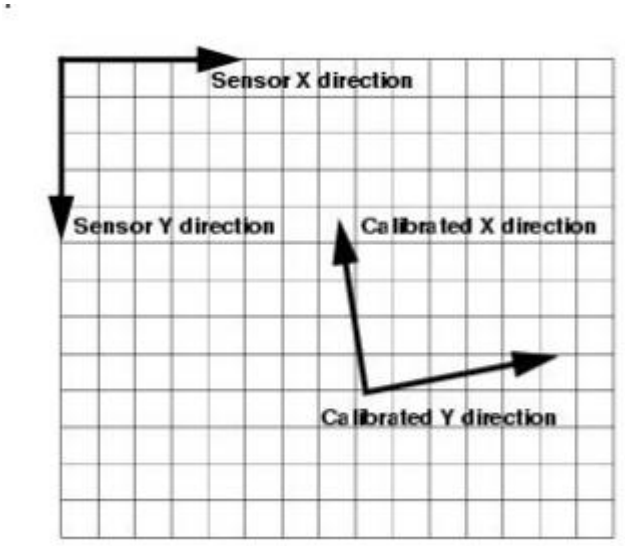
- Lý thuyết căn chỉnh
- Các bước để thực hiện căn chỉnh manual
- Các bước để thực hiện căn chỉnh tự động

4.1 Threory – Lý thuyết về căn chỉnh

Trong hệ Machine Vision, căn chỉnh là quá trình tính toán ma trận chuyển đổi từ hệ tọa độ pixel của camera sensor sang hệ tọa tộ thực tế - world coordinate system. Ma trận chuyển đổi định nghĩa mối quan hệ giữa khoảng cách đo được theo pixel trong hệ tọa độ camera và khoảng cách theo miliment trong hệ tọa độ của đối tượng được chụp. Cảm biến của camera được chia thành các pixel riêng biệt. Hình ảnh được lưu lại trọng bộ nhớ của hệ thống vision là 1 mảng ghi nhận sự thay đổi được tạo ra tại các vị trí pixel sensor. Thông thường, hệ thống machine vision và xử lý ảnh sẽ sử dụng vị trí trên cùng bên phải như là điểm gốc của ảnh và từ đó tạo ra hệ trục tọa độ ảnh với trục X chạy theo hàng, trục Y chạy theo cột của các sensor.

Tất cả các đặc trưng trong ảnh như là điểm, đường, góc, cạnh sẽ được biểu diễn và tính tỉ lệ theo hệ tọa độ ảnh. Điều này có nghĩa là, nếu thiếu đi quá trình căn chỉnh, tất cả các vị trí sẽ là trong hệ tọa độ pixel tính tương đối với góc ngoài cùng bên phải và khoảng cách sẽ tính bằng pixels. Quá trình căn chỉnh sẽ cho phép chúng ta xác định 1 hệ tọa độ khác có ý nghĩa thực tế hơn. Điều đó dẫn tới kết quả của quá trình tính toán vision sẽ được biểu diễn trong hệ tọa độ mới này.

Quá trình căn chỉnh được hỗ trợ bởi CVAIO B+ sẽ cho phép chúng ta xác dịnh các đặc trưng quan trọng của hệ tọa độ căn chỉnh bao gồm các thành phần sau:

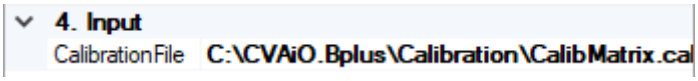


- **(1)** Vi trí của điểm gốc – Origin của hệ tọa độ căn chỉnh: Đây là vị trí (0,0) của hệ tọa độ căn chỉnh trong mối liên hệ với camera sensor (0,0)
- **(2)** Hướng các trục chỉnh của hệ tọa độ căn chỉnh. Đây là chiều dương của trục X, Y trong hệ tọa độ căn chỉnh
- **(3)** Góc giữa hệ tọa độ căn chỉnh và hệ tọa độ camera sensor. Đây là tổng lượng góc xoay giữa hệ tọa độ căn chỉnh và hệ tọa độ sensor
- **(4)** Tỉ lệ thu phóng của hệ tọa độ căn chỉnh. Đây là mỗi quan hệ giữa 1 đơn vị độ dài trong hệ tọa độ căn chỉnh và độ lớn 1 pixel trong hệ tọa độ camera sensor

Việc tính toán ma trận căn chỉnh được CVAIO B+ hỗ trợ thông qua việc sử dụng công cụ Find Calib Matrix với giá trị đầu vào là các giá trị tọa độ trong hệ tọa độ vision và hệ tọa độ thực tế như trình bày ở Section 3.3

Các cách sử dụng ma trận căn chỉnh trong CVAIO B+:

Cách 1: Gắn ma trận căn chỉnh vào hình ảnh ở công cụ Acquisition. Ma trận căn chỉnh sau khi được load lên sẽ đi cùng với hình ảnh trong suốt quá trình xử lý



Cách 2: Load vào trong công cụ Algorithm hoặc các công cụ ByUser

```
[Description("Calibration File"), Category("4. Input"), PropertyOrder(11)]
[Editor(typeof(CalibrationFileSelectionEditor), typeof(System.Drawing.Design.UITypeEditor))]
0 references
public string CalibrationFile
{
    get => calibrationFile;
    set
    {
        CalibMatrix calibMatrix = new Serializer().Deserializing(value) as CalibMatrix;
        if (calibMatrix == null) return;
        this.calibMatrix = calibMatrix;
        calibrationFile = value;
    }
}
[Browsable(false)]
0 references
public CalibMatrix CalibMatrix { get { if (calibMatrix == null) calibMatrix = new CalibMatrix(); return calibMatrix; } set => calibMatrix = value; }
```

Chuyển đổi giữa các hệ trục tọa độ:

- Vision Coordinate → World Coordinate:

  Step 1: Thêm thư viện: Project → References →Add Reference → Browse → CVAiO.Bplus.Core

  *Step2: Gọi function: World Coordinate = **AIO.VtoR**(Ma trận căn chỉnh, Vision Coordinate)*

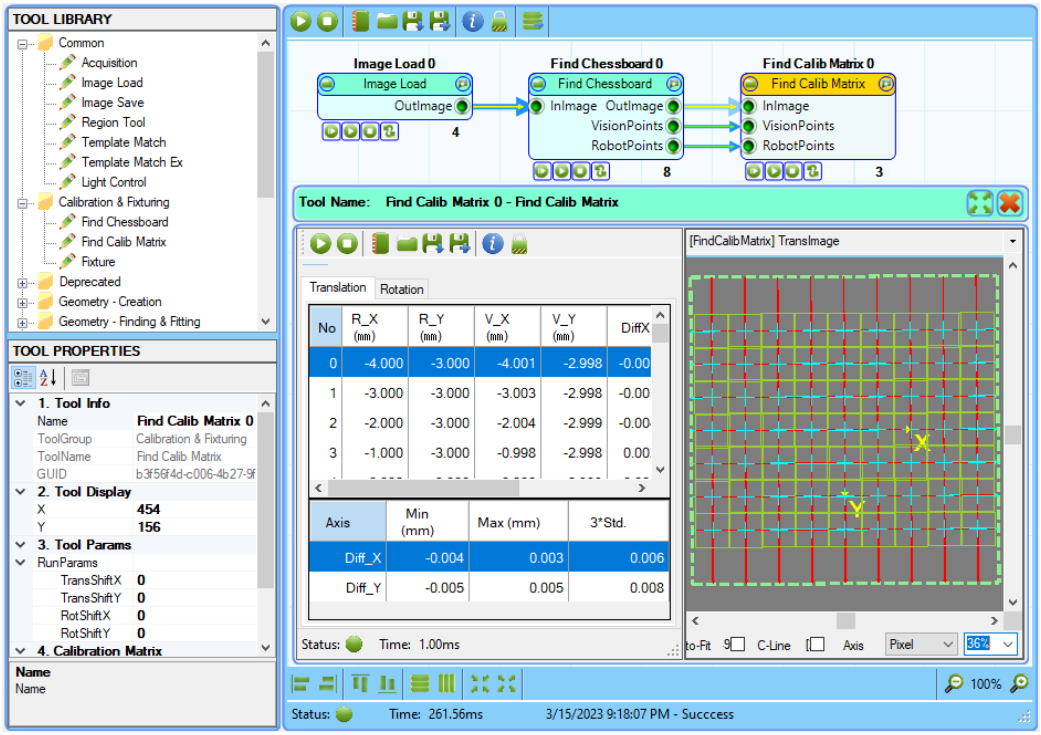
- World Coordinate → Vision Coordinate:

  Step 1: *Thêm thư viện: Project → References →Add Reference → Browse → CVAiO.Bplus.Core*

  *Step 2: Gọi function: Vision Coordinate = **AIO.RtoV**(Ma trận căn chỉnh, World Coordinate)*

4.2 Manual Calibration – Căn chỉnh bằng Chessboard

Căn chỉnh manual được sử dụng chủ yếu để tìm hệ số chuyển đổi từ pixel sang mm. Camera sẽ chụp hình 1 tấm hình ô bàn cờ được in ra với một kích thước nhất định tính theo mm. Công cụ Find Chessboard sẽ giúp tìm ra vị trí các góc của ô bàn cờ, sau đó tính toán vị trí góc theo mm dựa theo kích thước thực tế nhập vào. Tổng hợp kết quả sẽ được đưa vào công cụ Find Calib Matrix để tính toán ma trận căn chỉnh.



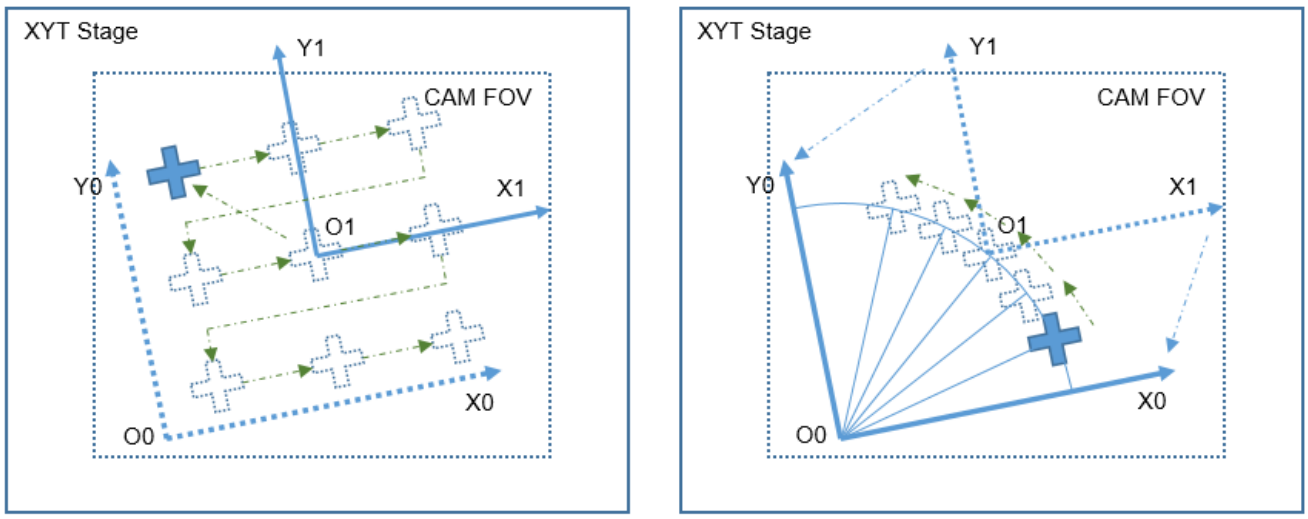
Các bước để thực hiện căn chỉnh manual.

- **Step 1:** Chuẩn bị tấm ô bàn: Số lượng các ô bàn cờ theo cả 2 chiều phải là số lẻ. Có thể download hoặc tự tạo các ô bàn có theo hướng dẫn ở OpenCV hoặc CVAiO
- **Step 2:** Thiết lập process căn chỉnh: Các công cụ được chọn và ghép lại giống như hình bên trên. Điều chỉnh lại thông số của các công cụ Find Chessboard Conner và Find Calib Matrix

- **Step 3:** Bấm nút run để chạy process căn chỉnh và lưu lại file căn chỉnh.
- **Step 4:** Kiểm tra kết quả sau căn chỉnh: Mở công cụ Find Calib Matrix và bấm Run để hiển thị kết quả căn chỉnh. Các thông tin cần kiểm tra bao gồm giá trị sái số nhỏ nhất ***min(mm)***, giá trí sai số lớn nhất ***max(mm)***, độ rộng vùng 3 sigma ***m*std*** của từng trục X,Y. Nếu các giá trị trên nhỏ tức là kết quả căn chỉnh có độ chính xác cao.

4.3 Auto Calibration – Căn chỉnh tự động

Căn chỉnh tự động được sử dụng để đồng bộ hóa hệ tọa độ vision và hệ tọa độ của robot hoặc hệ servo XYT. Ma trận căn chỉnh sẽ giúp tìm được vị trí của vật thể trong hệ tọa độ robot sau khi xác định được trong hệ tọa độ vision. Quá trình căn chỉnh được chia làm 2 phần riêng biệt: Xác định chiều dương của hệ trục OX, OY và xác định vị trí tâm xoay O. Để xác định chiều của hệ trục tọa độ OXY, chương trình vision sẽ gửi lần lượt thông tin 9 vị trí ***(RobotPoints)*** tính theo mm trong hệ tọa tộ oxy cho robot sau đó chụp hình để xác định lại 9 vị trí trong hệ tọa độ vision ***(VisionPoints)*** tính theo pixel. Để xác định vị trí của điểm gốc không O, chương trình vision sẽ gửi lần lượt thông tin 5 góc xoay tính theo độ trong hệ tọa độ oxy cho robot rồi sau đó chụp hình đề xác định 5 vị trí góc xoay trong hệ tọa độ vision tính theo pixel. Hình dưới đây mô tả lại toàn bộ quá trình căn chỉnh auto với robot 3 trục XYT:



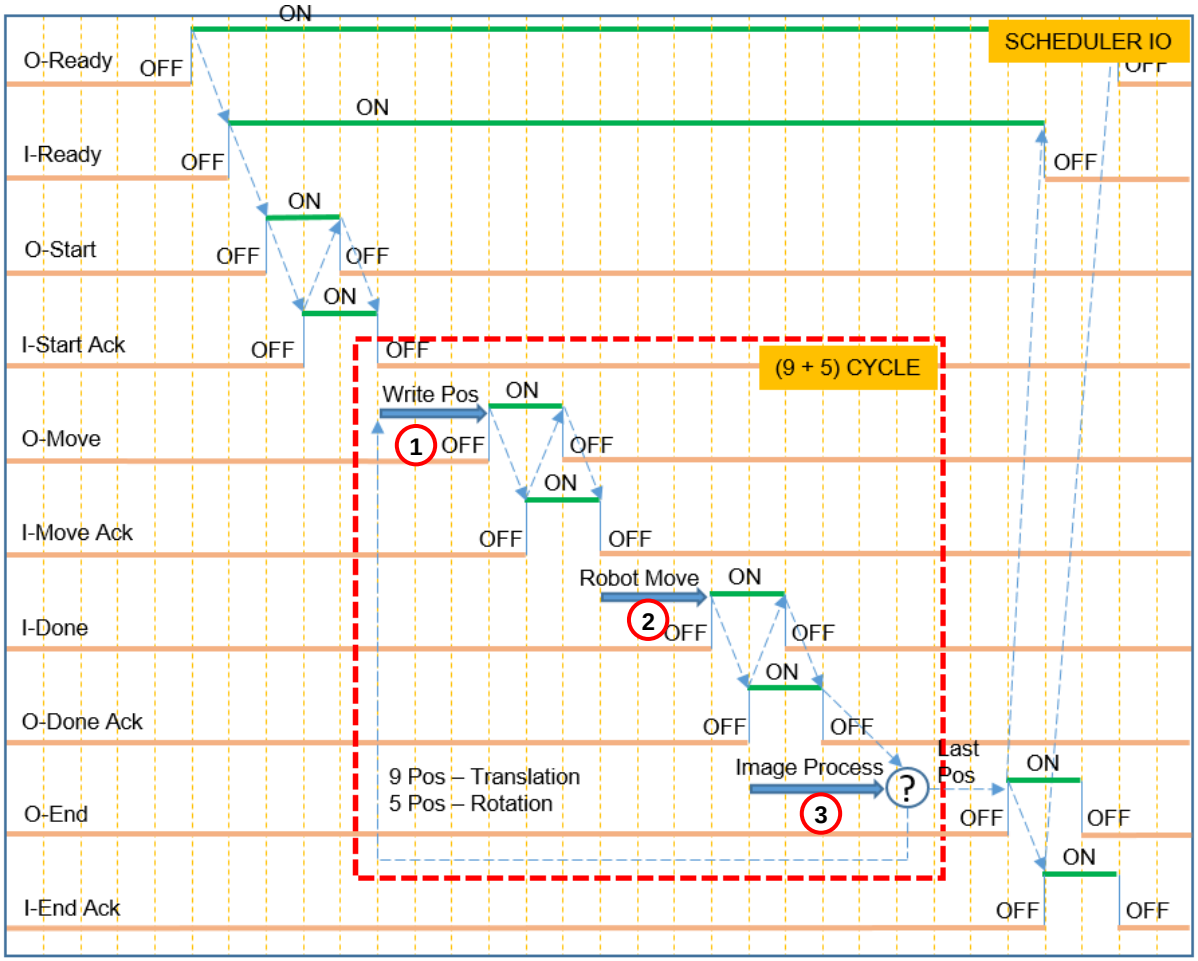
Trong toàn bộ quá trình căn chỉnh, chương trình vision sẽ giữ vai trò chủ động và robot (PLC) sẽ giữ vai trò bị động. Chương trình vision sẽ tính toán vị trí mà robot cần phải dịch chuyển sau đó gửi dữ liệu vị trí trước khi gửi tín hiệu IO yêu cầu robot dịch chuyển. Phía dưới mô tả chi tiết process căn chỉnh giữa chương trình vision và robot. Trong đó quá trình 3 steps

(1) – Gửi tọa độ vị trí robot cần phải dịch chuyển

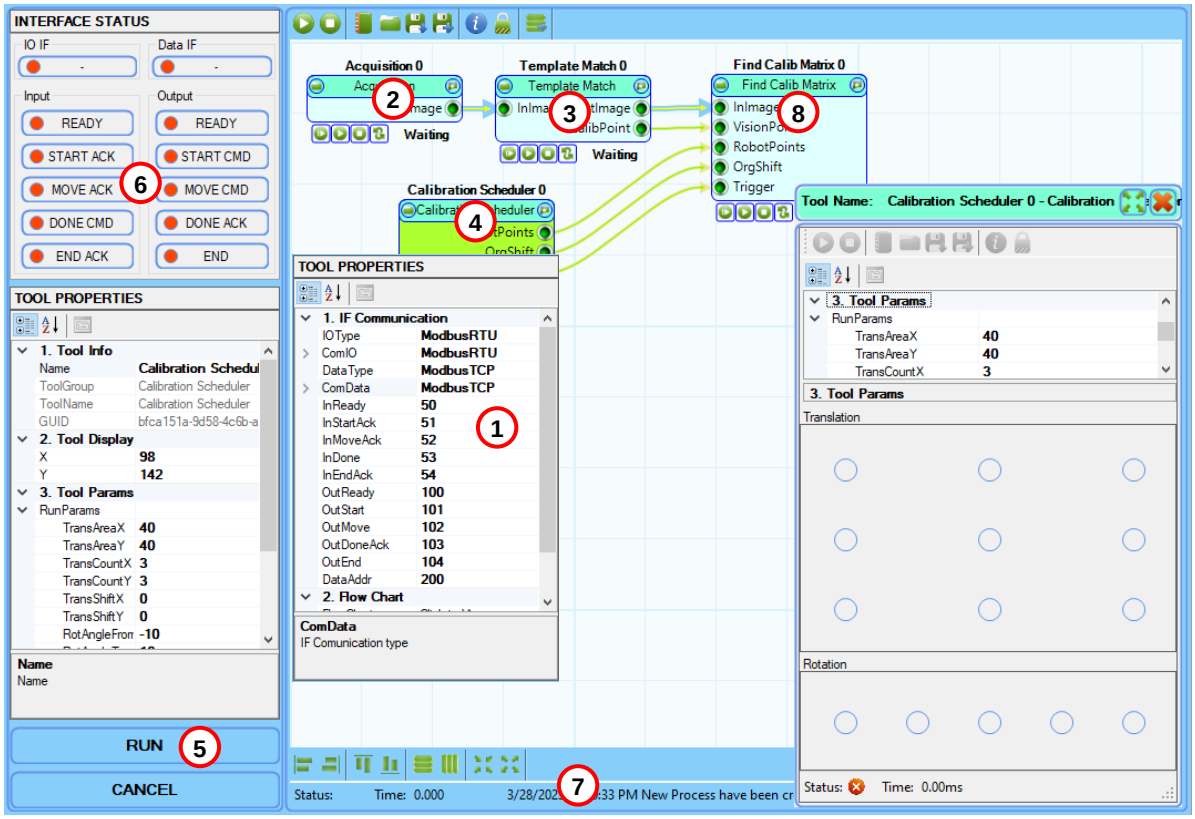
(2) – Robot dịch chuyển tới vị trí yêu cầu;

(3) – Vision tiến hành xử lý tìm vị trí điểm mark;

sẽ được lặp lại nhiều lần cho đến khi tổng số vị trì cần cho quá trình căn chỉnh hoàn tất. Trong lúc quá trình căn chỉnh tự động được thực hiện, người dùng có thể huy bỏ bằng cách bấm nút Cancel góc dưới bên trái. Việc giao tiếp giữa vision và robot phải tuân thủ chặt chẽ theo đúng process đưa ra ở hình bên dưới để đảm bảo quá trình căn chỉnh được chính xác nhất.



Các bước để thực hiện căn chỉnh tự động:



- **Step 1:** Cài đặt giao tiếp với PLC: Chuột phải vào màn hình chính để hiện thị các thuộc tính giao tiếp với PLC ở bên trái màn hình. Lựa chọn IOType, DataType sẽ được sử dụng để truyền nhận IO và dữ liệu vị trí robot sẽ dịch chuyển.
- **Step 2:** Lựa chọn camera sử dụng trong quá trình căn chỉnh
- **Step 3:** Setup công cụ template matching để tìm bắt điểm mark căn chỉnh, chỉ sử dụng main mark và sử dụng Max Accuracy algorithm để đạt được độ chính xác cao nhất trong quá trình tìm kiếm.
- **Step 4:** Lựa chọn tổng số điểm dịch chuyển trong quá trình căn chỉnh. Thông thường sẽ để giá trị mặc định là 9 điểm cho căn chỉnh hướng X,Y và 5 điểm cho căn chỉnh góc T.
- **Step 5:** Bấm Start để thiết lập kết nối với PLC và bắt đầu quá trình căn chỉnh.
- **Step 6:** Kiểm tra trạng thái giao tiếp của Vision và PLC
- **Step 7:** Kiểm tra các bước hiện tại của quá trình căn chỉnh
- **Step 8:** Kiểm tra kết quả sau căn chỉnh: Mở công cụ Find Calib Matrix và bấm Run để hiển thị kết quả căn chỉnh. Các thông tin cần kiểm tra bao gồm giá trị sái số nhỏ nhất ***min(mm)***, giá trí sai số lớn nhất ***max(mm)***, độ rộng vùng 3 sigma ***m*std*** của từng trục X,Y. Nếu các giá trị trên nhỏ tức là kết quả căn chỉnh có độ chính xác cao.

CHAPTER 5: HISTORY & USER LOGIN – LỊCH SỬ & QUẢN LÝ TÀI KHOẢN ĐĂNG NHẬP

Các nội dung sẽ được đề cập trong chương này:

- Cấu trúc của log file và cách phân tích
- Quản lý tài khoản đăng nhập của người dùng

5.1 History – Phân tích lịch sử

5.2 User Login – Quản lý tài khoản đăng nhập

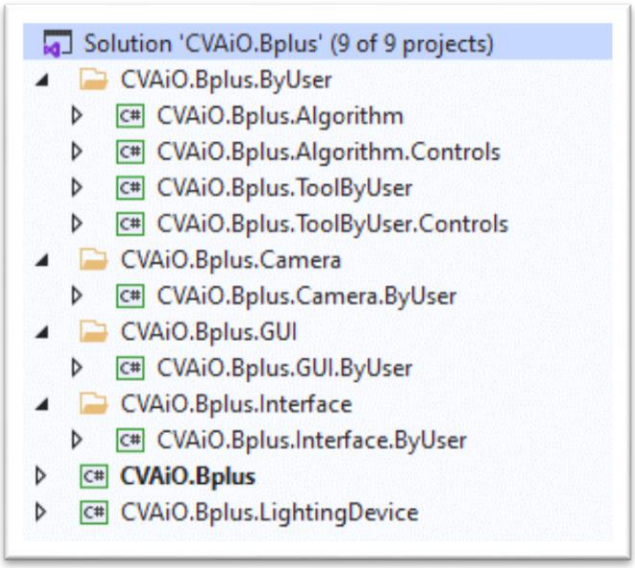
CHAPTER 6: TOOL BY USER – PHÁT TRIỂN CÔNG CỤ VÀ LUỒNG XỬ LÝ MỚI

Các nội dung sẽ được đề cập trong chương này:

- Giới thiệu về ý tưởng thiết kế mở cho người dùng
- Các bước để phát triển 1 công cụ xử lý ảnh mới
- Các bước để phát triển 1 luồng xử lý mới
- Các bước để tích hợp thêm camera mới
- Các bước để tích hợp thêm module interface mới

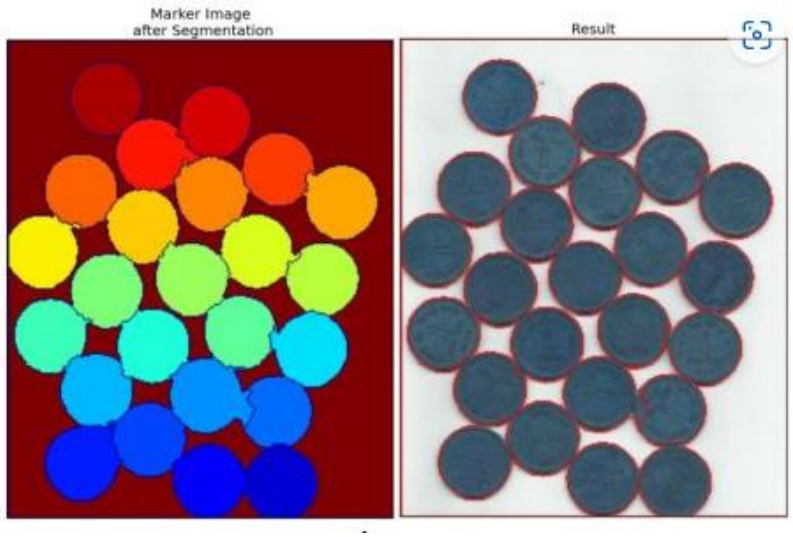
6.1 Overview - Giới thiệu chung

Các bài toán trong hệ machine vision rất da dạng nên không thể thiết kế một platform như một lời giải chung được. Do đo CVAIO B+ được đưa đến cho người dùng dưới dạng môt solution của visual studio c# để người dùng có thể dễ dàng cấu trúc lại cho phù hợp với yêu cầu của từng bài toán. Trong đó có 2 projects sử dụng để phát triển các công cụ xử lý ảnh, 2 projects sử dụng để phát triển công cụ algorithms, 1 project để phát triển module tích hợp camera, 1 project để phát triển module giao tiếp PLC, 1 project để phát triển module điều khiển ảnh sáng thông qua RS232, 1 project để phát triển luồng xử lý ảnh. Chi tiết về các projects này sẽ được trình bày chi tiết từ section 6.2 đến section 6.5. Ngoài ra còn có 2 projects liên quan đến việc tích hợp Cognex Vision Pro và 2 projects liên quan đến tích hợp halcon sẽ được trình bày trong chapter 7.



6.2 Tool By User – Phát triển công cụ xử lý ảnh

Yêu cầu: Phát triển 1 công cụ xử lý ảnh để thực hiện việc phân đoạn hình ảnh sử dụng thuật toán Watershed như trình bày trong OpenCV ở đường link này.

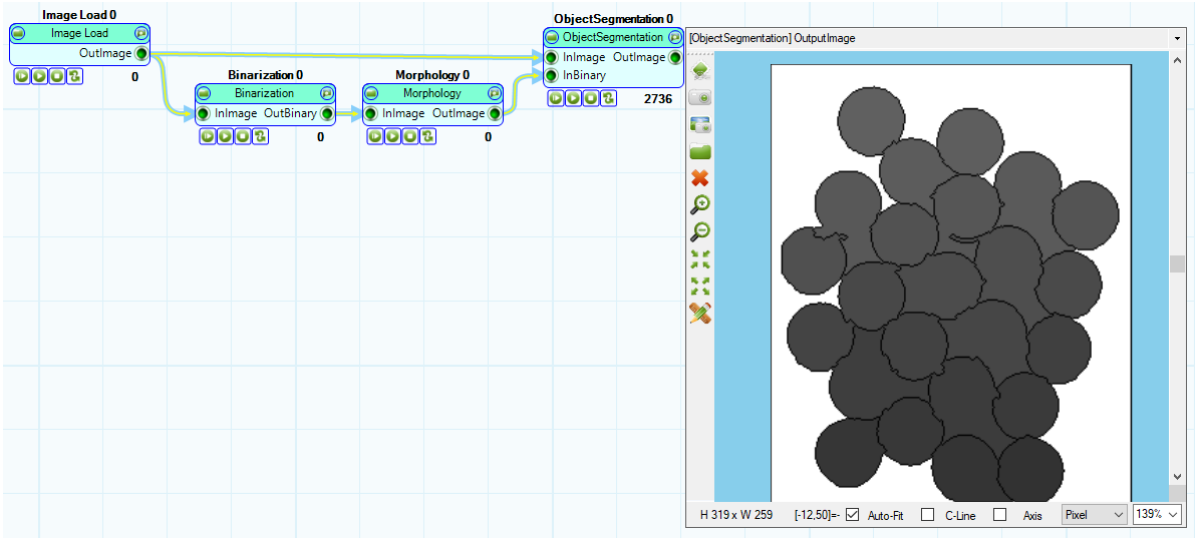


Giải pháp: Phát triển thêm công cụ xử lý ảnh sử dụng project ToolByUser

- **Step 1:** Xác định đầu vào: Công cụ sẽ sử dụng 2 hình ảnh đầu vào 1 – Hình ảnh Gray Scale lấy trực tiếp từ camera, 2 – Hình ảnh Binary sau khi đã xử lý thông qua các công cụ Binarization và Morphology.
- **Step 2:** Code các giá trị đầu vào

  Step 3: Xác định đầu ra
- **Step 3:** Code các giá trị đầu ra .
- **Step 4:** Xác định các giá trị Parameter setting
- **Step 5:** Code các giá trị đầu ra

Đánh giá: Đánh giá kết quả thực hiện được



6.3 Scheduler By User – Phát triển luồng xử lý ảnh

Yêu cầu: Đưa ra yêu cầu của bài toán

Giải pháp: Giải pháp là gì và các bước thực hiện

- **Step 1:** Chuẩn bị tấm căn chỉnh
- **Step 2:** Thiết lập process căn chỉnh.
- **Step 3:**
- **Step 3:** Chạy căn chỉnh .
- **Step 4:** Sau khi quá trình thu.

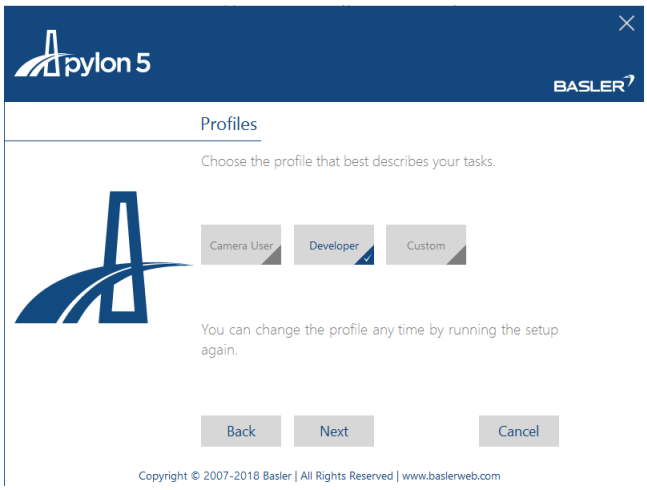
Đánh giá: Đánh giá kết quả thực hiện được

6.4 Camera By User – Phát triển tích hợp camera

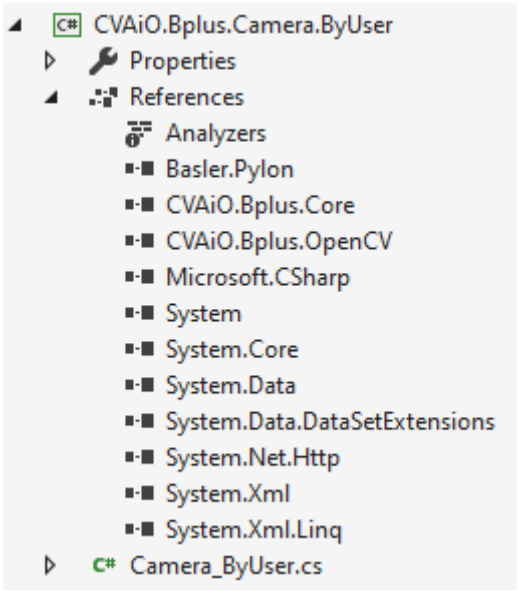
Yêu cầu: Phát triển module camera để có thể tích hợp camera basler Gige vào công cụ Aquisition

Giải pháp: Sử dụng khả năng tích hợp loại camera mới ở project ***CVAiO.Bplus.Camera.ByUser***

- **Step 1:** Cài đặt phần mềm Pylon 5 của nhà cung cấp camera basler tại Basler Web hoặc CVAIO B+. Lưu ý cài đặt sử dụng đường dẫn mặc định của nhà cung cấp và lựa chon **Developer Mode** để cài đặt toàn bộ các file thư viện cần thiết cho người phát triển.



- **Step 2:** Thêm thư viện được cung cấp bởi nhà cung cấp vào trong project bằng cách chọn: project → Reference →Add Reference→Browse→Basler.Pylon.dll



- **Step 3:** Thêm class và lựa chọn kế thừa từ interface Camera: Camera_ByUser : ICamDevice, Idisposable. Các lớp cần phải implement gồm có:

```
public interface ICamDevice : IDisposable
{
        int GetConnectedCamCount(); // Thu thập tổng số cam đang được kết nối đến
        string GetCameraSeiralNo(); // Lấy serial number của camera
        string GetCameraSeiralNo(int camNumber); // Lấy serial number của camera
        void CameraOpen(int camNumber); // Mở kế nối với camNumber
        Mat GrabImage(); // lấy hình và trả ra dưới định dạng Mat của OpenCV
        void CameraClose(); // Đóng kết nối với camera hiện tại
}
```

- **Step 4:** Các hàm sẽ được gọi trong công cụ Acquision gồm có CameraOpen(int CamNumber): Thiết lập kết nối với camera, GrapImage(): Thu thập hình ảnh từ camera và trả ra dưới định dạng Mat của OpenCV, CameraClose(): Đóng kết nối với camera và giải phóng tài nguyên đã sử dụng,

- Lưu ý: Người dùng nên tham khảo các code mẫu của nhà cung cấp để có thể hoàn thiện chương trình 1 cách chính xác nhất hoặc liên hệ với AIO MATRIX để được hỗ trợ.

Đánh giá: Kiểm tra kết quả bằng cách chuyển đổi camera trong công cụ Acqusition thành ByUser, thực hiện kết nối và thu thập hình ảnh từ camera

6.5 Interface By User – Phát triển module giao tiếp

Yêu cầu: Phát triển module giao tiếp với PLC sử dụng chuẩn SLMP

Giải pháp:

- **Step 1:** Tìm hiểu về SLMP protocal
- **Step 2:**

Đánh giá: Đánh giá kết quả thực hiện được

CHAPTER 7: COGNEX VISION PRO & HALCON MVTEC

Các nội dung sẽ được đề cập trong chương này:

- Tích hợp thêm công cụ xử lý ảnh của Cognex Vision Pro
- Tích hợp thêm công cụ xử lý ảnh của Halcon MVTec

7.1 Cognex Vision Pro

7.2 Halcon MVTech

APPENDIX

Appendix 1

Appendix 2