EUCALL

The European Cluster of Advanced Laser Light Sources

Grant Agreement number: 654220

Work Package 6 – HIREP

Deliverable D6.3

Beta version of sample identification software

Lead Beneficiary: ELI

Authors: David Watts, Carsten Deiter, Joachim Schulz, Dragos Popescu, Mihail Cernaianu, Boris Odložilik, Ondřej Janda, Pavel Bastl, Antonín Fajstavr, Tuomas Wiste, Daniele Margarone

Due date: 31.03.2017
Date of delivery: 31.03.2017

Project webpage: www.eucall.eu

Deliverable Type	
R = Report DEM = Demonstrator, pilot, prototype, plan designs DEC = Websites, patents filing, press & media actions, videos, etc. OTHER = Software, technical diagram, etc.	DEM
Dissemination Level	
PU = Public, fully open, e.g. web CO = Confidential, restricted under conditions set out in Model Grant Agreement CI = Classified, information as referred to in Commission Decision 2001/844/EC	PU

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

Contents

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

1. Software for target identification and localization *(European XFEL)*

1.1 Introduction

Due to the high repetition rate of the XFELs and high power laser facilities, which are in the order of 0.1 Hz to 120 Hertz, image analysis algorithms will not be able to reliably identify suitable targets between two shots and hence target locations need to be determined beforehand in an automated sample screening workflow and given to the beamline's ultra-fast sample handling system as a list of coordinate positions. To accurately position targets into the focus of the instrument the coordinates must be given with the precision of micrometres relative to a reference coordinate system as defined by the fiducial marks of the sample frame.
The described software "Targeter" for computer based target identification and localisation is part of this automated sample screening workflow (Task 6.1). The workflow consists of two separate parts:

- One system (Pre-Imager) will be concerned with obtaining images of the sample frame at micrometre resolution.
- The other (Targeter) will be concerned with analysing the images and finding the suitable targets.

This separation allows the user to choose any commercial solution as imaging system and to analyse the obtained images regardless which microscope is used. The other advantage with separating the software is that target detection can be performed elsewhere and at a later time without blocking usage of the computer connected to the microscope. In order to demonstrate a working system a prototype of a pre-investigation imaging device will be developed at the European XFEL. The system will provide an experimental set of real world example images for Targeter and be a cost effective design that users can easily build themselves using off the shelf parts. The design considerations of the imaging system should meet the requirements of the automated sample screening workflow and be compatible with the standard sample holder system (Deliverable 6.1). The imaging system will automatically identify the bar code identification of the sample; find the fiducial marks and the region(s) of interest by an overview camera. With this information the system will precisely localise the fiducial marks with micrometre resolution and raster scan the entire region of interest with the high magnification objective.
The software Targeter will then process each image and locate targets within the image, the target positions will be specified in micrometres relative to the fiducial marks. The target location information is contained within a structured XML file, as well as links to images of the targets. The file will be read by the beamline control system and enable the beamline stages to move to the target positions with micrometre accuracy at the high repetition rates required. The target detection software requires the input of the user to define regions of the image in which the targets are to be found. Information about the sample should also be entered by the scientist since this constitutes the beginning stage of the analysis experiment. Example good targets are required to train the software and therefore users need to locate

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

good targets within the image. Based on these good targets the software will search the sample area for other similarly good targets that meet the user's requirements. The scientist should always have the final say on the selection of targets and this may require changing of parameters or types of analysis algorithms to fine tune the target selection results.

1.2 Requirements for the target identification and localisation software Targeter

Based on the considerations given above, the software should be capable of

- Reading an XML sample description file containing all relevant information about:
  - The identification of the sample.
  - The type and the hierarchy of inner target frame.
  - The coordinate system of the sample.
  - The region(s) of interest.
  - The locations of the overview image files.
  - The locations of the high magnification image files.
  - The coordinates of the high magnification images.
- Handling user input of sample information if not available from imaging device
  - The identification of the sample.
  - An additional Sample description.
  - The size of one image pixel in micrometres.
  - The type and the hierarchy of the inner target frame.
- Defining regions of interest - including or excluding regions of the sample for the analysis
  - The area in which targets are detected by the program is restricted to the region of interest.
  - By reading the type of the inner sample frame or the identification of the sample regions of interest can be pre-masked by the software.
  - Given an overview image of the sample frame the user should be able to define regions of interest by drawing rectangles, ellipses or polygons manually.
- Defining good exemplar targets within the image:
  - The software needs examples of 'good targets'.
  - The user must identify 'good targets' in the image by drawing rectangles, ellipses or polygons around 'good target' regions in the image.
  - This bounding box of the drawing object defines the target sub-image, a binary mask image defines which areas in the target sub-image are not the target and therefore can be excluded from the training of the analysis algorithms.
- Target Detection - Finding all 'good targets' in the image
  - Several image analysis methods should be tested for their ability to accurately find targets within the image.
  - The algorithms should be tuneable to find either more or less of less good targets by adjusting parameters.

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

    - Once the targets have been detected they should be highlighted in the image and their locations should be saved to the XML sample description file.
    - The code should be structured in such a way that additional target detection algorithms can be easily incorporated into the software at a later date.
- Length calibration in case of a missing sample reference coordinate system
    - The image position of the fiducial marks and the size of one pixel should be used to determine the position of a given pixel in micrometres in respect to the fiducial marks.
- Software Development
    - The software should be open source and available to other developers through an open access software repository such as GitHub.
- Development of an imaging system
    - Development of a microscope imaging system with motorised focusing and motorised XY stage movement
    - Sample frame overview image by long working distance camera
    - Sample plate holder for microscope
    - Illumination by ring light and coaxial kohler illumination
    - Stage control such that the sample are can be scanned at micrometre resolution
    - Focusing control to bring out of focus regions of the sample into focus including fiducial marks on outer frame.
    - Focus image stack processing.
    - Acquisition of images from digital camera and further processing by pre-investigation software.

1.3 Requirements for the beta version of the software Targeter

The beta version of the software should perform target detection on a sample image. This requires a GUI in which the origin image, intermediate images and the result image can be displayed. The software needs the ability to draw on images in order to define target and mask regions, and an image processing pathway to detect targets within the test images. Several image analysis algorithms should be included since specific detection methods work better with certain types of targets. The obtained target positions should be written into the XML target description file.

1.4 Description of the beta version of the software Targeter

The user interface of Targeter is shown in Figure 1. An example image has been loaded into the program and is shown along with its thumbnail image. Previously loaded images are shown within the file menu and can be easily reloaded. Due to a Window bug when opening files over a network drive the loading of files is performed using the QT file upload dialog, multiple files can be selected and loaded into the program. The plus and minus buttons (along with the mouse wheel) allow the user to zoom in and out of the image. The zoom

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

point is the mouse position in the window and therefore target regions of the image can be zoomed into by ensuring that the mouse pointer is over the target (the zoom functions in a similar way to that of Google Maps). The image also can be panned by clicking on the hand icon and dragging the image. At any level of zoom the image can be drawn on using the rectangle, ellipse or polygon tools. Figure 2 shows an example of a zoomed image with a polygon region round a suitable target. The polygon shape has been drawn by clicking around the target. When the user clicks again on a polygon vertex the polygon is closed and the bounding box region is displayed as a rectangle.

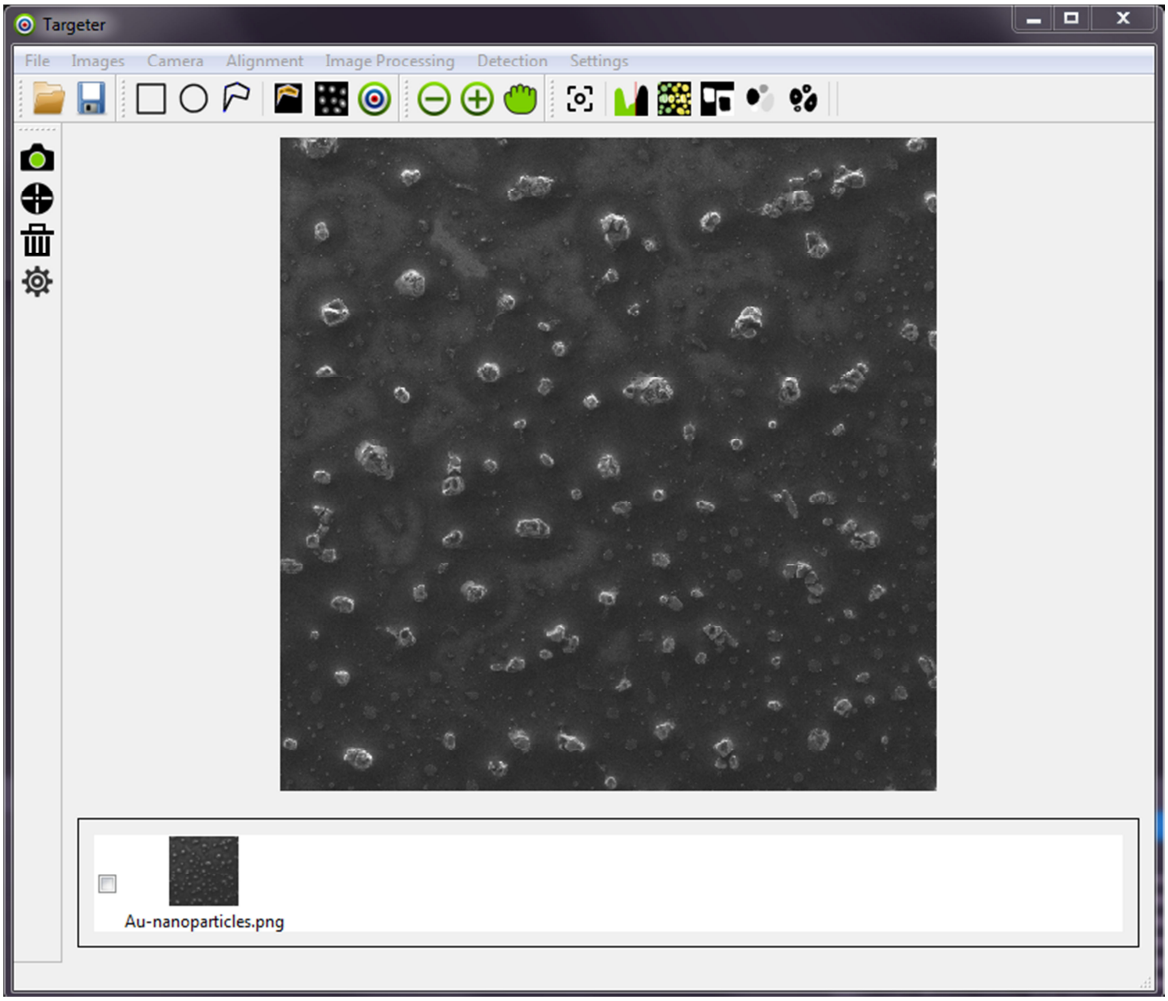

Figure 1: GUI interface of Targeter, showing example target detection image of Gold Nanoparticles.

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

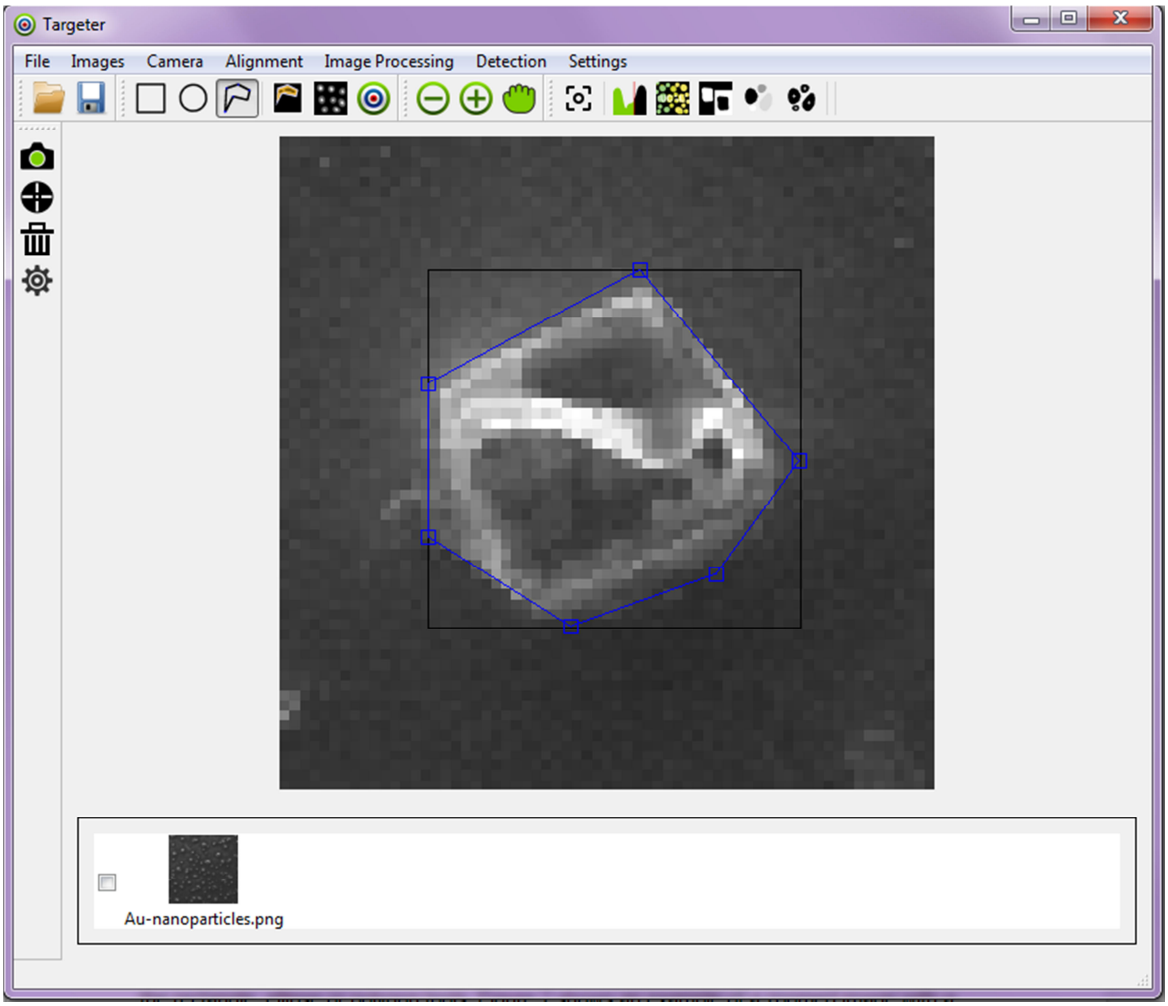

Figure 2: Zoomed/Panned image with target region selected by polygon.

By clicking on the 'create target image' icon  a target image is created as well as its binary mask (Figure 3). The images are also added to the thumbnail list and annotated accordingly in this case the sub-image is automatically labelled as 'target image'. By right clicking on the image thumbnail a sub-menu appears in which the user can define the image as a target or a detection image. In the example shown in Figure 3 the user has selected the original Au-nanoparticles.png image and set it as a 'detection image', this image will be used to find good targets with an example good target being given by the 'target image' and its mask image. Other 'good target' images may be also created in a similar way and also used in the target detection analysis.

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

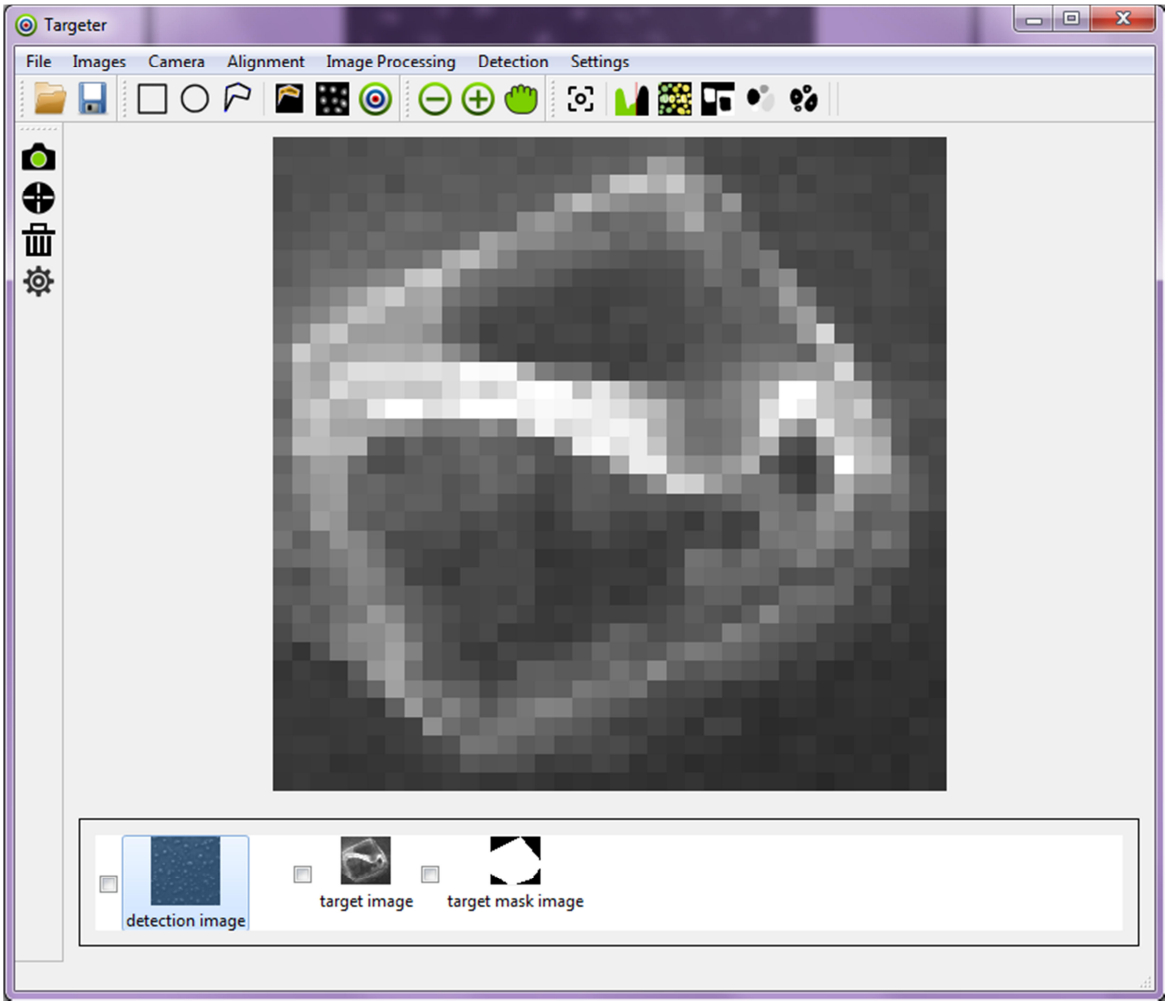

Figure 3: Mask sub-image and its binary mask.

The target detection algorithm proceeds by producing an intermediate score image, by thresholding this score image targets can be selected according to how good a target it is. A score image can be generated (providing a target and detection image are predefined) by clicking on the score image button  . A number of algorithms can be selected, these are given in the 'Settings' dialog which can be shown by clicking the settings button  . The 'Settings' dialog is shown in Figure 4, the user can select from COOC and the OpenCV algorithms SQDIFF, CCORR and CCOEFF and their normalised versions. The Color Co-occurrence Matrix algorithm (COOC) uses the method of Chang and Krumm [1]; it gives more accurate results than the OpenCV methods however with a slower run time. A hierarchical clustering method by Arifin and Asano [2] was used instead of k-Means clustering. This algorithm outperforms k-Means in colour/greyscale quantisation since it clusters a histogram rather than the image pixels. Other methods of speeding up the method are also possible however have not been implemented yet in this beta version of the software. The result of using the Colour Co-occurrence method (COOC) on the detection image is shown in Figure 5. The 'score threshold' setting determines the 'good target' score threshold, thresholding the image above this value produces a binary mask image of these good

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

targets. The score image represents an intermediate stage of analysis, the full analysis is made by clicking on the 'find targets' button  . Each target region is identified and labelled using a connected component algorithm and the centroid positions of the isolated image regions determined. Figure 6 shows the result of the 'Find Target' method target region of interest is shown by a green rectangle with the target center position being shown by a red cross (It may be necessary to maximise the image in order that the lines are drawn to resolution properly). The detected centre positions are the 'good target' positions and are written to a XML file along with the detected region of the target. Figure 7 contains an example of the XML output - in this beta version, distances are given in terms of pixel position from image top left and later versions will specify distance in micrometres from the fiducial marks. Figure 8 shows the result of processing another target image this time of orange polystyrene nanoballs. A bright ball was chosen as a good target and the algorithm found other bright balls in the image. While the algorithm finds bright regions within the image the accuracy of the detected areas should be higher so that the detection is better centered on the target. In future versions of the software the accuracy of the target detection center and region will be improved with regards to coloured images in which colour intensity (value) as well as hue should be taken into consideration in the COOC algorithm.

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

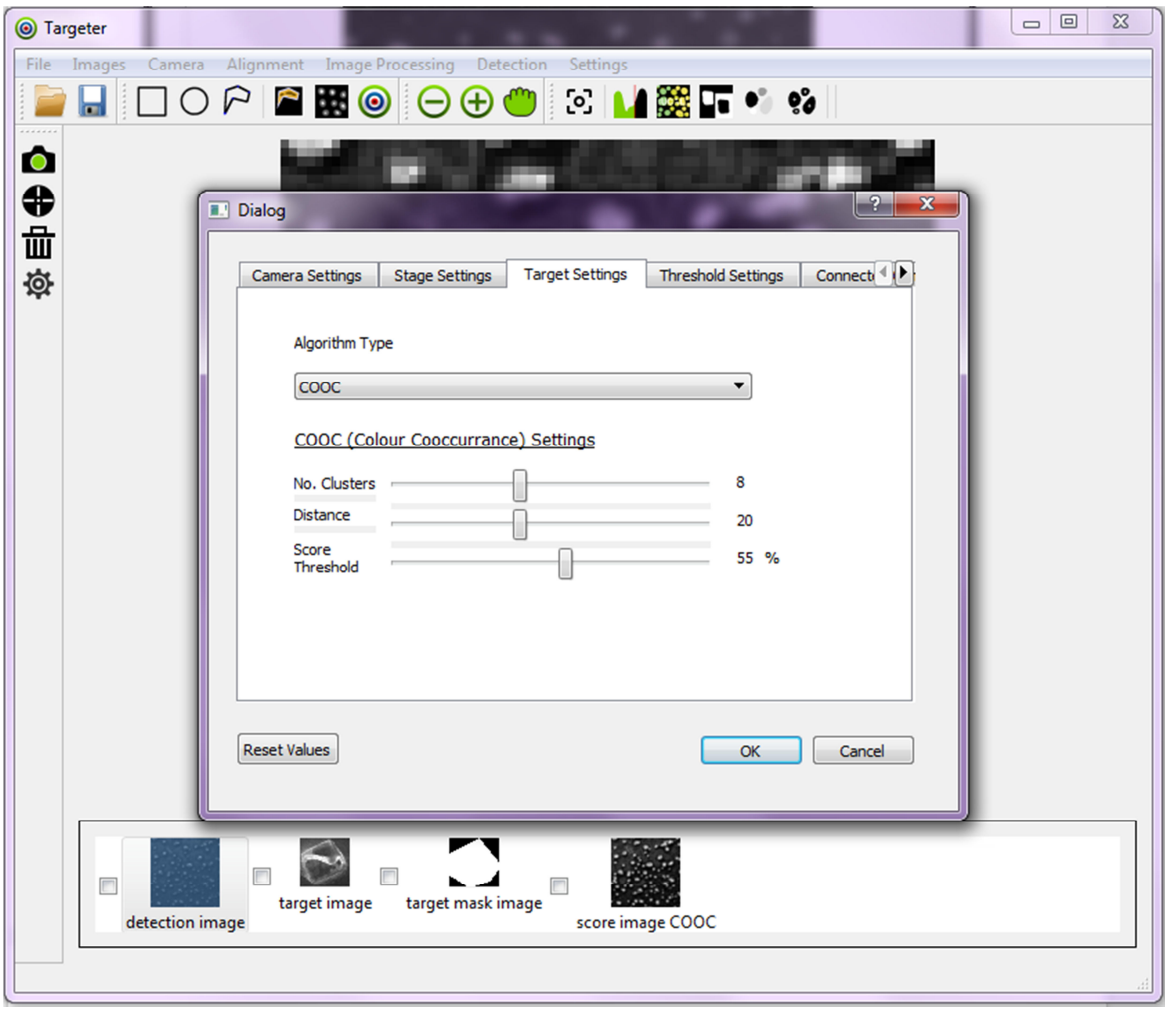

Figure 4: Targeter Settings Dialog showing the Target Settings tab.

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

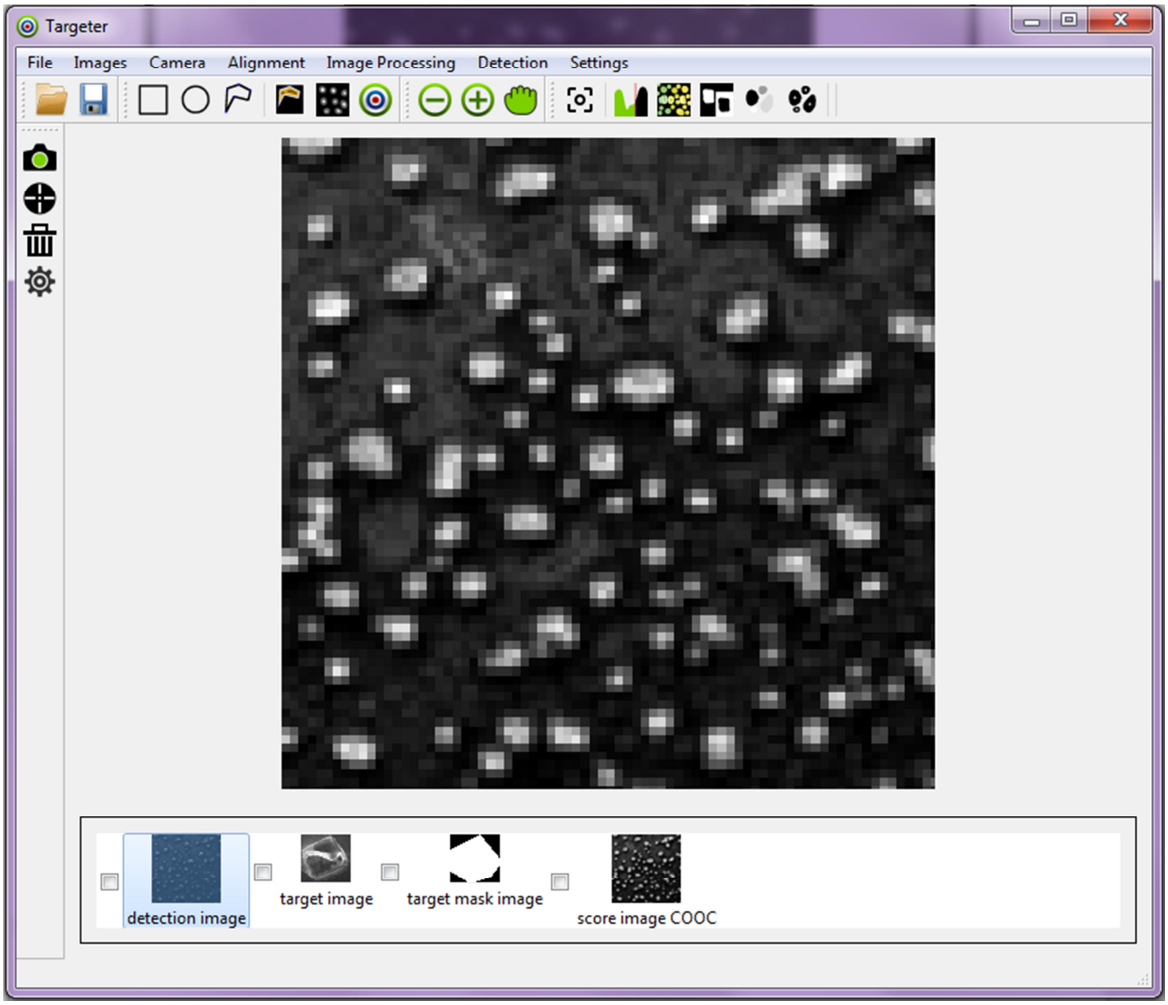

Figure 5: Score image of 'good target' positions within the image

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

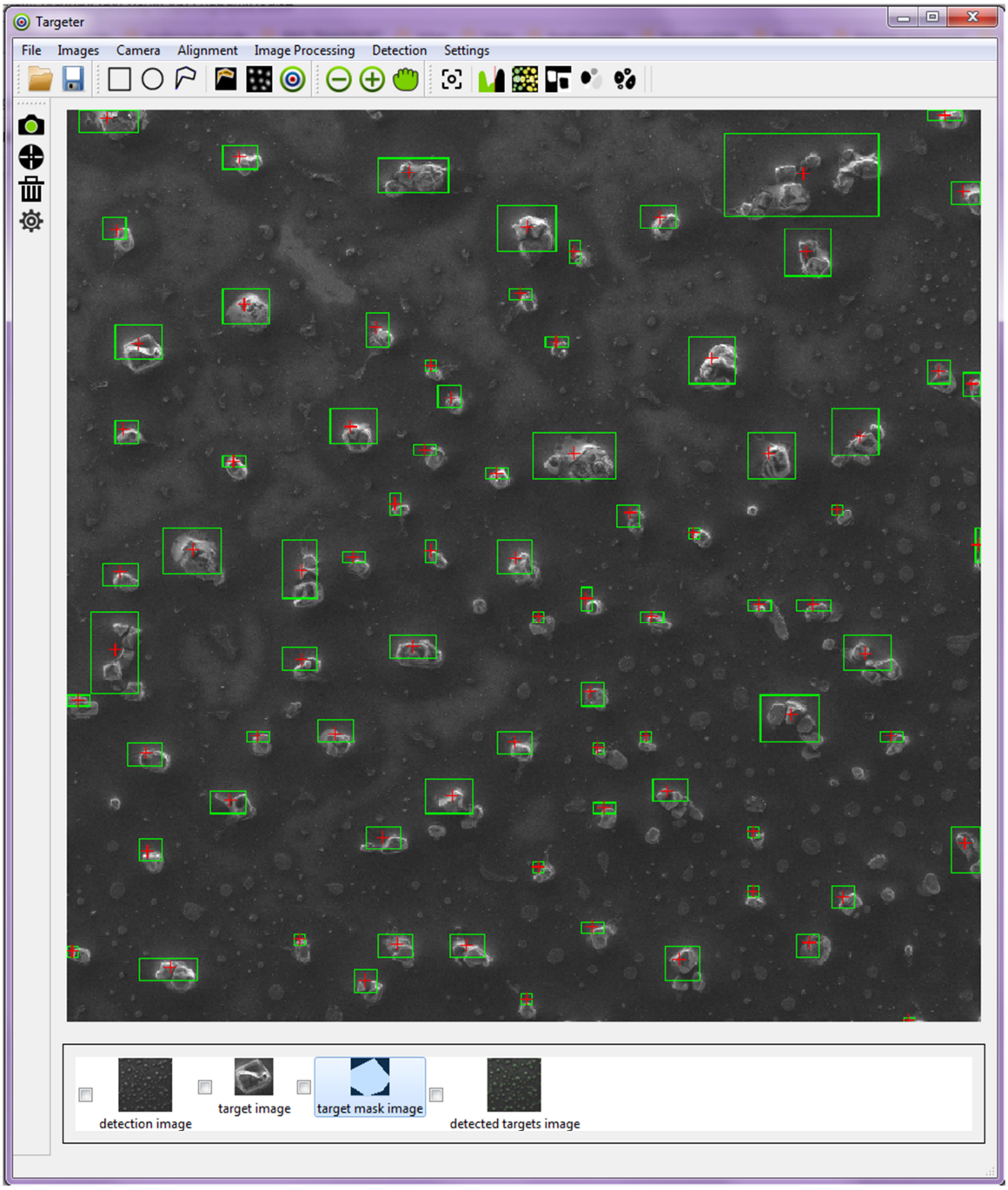

Figure 6: Detected good targets regions with target center position marked by red cross.

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

```
<target_coordinates>
  <target>
    <centroid x="35.5" y="8.5"/>
    <bounding_rectangle top="0" left="10" width="60" height="20"/>
  </target>
  <target>
    <centroid x="249.5" y="4.5"/>
    <bounding_rectangle top="0" left="240" width="20" height="10"/>
  </target>
  <target>
    <centroid x="732" y="7"/>
    <bounding_rectangle top="0" left="720" width="30" height="20"/>
  </target>
  <target>
    <centroid x="615.663" y="54.7326"/>
    <bounding_rectangle top="20" left="550" width="140" height="70"/>
  </target>
  <target>
    <centroid x="147.833" y="39.5"/>
    <bounding_rectangle top="30" left="130" width="40" height="20"/>
  </target>
  <target>
    <centroid x="283.447" y="52.9211"/>
    <bounding_rectangle top="30" left="250" width="70" height="40"/>
  </target>
  <target>
    <centroid x="209.5" y="54.5"/>
    <bounding_rectangle top="50" left="200" width="20" height="10"/>
  </target>
```

Figure 7: Sample XML coordinates output from Targeter

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

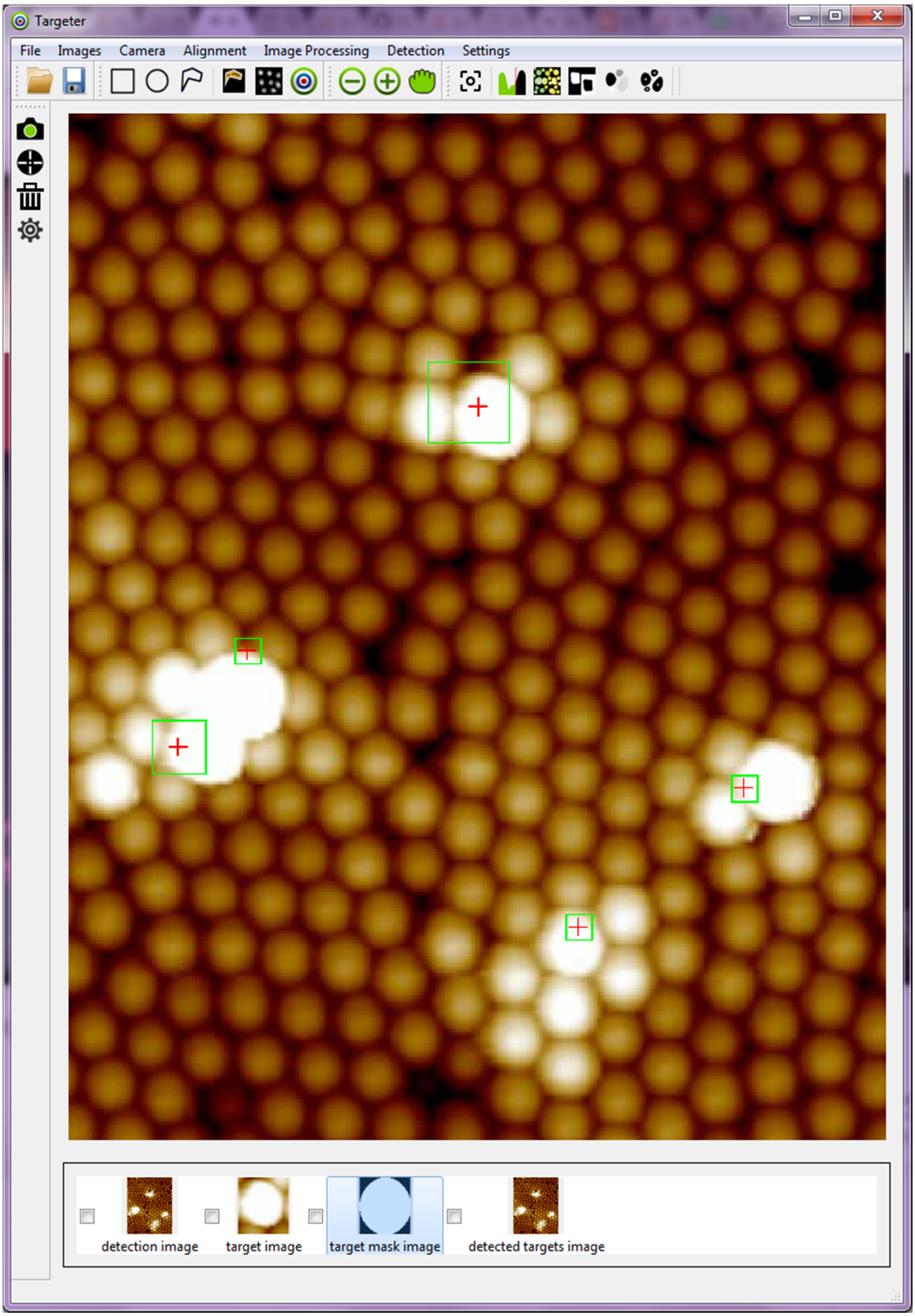

Figure 8: Polystyrene nanoballs, high brightness targets detected by the software.

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

1.5 Software repository

The software is available for the Windows platform since it is intended that the software works in conjunction with stages and CCD cameras of which Windows in generally is better supported. Windows source code of the beta release is given in the github project which is provided at the URL.

https://github.com/eucall-software/targeter

Doxygen documentation of the source code is provided at

https://eucall-software.github.io/targeter/

A listing of all classes and functions is given along with UML diagrams and caller graphs. An example of a caller graph for the DrawImage function is shown in Figure 9.

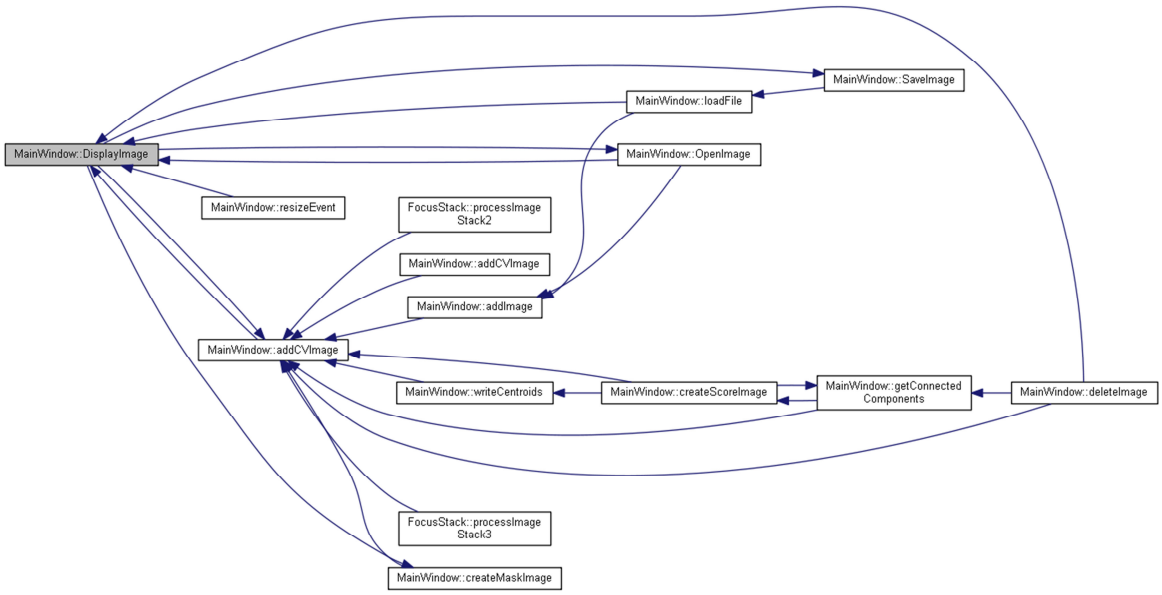

Figure 9: **Caller graph for the DisplayImage function**

1.6 Future developments of Targeter

There are a number of ways in which good target detection can be improved in future developments. The best performing algorithm, that based on co-occurrence matrices, is rather slow to compute. In this algorithm a window is slid over the image in a raster fashion and a co-occurrence matrix calculated by comparing each pixel with each other pixel. A speed up is possible by processing the image using a 1D vertical stripe and using an intermediate matrix to which values are added and subtracted during the raster scanning process. This technique will also allow more detailed spatial resolution of the score in the

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

image as the window can be stepped every pixel rather than the overlapping windows used presently.
At present the possible target positions are determined by a thresholding of the score image. The window size is generally smaller than the example target and therefore the best scoring regions may not cover the entire object. The window size can be enlarged to be the size of the object of interest however if this is performed then the processing time will increase considerably. More accurate target positions could be obtained if the score values are summed inside a sliding window which is the same size as the good target object bounding box or by combining score values over a larger neighbourhood using a Gaussian smoothing approach.
The detected good target regions will be further refined by comparing their size and morphology with that expected given the good target image. The method will also be used to perform a size based target detection based on an optimal thresholding of the object or its boundary as detected by a canny edge filter. Either the shape of the example target could be compared or candidate regions compared to a set of user specified measurement properties such as length, breadth, circularity etc.

1.7 Requirements for the pre-investigation microscope system Pre-Imager

In order to locate the inner sample section within the sample frame as well as the approximate position of the fiducial marks, an image of the entire sample frame will be obtained using an overview camera. The fiducial marks will then be localised to sub-micrometre resolution using the high magnification objective and used as a basis for the coordinate system of the sample. The regions containing the inner sample section with the user's targets will be masked as a region of interest and then used for the high resolution raster scanning by the high magnification objective. The obtained image and its location in respect to the coordinate system of the sample will be saved and used for the software Targeter. Due to the sample depth profile the focus will change during the raster scan and has to be scanned (focus stack) and adapted by focusing algorithms. The repeated use of these algorithms allows to achieve a depth profile of the entire region of interest. The pictures of the focus stack can be combined into one in-focus image of the entire sample.

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

2. Target identification software using TAG barcode *(ELI-NP)*

2.1 Description

The purpose of this software application is to identify and decode a TAG barcode which is used for target identification. The TAG, as defined in the previous deliverable D6.1, is a CODE39 barcode type that can be milled/printed/laser engraved/attached onto the target frame or target wafer itself, depending on the specific experimental configuration and constraints. Data from the barcode consist of 16 alphanumeric characters divided in three parts:

- 5 alphanumerical characters (A-Z & 0-9) encoding the registering facility (*ex:* EXFEL, ELINP, ELIBL, DESY1, etc.)
- 3 alphanumerical characters (A-Z & 0-9) to identify the design of the target frame in the database (*ex: 4IN, 9X9, FOI, etc.)*
- 6 digits encoding multiple frames unique serial numbers (*ex: 000000 – 999999)*

An example of a barcode having EXFEL4IN000001 encoding is described below:

2.2 Software application interfaces

The software is executed as a Command Line application having as arguments the name and extension of the input image containing a barcode and another optional argument for displaying an output image with the identified barcode. The encoding of the barcode is returned in the Command Line. It is also returned in a text file that is created having the same name as the input image but with .TAG extension.

2.3 Implementation

The application is built around two open-source libraries:

- **OpenCV** - http://opencv.org/
- **Zbar** - http://zbar.sourceforge.net/

OpenCV library is used for handling input image reading and output image display. The Zbar library is used for searching and decoding the barcode from the input image.
The application is developed in C++ and all data processing is CPU based.

2.4 Characteristics

Some benchmark tests have been made using the application to highlight decoding precision and execution time. We came up to the following conclusions:

- The barcode should be at least 400px long in the input image for a successful decoding.Auth

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

- The ambient light doesn't affect the decoding process as long as the barcode is visible.
- The image resolution increases the execution time.

3. Control software for target alignment *(ELI-Beamlines)*

3.1 Target alignment, centering and imaging

The correct positioning of a target/sample in its holder has to be verified directly in the laser irradiation conditions, i.e. in the so-called "target chamber" after placing the target holder/frame in the tower of the particular facility. Although the target tower is different in each facility, the target frame has to be precisely aligned before and during an experiment. Beside manual alignment procedures, software for automatic alignment has to be developed in order to shorten the time for the sample preparation before the experiment, as well as reducing human error.
Furthermore, in order to avoid wasting any laser shot during the experiment, it is also necessary to verify the real state of the target in air before pumping the vacuum chamber down (damages occurring during transport and mounting process), in vacuum (damages occurring during the pumping process) and after laser firing (neighbouring "fresh" surface damages). Thus, if pictures of individual target regions are captured during the alignment and centering procedure, the user/operator can select the undamaged targets.
If a manual alignment procedure is followed there is always human error to be taken into account within the errors present even with a fully automatic alignment procedure mainly coming from the evaluation of the best focus (or sharpest image) which has to be considered as subjective. Additionally a semi-automatic or fully automatic alignment procedure can significantly reduce the required time. The steps normally followed for a manual alignment procedure are described below.

Alignment of the x axis
1. Start at the top left corner of the frame (square 1 in Figure 10).
2. Find focus by adjusting in z axis (focus is in z1).
3. Move on the x axis to the top right corner of the frame (square 2 in Figure 1).
4. Find focus by adjusting in z axis (focus is in z2).
5. Evaluate and remember a value of $\Delta z = z1 - z2$.
6. Use pitch to move accordingly (rotate left or right).
7. Find focus by adjusting in z axis (focus is in new z2).
8. Go back to the previous corner and find focus in z axis (focus is in new z1).
9. Evaluate and compare old and new Δz to determine the pitch direction.
10. Repeat iterations 6-9 until you reach the required precision.

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

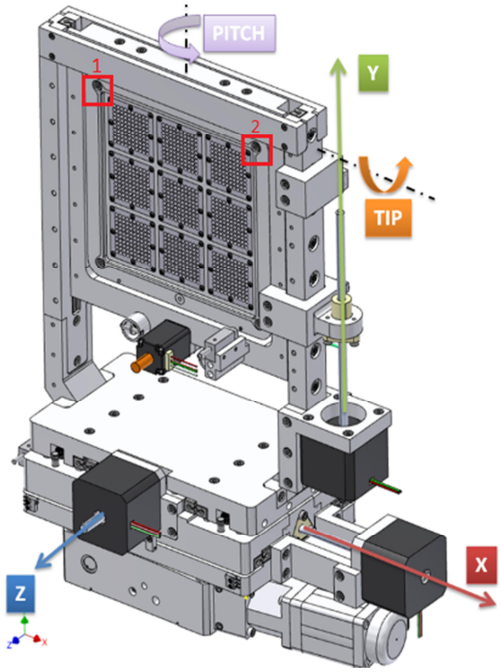

Figure 10: Manual alignment in axis x.

Alignment of the y axis

1. Start at the top left corner of the frame.
2. Find focus by adjusting in z axis.
3. Move on the y axis to the lower left corner of the frame.
4. Find focus by adjusting in z axis.
5. Evaluate and remember a value of $\Delta z = z1 - z2$.
6. Use tip to move accordingly (rotate left or right).
7. Find focus by adjusting in z axis.
8. Go back to the previous corner and find focus in z axis.
9. Evaluate and compare old and new Δz to determine the tip direction.
10. Repeat iterations 6-9 until you reach the required precision (Δz).

The overall target tower frame alignment is a combination of aforementioned alignment steps both in the x and y axes. In fact, the fine alignment is carried out by repeating the same procedure for individual target modules (each module can have different alignment settings and can contain different target). During the alignment of each target module, any individual target can be visualized and a snapshot can be captured, thus any potential damage can be identified).

3.2 Software for target alignment

The software developed for the alignment of the target after the positioning in the target tower is compatible with the central control system currently being implemented at the ELI Beamlines facility. The design is modular and uses a server-client approach. As much as

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

possible the C++ language is used and the Linux OS is used. The transport layer, providing communication between server and client, is based on the Tango package and the GUI is built using the QT package.
The target tower is considered as the "server side". This means that the control software runs on a control computer near the tower itself. However, the user can operate the tower remotely through a standard Ethernet network by using the specifically developed GUI (client). The flexibility of the network/transport layer allows to run the GUI in any physical location where the control network is present.

3.2.1 Low level packages

The software approach is based on a modular design. Each module comes in a form of a package designed to provide certain functionalities:

- Control the movement (motion package)
- Acquire image data from the camera (camera package)
- Process the received images (image package)

The following subsections provide a brief description of the main packages used for the control of the target tower.

The motion package

The motion package has four virtual application programming interfaces (API) and one common ancestor. These four APIs are then used to implement a control over various motion hardware. All the implementations share the same interface and are "transparent" for the software layers above them. The current motion of the tower is handled by Phidget stepper drivers, which are good enough for quick prototyping.

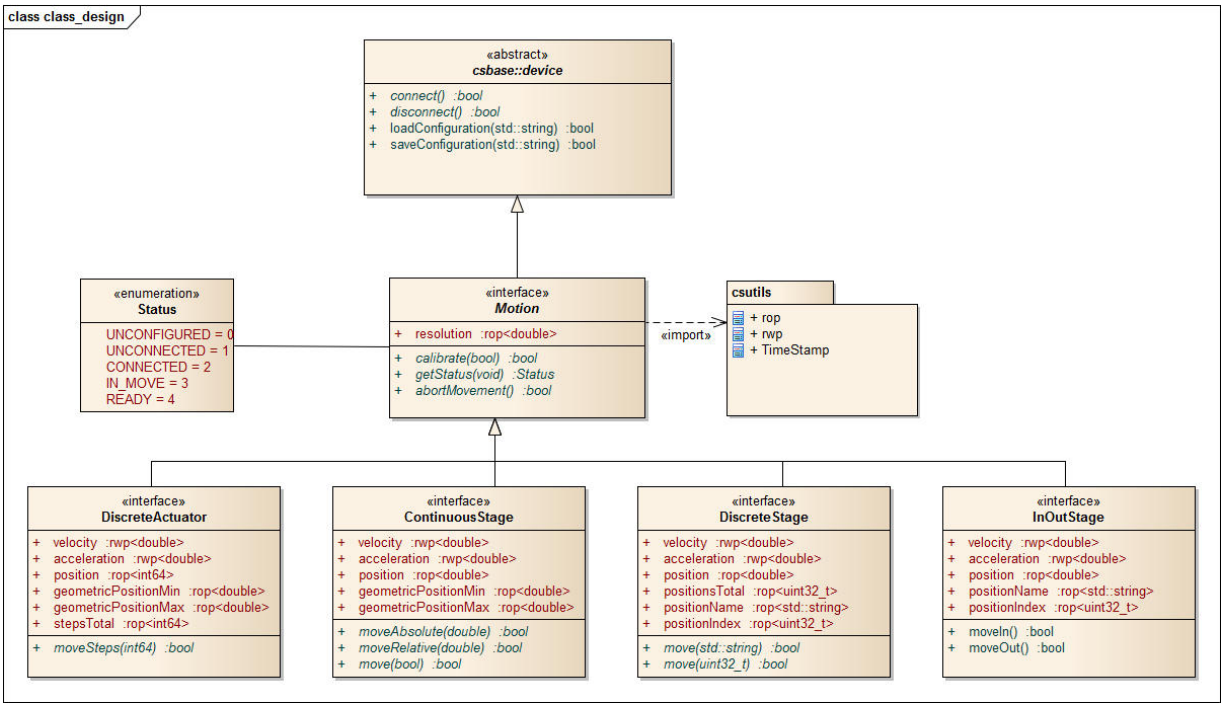

Figure 11: The motion package

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

The camera package

Another important package used for the tower control is the camera package (Figure 12). This package provides only one interface (API), but on the other hand allows to control all the GenICam compatible cameras, especially those with GigE or USB 3.0 physical interface. Using the package it is possible to fully access a camera in terms of configuration and data acquisition.

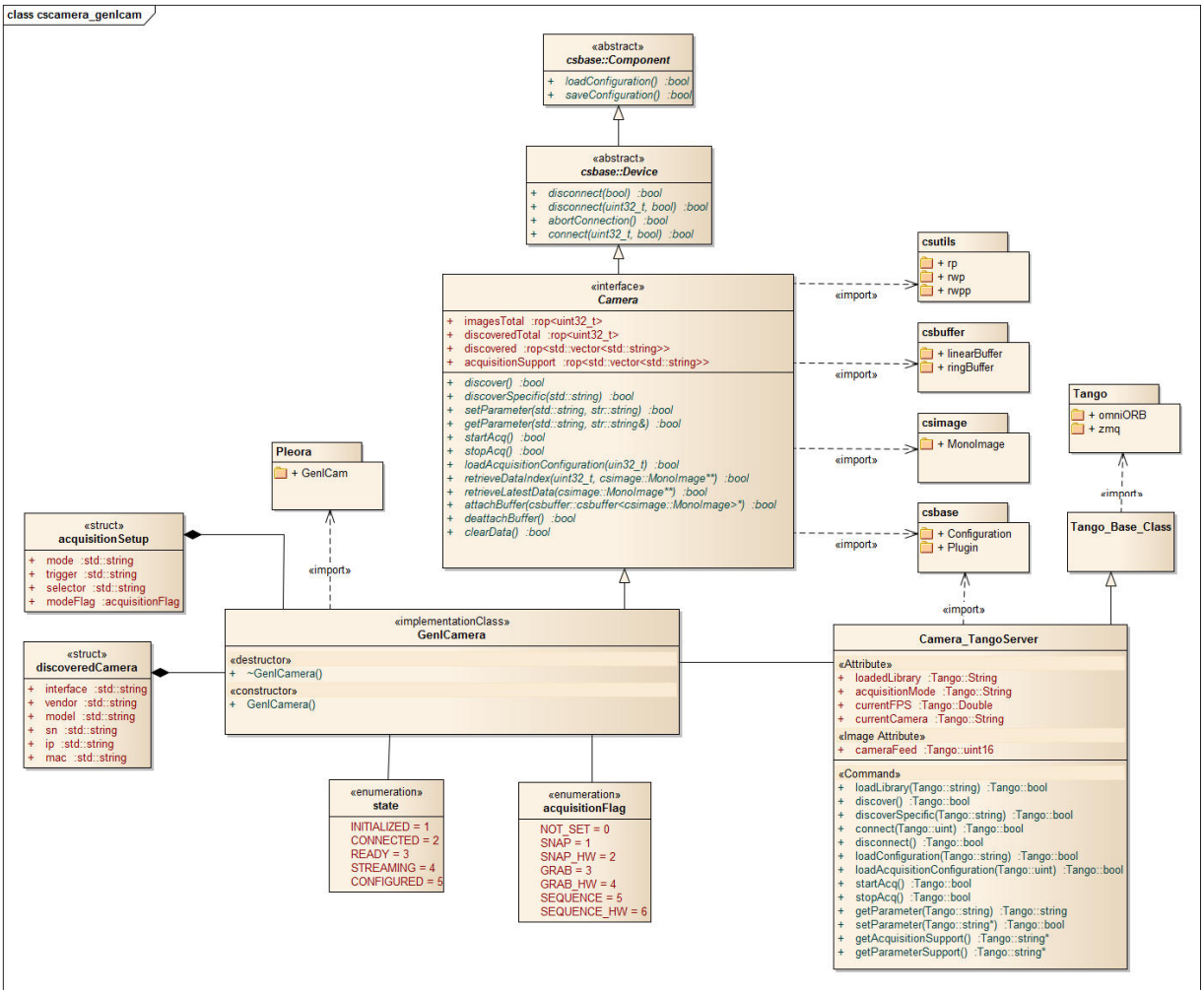

Figure 12: The camera package.

The image package

This purely software package provides two main features: image container and image processing engine. The image container is simply a wrapper around a data buffer of the image. The buffer is in default an array of unit values, which allows storing images up to 16 bit depth. The goal is to have a common image data object for the whole control system.

The second feature is the image processing engine which delivers multiple image processing operations from several different categories, for example geometrical transformations, morphing transformations, thresholding, Fourier transformations, edge detections and shapes/contour detection.

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

3.2.2 API and the state machine

The design of the API and the state machine for the Target Tower can be broken down into two parts: technological/system requirements and analysis of the user cases. The first category is mainly related to the motion system (homing) and configuration of the tower software. On the other hand, the user cases reflect what is expected from the tower from the experimentalist point of view - manual movement (absolute, relative and run) and finding the local focus, large frame alignment (in air and in vacuum), small frame (or holder) alignment (in air and in vacuum) and centering on a sample. The API (scheme shown in Figure 13) aims for minimalistic and simple public interface with the following functions:

1. loadConfiguration()/clearConfiguration();
   - This call loads an xml configuration file which holds information about the used motion and camera hardware, constants and settings for the following homing and aligning processes.
   - Once the file is loaded, the hardware is initialized and ready to be used.
   - To clear configuration the 'clearConfiguration' function can be used. All the previously connected hardware is disconnected
2. home()/abort();
   - This call starts the homing process described later.
   - Abort cancels the homing process.
   - NOTE: the homing is done as a background thread.
3. moveRelative()/moveAbsolute()/run()/abort();
   - This set of functions allows the manual control of the tower. The input is in millimetres as it is required by the used motion system package.
   - Any motion can be interrupted by calling the 'abort' function.
   - NOTE: the homing is done as a background thread.
4. align();
   - This function can be used to run the alignment process for both frame and holder.
   - Can be interrupted by calling the 'abort' function.
   - NOTE: the alignment process of any kind runs in a background thread.
   - A detailed description is provided in the next subsection.
5. analyze();
   - Can be called once the align function is done. In order to analyze a specific sample holder the alignment process has to be finished for the sample holder.
   - Can be interrupted by calling the abort function.
   - NOTE: the analyze process of any kind runs in a background thread.
   - Details about the analyze process can be found in the next subsection.
6. focus();
   - This call will find the best focus by moving through the z axis and calculating the

'focus' as is described in the next subsection.

NOTE: the focus runs in a background thread.

Can be interrupted by calling the 'abort' function.

7. disconnect();

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

- Can be used after the 'homing' function is finished to disconnect from the towers hardware. The call includes the 'clearConfiguration' function.

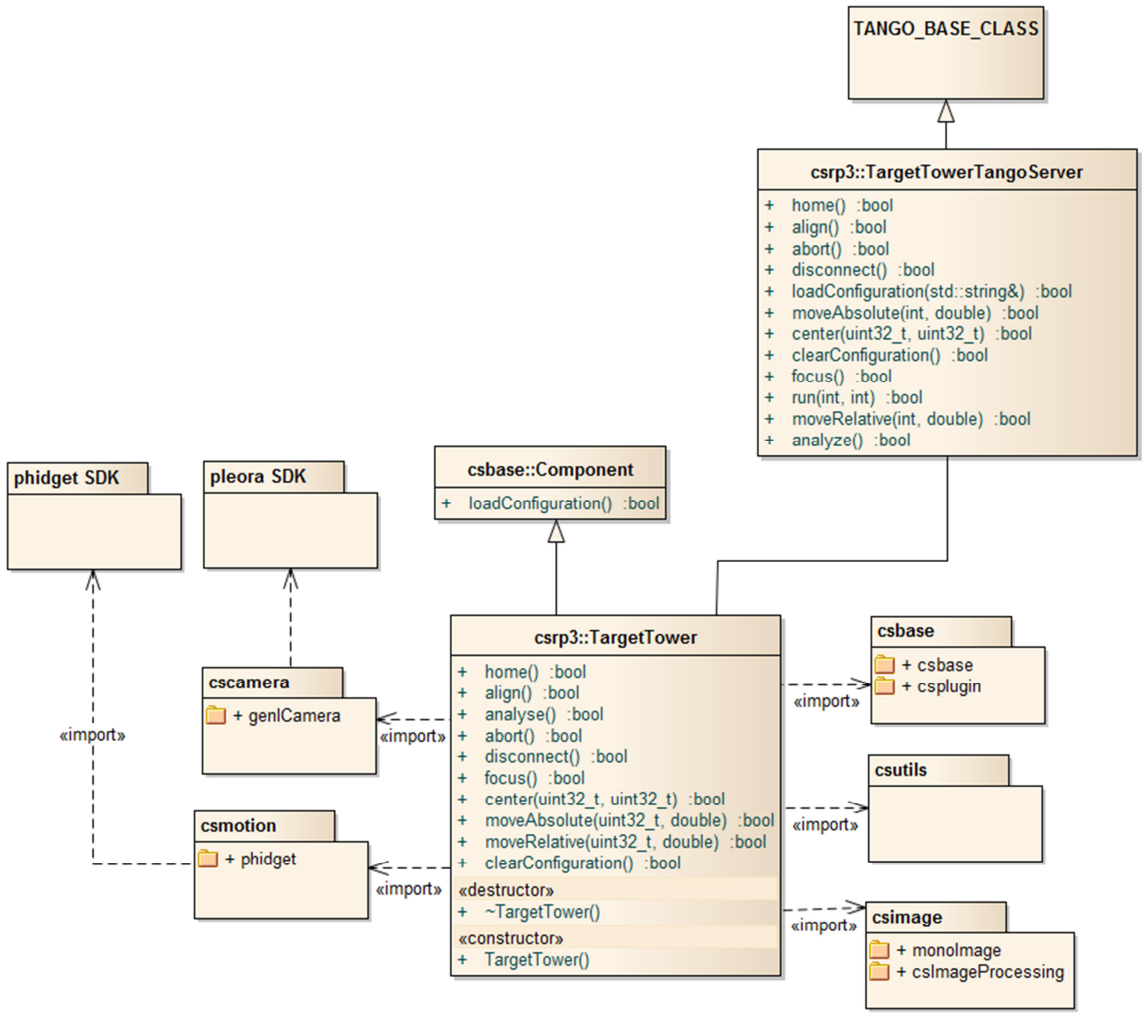

Figure 13: The Target Tower API scheme.

The availability of the functions is determined by the state machine of the target tower with exception being the 'analyze' function which is further conditioned by the alignment process for specific sample holder being finished.

When the Target Tower object is instantiated we are in the Initialized state. Here, calling the 'loadConfiguration' will transition the tower to the Configured state. From here we can fall back to the Initialized state ('clearConfiguration') or call the 'homing' function to run the homing process. If successful (or not aborted) the object moves to the Ready state. Here various functionality is available:

- Manual movement → during movement the tower is in the Moving state.
- Alignment process → during alignment the tower is in the Aligning state.

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

- Focus → during finding the focus for current position the tower is in the Aligning state.
- Analysis → during analysis process the tower is in the Analyzing state.

The tower will move out of each of these states (Moving, Aligning, and Analyzing) once the corresponding action (movement, alignment process, focusing, analysis) is finished and falls back to the Ready state. From the Ready state it is possible to transition to the Initialized state by calling the 'disconnect' function.

Tango server is a part of the Target Tower software. This server is a representation of the Target Tower on the control network. It allows to make remote calls of the tower control software and sends back (to a client) results of calls, image data and positions of the axes of the Target Tower.

3.2.3 Homing process

The homing process allows the tower software to map all the possible movement ranges for all the axes. First the pitch axis (rotational) is homed. The reason for this is to prevent potential collisions, for example when the tower is rotated to positive maximum and in the same time axis z and x are moving toward their maximums.
After the pitch axis is homed the tower moves to a default pitch position and the remaining axes are free to be homed all at once.
Once homed, the tower is put to a default position for all the axes. This position can be configured by loading the configuration file right after starting the software.

3.2.4 Automatic alignment process

This process is based on the manual alignment process described above and is the same for the frame and sample holder alignment (also the same in air and in vacuum). We can select points dedicated to the tower frame alignment (three points are enough) and points for the sample holder alignment (three are enough). The coordinates of the points are configurable by the configuration file loaded right after starting the software. The tower first aligns the x axis and then the y axis. The steps are the same only the points are changed. The steps are as follows:

1. Move to the first point.
2. Find focus (described in next subsection) → now we have a z1 position.
3. Move to the second point.
4. Find focus (described in next subsection) → now we have a z2 position.
5. Calculate Δz. If it is good enough, the axis alignment is finished.
6. Make correcting movement in the pitch axis (or tilt axis for y axis alignment).
7. Go to the first step.

An important part of the alignment process is the find focus function. This function works via two iterations. First it is necessary to find the "rough focus" by moving with a reduced

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

velocity through the whole range of z axis and every few steps the focus is evaluated. Then it is necessary to move to the z position of the best focus found (largest number when evaluating the contrast).
The second part is the “fine focus” we move in a small range around the previously found best focus with highly reduced velocity and again we evaluate the focus. The best found focus is the resulting focus. The focus evaluation is based on contrast evaluation. Basically a kernel is moving through the whole image and calculates the ’focus’ value.

3.2.5 Analysis process

The analysis is a process where we move from one sample of a sample holder to another until we scan all the 100 samples. On each step we zoom inside the ”sample hole” and focus on the sample itself. At this point we save both the image of the sample for later analysis and the corresponding coordinates. The position of the first sample to analyze is loaded in the configuration file.
The distances between samples are known for both x and y axis. Still it is better to ensure that we are properly centered on each sample before saving an image of the sample. This is done by a simple image processing sequence when we zoom on the sample and then we perform a correction movement. The sequence is as follows:

1. Resize image by 0.3 in both dimensions.
2. Laplacian edge detection with high kernel size.
3. Clear the image by applying morphological operation ’open’.
4. Invert the image, just for better visual representation.
5. Circle detection by Hough transformation with min/max radius set to be close to the expected radius of the sample.
6. Extract the circle center coordinates.

In order to center the sample we calculate the offset using the detected sample center and the center of the reference (the camera image). Then we simply make the correcting movements in the x and y axis.

3.2.6 GUI

The Target Tower GUI (Figure 14) acts as a Tango client. It is connected to the tower Tango server through a control network. The GUI itself is written in the qml language, which belongs to the QT package. The GUI is internally driven by the same state machine as the control software. If functionality is not available the corresponding button or field is greyed out.

This project has received funding from the *European Union’s Horizon 2020 research and innovation programme* under grant agreement No 654220

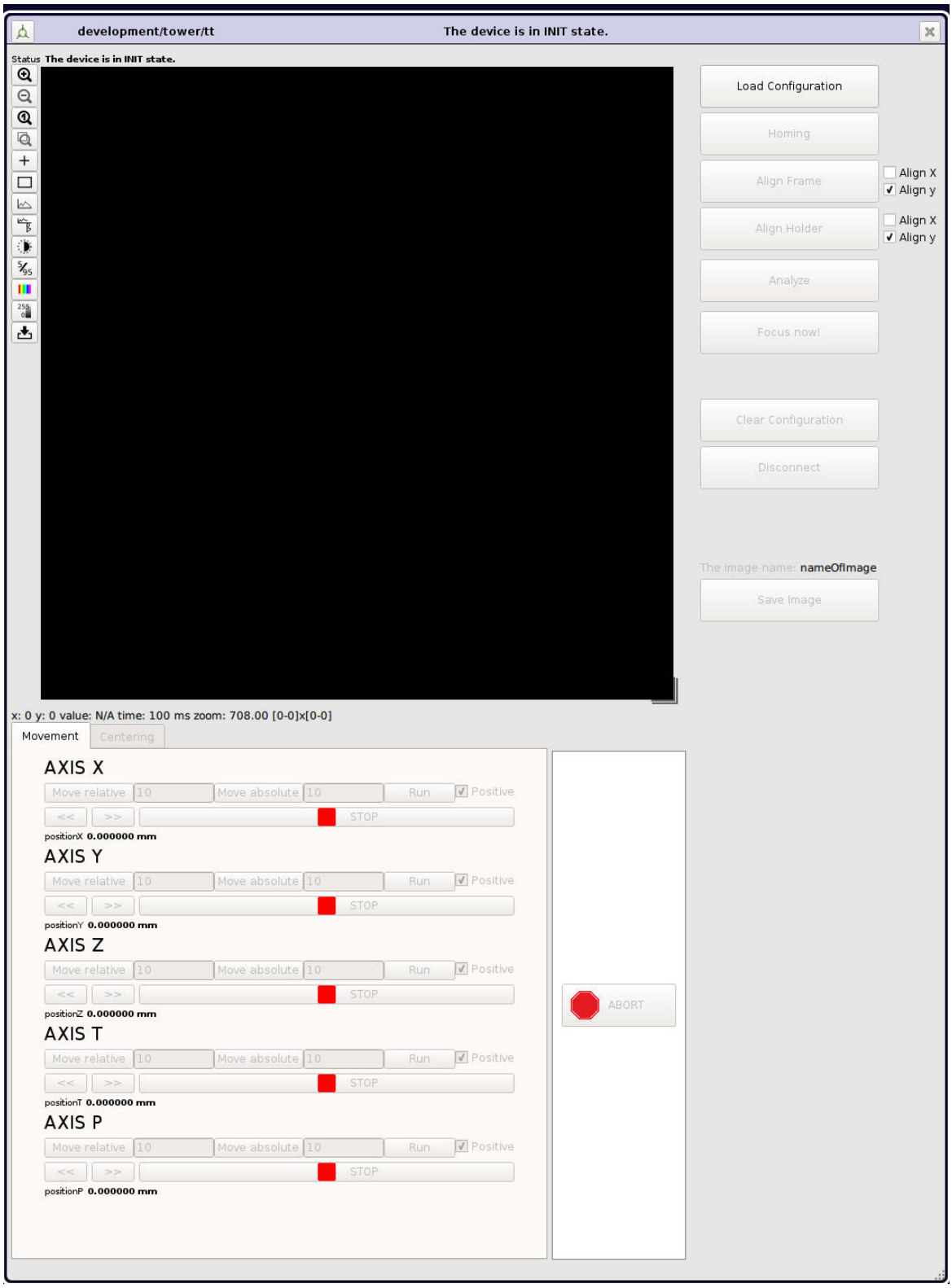

Figure 14: The GUI of the Target Tower in the initialized state.

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

4. Software repository

The code for Targeter can be found at:

https://github.com/eucall-software/targeter

while the Target identification software using TAG barcode can be found at:

https://github.com/eucall-software/ELI-NP-software-for-barcode-reading

5. References

[1] P. Chang and J. Krumm, *Object Recognition with Color Co-occurrence Histograms*, IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Vol. 2, pp. 504-509, (1999).

[2] A. Z. Arifin and A. Asano, *Image segmentation by histogram thresholding using hierarchical cluster analysis*, Pattern Recognition Letters, Vol. 27, No. 13, pp. 1515-1521, (2006).

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220