# Application Note:
# Field-of-View Expansion

# Contents

# 1. Introduction

Machine vision systems have fundamental limitations with respect to resolution, field of view (FOV), and depth of focus (DOF). Angular resolution and FOV are conflicting specifications, limited by the size and resolution of the image sensor.

The field of view of an imaging system is given by the focal length $f$ of the objective and the image sensor size $S$:

$$\mathrm{FOV} = 2\tan^{-1}\frac{S}{2f}$$

The DOF of an imaging system with focal length $f$, focus or object distance $d$, f-number $N$ and circle of confusion $c$ is given by:

$$\mathrm{DOF} \approx \frac{2d^2Nc}{f^2}$$

Extending the FOV while maintaining high resolution is crucial for a variety of applications. For example, in public surveillance of train stations or airports, a large scene needs to be imaged with a high resolution to do detailed object detection or face recognition (Figure 1). Another example is traffic sign detection in autonomous driving, which benefits from FOV expansion by enabling to read signs at a larger distance.

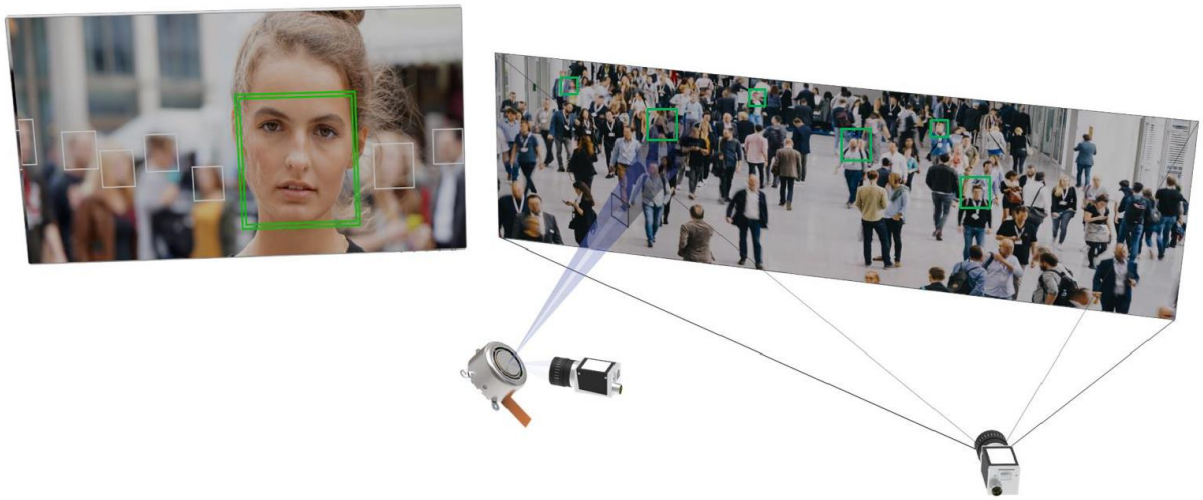


*Figure 1: FOV expansion and area of interest selection using a 2D fast steering mirror.*

Other applications where good resolution and a large FOV are beneficial are barcode scanning and iris recognition. To maintain high resolution, the barcode or the human eye need to be accurately positioned to be within the camera's FOV. This limits the ease-of-use of such imaging systems or may make it necessary to use multiple devices to cover the necessary FOV.

An elegant way to satisfy the extreme resolution and field-of-view requirements that these applications pose is to combine a 2D fast steering mirror with a liquid lens. The mirror is used to select a small area of interest within a large FOV, while the liquid lens allows one to quickly adapt the focus.

Optotune's 2D fast steering mirror MR-15-30 with its 15 mm aperture and large ±25° scan range as well as the tunable lens EL-16-40 with its unrivalled 16 mm clear aperture are both ideally suited for this application, allowing for a compact and flexible solution. The MR-15-30 has the advantage over 2D-MEMS mirrors in terms of both travel range and clear aperture, and over galvo mirrors in terms of compactness, requiring a single device to

cover two axes of rotation.The main technological benefits of the MR-15-30 are shown together with the working principle of our liquid lenses in Figure 2.

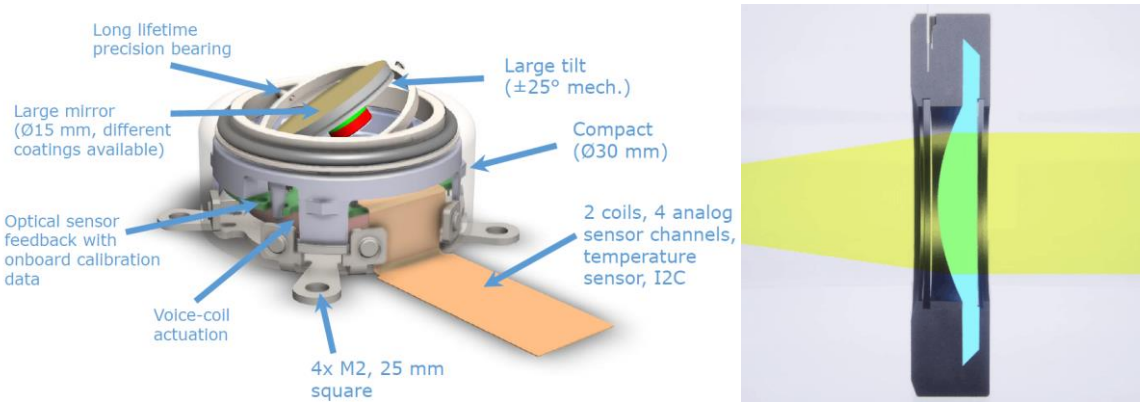



*Figure 2: Main technological benefits of Optotune's MR-15-30 fast steering mirror and working principle of a focus tunable lens.*

The combination of the MR-15-30 and the EL-16-40 is implemented in the Optotune's FOV Expansion Development kit (datasheet). The complete kit is shown in Figure 3. It is a plug-and-play solution for high-resolution imaging across a wide, 100° optical angle FOV. The kit comes with software support in our Optotune Cockpit, with built-in widgets for both face recognition and area-of-interest selection.


*Figure 3: Hardware components included in the FOV Expansion Development kit.*

## 2. Camera calibration

The simplest and most widely used camera model is the that of a pinhole camera. Figure 4 shows how points in the target or focal plane $(x_t, y_t)$ get mapped onto the image plane $(x_c, y_c)$ of the camera. For simplicity, the image plane is thought to be in front of the pinhole, at a distance equal to the focal length.

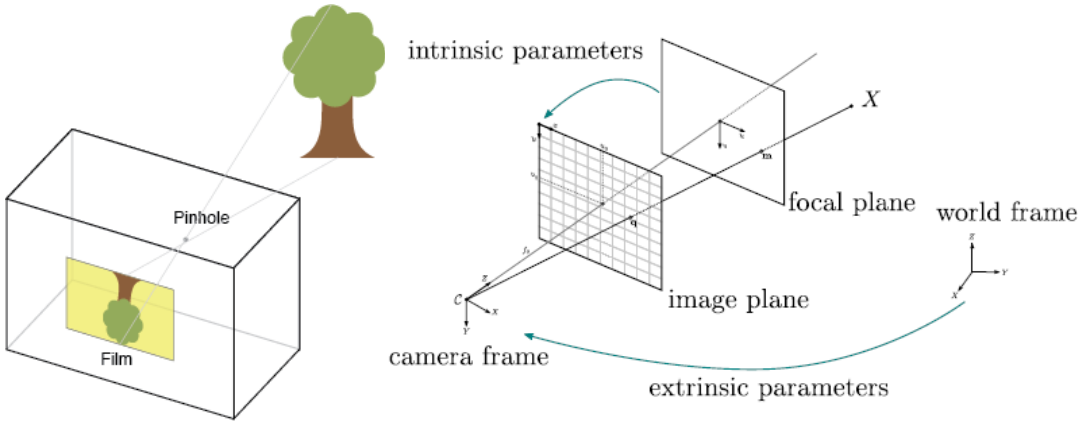



*Figure 4: Pinhole camera model on the left and the associated intrinsic and extrinsic calibrations on the right.*

This simple model can be represented by the following relation written in homogenous coordinates

$$\begin{bmatrix} x_c \\ y_c \\ 1 \end{bmatrix} = \boldsymbol{K} \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$\text{where } \boldsymbol{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \boldsymbol{R} = \boldsymbol{R}^{-\mathsf{T}} \text{ and } \boldsymbol{t} \in \mathbb{R}^3$$

The extrinsic parameters, i.e. the rotation matrix $\boldsymbol{R}$ and the translation vector $\boldsymbol{t}$, describe the location and orientation of the world frame with respect to the camera frame. We can simplify the above equation by selecting normalized coordinates $(x_t, y_t)$ on a target plane perpendicular to the optical axis at unit distance:

$$\begin{bmatrix} x_c \\ y_c \\ 1 \end{bmatrix} = \boldsymbol{K} \begin{bmatrix} x_t \\ y_t \\ 1 \end{bmatrix}$$

This parametrization with only two coordinates makes sense because a camera on its own cannot determine how far an actual object is away from the image plane. Without knowing actual object dimensions, just the angles to the object are known.[1]

While for an ideal pinhole camera $f_x = f_y = f$, it makes sense to allow for a small difference in this parameter due to an imperfect sensor or lens misalignment. The factors $c_x$ and $c_y$ are usually close to half of the image width and height, but can differ due to lens decentering.

For wide-angle imaging systems, this linear camera model does not provide enough fidelity. Therefore, additional nonlinear terms need to be added to account for lens distortion. A common model is

$$x_{c,\text{distorted}} = x_c(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + 2p_1 x_c y_c + p_2(r^2 + 2{x_c}^2)$$

$$y_{c,\text{distorted}} = y_c(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + p_1(r^2 + 2{y_c}^2) + 2p_2 x_c y_c$$

with $r = \sqrt{{x_c}^2 + {y_c}^2}$ and $k_1, k_2, k_3$ describing radial distortion (pincushion and barrel distortion) and $p_1, p_2$ describing tangential distortion.

A standard and practical technique to fit these parameters to a set of measurements was found by Zhang[2]. All that is required is a set of pictures of a flat checkerboard taken from a variety of positions and angles. An example implementation can be found following this link, with an example picture shown in Figure 5. Typically, at least 10 views of the checkerboard are required for a precise model fit.

In the case of the FOV Expansion Development kit, Optotune Cockpit allows importing a calibration matrix and distortion vector as defined above for the face recognition program. The condition $c_x < c_y$ has to be fulfilled due to the portrait format of the wide-angle camera.

*Figure 5: Example image of a checkerboard used to calibrate a wide-angle camera to correct for lens distortion.*

[2] Zhang, Zhengyou. (2000). A Flexible New Technique for Camera Calibration. Pattern Analysis and Machine Intelligence, IEEE Transactions on. 22. 1330 - 1334. 10.1109/34.888718.

# 3. Mirror coordinate system

## 3.1. Definition of the XY coordinate system

A coordinate system is defined to calibrate the internal optical feedback mechanism and relate it with a physical mirror position. This coordinate system is a Cartesian coordinate system with axes X and Y. The X-axis is perpendicular to the cable protruding from the mirror head and the Y-axis is parallel to this cable. The numerical values on the axes are unitless and defined via the maximum deflection of the mirror in optical angles, i.e. 50°. Along each axis, a deflection of +50° corresponds to a value of +1 and a deflection of -50° corresponds to a value of -1. This corresponds to the projection observed on a screen if the mirror reflects a laser beam travelling along the z-axis, incident on its center. For example, the analytical relationship between deflection angle θ and coordinate system value $x$ along the x-axis is:

$$\frac{\tan(\theta)}{\tan(50°)} = x$$

Figure 6 below illustrates the relationship between the deflection on-axis and the X-coordinate using three examples.

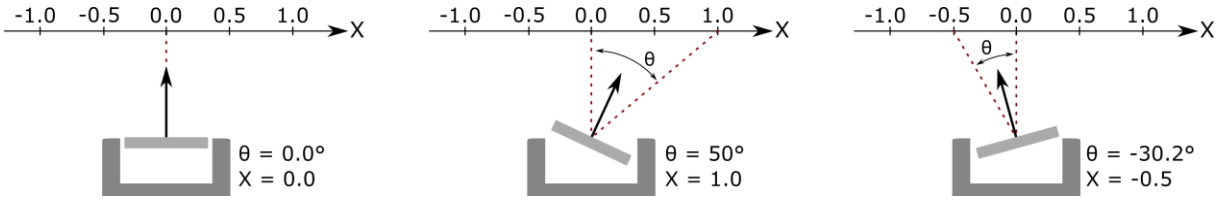



*Figure 6: Example mirror deflections and the corresponding X-coordinate according to the mirror coordinate system definition.*

The mirror has a maximum mechanical deflection of 25° in every direction. Therefore, in the XY coordinate system, a unit circle with radius 1 contains all accessible values, as shown in gray in Figure 7.

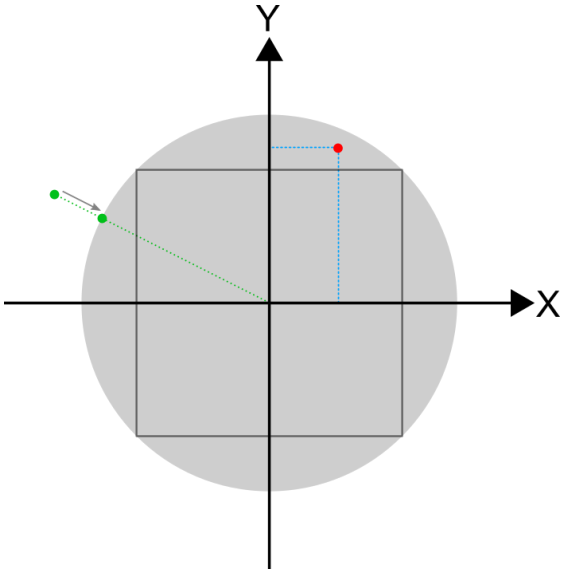



*Figure 7: Mirror coordinate system and accessible positions.*

The mirror can access every possible combination of X and Y values for both X and Y < 0.7 (black square in Figure 7). If one of the XY values exceeds 0.7, the other value must decrease, so that at any time $X^2 + Y^2 \leq 1$ (the red dot in the figure above is an example). The firmware of the mirror controller automatically reduces XY positions outside the accessible unit circle by moving them to the nearest edge of the unit circle (green dots in the figure above). This behavior prevents mirror and driver damage.

When driving the mirror in open loop mode, it is possible to reach angles larger than 25°, which correspond to XY values bigger than 1. However, these angles are outside the guaranteed and calibrated range of the mirror and calibration accuracy can vary significantly.

## 3.2. Transformation between different Cartesian projection coordinates

When using the mirror in a scanning system, typically the arrangement with 0° AOI is not practical, since incident and reflected beam paths would overlap. Figure 8 shows the general case: the mirror with its middle point $M$ and normal vector $\boldsymbol{n}_m$ rotates around the center of rotation $C$ and reflects an incoming beam, specified by a point $P_0$ and a unit vector $\boldsymbol{n}_0$. The reflected beam starts at point $P_1$ and has the direction of the unit vector $\boldsymbol{n}_1$. Finally, the reflected beam hits a target plane at the point $P_2$, where the target plane is located at a distance $D$ from the undeflected mirror center and a has the normal vector $\boldsymbol{n}_t$.

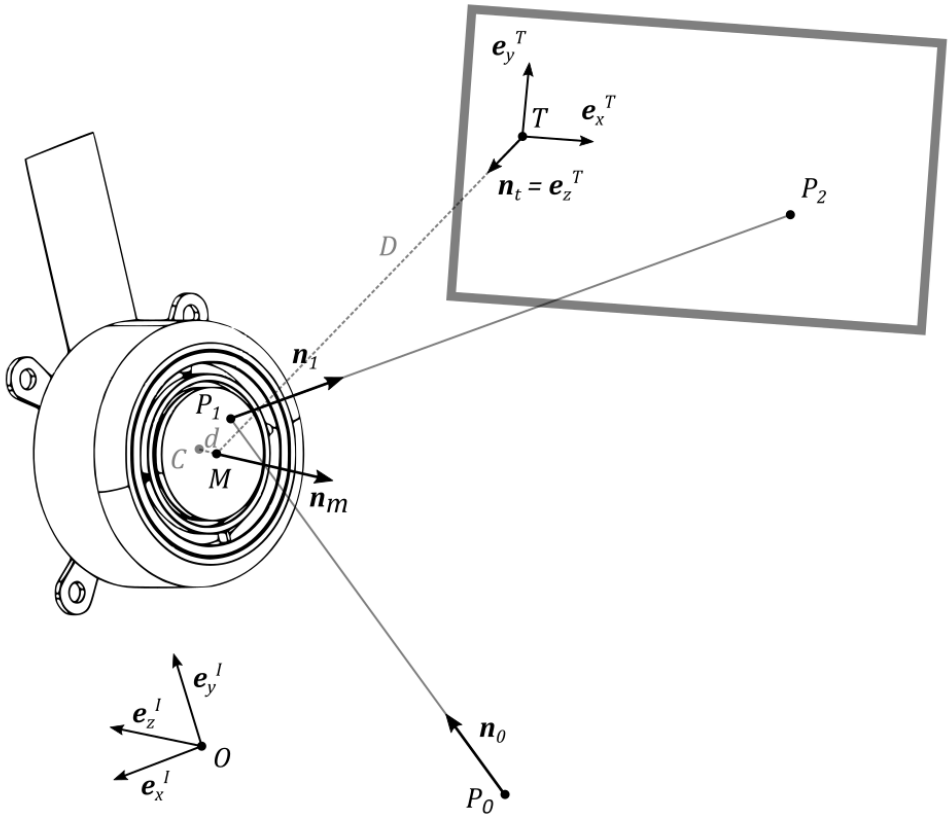



*Figure 8: Coordinate system definition of the mirror and target plane.*

This general arrangement captures distortion effects due to

- AOI of the incoming beam
- Rotated target plane
- Off-centered incoming beam
- Distant center of rotation

The vector analysis to calculate the beam path is straightforward. First, inserting the equation for the incoming beam $\boldsymbol{r} = \boldsymbol{r}_{OP_0} + t \cdot \boldsymbol{n}_0,\ t \in \mathbb{R}$ into the equation for the mirror plane $(\boldsymbol{r} - \boldsymbol{r}_{OM}) \cdot \boldsymbol{n}_m = 0$ yields the intersection point $P_1$

$$\boldsymbol{r}_{OP_1} = \boldsymbol{r}_{OP_0} + t_1 \cdot \boldsymbol{n}_0$$

$$\text{where } t_1 = \frac{(\boldsymbol{r}_{OM} - \boldsymbol{r}_{OP_0}) \cdot \boldsymbol{n}_m}{\boldsymbol{n}_0 \cdot \boldsymbol{n}_m} \text{ and } \boldsymbol{r}_{OM} = \boldsymbol{r}_{OC} + d \cdot \boldsymbol{n}_m$$

Then, the reflected beam is obtained by applying the law of reflection

$$\boldsymbol{n}_1 = \boldsymbol{n}_0 - 2 \cdot (\boldsymbol{n}_0 \cdot \boldsymbol{n}_m) \cdot \boldsymbol{n}_m$$

Finally, we calculate the intersection point with the target plane $P_2$

$$\boldsymbol{r}_{OP_2} = \boldsymbol{r}_{OP_1} + t_2 \cdot \boldsymbol{n}_1$$

$$\text{where } t_2 = \frac{(\boldsymbol{r}_{OT} - \boldsymbol{r}_{OP_1}) \cdot \boldsymbol{n}_t}{\boldsymbol{n}_1 \cdot \boldsymbol{n}_t}$$

We can now express the vector $\boldsymbol{r}_{TP_2} = \boldsymbol{r}_{OP_2} - \boldsymbol{r}_{OT}$ in target plane coordinates

$${}_T\boldsymbol{r}_{TP_2} = \boldsymbol{A}_{TI}\ {}_I\boldsymbol{r}_{TP_2} = \begin{bmatrix} x_t \\ y_t \\ 0 \end{bmatrix}$$

where $\boldsymbol{A}_{TI} = \boldsymbol{A}_{IT}{}^{-1} = \boldsymbol{A}_{IT}{}^{\mathsf{T}}$ is the orthogonal transformation matrix between the frames of reference $I$ and $T$.

In a simplified case, for an incoming beam hitting the mirror centered and $d$ assumed to be zero, we have $P_1 = M = C$ and one can explicitly calculate the mirror orientation from projected coordinates.

$$\boldsymbol{n}_m = \frac{\boldsymbol{n}_1 - \boldsymbol{n}_0}{\|\boldsymbol{n}_1 - \boldsymbol{n}_0\|}$$

$$\text{where } \boldsymbol{n}_1 = \frac{\boldsymbol{r}_{MP_2}}{\|\boldsymbol{r}_{MP_2}\|} \text{ and } \boldsymbol{r}_{MP_2} = \boldsymbol{r}_{MT} + \boldsymbol{r}_{TP_2}$$

Note that this simplification still captures the distortion introduced by the AOI of the incoming beam, which is by far the most important one to consider. For convenience, in the following, the origin $O$ is placed at the undeflected mirror center: $O = P_1 = M = C$.

The above equations can be used to transform between normalized mirror coordinates $(x,\ y)$ and target plane coordinates $(x_t,\ y_t)$:

$(x,\ y) \rightarrow \boldsymbol{n}_m \rightarrow (x_t,\ y_t)$:

1. By definition of the mirror coordinates, ${}_I\boldsymbol{n}_0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$ and ${}_I\boldsymbol{r}_{MP_2} = \begin{bmatrix} x\ D \tan 50° \\ y\ D \tan 50° \\ -D \end{bmatrix}$. So ${}_I\boldsymbol{n}_1$ is obtained by normalizing $\begin{bmatrix} x \\ y \\ -1/\tan 50° \end{bmatrix}$ and the mirror normal by calculating $\boldsymbol{n}_m = \frac{\boldsymbol{n}_1 - \boldsymbol{n}_0}{\|\boldsymbol{n}_1 - \boldsymbol{n}_0\|}$.
2. Redefine the actual incoming beam $\boldsymbol{n}_0$ and target plane ($\boldsymbol{A}_{TI}$ and $D$). Using the known mirror normal $\boldsymbol{n}_m$, calculate ${}_T\boldsymbol{r}_{TP_2}$ using the above equations and extract $(x_t, y_t)$, i.e. the first two components of ${}_T\boldsymbol{r}_{TP_2}$.

$(x_t,\ y_t) \rightarrow \boldsymbol{n}_m \rightarrow (x,\ y)$:

1. Specify the actual incoming beam $\boldsymbol{n}_0$ and target plane ($\boldsymbol{A}_{TI}$ and $D$). Calculate ${}_I\boldsymbol{r}_{MP_2} = \boldsymbol{A}_{TI}{}^{\top}\begin{bmatrix} x_t \\ y_t \\ -D \end{bmatrix}$ and normalize to get ${}_I\boldsymbol{n}_1$, then obtain the mirror normal from $\boldsymbol{n}_m = \frac{\boldsymbol{n}_1 - \boldsymbol{n}_0}{\|\boldsymbol{n}_1 - \boldsymbol{n}_0\|}$.
2. By definition of the mirror coordinates, redefine ${}_I\boldsymbol{n}_0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$ and $\boldsymbol{A}_{TI} = \mathbf{1}$. Using the mirror normal $\boldsymbol{n}_m$ from the previous step, calculate ${}_I\boldsymbol{n}_1 = {}_I\boldsymbol{n}_0 - 2 \cdot \left( {}_I\boldsymbol{n}_0 \cdot {}_I\boldsymbol{n}_m \right) \cdot {}_I\boldsymbol{n}_m$. Scale this vector with a constant factor to get the vector $\begin{bmatrix} x \\ y \\ -1/\tan 50° \end{bmatrix}$, from which $x$ and $y$ can be extracted.

As an example, consider the following arrangement:

The incoming beam in the $yz$-plane hits the mirror centered at a 45° incidence angle. The target plane is placed perpendicular to the reflected beam for an undeflected mirror, i.e. when $x = y = 0$.

$$ {}_I\boldsymbol{n}_0 = \frac{1}{\sqrt{2}}\begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix},\ {}_I\boldsymbol{r}_{OP_0} = \begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix},\ \boldsymbol{A}_{TI} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos 45° & -\sin 45° \\ 0 & \sin 45° & \cos 45° \end{bmatrix},\ D = 1700\ \text{mm},\ d = 1.3\ \text{mm} $$

Figure 9 shows the distortions introduced in this case. The 100° optical FOV is shown for comparison. This is the FOV the mirror would achieve for 0° AOI.

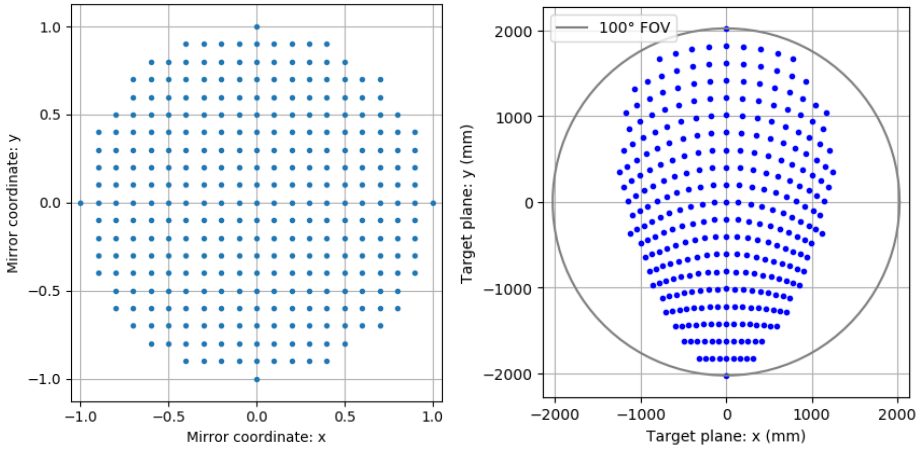



*Figure 9: Simulated FOV (right) for equally sampled mirror positions across the available angular range (left) for a 45° AOI. The 100° FOV resulting from 0° AOI is shown for reference.*

# 4. Image rotation

When the mirror is in the undeflected position as shown in Figure 10, the camera sees a vertically flipped, i.e. mirrored image. This is simple to correct and sometimes there is even a separate camera setting for flipping the image. However, when the mirror is scanned, objects in the resulting image will still appear rotated. The angle of rotation depends on the normal vector $\boldsymbol{n}$ of the mirror. The configuration below is identical to the configuration in the Optotune FOV Expansion Development kit, with the camera mounted vertically and the mirror mounted at 45°.

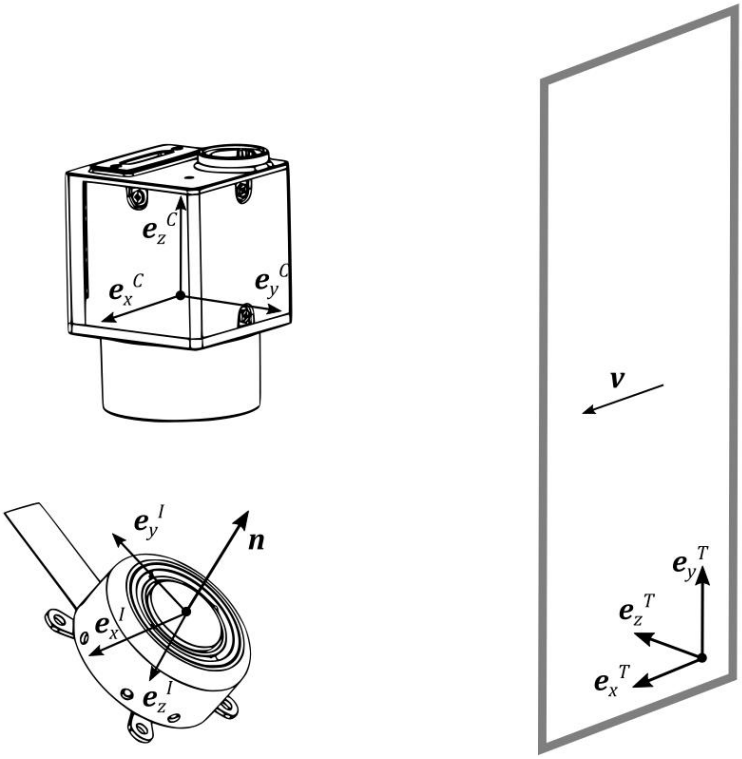



*Figure 10: Three different frames of reference - aligned with the object/target plane, the mirror housing, or the camera, respectively.*

Looking at how a certain vector $\boldsymbol{v}$ gets mapped onto the camera will allow calculating this angle. For simplicity, consider $\boldsymbol{v}$ aligned with the horizontal direction of the target frame $T$, which in turn is aligned with the mirror's x-direction, i.e. $\boldsymbol{v} = \boldsymbol{e}_x{}^T = \boldsymbol{e}_x{}^I$. When $\boldsymbol{v}$ is mirrored, it gets mapped to the vector $\boldsymbol{r}$ through the householder transformation

$$\boldsymbol{r} = (\mathbf{1} - 2\boldsymbol{n}\boldsymbol{n}^\top)\,\boldsymbol{v} = (\mathbf{1} - 2\boldsymbol{n}\boldsymbol{n}^\top)\,\boldsymbol{e}_x{}^I.$$

Expressing this equation in the frame $I$ yields

$${}_I\boldsymbol{r} = \begin{bmatrix} 1-2{n_x}^2 & * & * \\ -2n_xn_y & * & * \\ -2n_xn_z & * & * \end{bmatrix}\begin{bmatrix}1\\0\\0\end{bmatrix} = \begin{bmatrix} 1-2{n_x}^2 \\ -2n_xn_y \\ -2n_xn_z \end{bmatrix}$$

Where ${}_I\boldsymbol{n} = [n_x \quad n_y \quad n_z]^\top$ and all entries that do not contribute to the matrix-vector product are marked with *. By projecting this vector $\boldsymbol{r}$ onto the image sensor plane (Figure 11), the rotation angle can be calculated as follows

$$\varphi = \tan^{-1}\frac{r_y}{r_x}$$

$$r_x = {}_I\boldsymbol{r} \cdot {}_I\boldsymbol{e}_x{}^C = {}_I\boldsymbol{r} \cdot \boldsymbol{A}_{IC}\, {}_C\boldsymbol{e}_x{}^C$$

$$r_y = {}_I\boldsymbol{r} \cdot {}_I\boldsymbol{e}_y{}^C = {}_I\boldsymbol{r} \cdot \boldsymbol{A}_{IC}\, {}_C\boldsymbol{e}_y{}^C ,$$

where $\boldsymbol{A}_{IC}$ is found as a rotation around x by 135°:

$$\boldsymbol{A}_{IC} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos 135° & \sin 135° \\ 0 & -\sin 135° & \cos 135° \end{bmatrix}$$

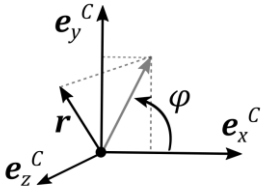



*Figure 11: Projection of $\boldsymbol{r}$ onto the image sensor plane. After being mirrored, the horizontal vector $\boldsymbol{v}$ appears rotated by $\varphi$.*

With this, the rotation for the setup shown in Figure 10 - corresponding to the angle $\varphi \in (-\pi/2\,,\, \pi/2)$ - becomes

$$\varphi = -\tan^{-1}\frac{\sqrt{2}\, n_x\, (n_y + n_z)}{1 - 2{n_x}^2}$$

Note that in order to rotate the image back, it is necessary to rotate the camera image by the angle $-\varphi$. When using the straightened image, it is necessary to either pad the corner areas or to crop the image. The cropping can for example be done as shown in Figure 12 by maximizing the area. Another choice would be to maximize the area while keeping the aspect ratio constant. It is obvious that for large rotation angles, a significant portion of the image gets discarded that way. However, for computer vision algorithms such as face recognition or barcode scanning, rotating the image back is typically not necessary.

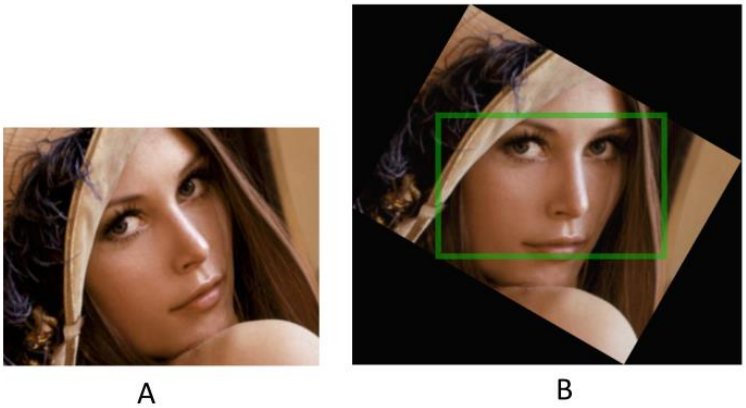



*Figure 12: Rotated image as captured by the camera (A) and the back-rotated image (B).*

# 5. Face detection

Figure 13 shows the program flow of the face detection demo program in Optotune Cockpit, available with the FOV Expansion Development kit. The demo algorithm detects a face in the wide-angle overview image, and provides a focused high-resolution image of that face using the tunable lens and the 2D mirror. The algorithm is based on a neural net by Linzai (link), ported to C# by Takuya Takeuchi (link).

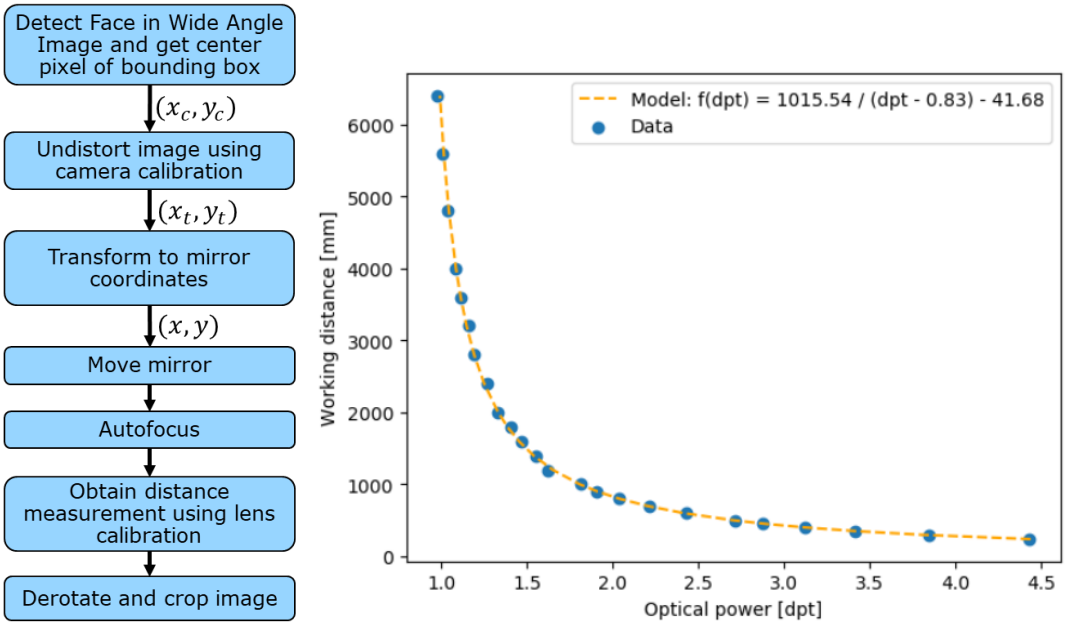



*Figure 13: Flowchart of the face detection demo program in Optotune Cockpit (left) and example of a distance-from-focus calibration (right).*

For more information on autofocus including distance measurements using Optotune's liquid lenses, please check this link. To find a good compromise between accuracy and speed, the lens settling time, lower and upper focal power limit, and fine and coarse step size can be adjusted.

The FOV Expansion Development kit is offered in two different versions, one with a 50 mm fixed focal length lens and one with a 75 mm lens. In Optotune Cockpit, two calibrations are available for the distance-from-focus measurement using the two different objectives. The calibration is valid only when the manual focus on the objectives is set to infinity. A custom calibration can be done by measuring the focal power $FP$ needed to focus to different working distances $WD$ and fitting the parameters $a, b, c$ to the data using the simple model:

$$\mathrm{WD} = \frac{a}{FP + b} + c$$

The complete list of parameters can be imported into Optotune Cockpit as a json-formatted text file. After closing Optotune Cockpit, the software will store an updated .json file to the user's Appdata folder (path=%appdata%). This file is automatically loaded at the next startup.

# 6. Image stitching

In addition to face recognition and area-of-interest selection, the FOV Expansion Development kit can be used to stitch multiple images into a single high-resolution, wide-angle image. An example is shown in Figure 14; another one can be found on our website.


*Figure 14: Example of stitched image with gigapixel resolution.*

## 6.1. Choice of mirror coordinates

Due to the projection distortion described in Section 3.2, a uniform spacing of mirror coordinates will not result in a uniform spacing of points in the scene. Instead, one needs to calculate the mirror coordinates that lead to a uniform spacing in the target plane. The sampling density should be chosen to yield a certain overlap between neighboring images, typically 20%-40%. A larger overlap increases the number of pictures to be captured and the computational effort to stitch the image. A smaller overlap makes the feature matching process more difficult.

An example set of mirror coordinates needed to create a uniformly sampled image plane is shown in Figure 15. As a landscape picture often is preferred, the setup can be rotated to get a wider horizontal- rather than vertical range.

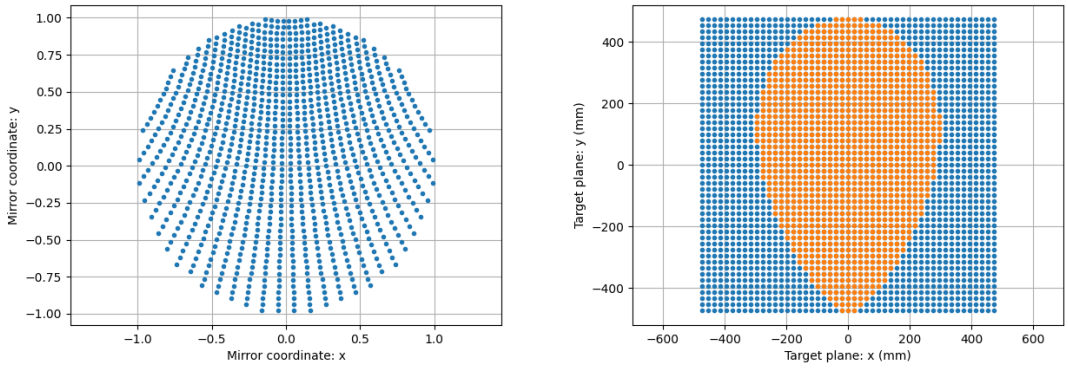



*Figure 15: A specific set of non-equally spaced mirror positions (left) results in equally spaced target plane coordinates (right, marked orange).*

## 6.2. Example Python script for image acquisition

Optotune provides a sample Python script to acquire images using the FOV Expansion Development kit, using the outlined coordinate transformations. The output of the script is

- A set of images uniformly acquired across FOV, numbered row by row
- A set of coordinates for each image (image position and image rotation) to be imported in the next step to facilitate the stitching process.

## 6.3. Stitching process

The stitching process is based on the free, yet powerful Hugin software.

1) Import all images. Save current Hugin project as project.pto.

2) (Optional) Get an initial picture distribution and image rotation according to mirror positions. This step allows to limit picture comparisons to neighboring images. From the command line use the command "pto_var --set-from-file filename --output=output.pto project.pto" with the file (filename="Hugin.csv") created with the python script taking the pictures. This should place all pictures in proximity to the final position after stitching. If the process results in a good result you can jump to point 7).

3) Create control points (CP): For the feature matching, create a new CP detector (cpfind.exe with argument "—linearmatch -o %o %s") which will detect matches between adjacent images (in the order of the import). This method will require an additional step to align the rows in the panorama, but the complexity of the CP detector will drastically decrease. The alternative CP detector *multirow* has higher computational complexity.

4) Create control points (connecting rows): Connect different rows by adding CP manually. It is advised to create CP for the pictures in the edges. There is no need to create CP along all the row as the *prealigned* method will do this work. Another method is to manually move rows in the preview window to align them correctly and run the CP detector with *prealigned* method 7).

5) Anchor the center image: Select the image and use the context menu to define the reference image.

6) Optimization step: this process will optimize the position of the pictures to minimize the mismatch in the control points. There is no need to optimize translation in the geometric parameters as long as the setup is stationary. Do not forget to previously anchor a picture close to the center.

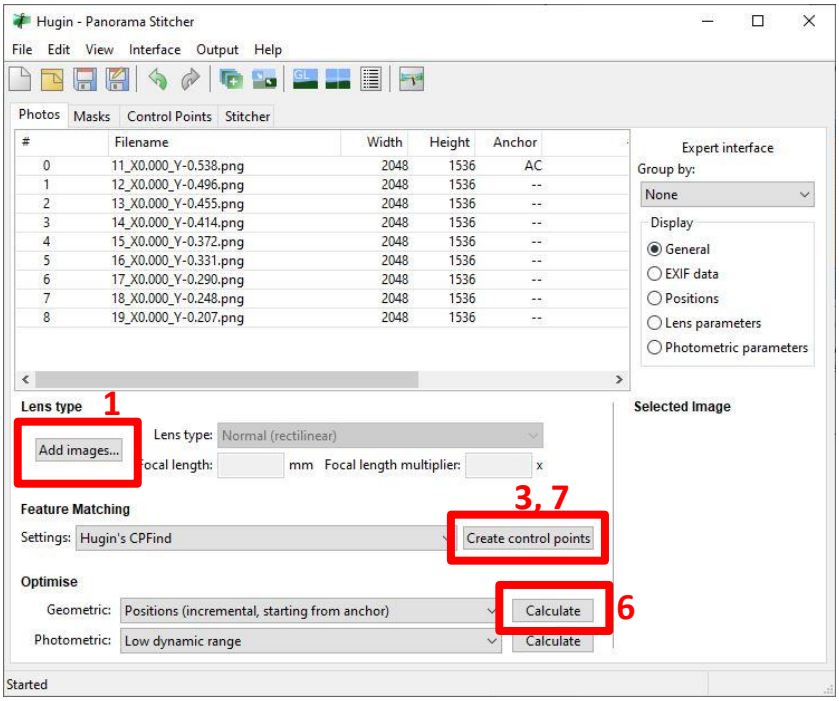
7) Autodetect control points of prealigned images (considering only overlapping images): Create a new CP detector with argument "--prealigned -o %o %s". This process will create CP between overlapping pictures which do not have CP in common. This step is to align images within rows, or also globally, if step 2) was successful.

8) (Optional) Create horizontal/vertical lines: This helps to avoid distortion in the stitched image. Add lines, e.g. edges of buildings, in different parts of the image for a better result, adding a lot of lines in same area of the FOV is not useful.

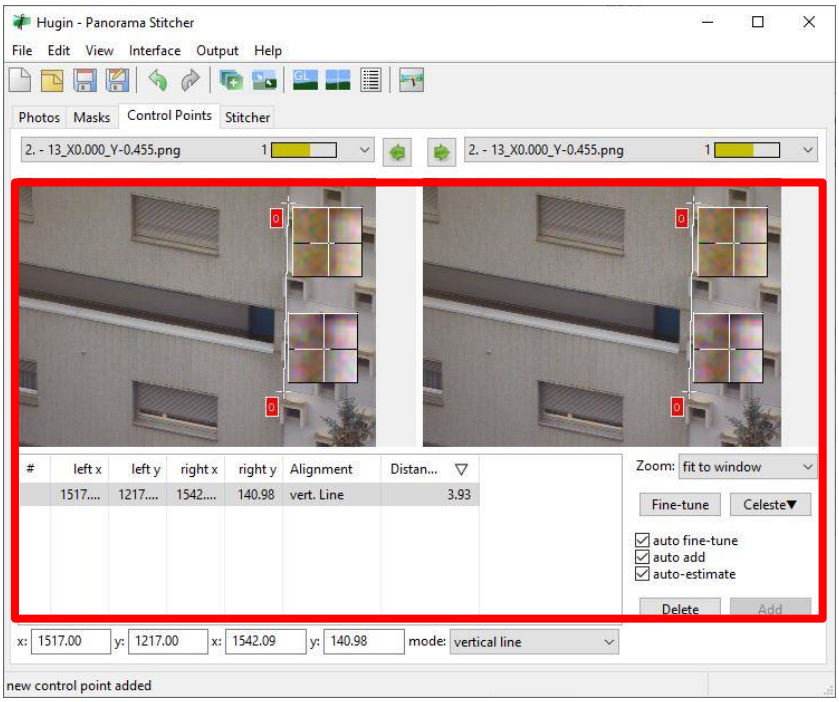
9) Verification + erasing bad CP: The bad CP can easily be identified in the CP table; usually they are the ones with large distances (sort accordingly). Verify those and delete the ones that are not correct (due to repetitive patterns in the pictures such as on building, a lot of bad CPs can appear). Be sure to redo the optimization step from time to time as you delete false control points.

10) Photometric optimization: Select multiple photometric parameters to optimize notably the vignetting[3] parameters. Do not forget to set as anchor for exposure a picture that is not over/under exposed. (Right click on Photos panel).

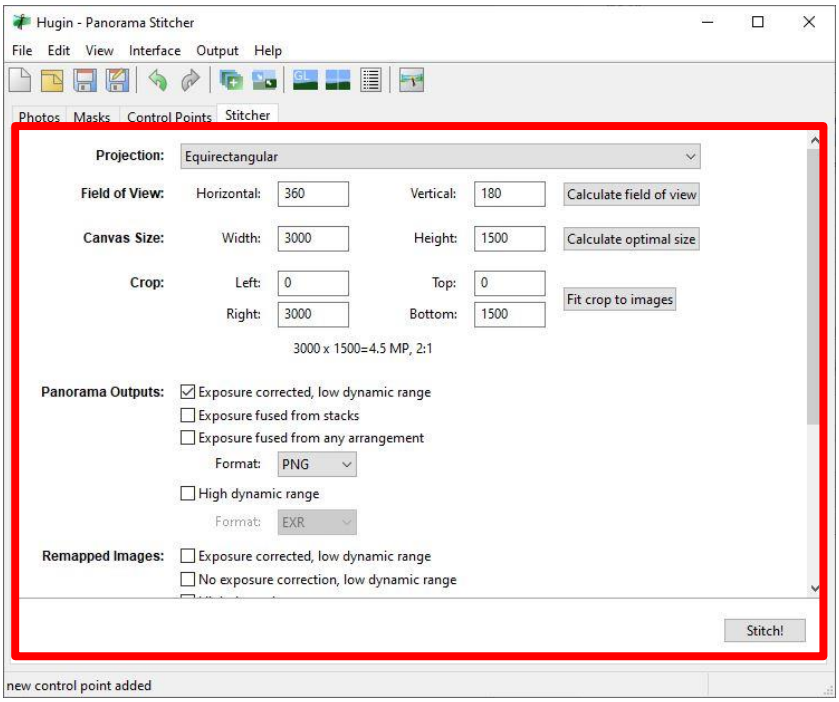
11) Stitching process: In *Stitcher* tab, calculate the FOV and optimal size before stitching. If you don't want to see the edges of the panorama and instead a rectangle, use the crop button. It is strongly advised to use *Multiblend* a significantly faster drop-in alternative to *Enblend* for the blending process (https://horman.net/multiblend/). "Due to Enblend's O($n^2$) complexity, compared to Multiblend's O(n) linear time complexity, this speed advantage increases to 300x for a gigapixel mosaic". The use of "--wideblend" argument is advised.




[3] Vignetting could be improved by having an entrance pupil of the objective tailored to the size of the mirror. Optotune may consider developing a custom lens for FOV expansion in the future.


Hugin - Panorama Stitcher
File Edit View Interface Output Help
Photos Masks Control Points Stitcher
2. - 13_X0.000_Y-0.455.png
2. - 13_X0.000_Y-0.455.png
# left x left y right x right y Alignment Distan...
1517.... 1217.... 1542.... 140.98 vert. Line 3.93
Zoom: fit to window
Fine-tune Celeste▼
auto fine-tune
auto add
auto-estimate
Delete Add
x: 1517.00 y: 1217.00 x: 1542.09 y: 140.98 mode: vertical line
new control point added

4, 8


Hugin - Panorama Stitcher
File Edit View Interface Output Help
Photos Masks Control Points Stitcher
Projection: Equirectangular
Field of View: Horizontal: 360 Vertical: 180 Calculate field of view
Canvas Size: Width: 3000 Height: 1500 Calculate optimal size
Crop: Left: 0 Top: 0 Right: 3000 Bottom: 1500 Fit crop to images
3000 x 1500=4.5 MP, 2:1
Panorama Outputs: Exposure corrected, low dynamic range
Exposure fused from stacks
Exposure fused from any arrangement
Format: PNG
High dynamic range
Format: EXR
Remapped Images: Exposure corrected, low dynamic range
No exposure correction, low dynamic range
Stitch!
new control point added

10