VIOSO ANYBLEND

VR & SIM

English version

Table Of Contents

1. Overview

VIOSO Anyblend VR & SIM is an extension to VIOSO Anyblend and enables you to blend multiple views created by image generators (IG's). It hereby does not matter whether there are multiple views rendered by one client machine, one view on multiple client machines each, or even a mix of both cases.

1.1 Extensions added

- Cluster calibration
- Mapping conversions: Create texture maps from a camera scan and a screen model.
- Model based warp: Create warp and blend from model and projector intrinsic parameters without a camera.
- Calculation of viewports according to screen and eye point: Create mappings for every IG channel to the screen.
- Templated viewport export: Export a settings file which can be included to IG configuration for automatic viewport settings
- 3D Mapping: Create sweet spot independent warp maps for head tracked CAVE™ solutions.
- Multi camera calibration: Use fast recalibration on screens where one camera can't see it at once.
- Master export: Collect all warp/blend maps on the master machine and copy them to the appropriate location on each client for VIOSO plugin enabled IG's.

1.2 Use Cases

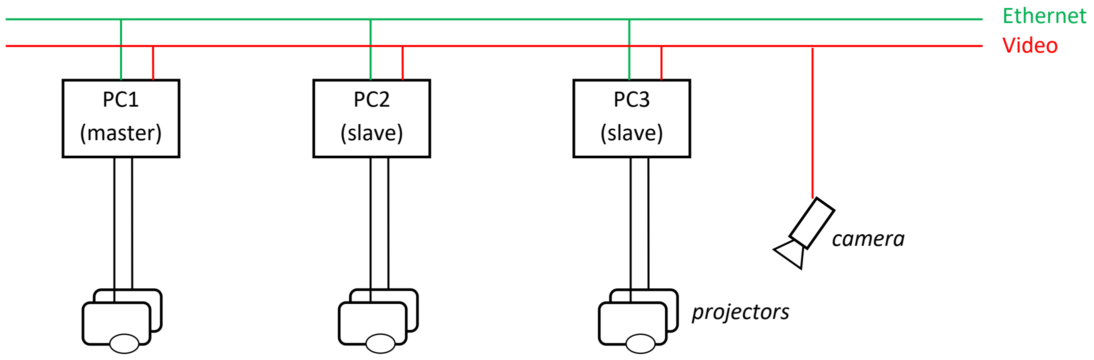
- Simulator setup on half dome, cylindrical panorama or irregular screens
- Projective Interior (virtual museum)
- Façade projections
- Tracked CAVE™ setups
- Projection mapping, augmented reality using textures projected on surfaces

1.3 Cluster Calibration

Use one or more cameras to calibrate a projection system residing in several computers. In most simulator or cave setups there are several renderers with single or multiple outputs.

To calibrate such a system, we use a given Ethernet connection to calibrate a cluster. Important is that:

- Every projector is "seen" by one camera completely
- The camera used to calibrate a connected projector must be connected to that PC, use network camera(s) if you need a camera fur multiple PCs



1.4 Camera to screen

Like every camera based projector calibration, the basis of all calculations is a camera scan which registers every projector pixel to the camera. With a few extra parameters this scan enables us to register the screen to the camera and respectively the projectors to the actual screen.

What benefits arise from that? Well, if you know every projector pixel's 3D coordinate in the real world, you can calculate viewports, warp and blend maps, and even eye point dependent transformations.

1.5 Projector pose to screen

This is the opposite case to a camera scan. What if we already know, where every projector pixel points to? Of course we can take advantage to that and place a 3D model in the ray fields of the projectors and calculate warping and blending from that. So quite all features known from Anyblend can be applied to a screen or every other surface without any camera!

1.6 Content Spaces

A content space is the image to be projected. There can be several content spaces in one setup. Content spaces are texture maps and view ports.

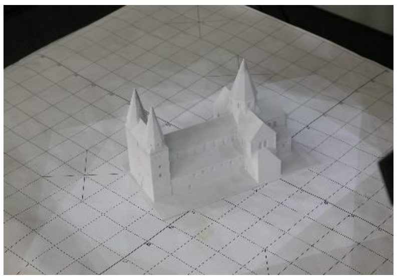
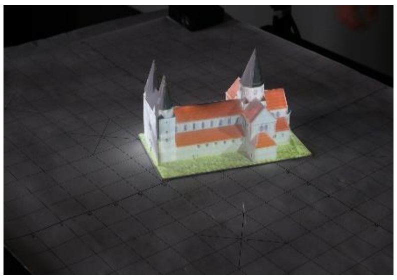
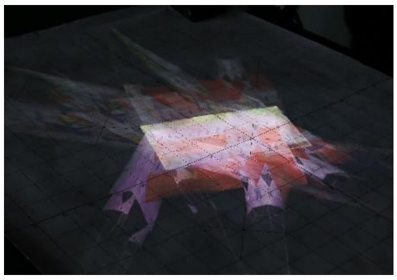
In a basic projection setup, the content space is the camera image plus the warping grid. Once the setup is done, all projectors are blended and there is a rectangular content image back projected through all projectors. This is the standard case for all flat surface projections, where the camera is positioned where the audience is or will be. You could also adapt the projection using our virtual canvas warp grid (VC) to adapt slightly curved screens or move the viewer's position by adding or removing keystone characteristics.

But what, if that is not enough? Content spaces enables you to transform a mapping to fit all kind of special purposes.

1.7 Projection mapping

How can I project something on a specified place on the screen? This technique is called "projection mapping" this is where an image or "texture" is warped to be projected on the screen surface. This method is similar to texturing 3D objects in virtual worlds.

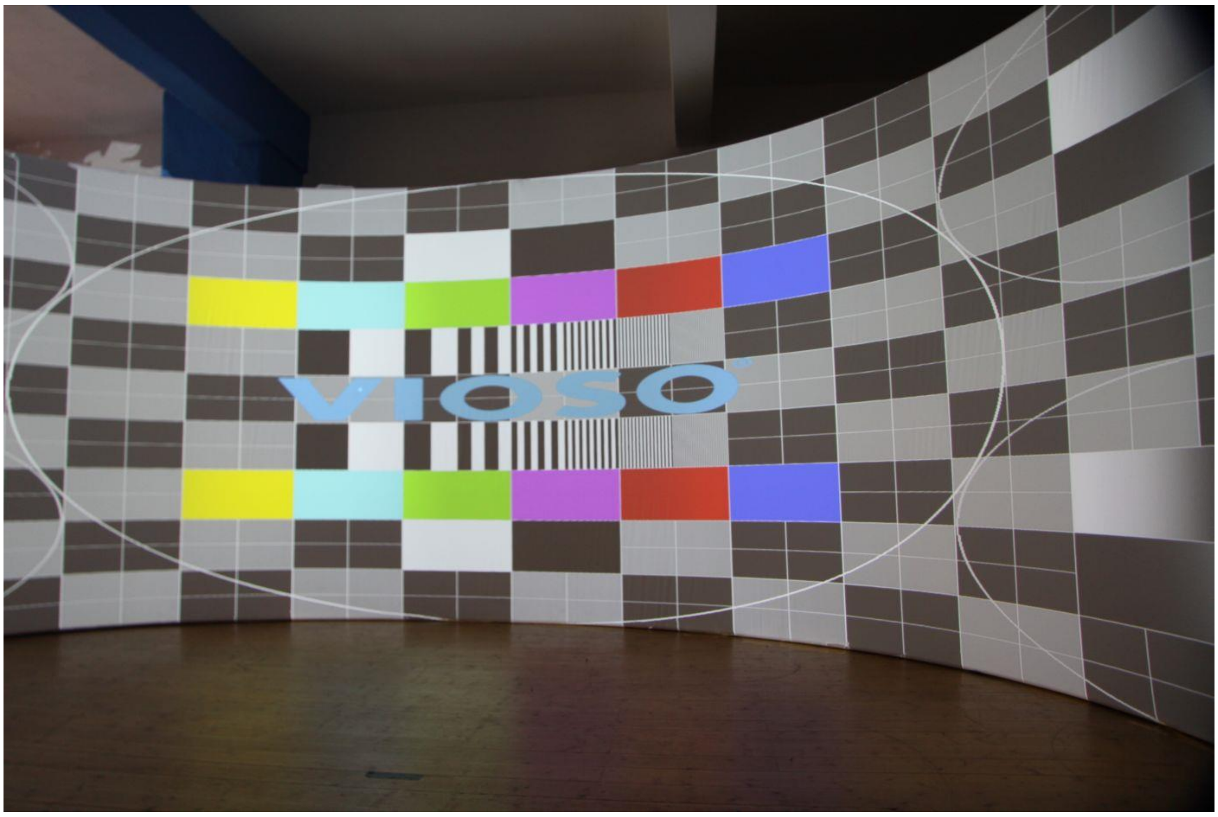
For that we need a model of the screen with texture coordinates for every surface. So once we've matched the model with the camera, respectively the projectors, we can read this texture coordinate from the 3D surface. Now we can sample in that image and project the image back to the real surface.

1.8 Virtual cameras

Virtual cameras are view ports to virtual reality environments. They are defined with an eye position, view direction and rotation, plus fields of view related to the virtual world.

Our projection screen can be seen as window to such a virtual world. First step is to match the screen with the simulated environment. This is done by matching a virtual screen model on the actual screen (augmented reality approach). In other words, if we can project a virtual screen on the real screen and they match, we know what projector pixel is affected by every view port's virtual camera image.



Then we can create a mapping which transforms a viewport's image to be displayed by a projector.

2. Performing Multiclient Autocalibration

2.1 Hardware setup

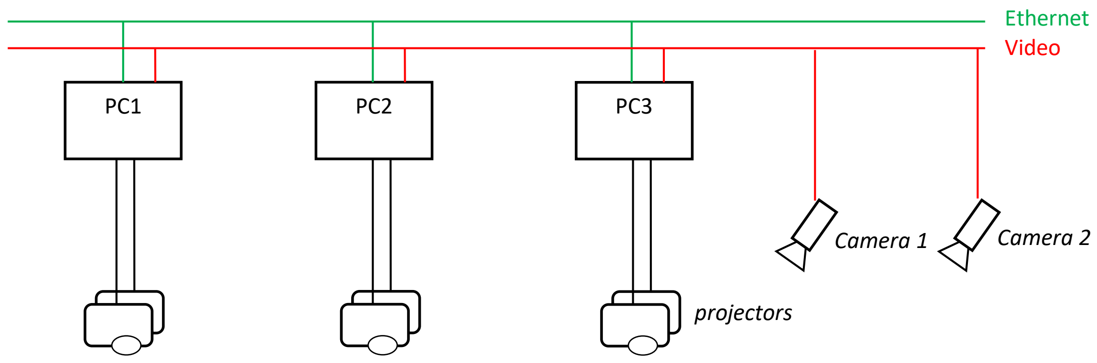
In general, the hardware setup consists of several PC's running the same version of VIOSO Calibrator.

In case of a camera based auto calibration we need of course one or more cameras. Best practice is to use as few cameras as possible, because the cameras have to be extra warped. This is due to the fact, that intrinsic parameters do not describe the lens completely.

There are no restrictions in brand or lens, but there are a couple of things to mind:

- All cameras must use DirectShow or Datapath Vision® interface
- Use same model and lens in case you want to use automatic brightness and color matching
- Every projector must be "seen" completely by one camera
- The resolution should not be less than a quarter of the corresponding projectors
- Each PC needs the camera signal of all cameras needed for its projector(s)
- Here are a few recommendations:
    - Logitech Webcam C930e (cannot be connected to multiple PC's!)
    - IDS µEye GigE using separate network adapter and switch
    - 3G/HD-SDI via splitter and Datapath Vision®

Example setup:



Please use gigabit or better network adapters as all data from the client has to be transferred to the master while calculating the blend. Please be sure the master has enough RAM to handle.

2.2 Preparations

Configure VIOSO Calibrator on every PC to use a IP address of the same subnet. I.e. 192.168.0.1 to 192.168.0.n enumerated continuously. Make sure to configure your network adapter's IPv4 or your DHCP server appropriately.

Make sure there your Windows Firewall is set to notify for new programs or add a rule for SPCalibrator.exe.

Start VIOSO Calibrator on each client. Check the settings by going to Options->Settings:

- Program->Common: select or mind the data path.
- Calibration->Multi client: select your network adapter and enable it.

In opposite to a single client calibration, you cannot split the displays while doing the calibration. So you have to split your Mosaic® or Eyefinity® groups according to the physical outputs. Do this on every client's SPCalibrator via Options->Screen split.

Before starting a calibration, you must figure the correct settings for all cameras. You can do this client by client, projector by projector. Make sure all cameras use the SAME settings and the brightest projector gets measured in brightness to 75-99%. If the darkest lies beneath 40% you should reconsider your setup, i.e. by moving it towards the screen. See manual_spcalibrator_en.pdf for details.

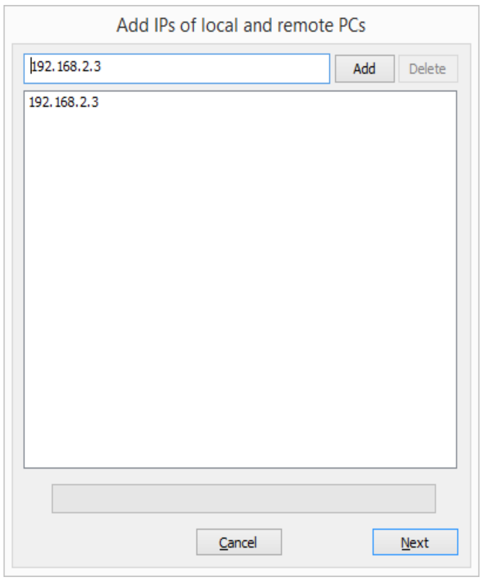
Keep in mind that all best practice considerations of a single client calibration apply here as well.

2.3. Starting a Multi client autocalibration with one camera

VIOSO Calibrator does not work on a classic master-slave architecture. So some things are a little different.

- You need to start VIOSO Calibrator on every client, you can even load it minimized (command line option /m).
- Data is kept on every client itself. Be sure to configure data directory properly as multi-client calibration files go there.
- If you load a settings file on some PC and press Preview with the Super Compound selected, the settings are loaded, if not already available.
- You have to load the settings file on all clients if you want to recalibrate! You can do that by adding a command line parameter "/l:D:\path\to\settings.sps". Keep the quotes, as white spaces will be interpreted as parameter separator.

Go to Calibration->Start Calibration or click the Calibrate button. Click on "multi client calibration".

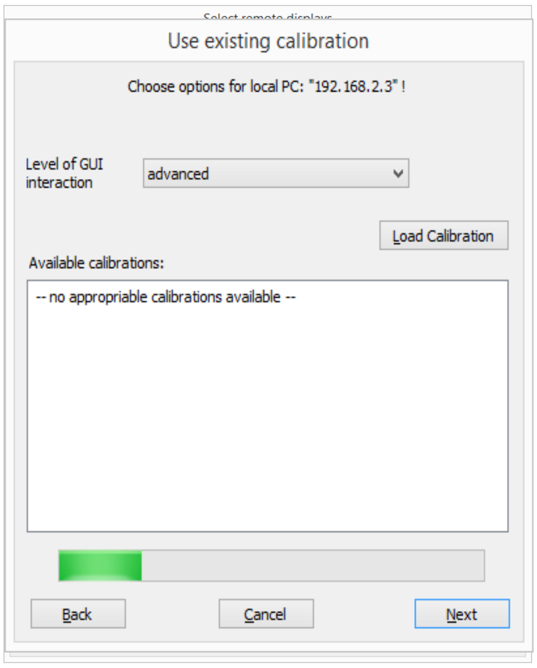


Add IPs of all participating PCs and the IP of the master by entering the IP-address or computer name to the text field and press Add. Once finished press Next.

You will get an error message if not all clients can be accessed. In case please double check the clients' network and firewall settings. Network communication will not work between clients with different versions!

Next you can select participating displays. The displays are fired up with a checker board to indicate their use.

Please double check split settings here. Every projector must have its own display in the list. You cannot calibrate parts or multiple projectors in this process. If the configuration is wrong, please cancel and go back to Options->Scree split on each affected client.



Choose options for local PC: "192.168.2.3" !

Level of GUI interaction: advanced

Available calibrations:

-- no appropriable calibrations available --

Now select preset options for the setup. Mind that the Setup method cannot be changed later. Choose a name for your setup and press Next.

In case a calibration already exists on the client it will be listed now. If you keep it selected, no measurements will be done.

This mode enables you to quickly combine calibrations separately done or loaded on the clients. This is an easy way of blending several clients measured by the same camera.

If you want to redo the measurement unselect all calibrations. Then press Next.

Now a single client calibration is started on the client. See VIOSO Calibrator manual.

The user interface is mirrored to the master, so you can do everything except camera settings from here.

3. Intrinsic and Poses

3.1 Camera intrinsic parameters

“Intrinsic parameters” describe the internal parameters of a camera. These values never change in the lifetime of a camera. They depend on the aperture and the lens. There is a set of intrinsic for every resolution, zoom level, focus and iris (shutter) setting. So these parameters must be constant on a setup.

Thus every camera must be tested on the setup to capture the right scene by adjusting focus and iris and then calibrated using exactly the same settings.

Camera calibration enables us to know which ray is captured by which camera pixel. The parameters are:

- Resolution width and height in pixels
- Focal length in width and height in pixels
- Principal point (image centre) in pixels
- Radial and tangential distortion

Intrinsic parameters are calculated from a set of test patterns presented to the camera. Just print out the provided pattern, save the captured images and calculate the intrinsic parameters using Calibration->Calculate camera intrinsic.

1. Setup the camera and find the best settings for resolution, focus, zoom and iris.
   These values must never be changed again or you have to recalibrate the camera!

2. Print out the chessboard patterns provided in %installpath%\TestPatterns. DO NOT RESIZE! DOUBLE CHECK PRINTED SQUARE SIZE! The process depends on real sizes. So all checkers have to be square shaped with known size in mm. It is recommended to fix them on a sturdy cardboard as they have to remain perfectly flat throughout the process.

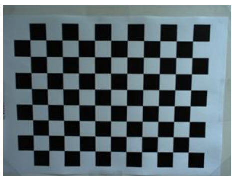
3. Open your favourite camera capture utility. You can use IDS Cockpit or VLC.

4. Create a directory with the same name as your camera.
   Example: “D:\Logitech Webcam C930e” or “D:\uEye capture device 1”

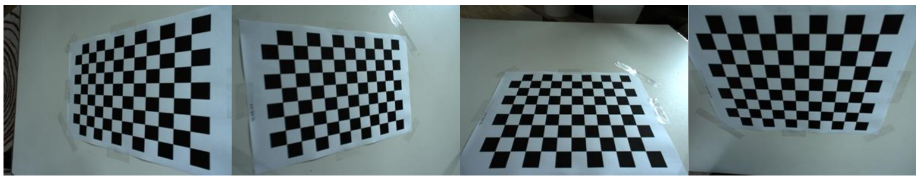
5. Create directories named for every test pattern: COLUMNSxROWS_SIZE
   Example: For the provided test image “pattern_A3_12x9_30.png”, which has 12 by 9 checkers sized 30mm, create a directory named “12x9_30”

6. Save a set of 9 pictures for every pattern this way (use png or 24-bit bitmap format):

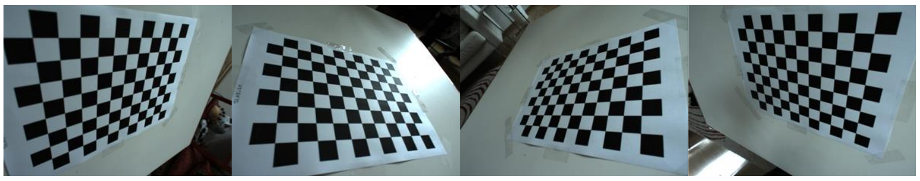
   a. Central, filling

   b. 4x edge tilt

   c. 4x corner

7. Use at least two pattern types!

8. Go to “Calibration->Intrinsics and poses”, select Camera intrinsic and press next. Browse for the directory named after your camera.

9. After about 10 seconds (1/2 sec per image), you’ll get a display of the intrinsic parameters. Check the back projection error, it should be less than 1 for a high quality camera.

10. You can inspect the calibration results in the data exchange TEMP directory. There are log files as well as back projection test images.

Once the intrinsic parameters are calculated, we can triangulate the camera’s pose from known 3D coordinates in the real world using “Calibration->Intrinsics and Poses, Camera pose”.

3.2 Calculate a camera view pose

By calculating a camera pose, you register the view rays of each camera pixel to the real world, respectively your designated screen. You need:

- at least 6 points in real (on or around your screen), you have 3D coordinates from. They should correspond to your model used, if any.
- to make those points visible to your camera. Use big and contrastful markers, or augment features by using a laser pointer/crosshair.
- calculated intrinsic parameters of that camera

Put markers on or around the screen. Measuring real coordinates works best, and be as accurate as possible! If you calculate 3D points on a cylinder/dome measure the cords to obtain the angles. A protractor often is not accurate enough, as you'll need +/-1 arc minutes, which is +/-0.0166 degree. That is about +/-1mm at 3 metres radius.

Edit the SPMarkerDef.ini found in your data directory. Enumerate from 1, continuous, you can define several sets by choosing start and count of the markers. You can have a 3D coordinate multiple times. Here is how a definition file looks like:

```
<?xml version="1.0"?>
<VIOSO>
  <File version="1.0.0" />
  <Marker id="1"  X="-200.0" Y="-200.0" Z="0.0" />
  <Marker id="2"  X="-200.0" Y=" 200.0" Z="0.0" />
  <Marker id="3"  X=" 200.0" Y=" 200.0" Z="0.0" />
  <Marker id="4"  X=" 200.0" Y="-200.0" Z="0.0" />
  <Marker id="5"  X="   0.0" Y="   0.0" Z="0.0" />
  <Marker id="6"  X="-400.0" Y=" 300.0" Z="0.0" />
  <Marker id="7"  X="-400.0" Y="-300.0" Z="0.0" />
  <Marker id="8"  X=" 400.0" Y=" 300.0" Z="0.0" />
  <Marker id="9"  X=" 400.0" Y="-300.0" Z="0.0" />
</VIOSO>
```

Go Calibration->Intrinsics and Poses->Manual camera pose. Select the camera and a control monitor, enter first and number of markers. Click Next. Select the camera mode you have calculated intrinsic parameters for. Click Next.

Move the markers using mouse (left moves the marker, wheel zooms, right pans) and arrow keys. You always move the point next to the mouse cursor, indicated by a circle drawn around the cross.

Once all points match the markers, get you UI in front (use F2 or right click), select the camera intrinsic and press Calculate. Double check the pose, maybe save an image (saved in data directory) and press Finish.

3.3 Calculate projector intrinsic parameters

Projectors have intrinsic parameters as well. We need them, if we want to use a projector's ray field to do advanced projection mapping, like using several trackable objects to project on.

To calculate them, you need a flat screen. Register your camera to the screen to have 3D coordinates for every camera pixel. (see Register a projector to the screen)

In Calibration->Intrinsics and poses/Projector intrinsics. Select your projector and camera (you need intrinsic parameters!) and click Next.

Now use a tripod and project images from different angles, similar to a camera intrinsic measurement. Finish by pressing Calculate and Finish.

4. Content space management

Here we talk about virtual cameras, like IG views, and models that match your screen or other projection surface.

4.1 Create and edit poses

- Create a view port in Extras->Custom content spaces, name it and use the projector's resolution as pixel space
- Define all viewports from ONE eye-point. This is to be done in the real world. Use for example the real position of the driver's eye. We call this the "sweet spot".
- Define a default orientation. This is the combination of "forward" direction and the natural "up". This also gives us the sideways direction. The IG operates relative to that.
- Find that pose in our model's coordinate space and create a pose.
- For every projector do:
    - Create a new pose.
    - Pinpoint the projection centre and set it as look-at point.
    - Keep "up" as close to the natural up as possible.
    - Measure the size horizontally and vertically of the projection area.
    - Measure the distance between projector and screen.
    - Size/distance gives the view size, alas the field of view. It must cover the whole projection area. Rather make it a little bigger to compensate possible projector movement.
    - Most IG's use angles to define a viewport. So switch to angles view and round the angles, for making it easier to enter into the IG's settings.

4.2 Export view ports

By clicking "IG views…" on the bottom of the Custom content spaces dialog, you can export poses for your IG. You have to define a straight forward direction in model coordinates, which indicates the main moving direction of your simulator. Usually something like a vector from eye to the centre of the screen.

Enter or browse a destination file.

Optionally you can use a template file, to generate an individual settings file for your IG.

```
#example template for exporting view parameter
#this is a comment
#macros are in ${}
#macros are case sensitive, no white spaces allowed
#first comes the settings section, nothing is written to file
${!rotationOrder,YXZ}#rotationOrder,param=[XYZ,XZY,YXZ,YZX,ZXY,ZYX]
${!angles,degree}#angles,param=[degree,radian]
${!header}#start of file header, following is written first to the file
//Export VIOSO Calibrator views ${date,%Y%m%d%H%M%S}.#date,format=see
strftime
${!channel}#!channel starts the channels section, this is written for every
channel
#${!channel}
[${name}]#name the channel name
yaw=${aY,0.5}#aY,format=[digits.precision] rotation around Y axis, if
format is omitted it's set to "0.5"
pitch=${aX,0.5}
```

```
roll=${aZ,0.5}
FoVLeft=${fovL,0.5}#fovL,format field of view to the left
FoVRight=${fovR,0.5}#right,format
FoVTop=${fovT,0.5}#top,format
FoVBottom=${fovB,0.5}#bottom,format
FoVHorizontal=${fovH}#fovH,format entire horizontal field of view
FoVVertical=${fovV}#fovV,format entire vertical field of view
FoVCenterH=${fovHC}#fovHC,format horizontal center angle, 0 if symmetric
FoVCenterV=${fovVC}#fovVC,format vertical center angle, 0 if symmetric
dirX=${dirX}#dirX,format x-coordinate of view vector
dirY=${dirY}#dirY,format y-coordinate of view vector
dirZ=${dirZ}#dirZ,format z-coordinate of view vector
upX=${upX}#upX,format x-coordinate of up vector
upY=${upY}#upY,format y-coordinate of up vector
upZ=${upZ}#upZ,format z-coordinate of up vector
sizeH=${sizeH}#sizeH,format horizontal view size
sizeV=${sizeV}#sizeV,format vertical view size
centerH=${centerH}#centerH,format vertical view center, 0.5 is symmetric, 0
is entitrly left, 1 entirely right
centerV=${centerV}#centerV,format vertical view center
offsetH=${offsetH}#horizontal view offset 0 is symmetric, 1 is entirely
left, -1 entirely right
offsetV=${offsetV}#vertical view offset 0 is symmetric, 1 is entirely left,
-1 entirely right
aspect=${aspect}#view aspect ratio, sizeH/sizeV
${!footer}#starts the footer, this is written at the end of the file, once
all channels are written
```

4.3 Definie a screen model

If we know the camera's position to the real screen, we can match every camera image coordinate to a 3D coordinate on a virtual screen model. Use your favourite CAD (i.e. SketchUp®) and create a model of your screen. Be careful with the metrics. If you want to use Content space transformations, put on a texture map (ONE texture for the whole model, unique uniform coordinates for the whole surface!). Export to a VIOSO readable format (.obj, .dae or .3ds are best)

Now go to Extras->Custom content spaces and create a new model, use camera resolution

- Set a prepared model file
- Check invert "v-coodinates"
- Adjust settings if necessary and save

5. Content space transformation

We register projectors to cameras. This is the usual VIOSO way of calibrating a screen. The camera image gives us a way to relate projector pixels and map them to a content. This is what we call a content space. Literally it is more like a content plane, as we use 2D coordinates for the transformation.

The main difference in Anyblend SIM is, we can introduce other content spaces, as we know exactly where camera or projector pixels are pointing on the actual screen. So we can cast 3D coordinates to the camera's pixels. This enables us to do Observer views by reprojecting through a virtual camera or view pose.

If a model contains texture coordinates, we can build a new 2D content space from this. This gives us a way of warping/blending multiple registered cameras or do projection mapping.

A content space transformation describes the act of converting a camera scan to a custom content space, aka a virtual screen model, maybe with texture coordinates. We can go for two different approaches: Model and Markers or Smart test images.

5.1 Matching a model with the screen

We need a scan from a pose registered camera and a model. As we have registered every projector pixel with a camera pixel, and the camera to the model, we can transform the scan based on the model.

If texture coordinates were provided, the content space gets converted as well. After the scan we work in the content space of the camera, alas we back-project a camera image on the screen, maybe warped using the warping grid. We now replace the camera coordinates with the texture coordinates from the model. Now we can back-project a texture on the screen and do projection mapping. So if you would play your texture atlas it fits to your screen, even if that screen is some object or façade.

The next chapters describe the steps using a pose registered camera.

5.1.1 Camera content space translation

After that, you can edit the camera content space. While warping, the view vector of the camera is transformed, so you actually adjust the camera pose and intrinsic parameters. You can use this feature to micro align multiple cameras or do minimal adjustments of the camera pose and intrinsic.

5.1.2 Add VC to display geometry

This step is only necessary, if you have done changes in content space translation. After that step, the VC is resetted to "full screen" and all changes are applied to the mapping.

5.1.3 Content space conversion

Use a model inside the camera viewport to generate a 2D contentspace from the model's texture coordinates and generate a 3D map. The new content space is used from now on to distribute the content. Now we have assigned a 3D coordinate of the screen to every projector pixel and use this to calculate observer views or use dynamic eye-point warping.

5.2 Smart test images

What if we do not have camera intrinsic parameters or screen models. For flat, cylindrical and spherical setups we have developed a far easier way to register a projector to a screen. We need a test image that, once aligned to the screen, indicates the 3D coordinate implicitly!

Once the test image is positioned on the screen using the warping grid, and we know how to calculate these coordinates from the test image, we can transform them to the projectors.

So far there are 3 such conversions implemented:

- Camera to flat screen
- Fisheye to spherical screen
- Fisheye to cylindrical panorama

But there is no limit in doing things that way. If necessary, we can implement such kind of conversions fast if there is a simple formula or a map to generate 3D coordinates from the test image.

Even the content space is implicitly defined by the conversion method. The spherical conversion leads to a dome master content space commonly used by half domes i.e. planetariums, the cylindrical conversion to a panorama content space.

5.3 Calculate auto pose

It is hard to find the optimal pose for an IG channel. But once a scan has been converted and bears the 3D coordinate of every projector pixel, you can simply go to Calibration->Conversion tasks and select “Auto pose”. Select the channel you want to calculate an auto pose from. If you select multiple displays or (Super) Compound(s), a pose is calculated for every participating display.

Now just enter your observer’s eye-point (“sweet spot”) and press perform.

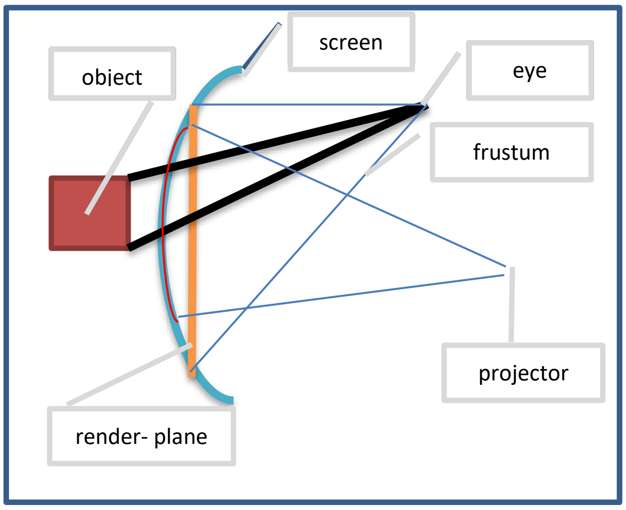
You’ll find the poses in Extras->Custom content spaces. They are named after the display they’re derived from.

If a pose with that name already exists, it will be overwritten! So if you do not want to stick with the calculated poses, create some with different names.

5.4 Observer conversion

In simulation environment, we have to create an immersive experience for the user. Hence the “window”, the user sees the simulated world through, needs to be as big as possible. So we use power walls, curved screens or even domes to cover a person’s sight to the max.

Whenever we need more to transport than one flat image, we have to go for multiple channels and poses. Clearly, we can render the virtual world from some virtual eye-point in every orientation and field of view. This is what we call a pose or a viewport. But how to match our projection channel with that?



The projected image covers a part of the screen, and we have to fill in the content. We know, to which 3D coordinate of the screen every projector pixel points to, by registering the projector to the screen. So rendering the 3D coordinates of the projector by a viewport leads to a coordinate on the render plane.

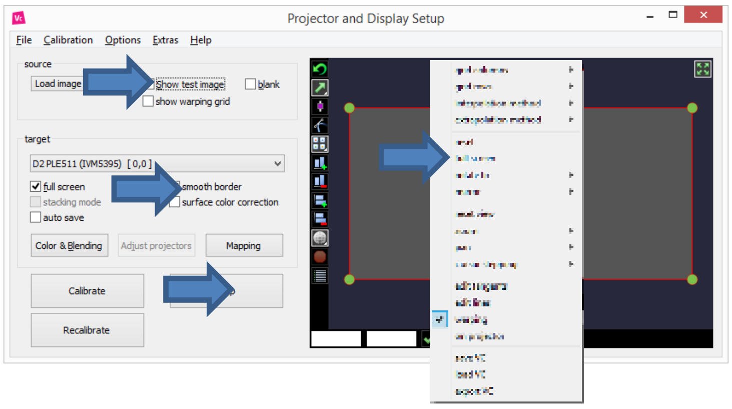
Using the same relative pose for the IG to generate the image, we have the content to be filled in.

An observer conversion simply generates a lookup map to fill the projector pixels with the content from a rendered image, based on the same viewport settings. Of course we have to use a fixed eye-point in the real world towards the screen, as it wouldn't move. The eye point and orientation in the virtual world may change however. It only has to be the same for all channels (world transformation).

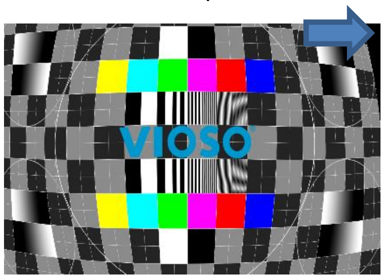
The hard thing about these conversions is to find the optimal pose to render from. See "Content space management" for further considerations.

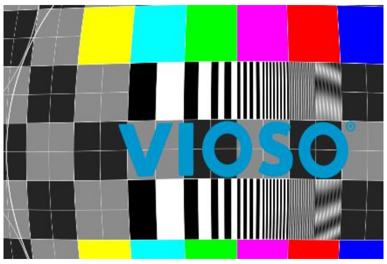
To test a pose, you have to save a 3D converted to a new name, remember we need the original scan as base for an auto recalibration!

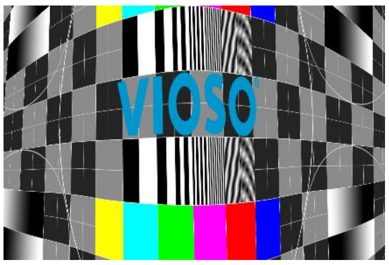
1. Do observer conversion with matching pose
2. Start preview on that channel
3. Fullscreen the VC
4. Check "show test image"
5. Uncheck smooth border



6. If the projector stays black, you probably have the wrong pose.
7. Look for problems like this:


Uncovered corner


too much loss


too much distortion

You can easily fix the first two problems by adjusting the viewport. The third problem can’t be solved in software. Consider changing the projector setup (i.e. use portrait mode) to get less distortion. Repeat the steps until the screen is optimally covered.

6. Model based functions

There is another way of registering projectors to the screen. We call it “Model based calibration”. The idea behind that kind of calibration, is to register a screen to a projector using projector intrinsic parameters and position. In other words, if we know each ray from each projector pixel, we can virtually place our screen into the ray field and trace the surface of the virtual model. This leads to a known 3D coordinate of every projector pixel. We can even blend the projectors on the virtual surface and use this information to generate viewports or projection mappings.

<insert vitrine manual>

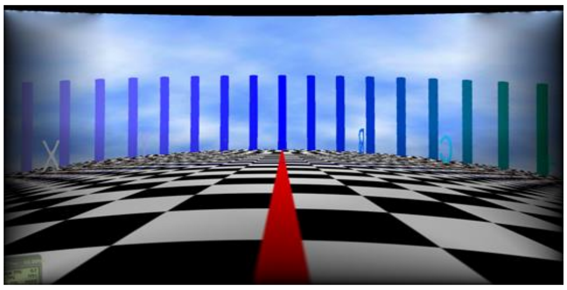
7. Use cases

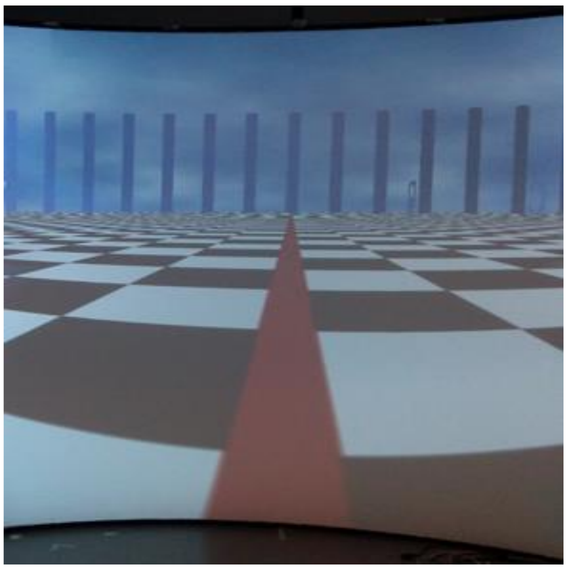
- Simulator setup on half dome, cylindrical panorama or irregular screens
- Projective Interior (virtual museum)
- Facade projections

7.1 Tracked CAVE™ setups

You can use 3D-Data to enable live eye-point-correction. For this purpose you need a post process shader for your application or connect Anyblend to a tracker.

The basic idea behind eye-point-correction is to get rid of the effects a curved projection surface has. Consider a simple cylindrical screen. It works like a curved mirror we look onto the virtual scene through. We can warp the image to appear correct from one eye-point. This is called the "sweet spot". If you look at the image which gets projected, you see why I say it works like a mirror, as it looks like a scene reflected by a curved mirror.

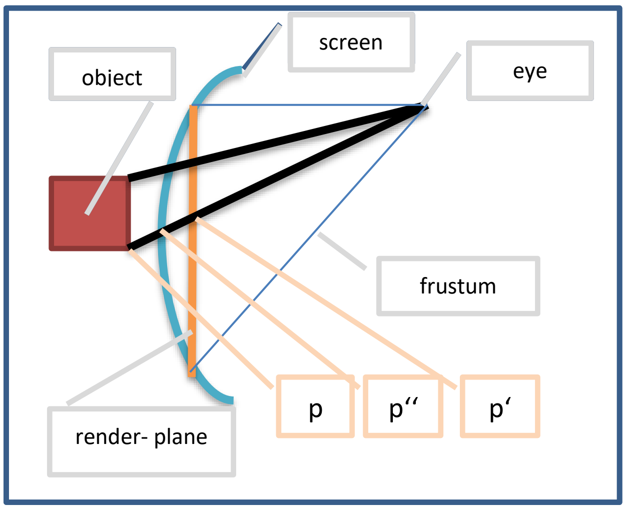


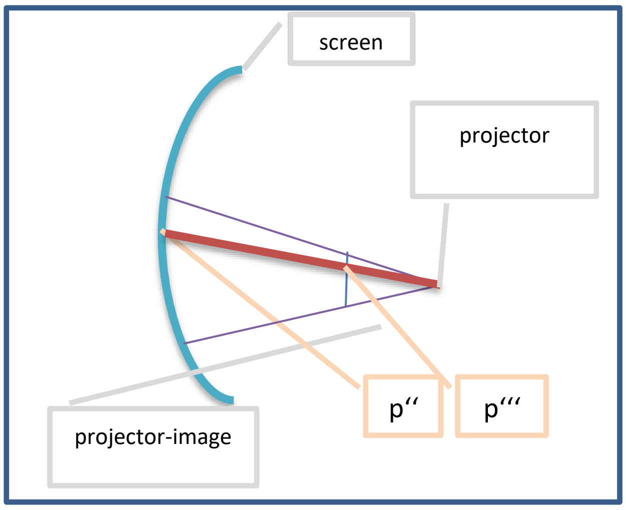


Looking from the "sweet spot" onto the scene on the screen you see that all lines, which should be straight, are straight. As the IG then renders a scene from the same fixed eye-point, which corresponds to the real eye-point, we can warp our image to look good. If you move away from the "sweet spot" you notice lines get bend and skewed. This is because the displayed perspective does not match your actual eye position.

Of course we simply can tell the IG to render from a different position. This works great as long as the projection surface is flat. A flat mirror would not distort the image it reflects. But this changes dramatically once we go curved.

For every dynamic eye-point setup there is a fixed render plane (rectangle in 3D) which defines the view **frustum** to the dynamic eye-point. The rendered image can be scaled to that render plane, once the IG updates it's projection-view-matrix M.





This gives us a correspondence of p and p'. If there is no match between the screen and the render plane (p'' != p') we have to answer another question: Where would I expect to see p on the screen (p'')?

If we take a look at the projection, we can see a correspondence between p'' and p'''. This is what we get from the Calibrator tool as a look-up-map.

Now we can solve all problems:

$p'' = L(p''')$ (1)
$p' = M^T * p$ (2)
$p' = M^T * p''$ (3)

(1)->(3)

$p' = M^T * L(p''')$ (4)

The equation (2) is solved by the renderer of the IG. Generating the image, it fills the render plane, so we know each content pixel.

Equation 4 can be solved by a simple pixel shader:

- p''' is the relative texel coordinate, given by the texture coordinates of the rendered quad.
- L() is a lookup texture giving us 3D of the screen
- p' is then resulting texture coordinate inside the content texture

There are a few things to mind:

- The render plane must be big enough to get the screen covered. That means, from every possible dynamic eye-point the whole projector image must be visible through that "window". So the best practice is to intersect it with the real screen and keep the projection "inside".

- Minimize the maximum distance between the render-plane and the actual screen. In other words: The bigger the gap, the heavier the distortion. Best practice is to place the projectors to cover only a small arc, i.e. using portrait mode or simply more projectors respectively render planes.

- We have to define the fixed render-plane. A convenient way to do this is to define a fixed frustum and a distance to define a view plane. The frustum setting can be derived by the Calibrator calculated from an eye-point and a scanned mapping.

8. Master Export

If you check “Export to Master” in File->Export menu and select a previously presented Super Compound, all data from all clients is collected and saved to the master (the client you are working on). Depending on the settings you will get one big .vwf file or one for every display.

8.1 Automatic conversion and delivery

If you check “use extended file definition”, all conversions and transfer tasks inside the file will be performed on the current scan data. That means you don’t have to perform conversions and export on the clients manually, you can just run it from the master after a recalibration. You even can overwrite Anyblend’s _Startup.vwf.

Here is an example, how such a file may look like:

```
<?xml version="1.0"?>
<VIOSO>
  <File version="1.0.0" />

  <CalibCommerceTask Typ="export" Format="vwf" >
    <CommonParam bSilent="1" bNoVC="0" bNoGeomCorr="0" bNoBlending="0"
bNoMask="0" bSeparatedSplitDisp="0"  bAllToMaster="1"
bBlankUnusedSplitDispParts="1" bExactFileName="1" bUse3D="0" />

    <Task Typ="convert" Format="pseudo 3D sphere" >
      <SpecialParam vParam1_X="1000.0"/>
    </Task>

    <TransferTask Format="vwf "  Core="192.168.100.5" >
      <Destination Core="192.168.100.5" FileName="D:\Export\EXPORT_Slave.vwf" />
    </TransferTask>
  </CalibCommerceTask>

</VIOSO>
```

9. Setup of a multi camera calibration

9.1 Creating screen model

To create a model of the screen, you need either a CAD of the screen or you have to build one.

9.1.1 Create a model from a CAD

If you already have a model, you need to strip everything obstructing the actual screen. Keep it as simple as possible, try to use a sane minimum of faces. Use real world units. We recommend mm.

The faces have to point towards the cameras, as we use face culling to reduce rendering and obstruction.

For a multi camera approach, it is necessary to distribute texture coordinates to the model. So distribute a textured material to the surface of the screen. Create a material from one big texture which has the dimension of the screen: In case of a cylindrical screen use its height and arc length of the screen. Once the material is applied, it must be distributed uniformly, like shown in next section.

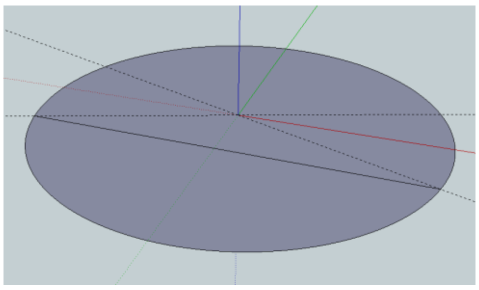
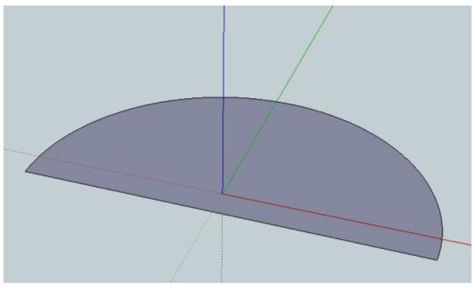
Export the result to either 3ds, dae or obj.

9.1.2 Create a new model using SketchUp®

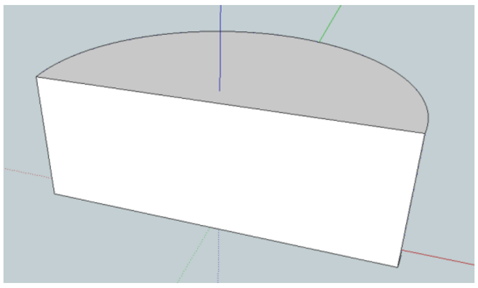
Here we show how to create a model of a cylindrical screen using SketchUp®.

Get radius, height and view angle from your screen. Use mm and degree.

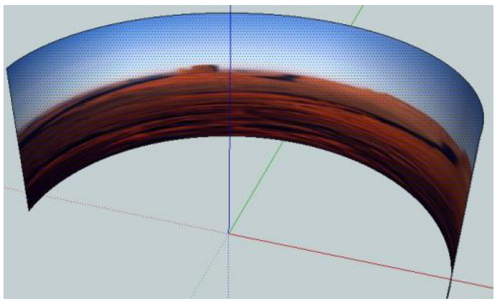
- Draw a circle in the x-y-plane. Key in the segments first, like 360. Then click to the center and key in the radius.
- Use protractor tool to find the view angles.
- Draw a line connecting the intersection points
- Remove ledgers and the unused arc

- Extrude to screen height



- Remove top and bottom line
- Reverse faces
- Create a material using a texture. Be sure to make its size like your screen: w = radius * viewangle * PI / 180



- Every part of the screen needs to be covered by the texture, no tiles. Every part of the screen needs its own texture coordinate.
- Make a component
- Rotate the component around x-axis to make x (red) point to the right, y (green) pointing up and z (blue) going back.
- Export to dae

9.1.3 Create markers

To align the cameras to the screen, we need at least 6 known features per camera. A feature is some unique place on (or off) the screen, we have the 3D coordinates of.

In case of a cylindrical screen, we just create them using angle functions. Add all points to a %programData\VIOSO\SPCalibrator\SPMarkerDef.ini (or your set data directory). On our 200° screen we use 22° step. We go up, then right. Enumerate continuously. In our example we use 5 cameras for 9 projectors, so the front camera needs points, 11° rotated.

Of course you have to find the markers on the screen itself, so put some next to the screen on the bottom and use a laser cross to find the upper edge.

```
<?xml version="1.0"?>
<VIOSO>
  <File version="1.0.0" />
  <Marker id=" 1"  X="-3792.723228" Y="0" Z="600.7083458" />
  <Marker id=" 2"  X="-3792.723228" Y="2885" Z="600.7083458" />
  <Marker id=" 3"  X="-3741.581049" Y="0" Z="-863.8120487" />
  <Marker id=" 4"  X="-3741.581049" Y="2885" Z="-863.8120487" />
  <Marker id=" 5"  X="-3145.54385" Y="0" Z="-2202.533516" />
  <Marker id=" 6"  X="-3145.54385" Y="2885" Z="-2202.533516" />
  <Marker id=" 7"  X="-2091.413894" Y="0" Z="-3220.494981" />
  <Marker id=" 8"  X="-2091.413894" Y="2885" Z="-3220.494981" />
  <Marker id=" 9"  X="-1438.489319" Y="0" Z="-3560.386002" />
  <Marker id="10"  X="-1438.489319" Y="2885" Z="-3560.386002" />
  <Marker id="11"  X="-732.7065422" Y="0" Z="-3769.448384" />
  <Marker id="12"  X="-732.7065422" Y="2885" Z="-3769.448384" />
  <Marker id="13"  X="0" Y="0" Z="-3840" />
  <Marker id="14"  X="0" Y="2885" Z="-3840" />
  <Marker id="15"  X="732.7065422" Y="0" Z="-3769.448384" />
  <Marker id="16"  X="732.7065422" Y="2885" Z="-3769.448384" />
  <Marker id="17"  X="1438.489319" Y="0" Z="-3560.386002" />
  <Marker id="18"  X="1438.489319" Y="2885" Z="-3560.386002" />
  <Marker id="19"  X="2091.413894" Y="0" Z="-3220.494981" />
  <Marker id="20"  X="2091.413894" Y="2885" Z="-3220.494981" />
  <Marker id="21"  X="3145.54385" Y="0" Z="-2202.533516" />
  <Marker id="22"  X="3145.54385" Y="2885" Z="-2202.533516" />
  <Marker id="23"  X="3741.581049" Y="0" Z="-863.8120487" />
  <Marker id="24"  X="3741.581049" Y="2885" Z="-863.8120487" />
  <Marker id="25"  X="3792.723228" Y="0" Z="600.7083458" />
  <Marker id="26"  X="3792.723228" Y="2885" Z="600.7083458" />
</VIOSO>
```

- Write down the sections for every camera:
    - Cam1 sees 6 markers, starting by 1
    - Cam2 sees 8 markers, starting by 5
    - Cam3 sees 10 markers, starting by 9
    - Cam4 sees 8 markers, starting by 15
    - Cam5 sees 6 markers starting by 21

9.1.4 Place your cameras

In the data directory you need SPDevices.ini, SPMarkerDef.ini and SPContenSpaces.ini. In the SPDevices you'll find the camera intrinsic. This in conjunction with the markers we can find the pose of the camera.

Start SPCalibrator and go to Calibration->Intrinsics and poses and select "Manual camera pose" from the dropdown. Select the camera and your observer screen (which should not be a projector).

Now put in the start marker and the appropriate number used for that camera. Press next.

Now adjust the camera to see the markers, respectively the laser line. Use "Format" button and adjust Exposure. Press Next.

Now move the points roughly to its positions, then position fine. Use mouse wheel to zoom and right button to pan. The point next to the mouse cursor is selected (encircled) and can be moved by arrow keys as well.

Once finished, press F2 to get the UI in front. Select the correct intrinsic parameter set from the dropdown and press “Calculate”. Save a screenshot for later use.

Press “Next”, if you are satisfied (perform sanity check on the values!).

Repeat for all cameras.

9.1.5 Create a model content space

Go to Extras->Custom content spaces. Click new and name it “screen”, it is a “model” and use camera resolution (2560x1920) as pixel space.

Select the model file (screen_mgp.obj) which should be copied to the data directory. Currently 0,0,0 is in the middle of the cylinder at bottom (not floor) level. Move the model to the eye point by moving it down (-y). Check “invert Tv”. Press “Apply” and “Save”.

Now you got all cameras aligned to the screen.

Close SPCalibrator on master.

9.1.6 Copy all settings to all clients

Copy SPDevices.ini, SPContentSpaces.ini and the screen model file from the data directory over to all clients’ data directories. Make sure there are no .ini’s except SPeASY.ini in the program directory.

Maybe use a startup script that copies the above and SPCalibrator64.exe, SPeASY64.dll and SPeASY.ini from master.

9.1.7 Run projector scans

Start VIOSO Calibrator on all clients. Make sure that network settings and firewall are configured.

Start a Multi-Client calibration on master. Add all IP’s to the listbox and click Next.

You have to make sure all cameras are same brand/make and are set physically and parametrical identical.

Select the correct screen mode (flat or curved screen) and give it a convenient name. This file will be saved on all clients in its data directory, as all clients hold their data themselves.

Check “no blending calculation” and click Next.

Now you are asked to perform a calibration on each client. Do not load or reuse an existing calibration. (This is why we restarted all Calibrator instances on all clients)

Use same camera settings on all scans. (write them down) Perform white balance once and write down the values. Then disable white balancing and auto exposure before doing each scan.

Unfortunately you have to remote to the clients and set up the camera (this is going to be saved, so once you load a successful scan, the system uses these parameters).

You have to use the correct camera intrinsic parameters, this is where the camera position comes from.

Draw a mask slightly inside the screen where the projection is cut off by the canvas or obstruction.

If any of the scans fail, just go on and restart the process. Now you can skip good scans by using them (keep the existing calibration selected) or even load, if you had to restart VIOSO Calibrator.

9.1.8 Conversions

Now we have a correspondence between camera(s) and screen. We need to calculate the 3D data, every projector pixel points to. This is done going the following steps.

a) **Camera content space translation**

   On every client go Calibration->Conversion tasks, select the Compound, Camera content space translation and your screen model. Then press perform.

   After that, please start preview of all Compounds on all clients.

   Now we look at a back projected rendering of the model, which is the virtual screen.

   You will notice that where camera sections overlap, the image is not matching perfectly. This comes from measurement and intrinsic errors summing up on the screen. This can be corrected using our warping tool. In this state we transform the camera. Just like adapting the intrinsic parameter.

b) **Add VC to display geometry**

   Once you are satisfied with the overall geometry you go Calibration->Conversion tasks and select “Add VC to display geometry”. Then the warping will be connected to the camera. So from now on, no warping must be applied on recalibration.

c) **Custom content space**

   The last step is to convert the projectors to their new content space. Now the 3D points and the texture coordinates are calculated to every pixel.

9.1.9 Redo blending

Now you have to do a recalibration. Click on the “Calibrate” button and select multi client calibration. Add only all IP addresses for one eye if you have stereoscopic mode, as they are to be blended together. Repeat for the other eye, in case. Click “Next” and select all projectors. Keep all settings except check “auto content space convert” to be able to rescan without doing conversions again. Then click “Next”.

Now confirm every available calibration and wait (be patient!) for the blending being calculated.

9.1.10 Export

The last step is to export the mappings. Go Files->Export warping and blending, and select the Super Compound. Check export to master.

a) Camera content space translation

To get the exact pose for every IG, you’ll need to calculate it from the collected data. Go Calibration->Conversion tasks and select Display auto pose. This leads to a new pose you’ll find in Extras->Custom content spaces.

b) Export pose

You can export it by clicking in IG views… Just key in the main view as follows:

Eye: 0, 0, 0; Dir: 0, 0, 1, and up: 0, 1, 0. Enter the location of the individual .ini and use the supplied template to generate the settings file for the plugin.

10. Examples and cases

10.1 Export for Barco WB2560 (MIPS)

Make sure that the warpboxes (WBs) are connected to the same network as the computer with Vioso software. It is recomended that they´d be in the same subnet and domain.

Usually for the monitoring and control of WBs the ProNet software is used which can be downloaded from official Barco web-site. This software can be also used for EDID selection for the WBs, rebooting, and status change (Bypass/Online).

Start Vioso Calibrator and carry out a calibration. Check the result, adjust the mapping and warping, and save the calibration into a file.

Go to menu File -> Export Warping Blending

Select the calibrated Display compound (it should be selected by default),

check „use extended file definition“ and press Edit to open the SPExtCalibCommOpDef.ini file.

This will call the notepad for editing the ini-file - the file is being used for the export definition.

Here is the example of this file that can be used as the layout for different installations.

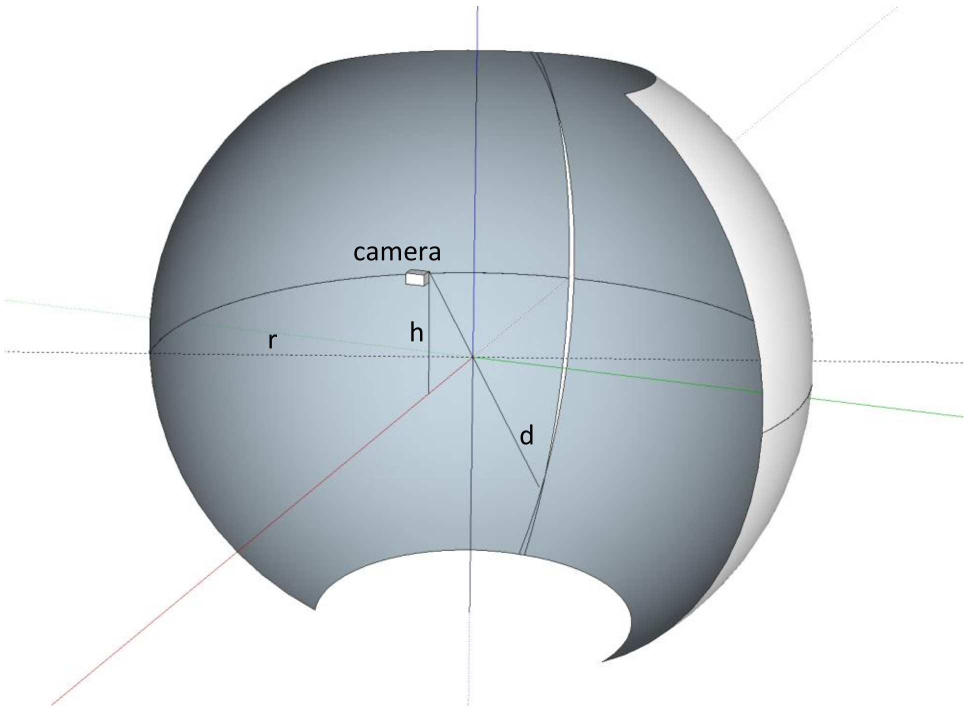
```
<?xml version="1.0"?>
<VIOSO>
  <File version="1.0.0" />

  <CalibCommerceTask Typ="export" Format="vwf" >
    <CommonParam bSilent="0" bNoVC="0"  bNoGeomCorr="0" bNoBlending="0" bNoMask="0" bSeparatedSplitDisp="1"      bAllToMaster="1"      bBlankUnusedSplitDispParts="0" bExactFileName="0" bVirtualContentRect="1" FileName="Display Group" />
    <!-- Insert the name of the calibration file here-->
    <SpecialParam qGridDimX="21" qGridDimY="21" vParam0_X="2915" vParam0_Y="1080" />
<!-- In the vParam0_X value Insert the width of the calibrated content space. Check below how to get this number „2915“ in Addition 1.  -->
<!-- In the vParam0_Y value Insert the vertical projection resolution. -->
    <TransferTask  Format="mips"  Device="D2  PROJECTOR  (ADVEF2D)     [  0,0  ]" vCntOffsetX="0" vCntOffsetY="0">
<!-- In the Device value Insert the projector names or their splitted parts exaclty as they appear in the calibrator. Note that spacebars are important. -->
<!-- In the vCntOffsetX and vCntOffsetX  value Insert the projector offset coordinates. This is the first left projector so the values are 0 and 0. Second projector has the offset values 995 and 0. -->
      <Destination Core="10.0.0.127"/>
<!—Destination Core insert the IP adress of the WB transmitting the signal of the following projector  -->
    </TransferTask>
    <TransferTask  Format="mips"  Device="D2  PROJECTOR  (ADVEF2D)     [  0,1  ]" vCntOffsetX="995" vCntOffsetY="0">
      <Destination Core="10.0.0.142"/>
    </TransferTask>
  </CalibCommerceTask>
</VIOSO>
```

10.2 Calibration of the cropped dome screen with an off-centre camera

In general, there is:

- The camera, describing VIOSO native coordinate system
  - By positioning the test image and entering radius and start angle, you name a 3D coordinate for every projector pixel on screen.
  - This is like a measured point in real world with some tolerance
  - View poses must use the same eye point
- A main view direction and eye point, describing the correspondence between IG and VIOSO
  - All views have same eye point in
  - There is no fix eye in IG, the eye can be everywhere in the virtual world
  - There is a main view direction, this is where we go “forward”
  - So every view is just an angular offset (yaw, pitch, roll angle) to that main view
- Offset angles to a main view
  - IG’s forward
  - View from eye “forward” to screen
  - Starting at SPCalibrator 4.3, it became easier, as we can use relative angles now

To position a camera somewhere else, here is the workflow how to position a fisheye camera at 2m height on a 4m diameter / 3m height cropped dome (220°x120°) screen.



- Use HQ calibration kit, it comes with 185° lens and very sensitive sensor
- Fix your camera at 2m height, facing the screen
- The central ray must go through the centre of the imaginary sphere, the screen is part of
- Go as far back as you need to see your whole screen, if upper left/right corner does not have any projection on it, you might move closer
- Adjust "Dome master" test image:
  - Pole is now where the central ray of the camera hits the screen
  - Image is now pitched down
  - Fill projection, make longitudes straight
  - Latitudes are circles
  - Do not try to line up outer circle with screen, just cover the screen using 4-point warp
- Now figure the angles
  - Find out which pitch the main view has

$$P = \arcsin\left(\frac{h}{d-r}\right);$$

where
d - distance from camera to screen, through sphere centre,
r - sphere radius,
h - off axis height, this is vertical distance from normal camera position (at sphere centre-height) to actual height,
Example, in meters:

$$r = 1.5;$$
$$h = 0.5;$$
$$d = 2.5;$$
$$P = \arcsin\left(\frac{h}{d-r}\right) = \arcsin\left(\frac{0.5}{2.5-1.5}\right) = \arcsin(0.5) = 30\,°;$$

  - ZERO (0,0,0) is still the sphere centre
  - Figure start angle
    - There are 10 latitude lines on our test image, numbered (0,) 10, 20, .. 90 (, 100), like percentage
    - Figure which would be the equator of a globe (you might even think of some extra line somewhere between).
    - The number *E* gives a ratio:

$$\frac{E}{90°} = \frac{100\%}{90° - s};$$

$$s = 90° - \frac{100\% * 90°}{E};$$

    - For example: equator (biggest circle) is the middle of line 70 and 80. This makes E=75%, thus

$$s = 90° - \frac{100\% * 90°}{75\%} = -30°;$$

    - Use this angle in "fisheye to spherical dome"-conversion
  - Figure view poses
    - Eye is always 0,0,0

- View poses are basically the same as with standard camera position, but pitched
- Create “main view”, which is pitched (rotation around x) by 30°, -y is up, that makes roll (rotation around z) 180°
- Do auto pose
- Use “main view” as relative view (SPCalibrator Version 4.3)
- Make them nice. They should be the same like the ones used in VRSG before, remember: screen, projector position and “forward” has not changed!

11. Notes

12. VIOSO Contacts

Should there be any questions which cannot be answered in the help section, please tell us about them. Please use the Support Function if there are any problems or errors. Wings Platinum can be updated via function Software Update. If you have any questions or suggestions, you can reach also us on the phone on weekdays from 9.00 am to 18.00 pm.

VIOSO GmbH
Ronsdorferstr. 77a
D-40233 Düsseldorf
Tel. +49 211 544 75 33 – 0
Fax: +49 211 544 75 33 – 33
E-mail: info@vioso.com
Internet: www.vioso.com

Documentation Status

Last review: **20.10.2016**
VIOSO Calibrator version: **4.3.1**
VIOSO Player version: **1.7**
VIOSO Anyblend Version: **4.3.1**