http://gmv.cast.uark.edu A Method Store for Advanced Survey and Modeling Technologies Mon, 01 Apr 2013 03:29:18 +0000 en-US hourly 1 http://wordpress.org/?v=3.5.1 http://gmv.cast.uark.edu/gps/acquire-external-control-for-close-range-photogrammetry-with-trimble-survey-grade-gps/ http://gmv.cast.uark.edu/gps/acquire-external-control-for-close-range-photogrammetry-with-trimble-survey-grade-gps/#comments Fri, 15 Mar 2013 09:54:36 +0000 steph http://gmv.cast.uark.edu/?p=13098

This page is a guide for acquiring external control for close range photogrammetry using Trimble survey grade GPS.
Hint: You can click on any image to see a larger version.

Prepare for Survey

1. Begin metadata process
   1. Choose a method for documenting the project (e.g. notebook, laptop)
   2. Fill in known metadata items (e.g. project name, date of survey, site location, etc.)
   3. Create a sketch map of the area (by hand or available GIS/maps)
2. Choose and prepare equipment
   1. Decide what equipment will best suite the project
   2. Test equipment for proper functioning and charge/replace batteries

Equipment Setup

1. Base station
   1. Setup and level the fixed height tripod over the point of your choice
   2. Attach the yellow cable to the Zephyr antenna
   3. Place the Zephyr antenna on top using the brass fixture and tighten screw
   4. Attach the yellow cable to the 5700 receiver
   5. Attach the external battery to the 5700 receiver (if using)
   6. Attach the data cable to the TSCe Controller and turn the controller on
   7. Create a new file and begin the survey
   8. Disconnect TSCe Controller

2. Rover
   1. Put two batteries in the 5800
   2. Attach the 5800 to the bipod
   3. Attach TSCe Controller to bipod using controller mount
   4. Connect data cable to 5800 and TSCe Controller
   5. Turn on the 5800 and controller
   6. Create a new project file (to be used all day)

Collecting Points

1. Have documentation materials ready
    1. As you collect points, follow ADS standards
2. Base station
    1. Once started, the base station will continually collect positions until stopped
    2. When you're ready to stop it, connect the TSCe controller to the receiver and end the survey
3. Rover
    1. When you arrive at a point you want to record, set the bipod up and level it over the point
    2. Using the controller, create a new point and name it
    3. Start collecting positions for the point and let it continue for the appropriate amount of time
    4. Stop collection when time is reached and move to next position

Data Processing

1. Have documentation materials ready
    1. As you process the data, follow ADS standards
2. Transfer data
    1. Use Trimble Geomatics Office (TGO) to transfer data files from the TSCe Controller and the 5700 receiver to the computer
3. Calculate baselines
    1. Use TGO to calculate baselines between base station and rover points
    2. Apply adjustment and export points

]]> http://gmv.cast.uark.edu/gps/acquire-external-control-for-close-range-photogrammetry-with-trimble-survey-grade-gps/feed/ 0 http://gmv.cast.uark.edu/uncategorized/microsoft-kinect-setting-up-the-development-environment/ http://gmv.cast.uark.edu/uncategorized/microsoft-kinect-setting-up-the-development-environment/#comments Tue, 11 Dec 2012 11:59:04 +0000 Matt T http://gmv.cast.uark.edu/?p=11061

Using Eclipse IDE

Since there is a plethora of existing tutorials guiding how to set up various development environments in C++, I will show you how to set up the 32-bit OpenNI JAR (OpenNI Java wrapper) in Eclipse IDE and to initialize a production node to begin accessing Kinect data via the Java programming language.
To continue we will be working with the open-source and fantastic piece of software known as Eclipse that you can find here: www.eclipse.org. You will want to download the IDE for Java programmers located on their "downloads" page(about 149 mb). Take note of the many other software solutions that they offer and the vast amount of resources on the site.

> NOTE: Even though we are downloading the "Java" Eclipse IDE you can easily add plugins to use this same piece of software with Python, C/C++, and many other applications.

Additionally, we are assuming that you have already gone through the OpenNI installation located here.

You also need to have the Java JDK installed (www.oracle.com).

Finally, to gain access to one of the best open-source computer vision libraries available, you will need to download and install OpenCV(http://opencv.org/) and the JavaCV(http://code.google.com/p/javacv/). The installation instructions located on each of these sites are excellent.

Setting Up Eclipse with OpenNI: Before You Start

Important Note: As you may already be aware, these tutorials are focused on the Beginner Level user, not only to using the Kinect but also to programming. Before going any further I should also remind you

that if jumping “head first” into the new domain of programming isn’t something for which you have the interest or the time, there are many things you can accomplish with the “ready to use software” solutions located here.

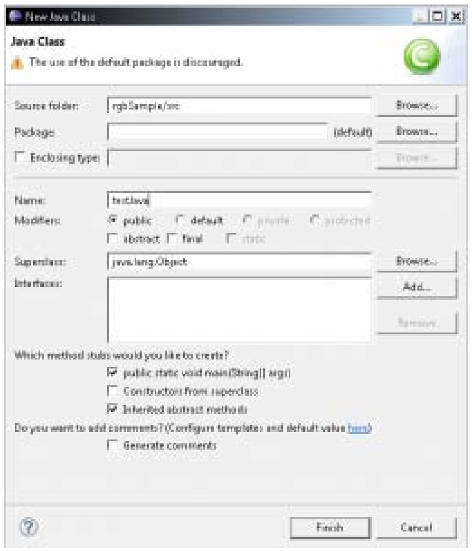
Also, before starting, make sure that you are using the same platform (32 –bit to 32-bit/64 to 64) on the Eclipse IDE, Java JDK, and OpenNI installation.

Eclipse with OpenNI: Starting a New Java Project

Starting a New Java Project .....

Once you have downloaded Eclipse, installed the java JDK and the OpenNI/Primesense, you will need to start a new Java Project. Following the wizard is the easiest way to do this.

Check the box that says “public static void main(String[] args)” so that Eclipse will add a few lines of code for us.



Java Class
The use of the default package is discouraged.

Source folder: rgbSample/src
Package: (default)
☐ Enclosing type:
Name: testJava
Modifiers: ● public ○ default ○ private ○ protected
☐ abstract ☐ final ☐ static
Superclass: java.lang.Object
Interfaces:

Which method stubs would you like to create?
☑ public static void main(String[] args)
☐ Constructors from superclass
☑ Inherited abstract methods
Do you want to add comments? (Configure templates and default value here)
☐ Generate comments

> **NOTE: For this tutorial I have kept the names fairly vague – be sure to use names that you will remember and understand. Remember that if you use a different naming convention than shown here, you will need to make corrections in the sample code to fit to your specifications.**

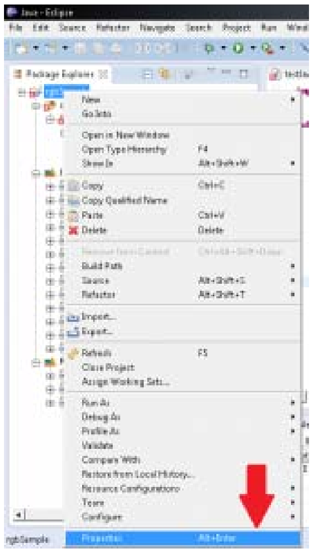
Eclipse with OpenNI: Adding the OpenNI Libraries Part 1

Adding the OpenNI libraries...

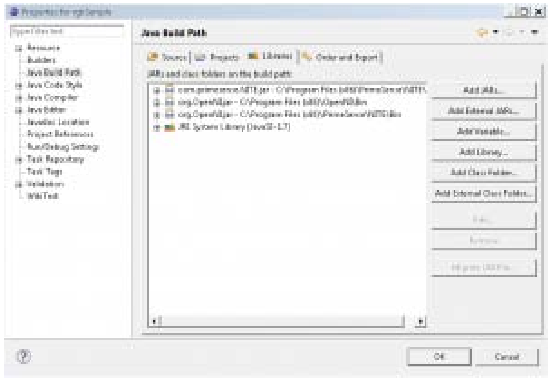
Next we will need to add the OpenNI libraries to the project. This is a pretty straight forward process in Java and Eclipse, simply being a matter of adding the pre-compiled JAR file from the “bin” folder of your OpenNI installation directory.

> NOTE: If you plan on using User Tracking or another Primesense middleware capability you will need to add the JAR in the Primesense directory

To do so right-click on the project we just created:



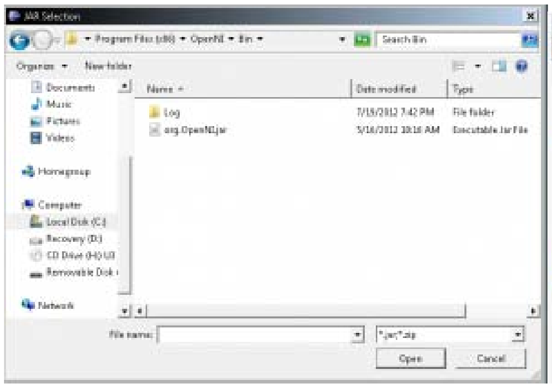
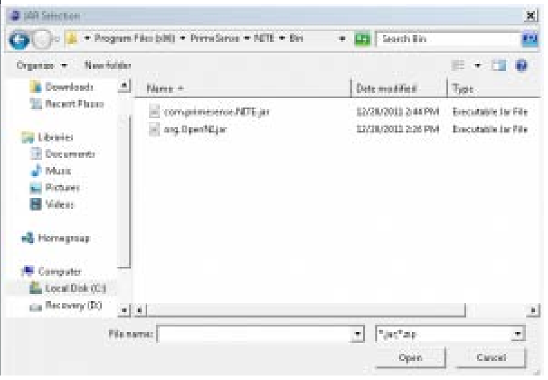
And select the “Properties” menu item.
Then we will want to select the “Java Build Path” and “Add External Jar’s” button.



Repeat the same steps as above for the JavaCV JAR’s that you previously installed somewhere on your machine.

Eclipse with OpenNI: Adding the OpenNI Libraries Part 2

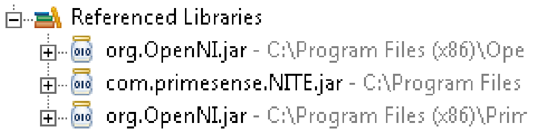
Navigate to the “bin” folders of the install directories for OpenNI and Primesense.
On my Windows 7 64-bit machine with the 32-bit install it is located here:

> **Note: There are <u>TWO</u> OpenNI “JAR” files – one in the bin folder of the OpenNI install directory as well as one in the Primesense directory. I haven’t noticed any difference in using one over the other; as long as your environment paths in Windows are set up to locate the needed files, they should both work.**

After this, you should see these files in the “Referenced Libraries” directory on the “Package Explorer” tool bar in

Eclipse.



Referenced Libraries
org.OpenNI.jar - C:\Program Files (x86)\Ope
com.primesense.NITE.jar - C:\Program Files
org.OpenNI.jar - C:\Program Files (x86)\Prir

Eclipse with OpenNI: Projects

We should now be able to access the Kinect via Java and Eclipse.

In the following projects we will introduce and attempt to explain the necessary steps for initializing the Kinect via OpenNI and for getting basic access to its data feeds in Java.

Each project goes through setting up the Kinect in OpenNI and includes comments to explain line-by-line what is going on.

For information on Visual Studio 2010 & Microsoft C# SDK....

Using the Microsoft SDK provides a lot of advantages and ease of access, but it is also only applicable to the "Kinect for Windows" hardware and not the Xbox Kinect (as of v1.5).

There are a lot of existing tutorials on the web about setting up your development environment with plenty of sample projects. Below is a list of links to a few of them in no particular order as to avoid reinventing the wheel.

1. http://channel9.msdn.com/Series/KinectQuickstart/Setting-up-your-Development-Environment
2. http://social.msdn.microsoft.com/Forums/el/kinectsdk/thread/7011aca7-defd-445a-bd3c-66837ccc716c
3. http://msdn.microsoft.com/en-us/library/hh855356.aspx
4. Power Point from Stanford

]]> http://gmv.cast.uark.edu/uncategorized/microsoft-kinect-setting-up-the-development-environment/feed/ 0 http://gmv.cast.uark.edu/uncategorized/working-with-data-from-the-kinect/ http://gmv.cast.uark.edu/uncategorized/working-with-data-from-the-kinect/#comments Fri, 06 Jul 2012 19:22:38 +0000 Matt T http://gmv.cast.uark.edu/?p=10459

RGB Image

The color image streaming from the RGB camera is much like that from your average webcam. It has a standard resolution of Height:640 Width:480 at a frame rate of 30 frames per second. You can "force" the Kinect to output a higher resolution image (1280×960) but it will significantly reduce it's frame rate.

Many things can be done with the RGB data alone such as:

- Image or Video Capture
- Optical Flow Tracking
- Capturing data for textures of models
- Facial Recognition
- Motion Tracking
- And many more....

While it seems silly to purchase a Kinect (about $150) just to use it as a webcam – it is possible. In fact there are ways to hook the camera up to Microsoft's DirectShow to use it with Skype and other webcam-enabled programs. (Check out this project http://www.e2esoft.cn/kinect/)

- For an Example Project Using the Kinect RGB feed ......

Depth Image

The Kinect is suited with two pieces of hardware, which through their combined efforts, give us the "Depth Image". It is the Infrared projector with the CMOS IR "camera" that measures the "distance" from the sensor to the corresponding object off of which the Infrared light reflects.

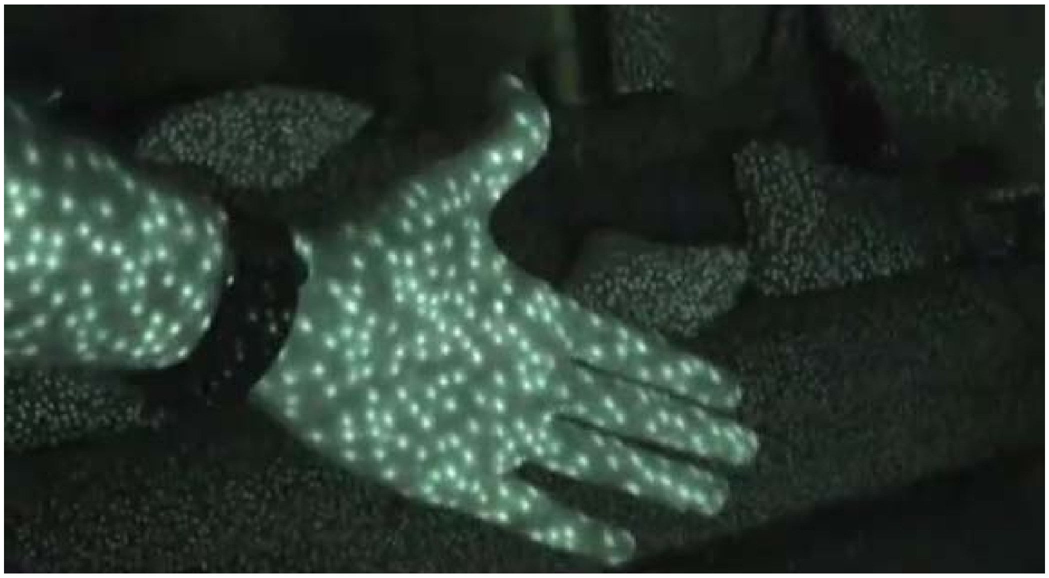
I say “distance” because the depth sensor of the Kinect actually measures the time that the light takes to leave the sensor and to return to the camera. The returning signal to the Kinect can be altered by other factors including:

- **The physical distance** – the return of this light is dependent on it reflecting off of an object within the range of the Kinect (~1.2–3.5m)
- **The surface** – like other similar technology (range cameras, laser scanners, etc.) the surface which the IR beam hits affects the returning signal. Most commonly glossy or highly reflective, screens (TV,computer,etc.), and windows pose issues for receiving accurate readings from the sensor.

The IR Projector

The IR projector does not emit uniform beams of light but instead relies on a “speckle pattern” according to the U.S. Patent (located here:http://www.freepatentsonline.com/7433024.pdf)

You can actually see the IRMap, as it’s called, using OpenNI. Here is a picture from Matthew Fisher’s website and another excellent resource on the Kinect (http://graphics.stanford.edu/~mdfisher/Kinect.html)



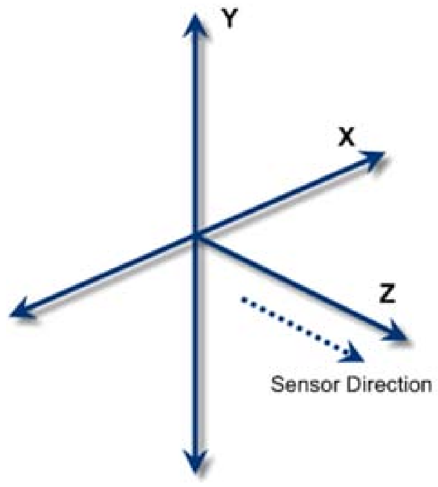
The algorithm used to compute the depth by the Kinect is derived from the difference between the speckle pattern that is observed and a reference pattern at a known depth.

> “The depth computation algorithm is a region-growing stereo method that first finds anchor points using normalized correlation and then grows the solution outward from these anchor points using the assumption that the depth does not change much within a region.”

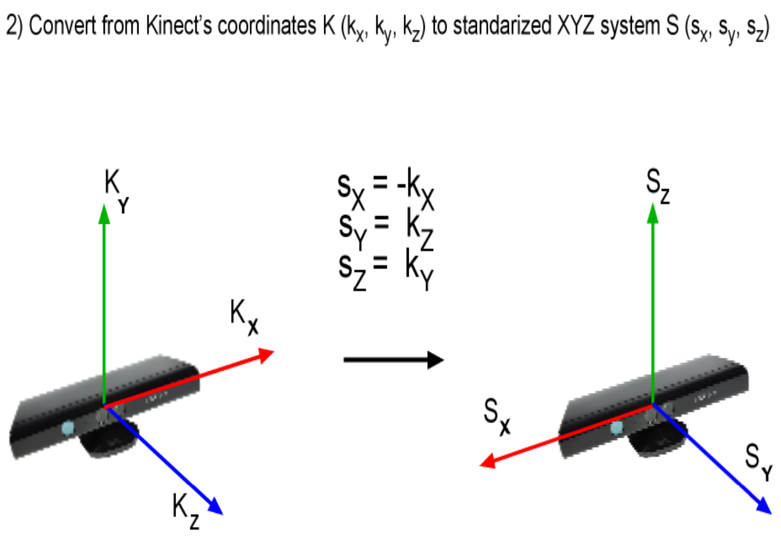
For a deeper discussion on the IR pattern from the Kinect check out this site : http://www.futurepicture.org/?p=116

Coordinate System

As one might assume the Kinect uses these “anchor points” as references in it’s own internal coordinate system. This origin coordinate system is shown here:



So the (x) is the to left of the sensor, (y) is up, and (z) is going out away from the sensor.


2) Convert from Kinect's coordinates K (k_x, k_y, k_z) to standarized XYZ system S (s_x, s_y, s_z)
K_Y
K_X
K_Z
$s_X = -k_X$
$s_Y = k_Z$
$s_Z = k_Y$
S_Z
S_X
S_Y

This is shown by the (Kx,Ky,Kz) in the above image, with the translated real world coordinates as (Sx,Sy,Sz).

The image is in meters (to millimeter precision).

Translating Depth & Coordinates

So when you work with the Depth Image and plan on using it to track, identify, or measure objects in real world coordinates you will have to translate the pixel coordinate to 3D space.OpenNI makes this easy by using their function (XnStatus xn::DepthGenerator::ConvertProjectiveToRealWorld) which converts a list of points from projective(internal Kinect) coordinates to real world coordinates.

Of course, you can go the other way too, taking real world coordinates to the projective using (XnStatus xn::DepthGenerator::ConvertRealWorldToProjective)

The depth feed from the sensor is 11-bits, therefore, it is capable of capturing a range of 2,048 values. In order to display this image in more 8-bit image structures you will have to convert the range of values into a 255 monochromatic scale. While it is possible to work with what is called the "raw" depth feed in some computer vision libraries(like OpenCV) most of the examples I've seen convert the raw depth feed in the same manner. That is to create a histogram from the raw data and to assign the corresponding depth value (from 0 to 2,048) to one of the 255 “bins” which will be the gray-scale value of black to white(0-255) in an 8-bit monochrome image.

You can look at the samples given by OpenNI to get the code, which can be seen in multiple programming languages by looking at their “Viewer” samples.

Accuracy

Another thing worth noting is the difference in accuracy of the depth image as distance from the Kinect increases. There seems to be a decrease in accuracy as one gets further away from the sensor, which makes sense when looking at the previous image of the pattern that the IR projector emits. The greater the physical distance between the object and the IR projector, the less coverage the speckle pattern has on that object in-between anchor points. Or in other words the dots are spaced further apart(x,y) as distance(z) is increased.

The Kinect comes factory calibrated and according to some sources, it isn’t that far off for most applications.

Links for Recalibrating

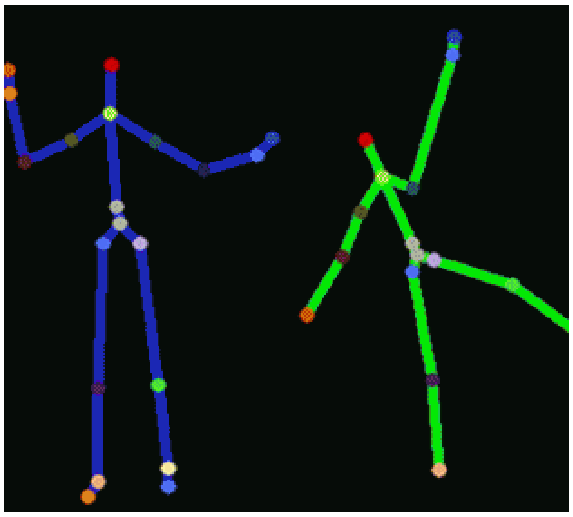
Here are some useful links to recalibrating the Kinect if you want to learn more:

- http://www.ros.org/news/2010/12/technical-information-on-kinect-calibration.html
- http://labs.manctl.com/rgbdemo/index.php/Documentation/TutorialProjectorKinectCalibration
- http://openkinect.org/wiki/Calibration
- http://nicolas.burrus.name/index.php/Research/KinectCalibration
- http://sourceforge.net/projects/kinectcalib/ (a MatLab ToolBox)

User Tracking

The Kinect comes with the capability to track users movements and to identify several joints of each user being tracked. The applications that this kind of readily accessible information can be applied to are plentiful. The basic capabilities of this feature, called “skeletal tracking”, have extended further for pose detection, movement prediction, etc.

According to information supplied to retailers, Kinect is capable of simultaneously tracking up to six people, including two active players, for motion analysis with a feature extraction of 20 joints per player. However,PrimeSense has stated that the number of people that the device can “see” (but not process as players) is only limited by how many will fit into the field-of-view of the camera.



Tracking 2 users **Microsoft

An in depth explanation can be found on the patent application for the Kinect here:http://www.engadget.com/photos/microsofts-kinect-patent-application/

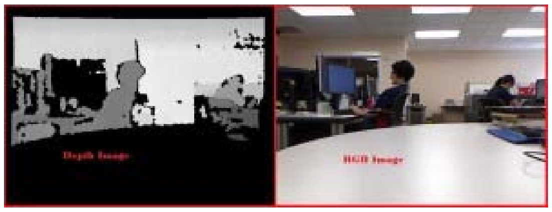
Point Cloud Data

The Kinect doesn't actually capture a 'point cloud'. Rather you can create one by utilizing the depth image that the IR sensor creates. Using the pixel coordinates and (z) values of this image you can transform the stream of data into a 3D "point cloud". Using an RGB image feed and a depth map combined, it is possible to project the colored pixels into three dimensions and to create a textured point cloud.

Instead of using 2D graphics to make a depth or range image, we can apply that same data to actually position the "pixels" of the image plane to 3D space. This allows one to view objects from different angles and lighting conditions. One of the advantages of transforming data into a point cloud structure is that it provides for more robust analysis and for more dynamic use than the same data in the form of a 2D graphic.

Connecting this to the geospatial world can be analogous to the practice of extruding Digital Elevation Models (DEM's) of surface features to three dimensions in order to better understand visibility relationships, slope, environmental dynamics, and distance relationships. While it is certainly possible to determine these things without creating a point cloud, the added ease of interpreting these various relationships from data in a 3D format is self-evident and inherent. Furthermore, the creation of a point cloud allows for an easy transition to creating 3D models that can be applied to various domains from gaming to planning applications.

So with two captured images like this:



Depth Map of the imaged scene, shown left in greyscale.

We can create a 3D point cloud.

Setting Up Your Development Environment

We will give you a few examples on how to set up your Development Environment using Kinect API's.

These are all going to be demonstrated on a Windows 7 64-bit machine using only the 32-bit versions of the downloads covered here.

At the time of this post the versions we will be using are:
OpenNI: v 1.5.2.23
Microsoft SDK: v 1.5
OpenKinect(libfreenect): Not Being Done at this time… Sorry

To use the following posts you need to have installed the above using these directions.

- For Information on setting up Eclipse and the OpenNI/Primesense Java ….

]]> http://gmv.cast.uark.edu/uncategorized/working-with-data-from-the-kinect/feed/ 0 http://gmv.cast.uark.edu/uncategorized/microsoft-kinect-additional-resources/ http://gmv.cast.uark.edu/uncategorized/microsoft-kinect-additional-resources/#comments Fri, 06 Jul 2012 19:13:53 +0000 Matt T http://gmv.cast.uark.edu/?p=10453

Links:Resources & Learning

Resources and Learning

1. **www.kinecthacks.com**

2. **www.kinect.dashhacks.com**

3. **www.kinecteducation.com**

4. **www.developkinect.com**

5. **www.scratch.saorog.com**

6. **www.microsoft.com/education/ww/partners-in-learning/Pages/index.aspx**

7. **blogs.msdn.com/b/uk_faculty_connection/archive/2012/04/21/kinect-for-windows-curriculum**

8. **dotnet.dzone.com/articles/kinect-sdk-resources**

9. **hackaday.com/2012/03/22/kinect-for-windows-resources**

10. **channel9.msdn.com/coding4fun/kinect**

11. **www.pcworld.com/article/217283/top_15_kinect_hacks_so_far.html**

Links: OpenNI

OpenNI

• openni.org – *Open Natural Interaction, an industry-led, not-for-profit organization formed to certify and promote the compatibility and interoperability of Natural Interaction (NI) devices, applications and middleware*

• github.com/openni – *Open source framework for natural interaction devices*

• github.com/PrimeSense/Sensor – *Open source driver for the PrimeSensor Development Kit*

Links: Tech

Tech

1. www.ifixit.com/Teardown/Microsoft-Kinect-Teardown/4066 – *Hardware teardown. Chip info is here. (via adafruit)*

2. kinecthacks.net/kinect-pinout – *Pinout info of the Kinect Sensor*

3. www.primesense.com/?p=535 – *Primesense reference implementation (via adafruit thread)*

4. www.sensorland.com/HowPage090.html – *How sensors work and the bayer filter*

5. www.numenta.com/htm-overview/education/HTM_CorticalLearningAlgorithms.pdf – *Suggestions to implement pseudocode near the end*

6. http://www.dwheeler.com/essays/floss-license-slide.html – *Which licenses are compatible with which*

7. http://www.eetimes.com/design/signal-processing-dsp/4211071/Inside-Xbox-360-s-Kinect-controller – *Another Hardware Teardown. Note this article incorrectly states that the PS1080 talks to the Marvell chip.*

8. http://nvie.com/posts/a-successful-git-branching-model/ – *Model for branching within Git*

9. http://git.kernel.org/?p=linux/kernel/git/torvalds/linux-2.6.git;a=blob;f=Documentation/SubmittingPatches – *Linux contribution procedure*

10. http://git.kernel.org/?p=git/git.git;a=blob_plain;f=Documentation/SubmittingPatches;hb=HEAD – *Git project contribution procedure*

Hardware Options

Well the first option is to build your own, here's a how-to:

http://www.hackengineer.com/3dcam/

But since not all of us have the time of skills to do that there are other options, like….

1- ASUS Xtion PRO:
Price: $140
Spec's: http://www.newegg.com/Product/Product.aspx?Item=N82E16826785030

2- Leap Motion:
Price: $70
Spec's: https://live.leapmotion.com/about.html

Of course there may be more and there is talk of Sony recently filling a patent resembling their own "Kinect-like" device.

Xbox Kinect vs. Kinect for Windows

As you may know there are actually two "Kinect" sensors out on the market today…

both under the Microsoft company but one was the original made for the Xbox 360 Game console, while the other is the recently released "Kinect for Windows".

Overview

As far as I can tell the two hardware stacks are identical except for the name plate on the front (XBOX or KINECT FOR WINDOWS) and the Windows version has a shorter power cord with a higher price tag due to licensing issues.

There are constant changes being done to the Kinect for Windows to distance itself from it's Xbox twin, like a firmware update to support "Near Mode" in the Windows SDK….

Microsoft even goes as far as saying the following:

> *'The Kinect for Windows SDK has been designed for the Kinect for Windows hardware and application development is only licensed with use of the Kinect for Windows sensor. We do not recommend using Kinect for Xbox 360 to assist in the development of Kinect for Windows applications. Developers should plan to transition to Kinect for Windows hardware for development purposes and should expect that their users will also be using Kinect for Windows hardware as well.'*

If you are currently using the Kinect for Xbox you will find that the automatic registration functions found with the Microsoft SDK will not recognize your Kinect and therefore kick out an error every time you try to run one of their samples.

As far as I know, you can however, still manually register the Kinect with the Microsoft SDK and utilize the functions already developed in the API AT THIS POINT with the XBOX version of the sensor. I wouldn't be surprised if this changes in the near future however.

Published Resources

Published Resources

1) Abramov, Alexey et al. “Depth-supported Real-time Video Segmentation with the Kinect.” Proceedings of the 2012 IEEE Workshop on the Applications of Computer Vision. Washington, DC, USA: IEEE Computer Society, 2012. 457–464. Web. 6 July 2012. WACV ’12.

2) Bleiweiss, Amit et al. “Enhanced Interactive Gaming by Blending Full-body Tracking and Gesture Animation.” ACM SIGGRAPH ASIA 2010 Sketches. New York, NY, USA: ACM, 2010. 34:1–34:2. Web. 6 July 2012. SA ’10.

3) Borenstein, Greg. Making Things See: 3D Vision with Kinect, Processing, Arduino, and MakerBot. Make, 2012. Print.

4) Boulos, Maged N Kamel et al. INTERNATIONAL JOURNAL OF HEALTH GEOGRAPHICS EDITORIAL Open Access Web GIS in Practice X: a Microsoft Kinect Natural User Interface for Google Earth Navigation. Print.

5) Burba, Nathan et al. “Unobtrusive Measurement of Subtle Nonverbal Behaviors with the Microsoft Kinect.” Proceedings of the 2012 IEEE Virtual Reality. Washington, DC, USA: IEEE Computer Society, 2012. 1–4. Web. 6 July 2012. VR ’12.

6) Center for History and New Media. “Zotero Quick Start Guide.”

7) Clark, Adrian, and Thammathip Piumsomboon. “A Realistic Augmented Reality Racing Game Using a Depth-sensing Camera.” Proceedings of the 10th International Conference on Virtual Reality Continuum and Its Applications in Industry. New York, NY, USA: ACM, 2011. 499–502. Web. 6 July 2012. VRCAI ’11.

8) Cui, Yan, and Didier Stricker. “3D Shape Scanning with a Kinect.” ACM SIGGRAPH 2011 Posters. New York, NY, USA: ACM, 2011. 57:1–57:1. Web. 6 July 2012. SIGGRAPH ’11.

9) Davison, Andrew. Kinect Open Source Programming Secrets: Hacking the Kinect with OpenNI, NITE, and Java. 1st ed. McGraw-Hill/TAB Electronics, 2012. Print.

10) Devereux, D. et al. “Using the Microsoft Kinect to Model the Environment of an Anthropomimetic Robot.” Submitted to the Second IASTED International Conference on Robotics (ROBO 2011). Web. 6 July 2012.

11) Dippon, Andreas, and Gudrun Klinker. “KinectTouch: Accuracy Test for a Very Low-cost 2.5D Multitouch Tracking System.” Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces. New York, NY, USA: ACM, 2011. 49–52. Web. 6 July 2012. ITS ’11.

12) Droeschel, David, and Sven Behnke. “3D Body Pose Estimation Using an Adaptive Person Model for Articulated ICP.” Proceedings of the 4th International Conference on Intelligent Robotics and Applications – Volume Part II. Berlin, Heidelberg: Springer-Verlag, 2011. 157–167. Web. 6 July 2012. ICIRA’11.

13) Dutta, Tilak. “Evaluation of the Kinect™ Sensor for 3-D Kinematic Measurement in the Workplace.” Applied

Ergonomics 43.4 (2012): 645–649. Web. 6 July 2012.

14) Engelharda, N. et al. “Real-time 3D Visual SLAM with a Hand-held RGB-D Camera.” Proc. of the RGB-D Workshop on 3D Perception in Robotics at the European Robotics Forum, Vasteras, Sweden. Vol. 2011. 2011. Web. 6 July 2012.

15) Francese, Rita, Ignazio Passero, and Genoveffa Tortora. “Wiimote and Kinect: Gestural User Interfaces Add a Natural Third Dimension to HCI.” Proceedings of the International Working Conference on Advanced Visual Interfaces. New York, NY, USA: ACM, 2012. 116–123. Web. 6 July 2012. AVI ’12.

16) Giles, J. “Inside the Race to Hack the Kinect.” The New Scientist 208.2789 (2010): 22–23. Print.

17) Gill, T. et al. “A System for Change Detection and Human Recognition in Voxel Space Using the Microsoft Kinect Sensor.” Proceedings of the 2011 IEEE Applied Imagery Pattern Recognition Workshop. Washington, DC, USA: IEEE Computer Society, 2011. 1–8. Web. 6 July 2012. AIPR ’11.

18) Gomez, Juan Diego et al. “Toward 3D Scene Understanding via Audio-description: Kinect-iPad Fusion for the Visually Impaired.” The Proceedings of the 13th International ACM SIGACCESS Conference on Computers and Accessibility. New York, NY, USA: ACM, 2011. 293–294. Web. 6 July 2012. ASSETS ’11.

19) Goth, Gregory. “Brave NUI World.” Commun. ACM 54.12 (2011): 14–16. Web. 6 July 2012.

20) Gottfried, Jens-Malte, Janis Fehr, and Christoph S. Garbe. “Computing Range Flow from Multi-modal Kinect Data.” Proceedings of the 7th International Conference on Advances in Visual Computing – Volume Part I. Berlin, Heidelberg: Springer-Verlag, 2011. 758–767. Web. 6 July 2012. ISVC’11.

21) Henry, P. et al. “RGB-D Mapping: Using Kinect-style Depth Cameras for Dense 3D Modeling of Indoor Environments.” The International Journal of Robotics Research (2012): n. pag. Web. 6 July 2012.

22) Henry, Peter et al. “RGB-D Mapping: Using Kinect-style Depth Cameras for Dense 3D Modeling of Indoor Environments.” Int. J. Rob. Res. 31.5 (2012): 647–663. Web. 6 July 2012.

23) Hilliges, Otmar et al. “HoloDesk: Direct 3d Interactions with a Situated See-through Display.” Proceedings of the 2012 ACM Annual Conference on Human Factors in Computing Systems. New York, NY, USA: ACM, 2012. 2421–2430. Web. 6 July 2012. CHI ’12.

24) Izadi, Shahram et al. “KinectFusion: Real-time 3D Reconstruction and Interaction Using a Moving Depth Camera.” Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology. New York, NY, USA: ACM, 2011. 559–568. Web. 6 July 2012. UIST ’11.

25) Jean, Jared St. Kinect Hacks: Creative Coding Techniques for Motion and Pattern Detection. O’Reilly Media, 2012. Print.

26) Kean, Sean, Jonathan Hall, and Phoenix Perry. Meet the Kinect: An Introduction to Programming Natural User Interfaces. 1st ed. Berkely, CA, USA: Apress, 2011. Print.

27) Kramer, Jeff et al. Hacking the Kinect. 1st ed. Berkely, CA, USA: Apress, 2012. Print.

28) LaViola, Joseph J., and Daniel F. Keefe. “3D Spatial Interaction: Applications for Art, Design, and Science.” ACM SIGGRAPH 2011 Courses. New York, NY, USA: ACM, 2011. 1:1–1:75. Web. 6 July 2012. SIGGRAPH ’11.

29) Li, Li, Yanhao Xu, and Andreas König. “Robust Depth Camera Based Eye Localization for Human-machine Interactions.” Proceedings of the 15th International Conference on Knowledge-based and Intelligent Information and Engineering Systems – Volume Part I. Berlin, Heidelberg: Springer-Verlag, 2011. 424–435. Web. 6 July 2012. KES’11.

30) Livingston, Mark A. et al. “Performance Measurements for the Microsoft Kinect Skeleton.” Proceedings of the 2012 IEEE Virtual Reality. Washington, DC, USA: IEEE Computer Society, 2012. 119–120. Web. 6 July 2012. VR ’12.

31) Melgar, Enrique Ramos, and Ciriaco Castro Diez. Arduino and Kinect Projects: Design, Build, Blow Their Minds. 1st ed. Berkely, CA, USA: Apress, 2012. Print.

32) Miles, Helen C. et al. “A Review of Virtual Environments for Training in Ball Sports.” Computers & Graphics 36.6 (2012): 714–726. Web. 6 July 2012.

33) Miles, Rob. Start Here! Learn the Kinect API. Microsoft Press, 2012. Print.

34) Mitchell, Grethe, and Andy Clarke. “Capturing and Visualising Playground Games and Performance: a Wii and Kinect Based Motion Capture System.” Proceedings of the 2011 International Conference on Electronic Visualisation and the Arts. Swinton, UK, UK: British Computer Society, 2011. 218–225. Web. 6 July 2012. EVA’11.

35) Molyneaux, David. “KinectFusion Rapid 3D Reconstruction and Interaction with Microsoft Kinect.” Proceedings of the International Conference on the Foundations of Digital Games. New York, NY, USA: ACM, 2012. 3–3. Web. 6 July 2012. FDG ’12.

36) Mutto, Carlo Dal, Pietro Zanuttigh, and Guido M. Cortelazzo. Time-of-Flight Cameras and Microsoft Kinect(TM). Springer Publishing Company, Incorporated, 2012. Print.

37) Panger, Galen. “Kinect in the Kitchen: Testing Depth Camera Interactions in Practical Home Environments.” Proceedings of the 2012 ACM Annual Conference Extended Abstracts on Human Factors in Computing Systems Extended Abstracts. New York, NY, USA: ACM, 2012. 1985–1990. Web. 6 July 2012. CHI EA ’12.

38) Pheatt, Chuck, and Jeremiah McMullen. “Programming for the Xbox Kinect™ Sensor: Tutorial Presentation.” J. Comput. Sci. Coll. 27.5 (2012): 140–141. Print.

39) Raheja, Jagdish L., Ankit Chaudhary, and Kunal Singal. “Tracking of Fingertips and Centers of Palm Using KINECT.” Proceedings of the 2011 Third International Conference on Computational Intelligence, Modelling & Simulation. Washington, DC, USA: IEEE Computer Society, 2011. 248–252. Web. 6 July 2012. CIMSIM ’11.

40) Riche, Nicolas et al. “3D Saliency for Abnormal Motion Selection: The Role of the Depth Map.” Proceedings of the 8th International Conference on Computer Vision Systems. Berlin, Heidelberg: Springer-Verlag, 2011. 143–152. Web. 6 July 2012. ICVS’11.

41) Rogers, Rick. “Kinect with Linux.” Linux J. 2011.207 (2011): n. pag. Web. 6 July 2012.

42) Shrewsbury, Brandon T. “Providing Haptic Feedback Using the Kinect.” The Proceedings of the 13th International ACM SIGACCESS Conference on Computers and Accessibility. New York, NY, USA: ACM, 2011. 321–322. Web. 6 July 2012. ASSETS ’11.

43) Sidik, Mohd Kufaisal bin Mohd et al. “A Study on Natural Interaction for Human Body Motion Using Depth Image Data.” Proceedings of the 2011 Workshop on Digital Media and Digital Content Management. Washington, DC, USA: IEEE Computer Society, 2011. 97–102. Web. 6 July 2012. DMDCM ’11.

44) Smisek, J., M. Jancosek, and T. Pajdla. “3D with Kinect.” Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference On. 2011. 1154 –1160.

45) Solaro, John. “The Kinect Digital Out-of-Box Experience.” Computer 44.6 (2011): 97–99. Web. 6 July 2012.

46) Stone, E. E, and M. Skubic. “Evaluation of an Inexpensive Depth Camera for Passive In-home Fall Risk Assessment.” Pervasive Computing Technologies for Healthcare (PervasiveHealth), 2011 5th International Conference On. 2011. 71–77. Web. 6 July 2012.

47) Sturm, J. et al. “Towards a Benchmark for RGB-D SLAM Evaluation.” Proc. of the RGB-D Workshop on Advanced Reasoning with Depth Cameras at Robotics: Science and Systems Conf.(RSS), Los Angeles, USA. Vol. 2. 2011. 3. Web. 6 July 2012.

48) Sung, J. et al. “Human Activity Detection from RGBD Images.” AAAI Workshop on Pattern, Activity and Intent Recognition (PAIR). 2011. Web. 6 July 2012.

49) Tang, John C., Carolyn Wei, and Reena Kawal. “Social Telepresence Bakeoff: Skype Group Video Calling, Google+ Hangouts, and Microsoft Avatar Kinect.” Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work Companion. New York, NY, USA: ACM, 2012. 37–40. Web. 6 July 2012. CSCW ’12.

50) “The Kinect Revolution.” The New Scientist 208.2789 (2010): 5. Web. 6 July 2012.

51) Tong, Jing et al. “Scanning 3D Full Human Bodies Using Kinects.” IEEE Transactions on Visualization and Computer Graphics 18.4 (2012): 643–650. Web. 6 July 2012.

52) Villaroman, Norman, Dale Rowe, and Bret Swan. “Teaching Natural User Interaction Using OpenNI and the Microsoft Kinect Sensor.” Proceedings of the 2011 Conference on Information Technology Education. New York, NY,

USA: ACM, 2011. 227–232. Web. 6 July 2012. SIGITE '11.

53) "Virtual Reality from the Keyboard/mouse Couple to Kinect." Annals of Physical and Rehabilitation Medicine 54, Supplement 1.0 (2011): e239. Web. 6 July 2012.

54) Webb, Jarrett, and James Ashley. Beginning Kinect Programming with the Microsoft Kinect SDK. 1st ed. Apress, 2012. Print.

55) Weise, Thibaut et al. "Kinect-based Facial Animation." SIGGRAPH Asia 2011 Emerging Technologies. New York, NY, USA: ACM, 2011. 1:1–1:1. Web. 6 July 2012. SA '11.

56) Wilson, Andrew D. Using a Depth Camera as a Touch Sensor. Print.

57) Xu, Yunfei, and Jongeun Choi. "Spatial Prediction with Mobile Sensor Networks Using Gaussian Processes with Built-in Gaussian Markov Random Fields." Automatica 0 n. pag. Web. 6 July 2012.

58) Zhang, Zhengyou. "Microsoft Kinect Sensor and Its Effect." IEEE MultiMedia 19.2 (2012): 4–10. Web. 6 July 2012.

]]> http://gmv.cast.uark.edu/uncategorized/microsoft-kinect-additional-resources/feed/ 0
http://gmv.cast.uark.edu/uncategorized/ready-to-use-software-for-the-kinect/
http://gmv.cast.uark.edu/uncategorized/ready-to-use-software-for-the-kinect/#comments Fri, 06 Jul 2012 17:19:17 +0000 Matt T http://gmv.cast.uark.edu/?p=10433

Getting Started Fast with the Kinect

Getting Started

There are a variety of software that are being or have been developed that provide you with access to the Kinect sensor through interactive GUI's. Some use different API's and libraries (OpenNI,Microsoft SDK, etc.) to do this communication with the Kinect and these may conflict with software that you have already installed. So read the sites before downloading and installing. Make sure that you do clean installs.

In this post we will provide you with a high level introduction to two excellent software suites provided on a free-to-use basis.

Instant Access

These provide instant access to the sensor data and help to get your project started whether it involves scanning, recording, viewing, or streaming from the Kinect.

Follow the links at the end of the pages to see an example workflow with each of the two software packages.

RGBDemo

RGB Demo was initially developed by Nicolas Burrus in the RoboticsLab. He then co-founded the Manctl company that now maintains it, helped by various contributors from the opensource community.

Current features

- Grab kinect images and visualize / replay them
- Support for libfreenect and OpenNI/Nite backends
- Extract skeleton data / hand point position (Nite backend)
- Integration with OpenCV and PCL
- Multiple Kinect support and calibration
- Calibrate the camera to get point clouds in metric space (libfreenect)
- Export to meshlab/blender using .ply files

- Demo of 3D scene reconstruction using a freehand Kinect
- Demo of people detection and localization
- Demo of gesture recognition and skeleton tracking using Nite
- Demo of 3D model estimation of objects lying on a table (based on PCL table top object detector)
- Demo of multiple kinect calibration
- Linux, MacOSX and Windows support

RGBDemo can be found here: www.labs.manctl.com/rgbdemo/index.php

Brekel Kinect

In my opinion the Brekel Kinect software is the best that I have seen for easily interfacing with the Kinect.

It uses the OpenNI framework, which we show on the GMV here, but the developer of this software also provides their own packaged OpenNI installer that comes with all the required dependencies.

Current Features

The Brekel Kinect only offers binaries for the Windows platform. It was developed by Jasper Brekelmans in his free time.

It allows you to capture 3D objects and to export them to disk for use in 3D packages. It also allows you to do skeleton tracking which can be streamed into Autodesk's MotionBuilder in realtime, or exported as BVH files.

The greatest things about the Brekel software are that it requires no programming expertise, it has an easy to use GUI, and it has the ability to export almost all of the Kinect's capabilities in a variety of formats.

The Brekel website offers a bunch of links to other resources, downloads, tutorials and answers to FAQ's. Check it out at:

Main Site: http://www.brekel.com

]]> http://gmv.cast.uark.edu/uncategorized/ready-to-use-software-for-the-kinect/feed/ 0 http://gmv.cast.uark.edu/uncategorized/openni-and-the-kinect/ http://gmv.cast.uark.edu/uncategorized/openni-and-the-kinect/#comments Fri, 06 Jul 2012 16:28:24 +0000 Matt T http://gmv.cast.uark.edu/?p=10411

Overview

The OpenNI Organization is primarily supported by PrimeSense, the company who originally developed the Kinect hardware for Microsoft.

The OpenNI framework is not specifically designed to work with the Kinect hardware – rather it is capable of interfacing with various sensors that all happen to be located in the hardware stack known as the Kinect.

OpenNI is primarily written in C++ but comes with Java and .NET wrappers.

Python wrappers do exist, however we have not used them yet.
Here are some links to those wrappers: https://github.com/jmendeth/PyOpenNI, http://code.google.com/p/onipy/

In the next slide we show you how to install the OpenNI modules.

Downloading for Installation 1

This procedure will be focused on a Windows 7 OS machine, with Visual Studio 2010. Although it is a 64-bit machine, we will be downloading the 32-bit versions of the software from OpenNI in order to maintain compatibility throughout our following workflows.

NOTE: It is generally good etiquette to use 64-bit software on 64-bit machines. However, in this case it seems to provide less pain in the long run to use the 32-bit version as we ultimately intend to deploy our code in programs of projects that use other 32-bit libraries or software. Whether 32-bit or 64-bit, the installation process is identical – you just need to make sure you install all of the right modules for your platform and that you not mix them up by accident.

Download the required packages from the OpenNI website located here:
http://www.openni.org/Downloads/OpenNIModules.aspx

These are the pre-compiled binaries offered by OpenNI for Mac, Windows, and Ubuntu(32 & 64-bit).

You will need to choose the builds you want (Stable or Unstable)and to keep this selection consistent. For assured performance we will be downloading the Stable builds.

Downloading for Installation 2

One last consideration before downloading is ensuring that you download all of the same editions (Development or Redistributable).

For the most part, either one will work. Since we are primarily going to use the Kinect for in-house projects, we are using the development editions.

So we are downloading the following executables:

- **OpenNI Binaries**–> Stable –>OpenNI Stable Build for Windows x86(32-bit) v1.5.2.23 Development Edition
- **OpenNI Compliant Middleware Binaries** –> Stable–> Prime Sense NITE Stable Build for Windows x86(32-bit)v1.5.2.21 Development Edition
- **OpenNI Compliant Hardware Binaries**–> Stable –>Prime Sensor Module Stable Build for Windows x86 (32-bit) v.5.1.0.41

In addition to these you will have to download these drivers:

https://github.com/avin2/SensorKinect/

***On a side note you can skip downloading the individual modules and download one of the packages which includes all three installers in one executable. However, we have had issues in the past with these installing incorrectly. Also note that even when using the packages, you will still need to download the Kinect driver located at the last link above.

Installation

First we need to make sure that the Kinect is unplugged from the computer and that all previous installations have been removed.

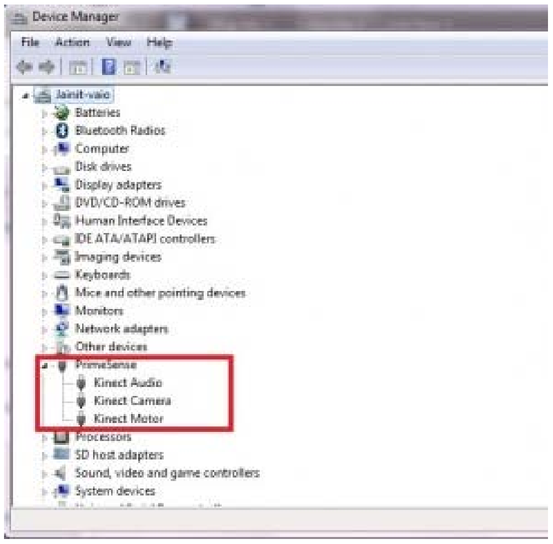
Previously, the order in which you installed these modules was important. For that matter, it may still affect the installation. Regardless, we will continue to follow the convention of installing in this order:

1. “openni-win32XXXX.exe”
2. “nite-win32XXXX.exe”
3. “sensor-win32XXXX.exe”
4. “SensorKinect093-Bin-Win32XXXX.msi” (note this is located in the bin folder of the .zip file we downloaded from https://github.com/avin2/SensorKinect/)

Test the Install

Okay, so now let’s check and confirm that it worked.

Plug in the Kinect and (assuming you are working on a Windows machine) pull up your “Device Manager”. Confirm that you see the following:



If you do not see this or if you have automatic driver installation (meaning that Windows might have installed drivers for the Kinect automatically), make sure that you re-run the installation above and that you completely uninstall the current set of drivers and software.

One final test if the installation is unsuccessful, is to pull up one of the samples provided by OpenNI and Primesense. These are located in the in the installation directory.

For more on using OpenNI check out the other posts on this site and the Additional Resources post.

]]> http://gmv.cast.uark.edu/uncategorized/openni-and-the-kinect/feed/ 0 http://gmv.cast.uark.edu/uncategorized/microsoft-kinect-resourcesprogramming-apis/ http://gmv.cast.uark.edu/uncategorized/microsoft-kinect-resourcesprogramming-apis/#comments Fri, 06 Jul 2012 05:44:35 +0000 Matt T http://gmv.cast.uark.edu/?p=10390

API's for the Kinect

There are three primary programming API's commonly used when working with the Kinect.

On this page a general overview of the API's is given and then we direct you to resources regarding each of those discussed.

Note that the pro's and con's of each API is really for you to find out. The work done with Kinect here at CAST has primarily been with the OpenNI API.

The capabilities of each API and their limitations will be project specific to your project.

So which API is the best?

While each of the presented API's offer differences in the way of community support, programming language options, and actual accessibility to the Kinect it would be impossible to state that one is the best or significantly better than the other ones across the board. Your program development skills and the situation of your application will dictate which is the route for you to choose.

Currently, the GMV series on the Kinect is not intended to be focused on development at an in-depth level. Rather, we hope only to provide a resource that pools existing materials in one location and gives you examples of a few of the many projects in which you can use the Kinect.

Furthermore, rapid updating and continual revisions to each of these projects require you to read up on their respective websites for the most current changes.

At the time of writing this page (summer 2012) the three primary options are the

Microsoft supported SDK (v1.5), OpenNI (v1.5.2), and the Open Kinect Project.

There are other means and routes to obtaining access to the Kinect hardware each requiring different expertise in programming knowledge, OS requirements, and licensing requirements.

A final note and a somewhat important one:

These libraries usually do not play well with each other, with each of them requiring their own driver's and dependencies for using them.
There are options to bridge these gaps but in general terms, it will be necessary to completely remove any legacy or conflicting installations before switching from one to the other. That includes drivers, .dll's, and registry/environment path settings.

Overview of OpenNI

About the OpenNI organization

The OpenNI organization is an industry-led, not-for-profit organization formed to certify and promote the compatibility and interoperability of Natural Interaction devices, applications and middleware. One of the OpenNI organization goals is to accelerate the introduction of Natural Interaction applications into the marketplace.

Their website is www.openni.org, where you can download the latest binaries and software packages available. They also offer documentation for their API, review sample projects, and provide a forum for user generated-projects (most of which come with source code).

Main Site: www.openni.org
DownLoads: www.openni.org/Downloads/OpenNIModules.aspx
Daily Build and Dev Edition: github.com/OpenNI/OpenNI
Community Site: arena.openni.org

GO TO MORE ON OPENNI ON THE GMV.....

Overview of Microsoft SDK

Kinect for Windows and Xbox Kinect

The Kinect for Windows SDK is provided free of charge to developers who wish to create applications using C++, C# or Visual Basic. Being formally supported by Microsoft gives you access to all of the capabilities of the Kinect including gesture and voice recognition.

An important consideration is that the Microsoft SDK is intended for the "Kinect for Windows", which is different than the "Xbox Kinect".

> While as of v.1.5 SDK the Xbox Kinect is still accessible for development purposes,THERE IS NO DISTRIBUTABLE CAPABILITY with this API due to licensing issues regarding the hardware. Which consequently is the only major difference between the Kinect for Windows and Xbox Kinect at this point. For more information on the differences between these seemingly twin sensors, please check out the links at the bottom of this slide.

Downloading

Microsoft prepackages drivers for using the Kinect sensor on a computer running Windows 7, Windows 8 Consumer

Preview, and Windows Embedded Standard 7. In addition, the download includes application programming interfaces (APIs) and device interfaces.

> Microsoft also offers their Development Toolkit which comes with about a dozen example projects that you can reuse in your own work.
>
> Current Version: 1.5, updated 05/21/2012
> Size: 221 MB
> Language: English

Main Site: www.microsoft.com/en-us/kinectforwindows
Download Page: www.microsoft.com/en-us/kinectforwindows/develop/developer-downloads.aspx
Documentation: www.microsoft.com/en-us/kinectforwindows/develop/learn.aspx
Gallery(User Forum): www.microsoft.com/en-us/kinectforwindows/develop/community_support.aspx

Overview of The Open Kinect Project

About OpenKinect

> *OpenKinect is an open community of people interested in making use of the amazing Xbox Kinect hardware with PCs and other devices. They are working on free, open source libraries that will enable the Kinect to be used with Windows, Linux, and Mac.*
>
> *The OpenKinect community consists of over 2000 members contributing their time and code to the Project. Members have joined this Project with the mission of creating the best possible suite of applications for the Kinect.*
>
> *OpenKinect is a true "open source" community!*
>
> *The primary focus is currently the libfreenect software. Code contributed to OpenKinect where possible is made available under an Apache20 or optional GPL2 license.*

Main Site: www.openkinect.org/wiki/Main_Page
Download: www.github.com/OpenKinect/libfreenect
Documentation: www.openkinect.org/wiki/Documentation

]]> http://gmv.cast.uark.edu/uncategorized/microsoft-kinect-resourcesprogramming-apis/feed/ 0 http://gmv.cast.uark.edu/scanning/hardware/microsoft-kinect-resourceshardware/ http://gmv.cast.uark.edu/scanning/hardware/microsoft-kinect-resourceshardware/#comments Fri, 06 Jul 2012 03:00:50 +0000 Matt T http://gmv.cast.uark.edu/?p=10354

Hardware Specifications

The Kinect provides convenient access to a stack of hardware sensors that are capable of working in sync with each other at an affordable price tag. A general overview on each of the sensors is given here when available.

Power Consumption: 2.25W
Field of View for Both Cameras:
Horizontal field of view: 57 degrees
Vertical field of view: 43 degrees
Tilt Motor Range: -27 to 27 degrees vertical (from base)

Connection Interface: The Kinect port is a Microsoft proprietary connector that provides power and data communication for the Kinect sensor. Ideally Microsoft would have used a USB port, but due to the additional load of the pivot motor, a standard Xbox 360 USB port could not provide enough power for this device. All of the newer Xbox 360 slim models include a dedicated Kinect port.

If you own an older Xbox 360 console you will need a spare USB port and an external power supply. This power supply is provided with the Kinect pack as standard.

“Microsoft Kinect Teardown.” iFixit, n.d.
http://www.ifixit.com/Teardown/Microsoft-Kinect-Teardown/4066/1.

Camera(RGB) Specifications

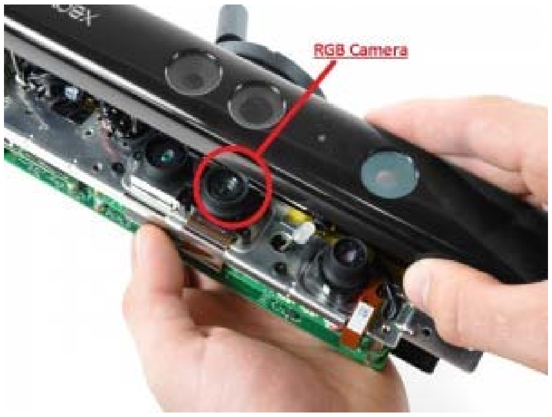
The Kinect camera is an RGB device with a resolution of 640×480 and a 24 bit color range (Red- Green- Blue channels).

Working at a rate of 30 frame captures per second this camera, this camera is similar to the run of the mill webcam or the sensors in your digital camera and it is, in most regards, very commonplace.

The camera is capable of capturing 1280×960 resolution however it seems that the fastest frames per second (fps) achievable at this higher definition is about 15fps. Conversely faster frame rates are possible by setting the resolution of the camera lower.

The Kinect sensor is limited in the distance that it can see and has a working range of between 1.2 and 3.5 meters. The horizontal field of view is 57° wide, which means at its maximum range it will be able to scan a scene 3.8 meters wide.

The sensor has a vertical field of view of 43° or 63 cm (25 in). This field of view is enhanced by the vertical pivot system that allows the sensor to be tilted up or down by as much as 27° in either direction.



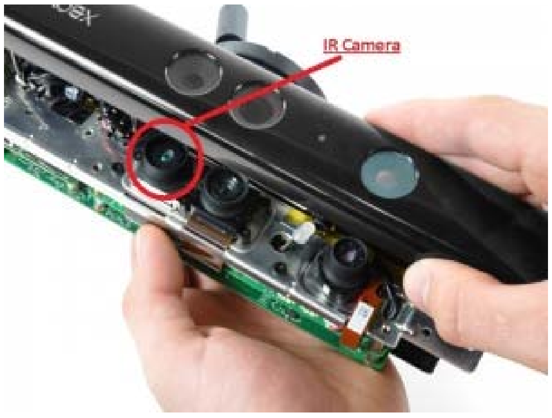
"Microsoft Kinect Teardown." iFixit, n.d. http://www.ifixit.com/Teardown/Microsoft-Kinect-Teardown/4066/1.

IR Depth Sensor Specifications

An infrared (IR) emitter and an IR depth sensor give the Kinect its depth measuring capabilities. The emitter emits infrared light beams and the depth sensor reads the IR beams reflected back to the sensor. The reflected beams are converted into depth information measuring the distance between an object and the sensor. This makes capturing a depth image possible.

The depth sensor system consists of an infrared projector and a monochrome CMOS sensor. Located to either side of the RGB camera, the depth sensor has a resolution of 640×480 with 16 bit sensitivity. ***Note that some sources say the depth feed is actually 11-bits.

This means it can see roughly 65,000 shades of grey. Like the RGB camera, the depth sensor also captures video at a rate of 30 frames per second. However depending on the machine you are running it on, complexity of the task being done, etc. dropping frames seems to be a common mishap.

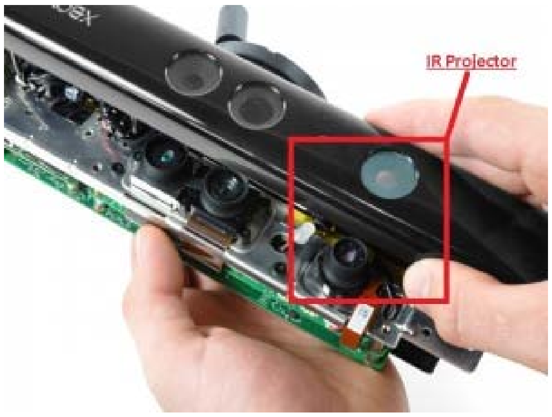


"Microsoft Kinect Teardown." iFixit, n.d. http://www.ifixit.com/Teardown/Microsoft-Kinect-Teardown/4066/1.

Some rough estimates of the accuracy of the depth sensor:

- **Range:** ~ 50 cm to 5 m. Can get closer (~ 40 cm) in parts, but can't have the full view be < 50 cm.
- **Horizontal Resolution:** 640 x 480 and 45 degrees vertical Field of View (FOV) and 58 degrees horizontal FOV. Simple geometry shows this equates to about ~ 0.75 mm per pixel x by y at 50 cm, and ~ 3 mm per pixel x by y at 2 m.
- **Depth resolution:** ~ 1.5 mm at 50 cm. About 5 cm at 5 m.

Noise: About +-1 DN at all depths, but DN to depth is non-linear. This means +-1 mm close, and +-5 cm far.



“Microsoft Kinect Teardown.” iFixit, n.d. http://www.ifixit.com/Teardown/Microsoft-Kinect-Teardown/4066/1.

Here are some useful papers reviewing the specifications of the depth image:

Khoshelham, K. and Oude Elberink, S.J. (2012) Accuracy and resolution of Kinect depth data for indoor mapping applications. In: Sensors : journal on the science and technology of sensors and biosensors : open access, 12 (2012)2 pp. 1437-1454.

Nathan Crock has also developed his own accuracy testing for the Kinect which can be viewed here: http://mathnathan.com/2011/02/03/depthvsdistance/

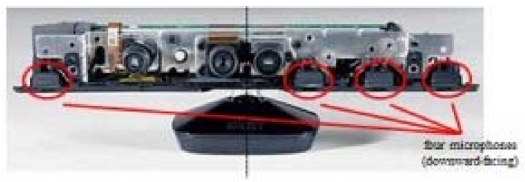
The ISPRS Laser Scanning Proceedings: http://www.isprs.org/proceedings/XXXVIII/5-W12/info.pdf

Microphone Array Specifications

The Kinect microphone is capable of localizing acoustic sources and suppressing ambient noise. This allows people to chat over Xbox Live without a headset.

Physically the microphone consists of an array of 4 microphone capsules. Each of the four channels processes sound at in 16 bit audio at a rate of 16 KHz.

This is an array of four microphones that can isolate the voices of the players from the noise in the room. This allows the player to be a few feet away from the microphone and still use voice controls.



“Microsoft Kinect Teardown.” iFixit, n.d. http://www.ifixit.com/Teardown/Microsoft-Kinect-Teardown/4066/1.

Power Consumption: 2.25W
Field of View for Both Cameras:
Horizontal field of view: 57 degrees
Vertical field of view: 43 degrees
Tilt Motor Range: -27 to 27 degrees vertical (from base)
]]> http://gmv.cast.uark.edu/scanning/hardware/microsoft-kinect-resourceshardware/feed/ 0
http://gmv.cast.uark.edu/scanning/guide-to-leveling-and-aligning-the-breuckmann-tripod-and-smartscan-he-for-calibration/ http://gmv.cast.uark.edu/scanning/guide-to-leveling-and-aligning-the-breuckmann-tripod-and-smartscan-he-for-calibration/#comments Tue, 10 Apr 2012 20:06:51 +0000 Rachel http://gmv.cast.uark.edu/4146/guide-to-leveling-and-aligning-the-breuckmann-tripod-and-smartscan-he-for-calibration/

This workflow will show you how set up the Breuckmann Tripod and SmartScan HE for calibration prior to beginning your scanning project. *Hint: You can click on any image to see a larger version.*

TRIPOD LEGS SETUP

Start with the tripod in the closed position (all three legs together) and check that the legs are the same length, adjusting them if necessary.

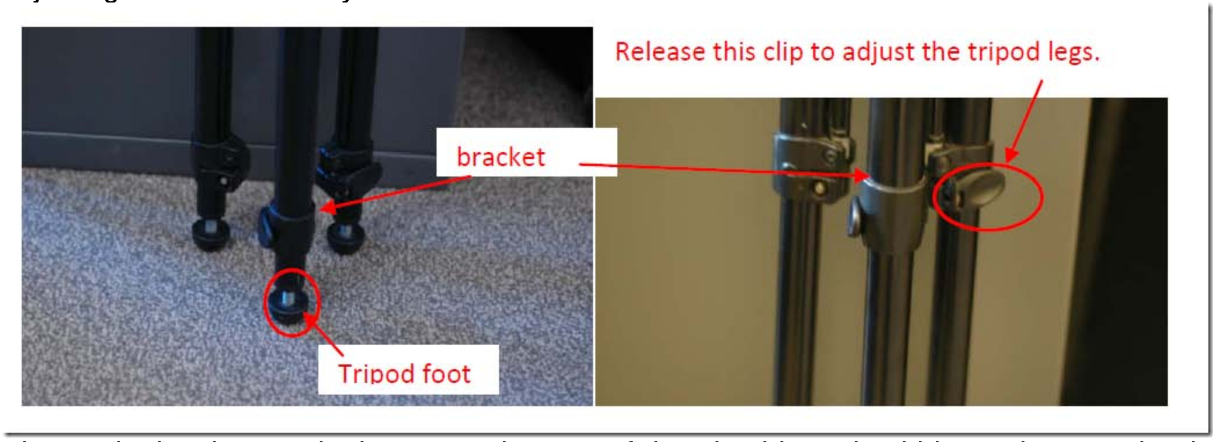


Fig. 1: The brackets at the bottom and center of the tripod legs should be at the same level when the legs are closed, and all the feet should touch the floor at the same time.

Open the tripod legs by loosening the wingnut on the central pole of the tripod and sliding the legs outward. Create a wide triangular base with the tripod legs and tighten the wingnut.

PLACE TRIPOD NEXT TO TABLE

Place the tripod so that two adjacent legs are flush and parallel with the edge of the table or desk on which you have placed (or will place) the calibration chart.

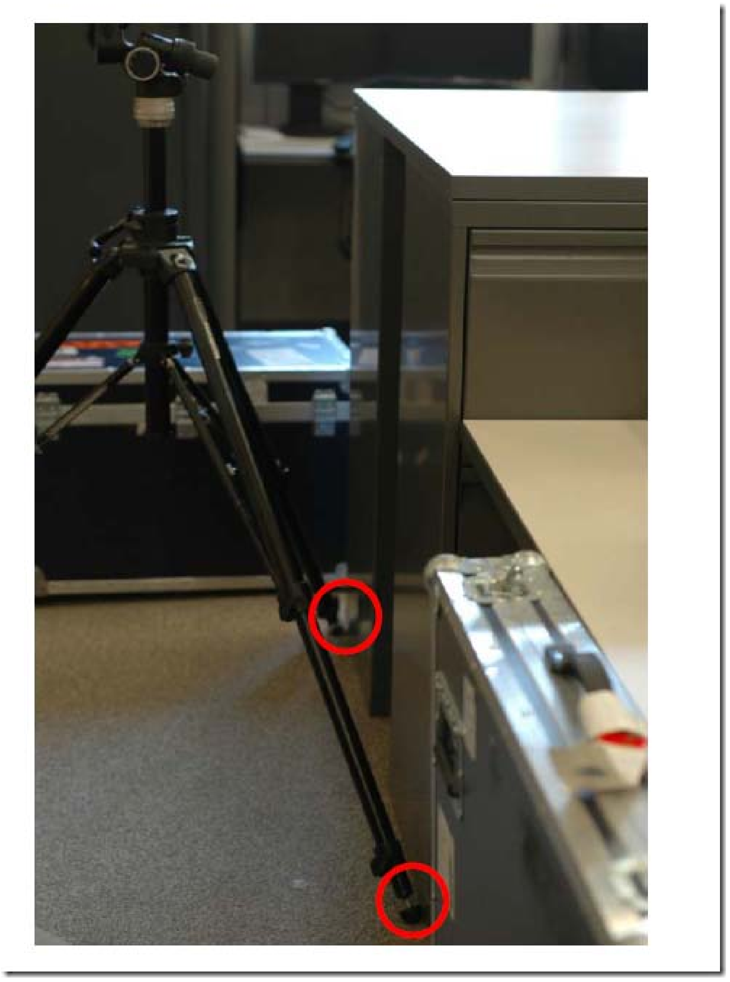


Fig.2: Tripod feet are level and flush with the base of the desk

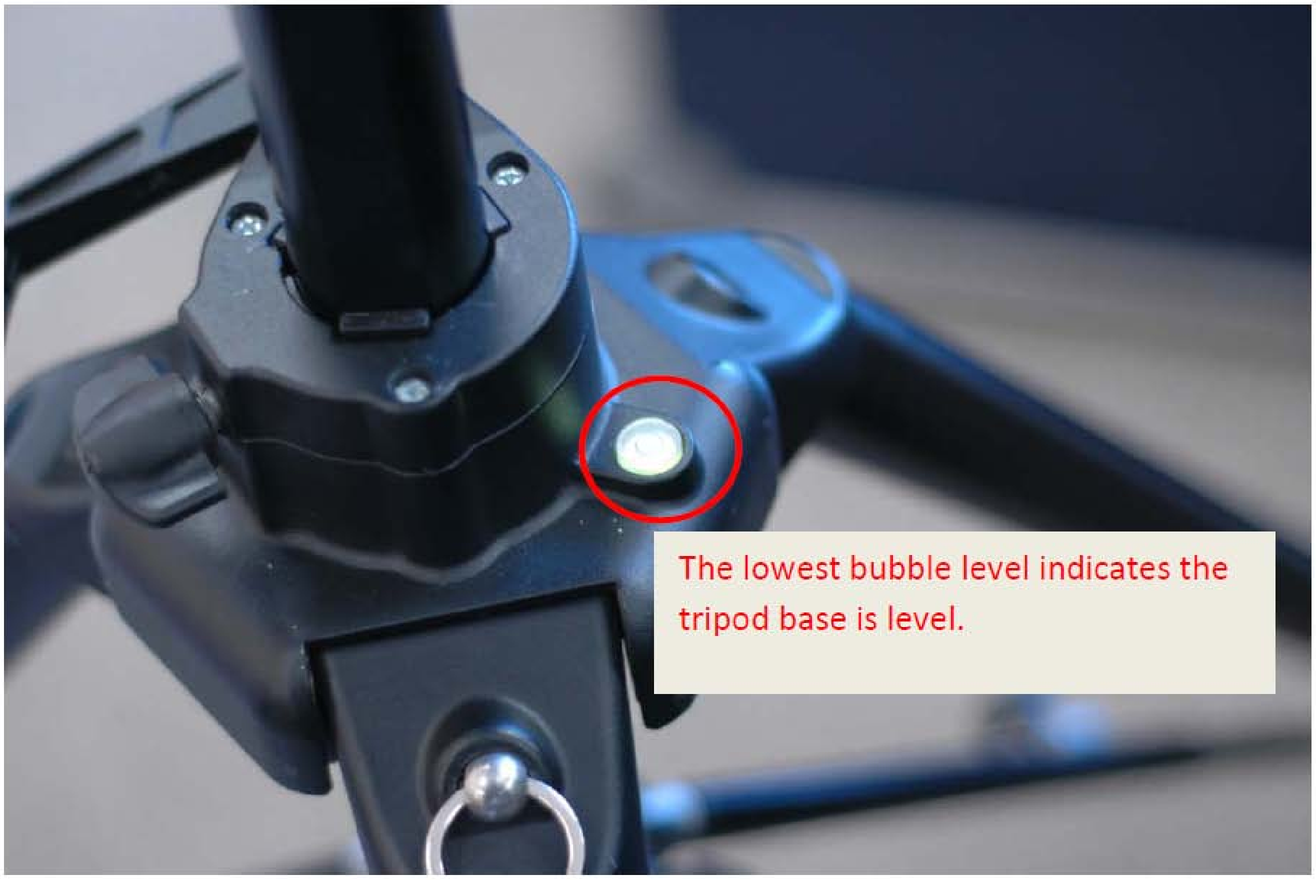


Fig.3: The lowest bubble level

If you are uncertain about which leg to adjust first, imagine drawing a line down the middle of each leg of the tripod extending through the bubble level. The line that passes closest to the center of the bubble is the leg you should adjust first. In this illustration where A, B and C mark the positions of the legs you should begin by adjusting leg C. To make the bubble move toward the center, level circle, leg C should be lengthened.

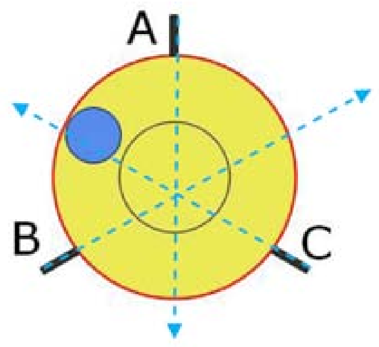


Fig.4: To move the bubble toward you parallel to any leg of the tripod, lengthen that leg. To move the bubble away from you, shorten that leg.

Continue adjusting the length of the legs following this logic until the tripod base is level.

ADJUST TRIPOD HEIGHT

5. Set the height of the tripod head so that it is roughly level with the center of the calibration chart by flipping out and then turning the handle, as shown below.

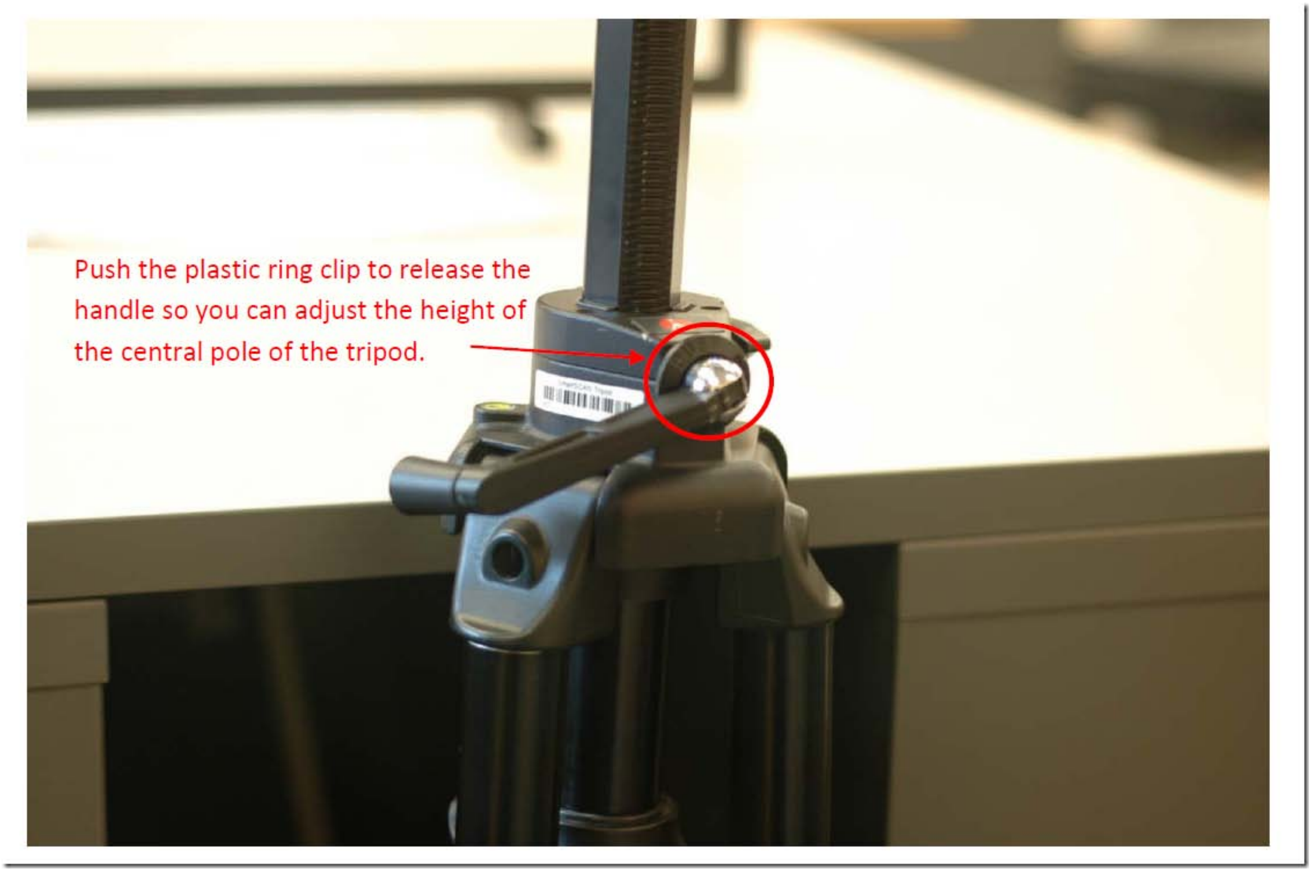

Push the plastic ring clip to release the
handle so you can adjust the height of
the central pole of the tripod.

Fig.5: The central pole height is adjusted by turning this handle.

LEVELING TRIPOD

6. Next, you want to adjust the tripod so that the second lowest level bubble is also within its level circle. This step is a little fiddly. The second level bubble essentially accounts for any deviations from level between the central pole, the bracket which attaches it to the tripod legs, and the tripod head. The goal is for the central pole to be just that – centered in the middle of the tripod base. Begin by checking that the tripod head is seated flat on the central pole. Loosen the two wingnuts that lock the position of the central pole. Shift the position of the central pole until the level bubble is within the circle and tighten the wingnuts. Be sure to check that this bubble is still indicating the tripod head is level after you attach the scanner to the tripod!

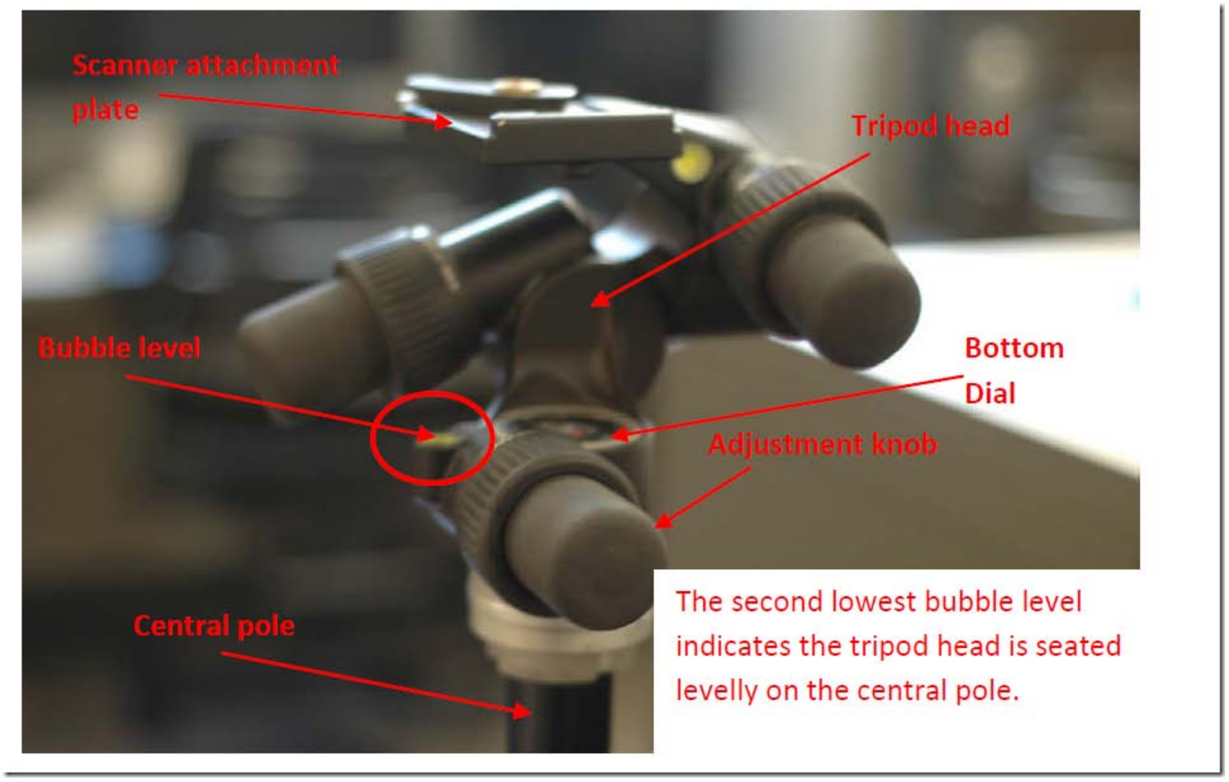

Scanner attachment plate
Tripod head
Bubble level
Bottom Dial
Adjustment knob
Central pole
The second lowest bubble level indicates the tripod head is seated levelly on the central pole.

Fig.6: The tripod head.

LEVEL TRIPOD 2- LOWEST ADJUSTMENT KNOB

7. Use the lowest adjustment knob to align the front edge of the scanner attachment plate parallel to the edge of the desk or table. On the CAST tripod, this should be approximately at the 30° mark on the bottom dial.

8. Finally, adjust the tripod head so that the top bubble is centered in the inner circle of the bubble level. The highest adjustment knob moves the scanner attachment plate along its long axis, the second highest adjustment knob moves the scanner attachment plate along its short axis.

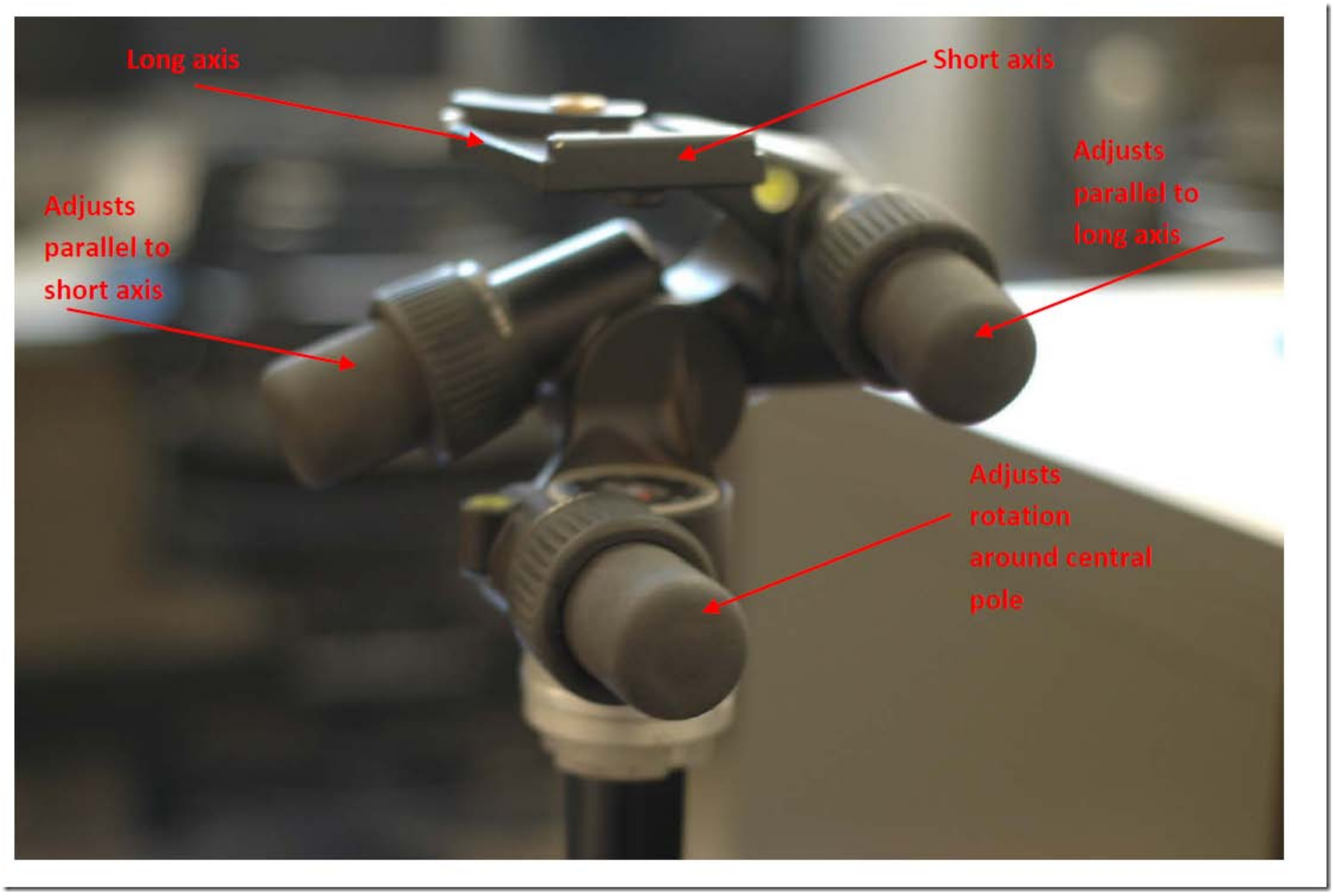

Long axis
Short axis
Adjusts parallel to short axis
Adjusts parallel to long axis
Adjusts rotation around central pole

Fig. 7: Tripod head adjustment guide.

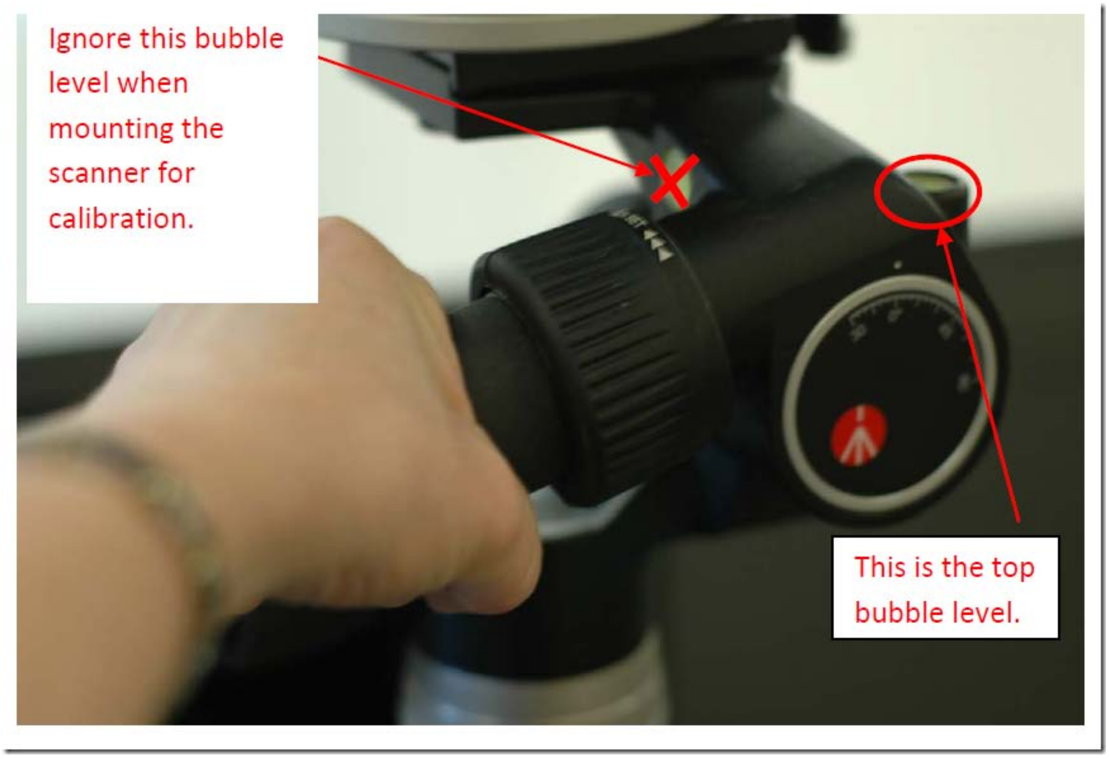

Ignore this bubble level when mounting the scanner for calibration.
This is the top bubble level.

Fig. 8: The top bubble level.

LEVEL TRIPOD 3- SIDE DIALS

In addition to the top level bubble, you can use the two side dials as guides for setting the level of the tripod head. On flat ground both these dials should read approximately 0°. There is a small silver dot above each dial marking the 0° position.

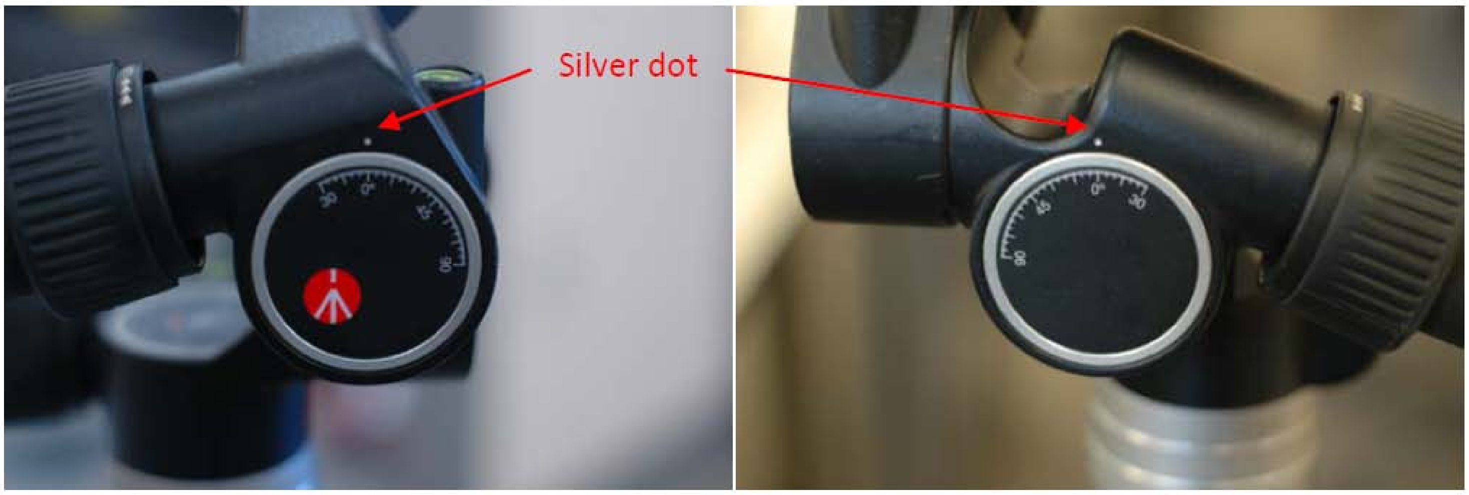


Fig.9: Side dials on the tripod head.

COARSE ADJUSTMENTS

(left) Quick, coarse adjustments are made by twisting the 'quick grip' and then pivoting the tripod head. Be careful using the 'quick grip' if the scanner is already mounted on the tripod head, as the uneven weight of the scanner can cause it to pivot faster than you might expect. (right) Fine adjustments are made by twisting the knob below the 'quick grip'.

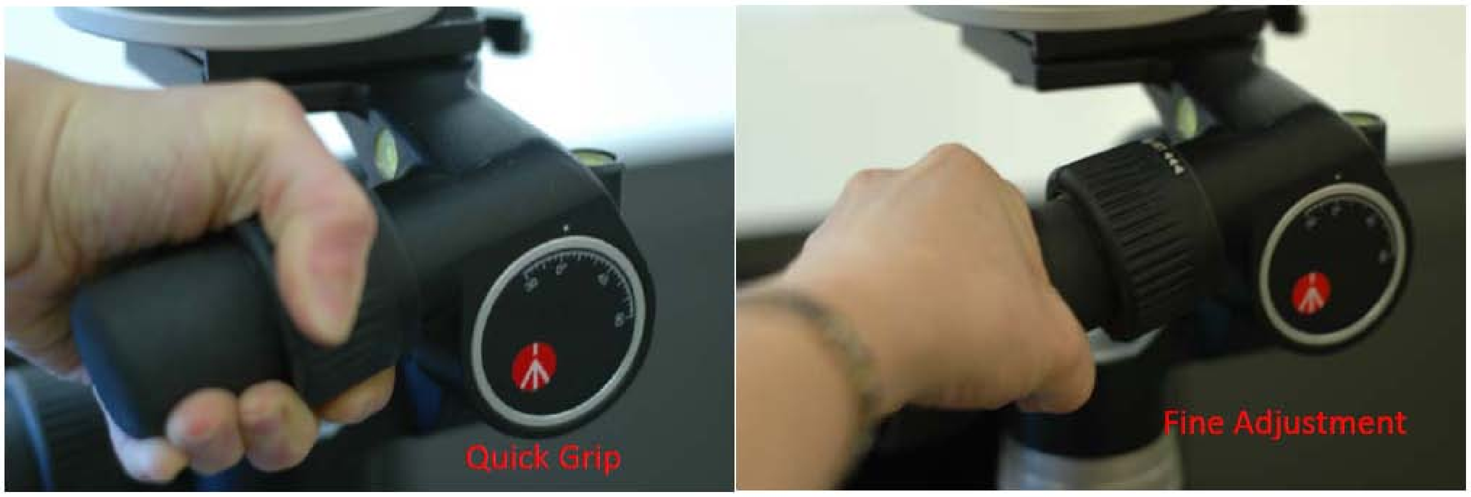


Fig.10: Quick grip and fine adjustment positions on the adjustment handles of the tripod head.

FINAL CHECK

9. After you have leveled the tripod, attach the scanner and check and adjust the level if needed. At this point you should only need to make fine adjustments.

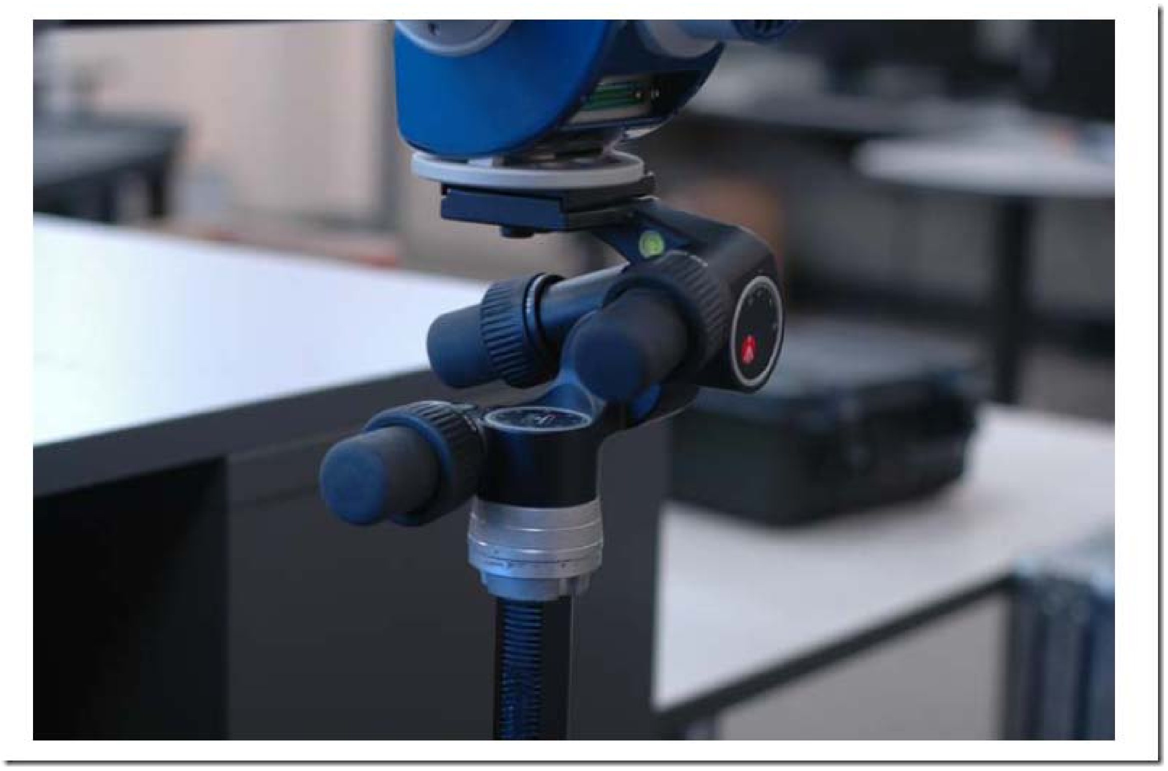


Fig. 11: The scanner mounted on the level tripod.

SCANNER SHEET IN SCANNER CASE

10. Once the scanner is positioned, put the chart placement guide on the table. Position the guide so its short edge is parallel to the edge of the table, and the side reading “SENSOR ” is pointed toward the scanner. Check the datasheet for the correct distance between the scanner and the center of the chart placement guide (approximately one meter). The datasheet and chart placement guide are stored in the lid of the scanner’s case.

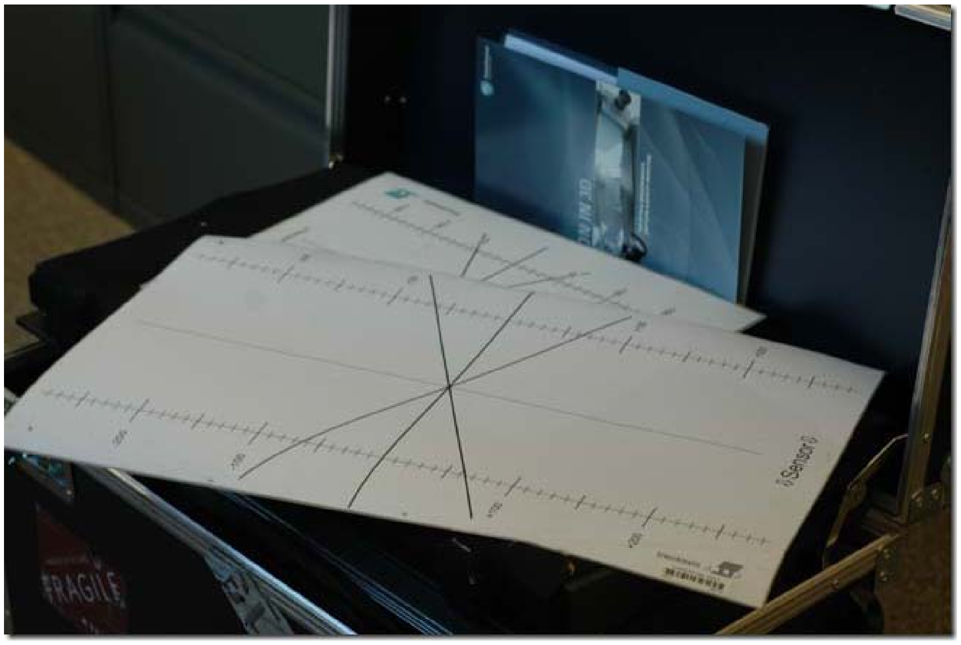
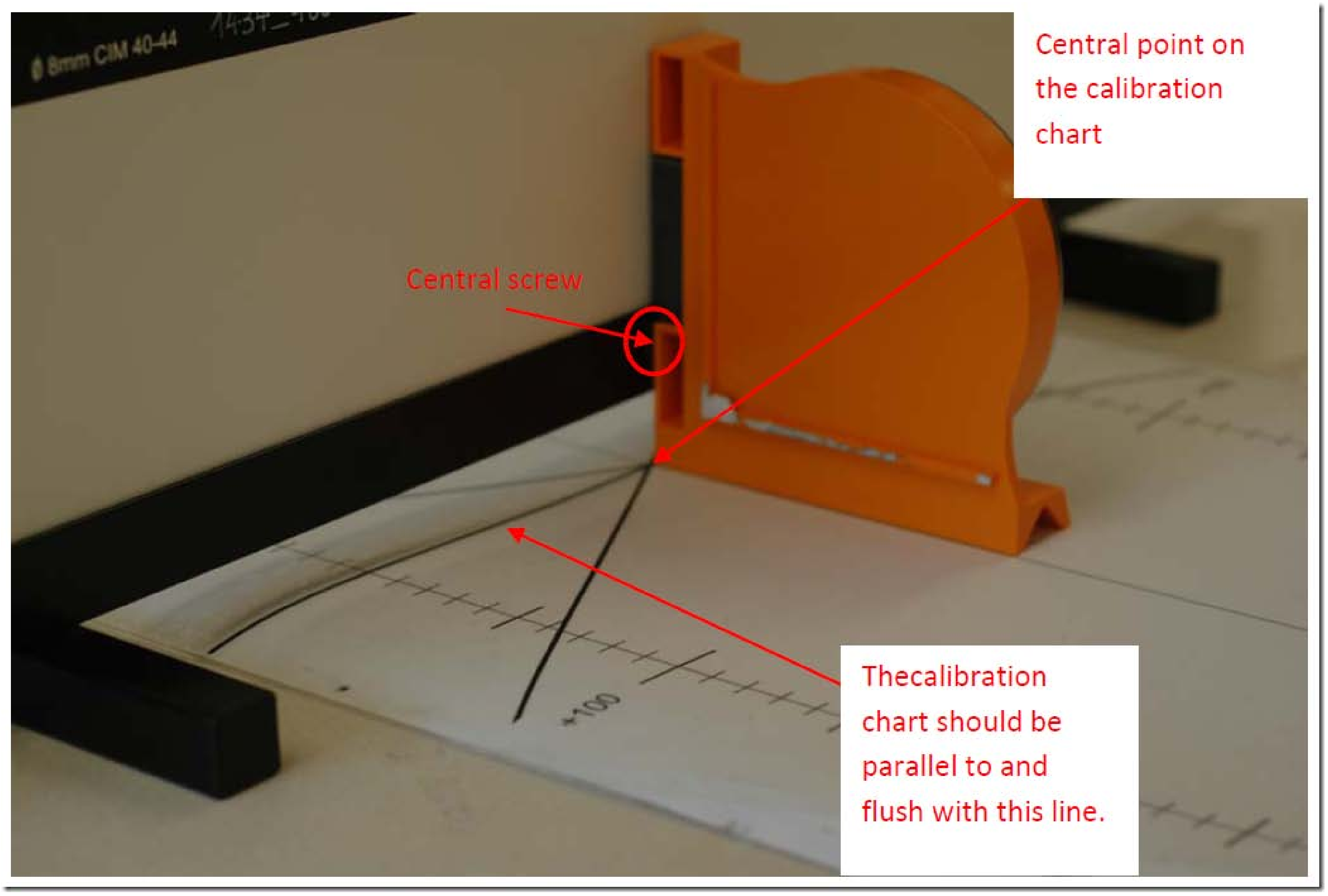


Fig.12: The datasheet and chart placement guide in the scanner case.

CHART PLACEMENT

11. Place the Chart over the placement guide and center it. There is a screw in the center of the chart's frame. This screw should be directly over the center line of the chart. You will probably want to use a
straight edge to ensure the calibration chart is parallel and flush with the center line on the placement guide.


Central point on the calibration chart
Central screw
Thecalibration chart should be parallel to and flush with this line.
8mm CIM 40-44
+100

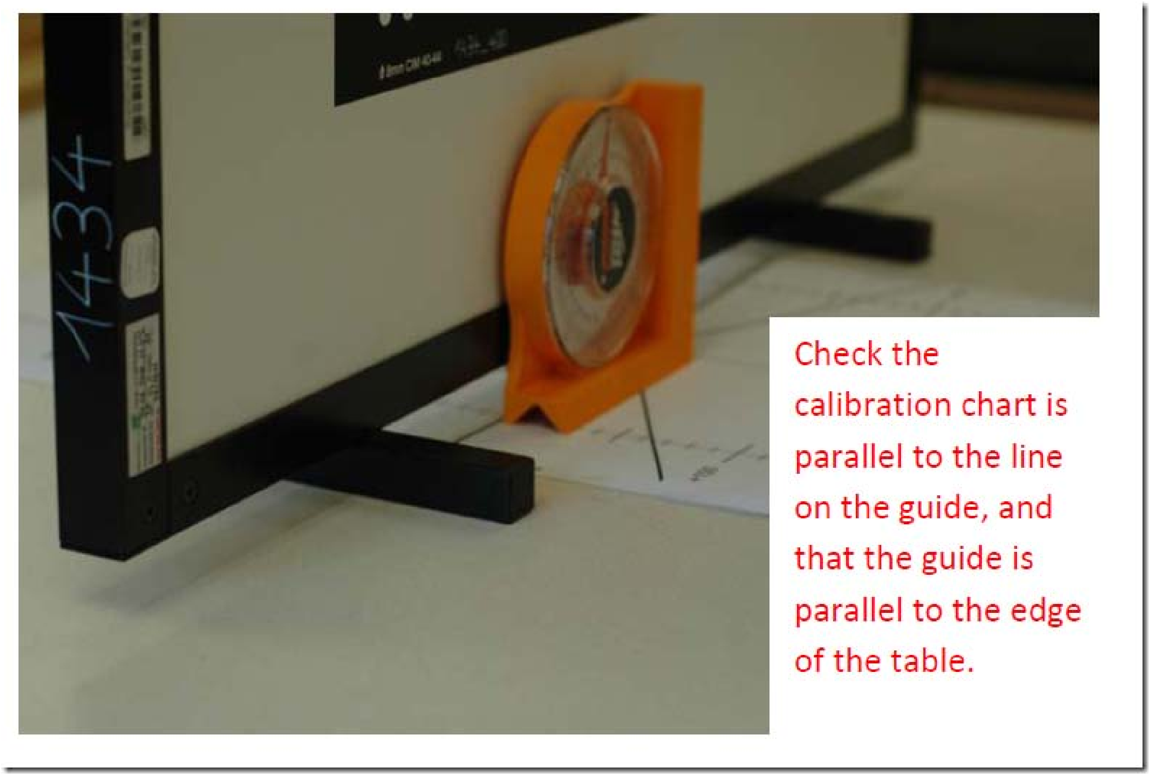

Check the calibration chart is parallel to the line on the guide, and that the guide is parallel to the edge of the table.

Fig. 13: Aligning the calibration chart to the chart placement guide.

When everything is aligned, tape the chart placement guide to the table so it won't move as you shift the calibration chart.

12. Alignment and leveling is complete!

13. You may now proceed with the calibration. (You may make some further fine adjustments to the height of the scanner and position of the chart once you have activated the lasers and live screen in the calibration.)

CONTINUE TO...

Continue to Section XXXXXX

]]> http://gmv.cast.uark.edu/scanning/guide-to-leveling-and-aligning-the-breuckmann-tripod-and-smartscan-he-for-calibration/feed/ 0 http://gmv.cast.uark.edu/scanning/software/leica-software/leica-cyclone/cyclone-workflows/leica-cyclone-creating-a-mesh-and-modeling-surfacetopography-3/ http://gmv.cast.uark.edu/scanning/software/leica-software/leica-cyclone/cyclone-workflows/leica-cyclone-creating-a-mesh-and-modeling-surfacetopography-3/#comments Tue, 10 Apr 2012 16:47:09 +0000 Angie 5 meters]]> http://gmv.cast.uark.edu/1787/leica-cyclone-creating-a-mesh-and-modeling-surfacetopography/

This series will show you how create a mesh and model surface topography in Leica's Cyclone
Hint: You can click on any image to see a larger version.

INTRODUCTION AND DEFINITIONS

In this guide, **a mesh** is considered to be a series of triangles that represents a surface. Cyclone generates meshes by using the points in a point cloud, vertices, polylines, or any combination of the three as vertices. For each adjacent trio of points in a cloud, a triangle is created. This has the effect of creating a visually coherent surface from the point cloud, and it is the primary method for modeling topography.

Break lines are lines or polylines that sub-divide the mesh; they represent the edges of a paved surface, a ridge, a channel, or any other topographic feature that the user wants to preserve. While mentioned in 2D topographic tracing, break lines become very important in meshing. Here, they have an accurate z-coordinate and they become the edges that the triangles in the mesh conform to, defining and controlling the smoothness and continuity of the mesh. They allow the user to break-up the mesh into reasonable “chunks” for future texturing and detailing.

Please see ‘Leica Cyclone – Creating a Basic CAD Ojects From Surface Topography (2D)’ as a supplement to this workflow.

OPEN A REGISTERED UNIFIED MODEL SPACE

1. Open a Registered Unified Model Space -> Create fence around ground plane –> Right Click –> Copy Fenced to new Model Space (NOTE: viewing from a standard side, front, or back view in orthographic mode assists in selection) -> Original MS may be closed

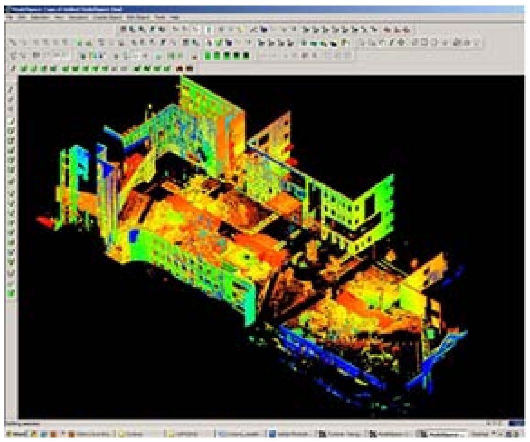
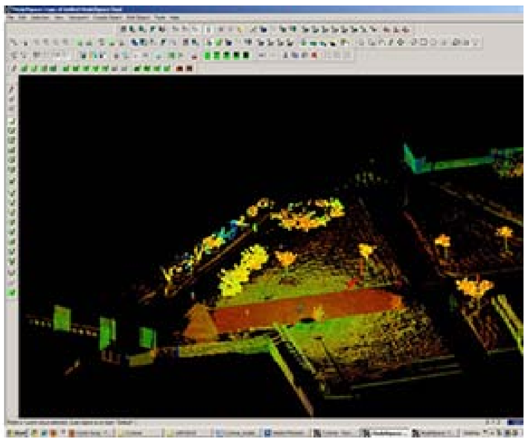



Figure 1 – (Left) Original registered scan world of plaza (Right) Point Cloud Sub-Selection (Select -> Right Click -> Point Cloud Sub-Selection) allows unneeded points, such as trees and vertical surfaces, to be deleted; Sub-selection allows the user to precisely choose and view points before deciding to delete

SELECT AND DELETE UNNEEDED POINTS

2. In the new Working MS -> Select and delete unneeded points (it’s best to eliminate as much vertical surface data as possible so that the ground plane to be modeled is isolated; objects in motion, such as trees/vegetation can be especially problematic and cleaning up areas where these are abundant is suggested; eliminating data that may be present inside buildings or areas that will not be modeled is also suggested)

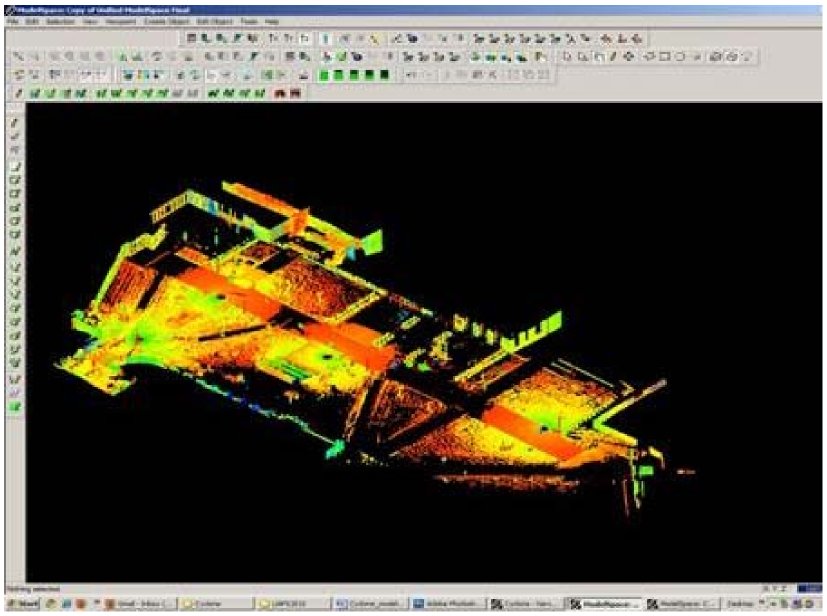


Figure 2 – The same plaza as Figure 1 now copied to a working MS and “cleaned” of unneeded vertical surfaces and vegetation leaving only the ground plane to be modeled.

CREATE NEW LAYERS FOR THE BREAKLINES AND FEATURES

3. Create new layers for the breaklines and features (Shift + L) -> Review the area to be modeled identifying where the surface changes and/or where the user wants a clean break or difference between adjacent surfaces.Create layers for primary and secondary features as needed.

4. Create lines to represent the topography and features, assigning an accurate x, y, and z coordinate -> There are several ways to make lines including (1) using 2D drawing methods (covered in 'Topography and Site Elements – 2D' above) (2) with pick points, (3) by extending an existing line (4) by snapping two existing lines together (5) by creating a polyline with Fit Edge command, and (6) by merging polylines together.

See Help -> Contents -> Search -> 'Make Lines, Polylines, and Breaklines' for steps for each of these methods.There are additional variations of each.BEWARE: Lines created by pick points cannot be used for break lines in meshing; if picking points, be sure to create a polyline rather than a line segment.

TIPS

5. Tips

A. If drawn in 2D, use Edit Object -> Move/Rotate to move the object to the correct z-coordinate; there are multiple options moving objects including by a standard axis and pick points.

B. Create additional break lines following the feature or edge of the break lines; place these to break up the mesh and around the boundaries, segmenting out areas where a topographic ground plane is not needed (ie: inside buildings)

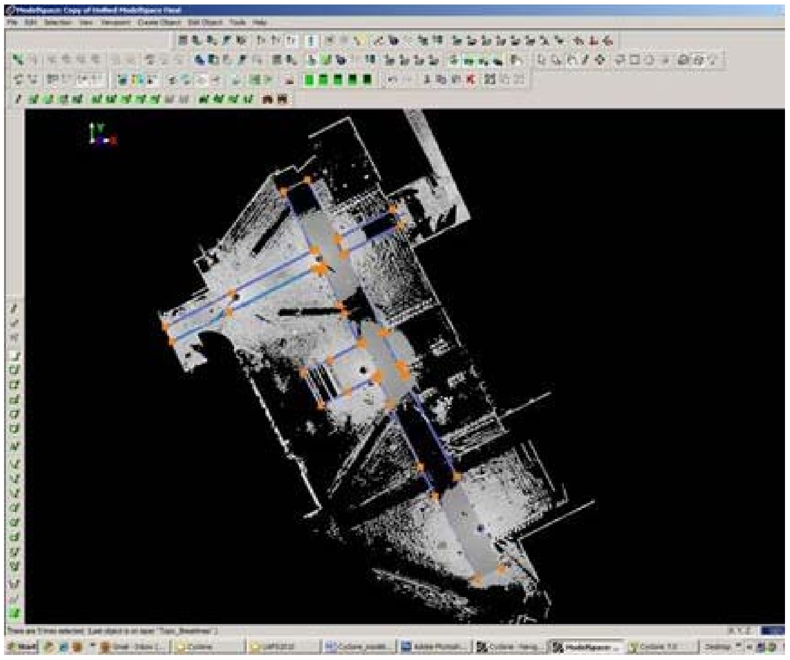


Figure 3 – Top view of plaza; Polylines have been created to outline where the sidewalk and grass meet; orange handles help highlight the polylines

MORE TIPS

More Tips:

C. Use multiple views to confirm line is properly located at all angles and handle constraints (Edit Object -> Handles constrain to -> Various options – NOTE: once placed in top view, constraining handles to z-direction helps placement in 3D)

D. Try different methods to see what works for you.With all, use multiple views to confirm the line is properly located at all angles

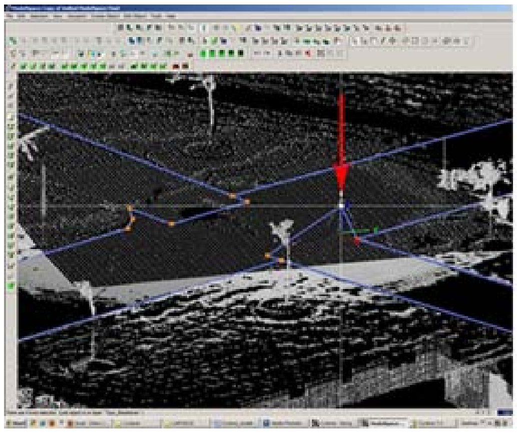
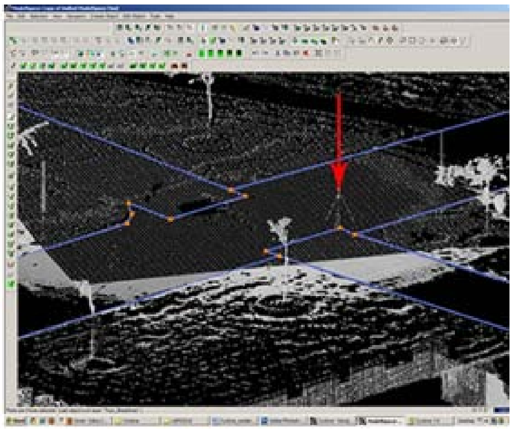



Figure 4 – (Left) Although from Top View, the breakline looks correct, inspection from a different angle reveals that it is not correct (Right) Handles and constraints (Edit Object -> Handles -> Constrain Motion) are used to pull the polyline into the correct position – every section of every breakline should be inspected from multiple views

FINAL STEPS

6. Create handles at places where polylines intersect (Select line -> ALT + Select point on line for new handle to help snap lines together).

7. Delete all points beyond boundary of mesh -> Select polyline representing boudary of mesh -> Right click -> Fence -> From Selection -> Fence -> Delete outside

8. Delete points within other breaklines to remove any remaining points that should not be considered in the creation of the mesh -> Copy final breaklines to original or temporary working space in case altered/needed in future

8. Unify points reducing spacing to 1 foot to start and adjust settings for desired results.

CONTINUE TO...

Continue to Creating the Topographic Mesh
]]> http://gmv.cast.uark.edu/scanning/software/leica-software/leica-cyclone/cyclone-workflows/leica-cyclone-creating-a-mesh-and-modeling-surfacetopography-3/feed/ 0