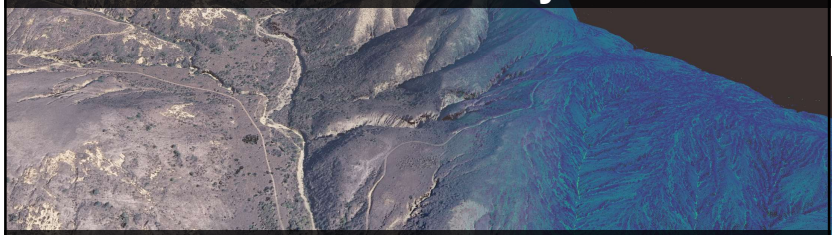
From point clouds and full-waveform data to DEM analysis



Ramon Arrowsmith, Chris Crosby, Fiona Clubb, Aljoscha Rheinwalt, Taylor Smith, Bodo Bookhagen

From point clouds and full-waveform data to DEM analysis - Schedule

Primary Source of information: github page @ https://github.com/UP-RS-ESP/PointCloudWorkshop-Oct2019

This will include all exercises, links to data and sources codes. Etherpad links (https://yourpart.eu/) to source codes and writings will be posted on github.

From point clouds and full-waveform data to DEM analysis - Schedule

Monday, Sept 30, 2019 start 9am

- Introduction to Point Clouds (Bodo)
  - Desktop based (Python driven) computational workflows (I) (Bodo + Aljoscha)
  - CloudCompare and aligning different point cloud datasets
  - Importing LAZ/LAS files into python: airborne lidar to photogrammetric SFM warping example PDF manual, python code
- OpenTopography introduction and production implementations (Ramon and Chris)
  - Point cloud selection and review of basic processing workflows
  - DEM processing using neighborhood and triangulation methods to produce DSM and DTMs
  - DEM processing for flow routing using TauDEM HPC
- 4 - 5pm *Evening student presentations*

From point clouds and full-waveform data to DEM analysis - Schedule

Tuesday, October 1, 2019

- Desktop based (Python driven) computational workflows (II) (Aljoscha + Bodo)
  - Point cloud classification and filtering
  - interpolation to grids using the most appropriate methods (from triangulation to IDW and fitting green's functions and splines and other surfaces) github: Lidar_PC_interpolation
  - Efficient computation with point clouds using python, cython, and KDTrees (Example for generating DEMs from dense pointclouds) github: LidarPC-KDTree
  - Other topics depending on time and participant's interests
- 4 - 5 pm *Evening student presentations*

From point clouds and full-waveform data to DEM analysis - Schedule

Thursday, October 3, 2019

- Topographic analysis with LSD TopoTools (Fiona)
  - Introduction LSDTT PDF manual
  - River-profile clustering github repository and Jupyter notebook and tutorial, publication
  - Ridge-top curvature
  - Making appealing maps with GMT

From point clouds and full-waveform data to DEM analysis - Schedule

Friday, October 4, 2019

- Full waveform lidar explanation and exploration with specific applications for geomorphology (Bodo + Aljoscha)
  - micro-surface roughness
  - better characterization of ground (and other) surfaces
  - material-surface characteristics
  - water-column characteristics
  - biomass estimation

LiDAR – Why use a laser?

ALS (Airborne Lidar System) take advantage of two of the unique properties of laser light:

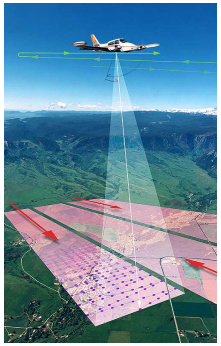
1. The laser is **monochromatic**. It is one specific wavelength of light. The wavelength of light is determined by the lasing material used, usually 532 or 1064nm.

Advantage: We know how specific wavelengths interact with the atmosphere and with materials.

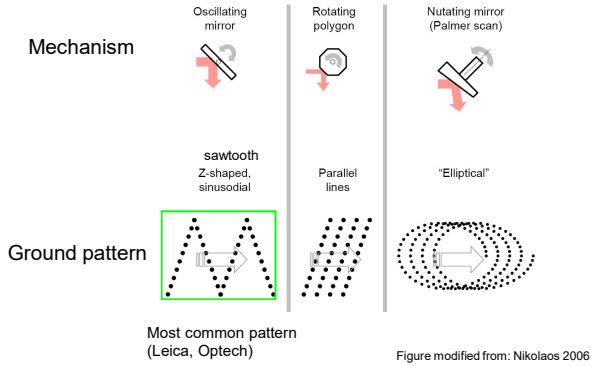
2. The light is accurate and **directional**. A laser has a very narrow beam which remains concentrated over long distances. A flashlight (or Radar) on the other hand, releases energy in many directions, and the energy is weakened by diffusion.

Advantage: The beam maintains its strength over long distances.

3mrad divergence = 30 cm at 1 km and 1.5m at 5 km



Scanning Mechanisms



Most common pattern (Leica, Optech)

Figure modified from: Nikolaos 2006

LIDAR – Two distinct types of ALS systems

Discrete-return systems (a.k.a. small-footprint)

SAMPLES the returned energy from each outgoing laser pulse in the vertical plane (Z) *(if the return reflection is strong enough)*

Most commercial lidar systems are discrete return, many different types are available

Waveform systems (a.k.a. large-footprint)

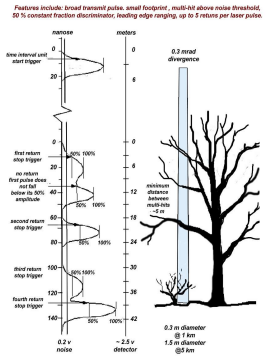
Records the **COMPLETE** range of the energy pulse (intensity) reflected by surfaces in the vertical dimension

Samples transects in the horizontal (X,Y) plane

Waveform systems designed to capture vegetation information are not widely available

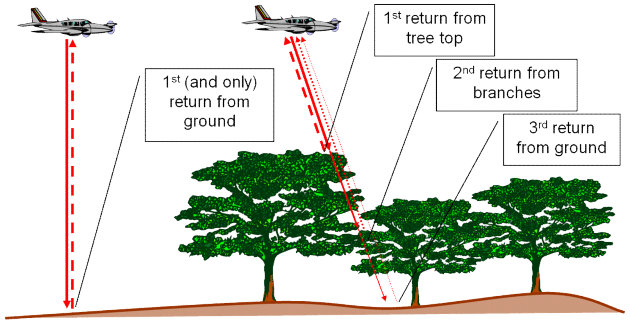
Waveform systems include SLICER, LVIS, ICESat

Discrete Return LIDAR

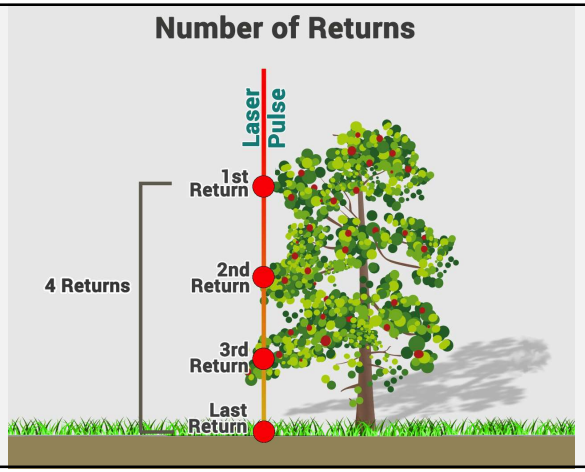


- Records data as X,Y,Z points
- Spatial resolution is expressed in terms of "post spacing" which is the avg. horizontal distance between points
- However, you *can* have multiple returns within the same pulse
- Returns are "triggered" if the laser reflects from a surface large enough to exceed a pre-set energy threshold
- New capability to record the intensity of point returns.

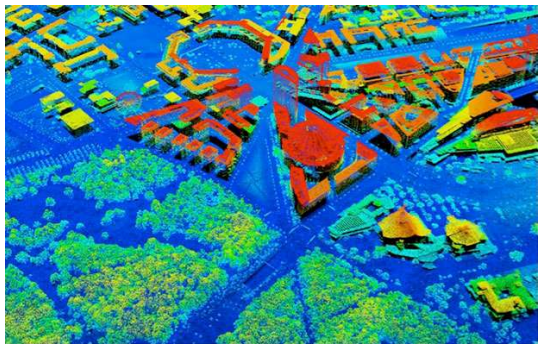
Point-cloud Returns



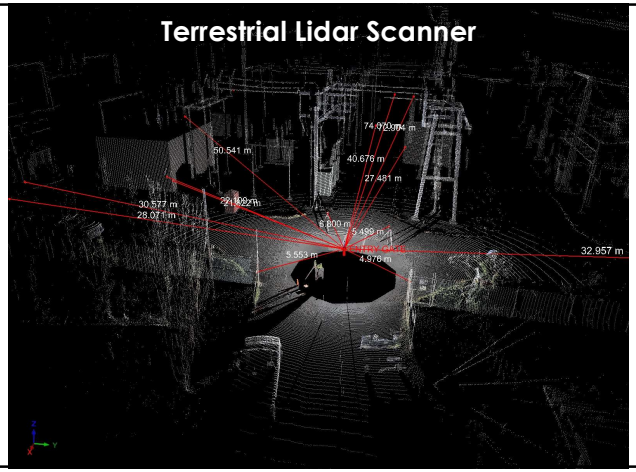
Number of Returns



Discrete return data: Millions of X, Y, Z points

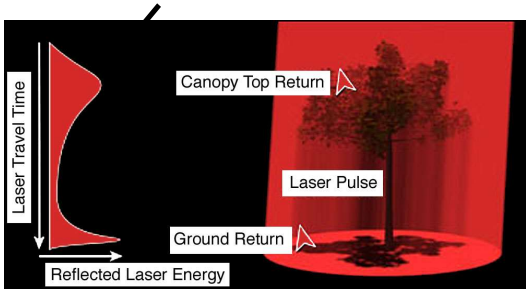


Terrestrial Lidar Scanner



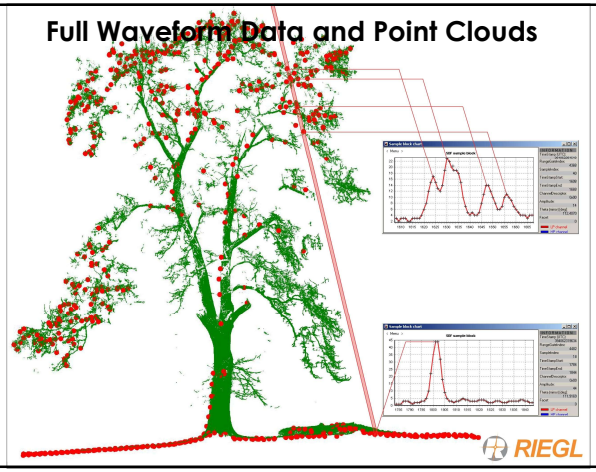
Full-Waveform LIDAR

This is the "waveform" graph of energy intensity



Notice that it follows the "shape" of the tree biomass

Full Waveform Data and Point Clouds



LiDAR Data

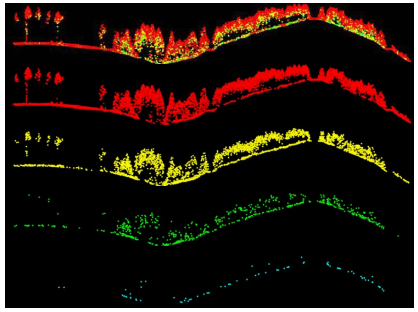
All returns

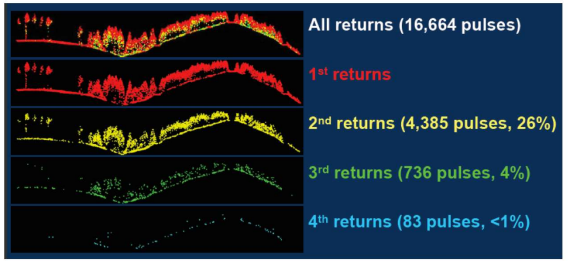
1st return

2nd return

3rd return

4th return





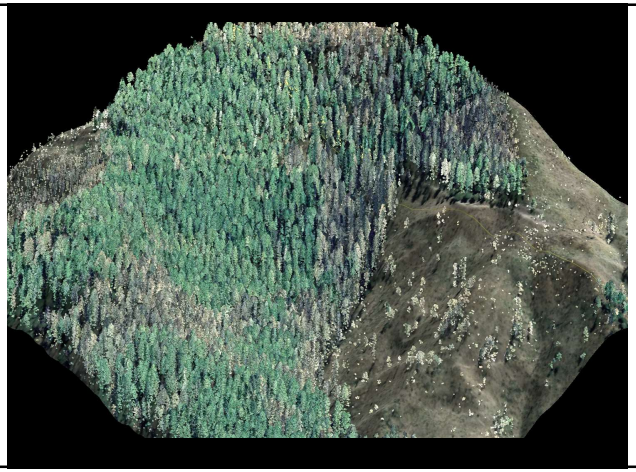
DEMs typically are made from last-returns, which **do not necessarily mean ground** but do have higher probability of reflecting from ground surface.

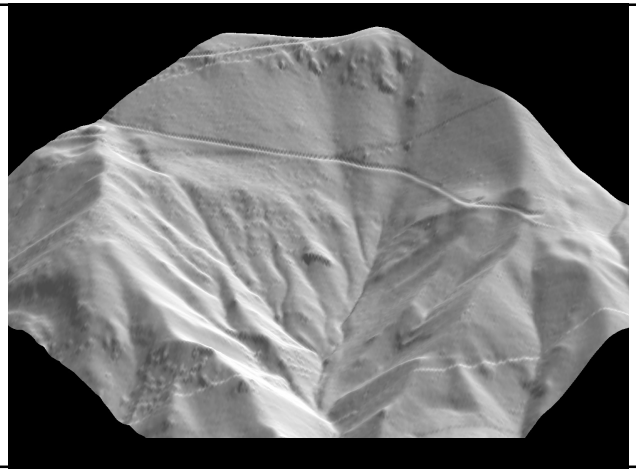
In this example, last-returns can be 2nd, 3rd, or 4th return.

*Note: pulses with only 1-return are considered a last-return

Image modified from USGS; Hans-Erik Anderson

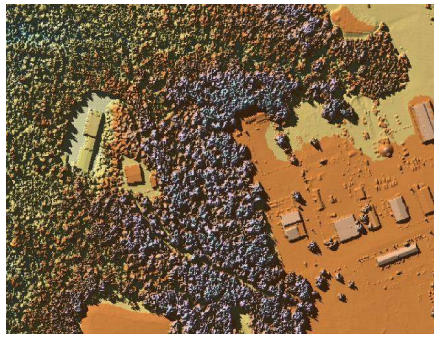
17





Bare Earth DEM Calculations

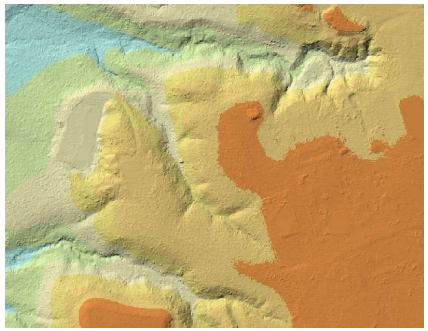
1. Identify man made structures
   - Determine ground elevation at edges
2. Identify vegetated areas
   - Search for ground points within area
3. Identify open areas
4. Interpolate between available points
5. Filtering and clean-up



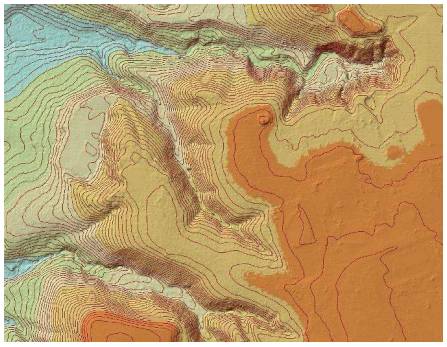
First Return DEM

Automatically extracted Bare Earth DEM

- **Ground surface with vegetation and man-made structures removed**
- **Critical for terrain and hydrology modeling**



Bare Earth DEM



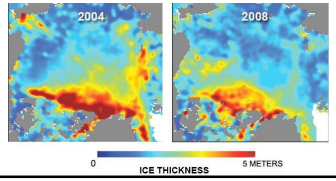
Bare Earth DEM with 1 meter contours

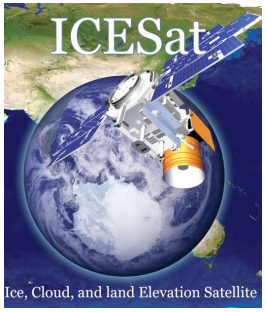
Spaceborne LIDAR: ICESat and ICESat2

Spaceborne waveform LIDAR systems

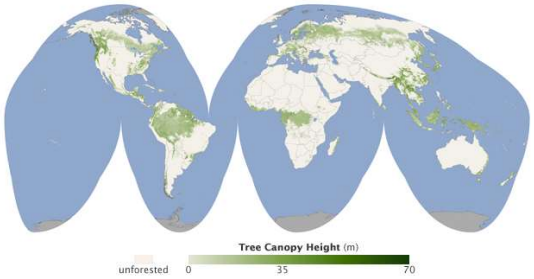
ICEsat launched in 2003 for survey of topography and polar ice, but stopped working in 2010

ICESat 2 is planned for 2016



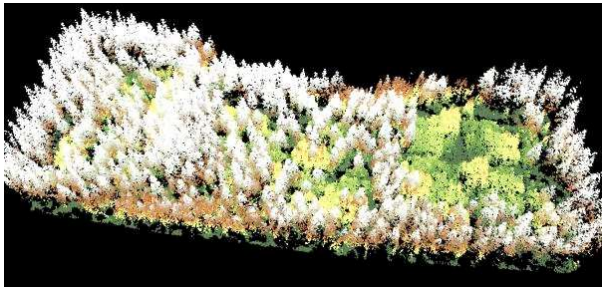


Global Tree Heights from Lidar

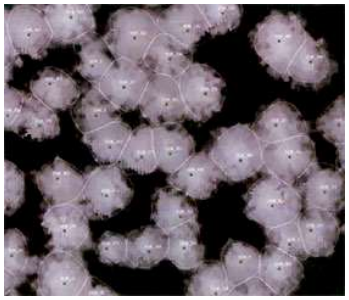


Developed with ICESat data by Michael Lefsky (2010)
http://www.nasa.gov/topics/earth/features/forest-height-map.html

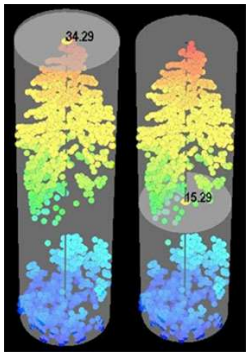
Lidar Applications

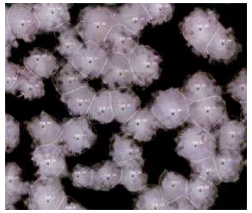


Lidar Applications

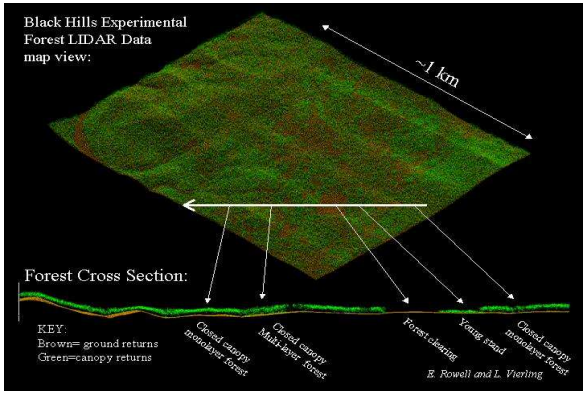


Lidar Applications

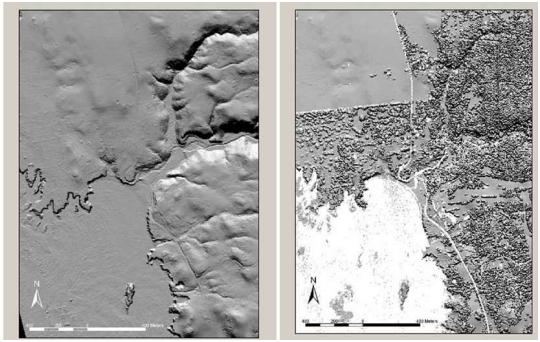




Lidar Mapping and Forest Structure



Point-Cloud Classifiations: DEMs: Bare Earth and Canopy



Point cloud classification

ASPRS:
American Society for Photogrammetry and Remote Sensing

Classification Value (Levels)	Meaning
0	Created, Never Classified
1	Unclassified 1
2	Ground
3	Low Vegetation
4	Medium Vegetation
5	High Vegetation
6	Building
7	Low Point (noise)
8	Model Key Point (mass point)
9	Water
10	Reserved for ASPRS Definition
11	Reserved for ASPRS Definition
12	Overlap Points 2
13-31	Reserved for ASPRS Definition

Point cloud classification

ASPRS:
American Society for Photogrammetry and Remote Sensing

Table 17: ASPRS Standard LIDAR Point Classes (Point Data Record Formats 6-10)

Classification Value	*Meaning*
0	Created, never classified
1	Unclassified[1]
2	Ground
3	Low Vegetation
4	Medium Vegetation
5	High Vegetation
6	Building
7	Low Point (noise)
8	Reserved
9	Water
10	Rail
11	Road Surface
12	Reserved
13	Wire – Guard (Shield)
14	Wire – Conductor (Phase)
15	Transmission Tower
16	Wire-structure Connector (e.g. Insulator)
17	Bridge Deck
18	High Noise
19-63	Reserved
64-255	User definable

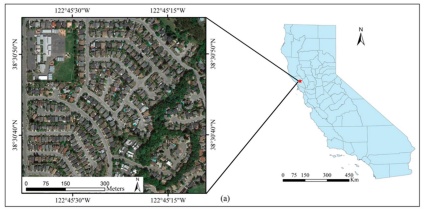
Official document describing LAS Format 1.4 and classes:
https://www.asprs.org/a/society/committees/standards/LAS_1_4_r13.pdf

Point storage and data

Table 18: Point Data Record Format 7

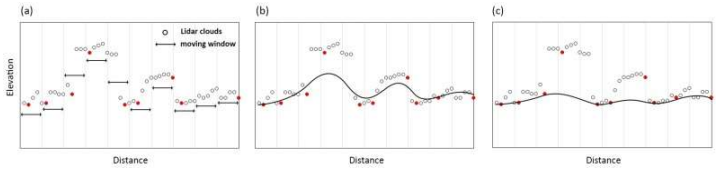
Item	Format	Size	Required
X	long	4 bytes	*
Y	long	4 bytes	*
Z	long	4 bytes	*
Intensity	unsigned short	2 bytes	
Return Number	4 bits (bits 0, 1, 2, 3)	4 bits	*
Number of Returns (given pulse)	4 bits (bits 4, 5, 6, 7)	4 bits	*
Classification Flags	4 bits (bits 0 – 3)	4 bits	
Scanner Channel	2 bits (4-5)	2 bits	*
Scan Direction Flag	1 bit (bit 6)	1 bit	*
Edge of Flight Line	1 bit (bit 7)	1 bit	*
Classification	unsigned char	1 byte	*
User Data	unsigned char	1 byte	
Scan Angle	short	2 bytes	*
Point Source ID	unsigned short	2 bytes	*
GPS Time	double	8 bytes	*
Red	unsigned short	2 bytes	*
Green	unsigned short	2 bytes	*
Blue	unsigned short	2 bytes	*

Point-cloud Classification



Ao, Z. et al. (2017): One-class Classification of …, Remote Sensing, 9(10)

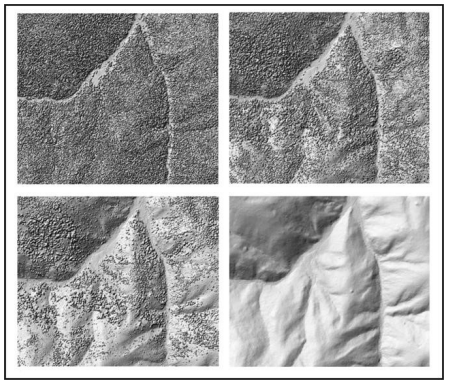
Surface-based ground classification



Schematic diagram of surface-based DTM generation methods. (**a**) A lowest point is selected for each cell; (**b**) A coarse surface is produced based on these pre-selected points; (**c**) A refined DTM is generated based on the residue between the coarse surface and the elevation of rest points.

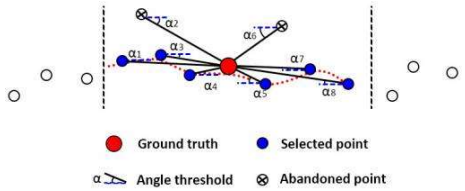
Chen, Z., Gao, B., & Devereux, B. (2017). State-of-the-Art: DTM Generation Using Airborne LIDAR Data. *Sensors (Basel, Switzerland)*, *17*(1), 150.

Getting Down to the Ground



Progressive Curvature Filter (Evans and Hudak 2007)

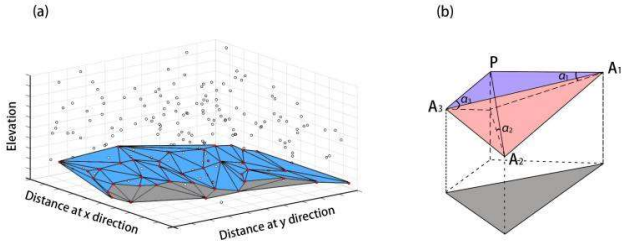
Morphology-based ground classification



Schematic diagram of morphology-based DTM generation method: If the slope between the ground point and a candidate point is smaller than a (global or local) slope threshold, then this candidate point is set as a ground point. Otherwise, this candidate point is set as a non-ground point.

Chen, Z., Gao, B., & Devereux, B. (2017). State-of-the-Art: DTM Generation Using Airborne LIDAR Data. *Sensors (Basel, Switzerland)*, *17*(1), 150.

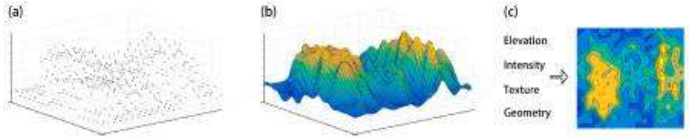
TIN/Morphology-based ground classification



Schematic diagram of morphology-based DTM generation method. (**a**) Some of the lowest points are selected as preliminary ground points and form a coarse TIN; (**b**) Rest points are examined using triangles within this TIN model. (used in TerraScan and lasground)

Chen, Z., Gao, B., & Devereux, B. (2017). State-of-the-Art: DTM Generation Using Airborne LIDAR Data. *Sensors (Basel, Switzerland)*, *17*(1), 150.

Segmentation-based ground classification

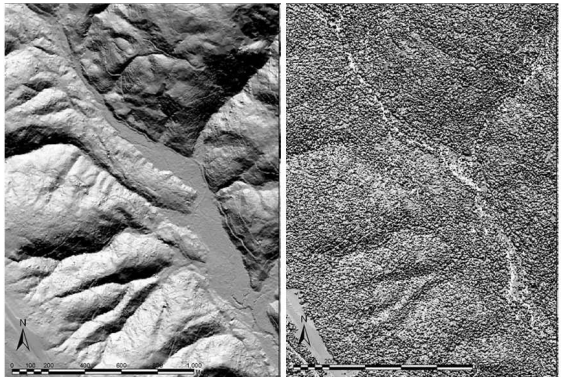


Schematic diagram of segmentation and classification-based DTM generation method. (**a**) Raw Lidar points; (**b**) Raster images produced using raw Lidar points; (**c**) After-image segmentation, unclassified segments are categorized into different land cover types by employing a diversity of features.

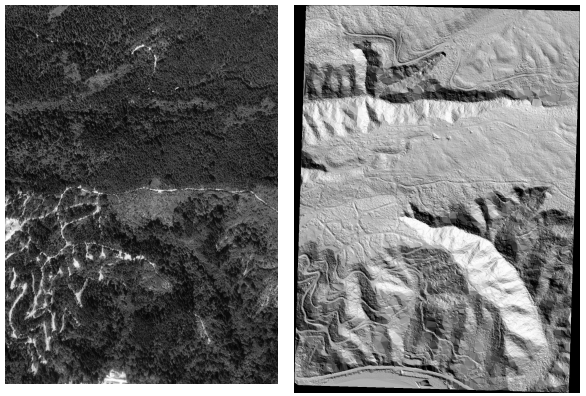
Chen, Z., Gao, B., & Devereux, B. (2017). State-of-the-Art: DTM Generation Using Airborne LIDAR Data. *Sensors (Basel, Switzerland)*, *17*(1), 150.

San Andreas Fault Zone

San Andreas Fault Zone



LIDAR Surfacing

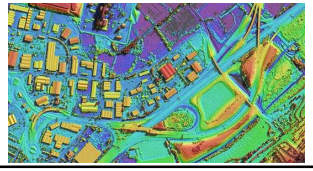


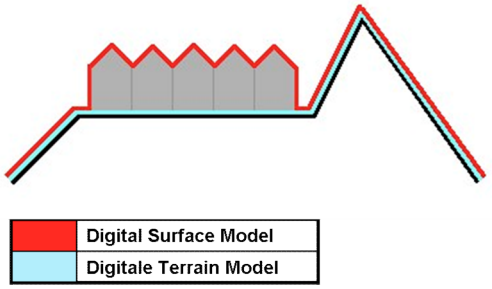
Digital Elevation Data

Definitions paraphrased from Maune et al, 2nd edition DEM Users Manual

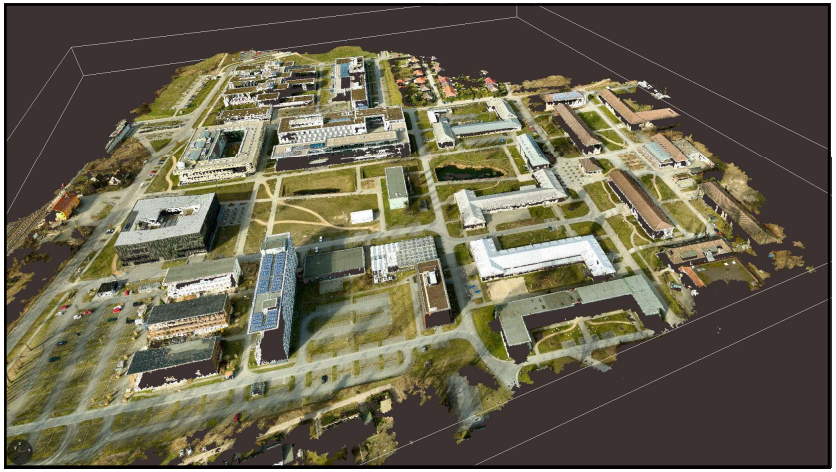
(Terms often used interchangeably)

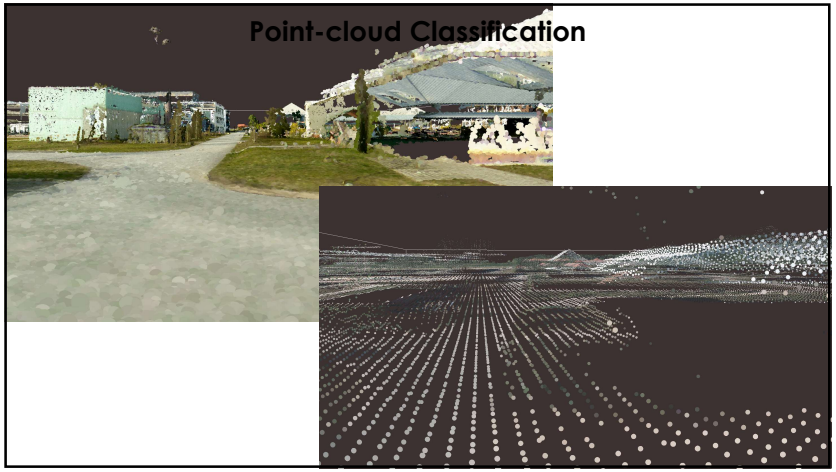
- DEM = Digital Elevation Model; Typically Bare Earth or terrain
- DTM = Digital Terrain Model; Similar to DEM with the addition of some elevations for significant topographic features on the land defined by mass points or break lines
- DSM = Digital Surface Model; Similar to a DEM or DTM, but shows the tops of all surfaces including buildings, trees, and other features above the bare earth

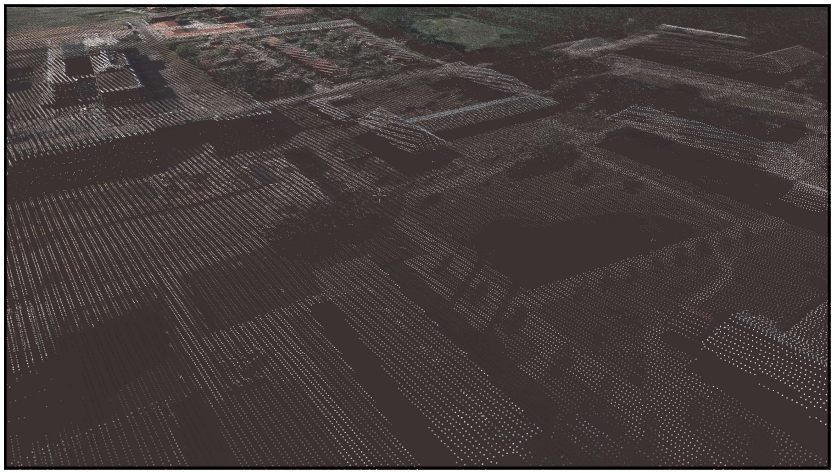


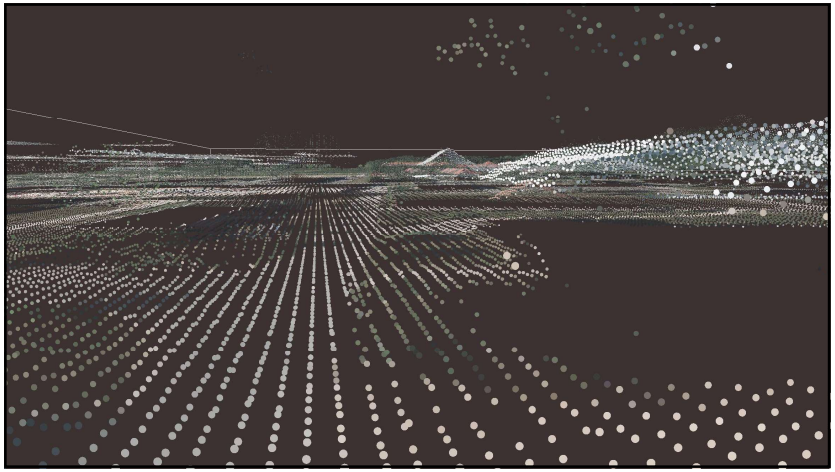


http://en.wikipedia.org/wiki/Digital_elevation_model



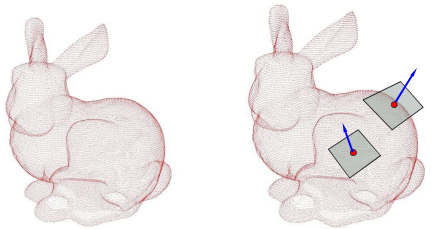

Point-cloud Classification





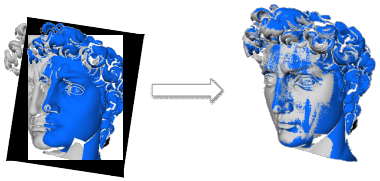
Point Clouds

- Simplest representation: **only points,** no connectivity.
- Collection of (x,y,z) coordinates, possibly with normals



Local Alignment

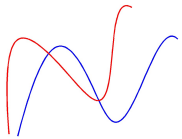
- Simplest instance of the registration problem



Given two shapes that are **approximately aligned** (e.g. by a human) we want to find the optimal transformation.

Iterative Closest Point (ICP)

- Approach: iterate between finding correspondences and finding the transformation:

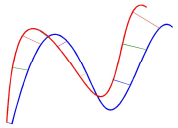


Given a pair of shapes, X and Y, iterate:

1. For each $x_i \in X$ find **nearest** neighbor $y_i \in Y$.
2. Find deformation $\mathbf{R}, t$ minimizing: $\sum_{i=1}^{N} \|\mathbf{R}x_i + t - y\|_2^2$

Iterative Closest Point (ICP)

- Approach: iterate between finding correspondences and finding the transformation:

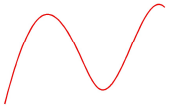


Given a pair of shapes, X and Y, iterate:

1. For each $x_i \in X$ find **nearest** neighbor $y_i \in Y$.
2. Find deformation $\mathbf{R}, t$ minimizing: $\sum_{i=1}^{N} \|\mathbf{R}x_i + t - y\|_2^2$

Iterative Closest Point (ICP)

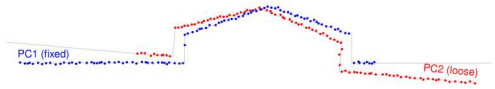
- Approach: iterate between finding correspondences and finding the transformation:



Given a pair of shapes, X and Y, iterate:

1. For each $x_i \in X$ find **nearest** neighbor $y_i \in Y$.
2. Find deformation $\mathbf{R}, t$ minimizing: $\sum_{i=1}^{N} \|\mathbf{R}x_i + t - y\|_2^2$

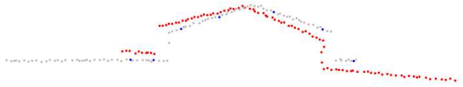
Iterative Closest Point (ICP) 0



The ICP algorithm is used to improve the orientation of a single **loose point cloud** with respect to a single **fixed point cloud**. The point clouds must be roughly aligned.

http://www.geo.tuwien.ac.at/downloads/pg/pctools/globalICPGeneralDescription.html

Iterative Closest Point (ICP) 1



Point Selection: A subset of points is selected within the overlap area in one point cloud. For this, points are first selected uniformly in object space, which leads to a homogeneous distribution of the points within the overlap area (**uniform sampling**). The mean sampling distance in each coordinate direction is defined by the parameter **`UniformSamplingDistance`**. The selected subset of points can be further refined with the parameter strategies **normal space sampling** or **maximum leverage sampling**.

http://www.geo.tuwien.ac.at/downloads/pg/pctools/globalICPGeneralDescription.html

Iterative Closest Point (ICP) 2

Matching: Find the closest points (= nearest neighbors) of the selected subset in the other point cloud. This step leads to a set of correspondences which possibly also includes some outliers.

http://www.geo.tuwien.ac.at/downloads/pg/pctools/globalICPGeneralDescription.html

Iterative Closest Point (ICP) 3

Rejection: Rejection of false correspondences (outliers) based on the compatibility of points. Correspondences are rejected if:

- the euclidean **distance** between corresponding points is larger than the value specified by the parameter **`MaxDistance`**
- the **roughness** attribute of one of the corresponding points is larger than the value specified by the parameter **`MaxRoughness`**. (Note: as roughness attribute the standard deviation in normal direction of the points used for plane fitting is used.)
- the **angle between the normals** of corresponding points is larger than the value specified by the parameter **`MaxDeltaAngle`**.

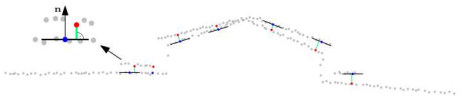
Iterative Closest Point (ICP) 4

Weighting: In this step a weight between 0 and 1 is assigned to each correspondence. Weights are based on:

- the **roughness** attribute of corresponding points
- the **angle between the normals** of corresponding points

http://www.geo.tuwien.ac.at/downloads/pg/pctools/globalICPGeneralDescription.html

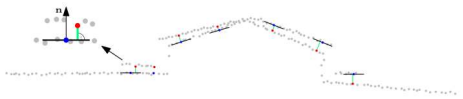
Iterative Closest Point (ICP) 5



Minimization: Estimation of the transformation parameters (for the loose point cloud) by a least squares adjustment. The adjustment minimizes the sum of squared **point-to-plane distances**. The point-to-plane distance of two corresponding points is defined as the orthogonal distance of one point to the fitted plane of the other point.

http://www.geo.tuwien.ac.at/downloads/pg/pctools/globalICPGeneralDescription.html

Iterative Closest Point (ICP) 6



Transformation: Transformation of the loose point cloud with the estimated parameters. The transformation model can be chosen with the parameter **`NoOfTransParam`**.

After step 6, a convergence criterion is tested. If it is not met, the process restarts with a new iteration from step 1. However, if subsets are defined by the parameter **`SubSetRadius`**, the iteration loop restarts from step 3 and the first two steps are only performed once. The total number of iterations heavily depends on the initial orientation of the points clouds, but typically, between 3 and 10 iterations should be sufficient to reach the global minimum.

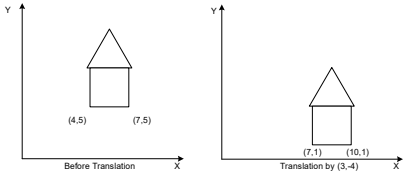
http://www.geo.tuwien.ac.at/downloads/pg/pctools/globalICPGeneralDescription.html

Iterative Closest Point (ICP) 7

Final Alignment.

http://www.geo.tuwien.ac.at/downloads/pg/pctools/globalICPGeneralDescription.html

Translation – 2D



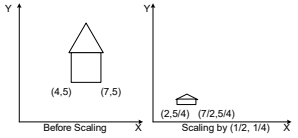
$x' = x + d_x$
$y' = y + d_y$

$$P = \begin{bmatrix} x \\ y \end{bmatrix} \quad P' = \begin{bmatrix} x' \\ y' \end{bmatrix} \quad T = \begin{bmatrix} d_x \\ d_y \end{bmatrix} \quad P' = P + T$$

Homogeneous Form

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & d_x \\ 0 & 1 & d_y \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Scaling – 2D



Types of Scaling:

- Differential (s_x != s_y)
- Uniform (s_x = s_y)

$$x' = s_x * x$$
$$y' = s_y * y$$

$$S \quad * \quad P = \quad P'$$

$$\begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} * \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x * s_x \\ y * s_y \end{bmatrix}$$

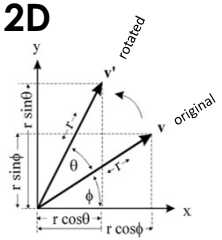
Homogeneous Form

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_x & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Rotation – 2D

$$\mathbf{v} = \begin{bmatrix} r\cos\phi \\ r\sin\phi \end{bmatrix}$$

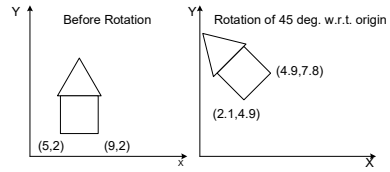
$$\mathbf{v}' = \begin{bmatrix} r\cos(\phi+\theta) \\ r\sin(\phi+\theta) \end{bmatrix}$$



$$\text{expand } (\phi+\theta) \Rightarrow \begin{cases} x' = r\cos\phi\cos\theta - r\sin\phi\sin\theta \\ y' = r\cos\phi\sin\theta - r\sin\phi\cos\theta \end{cases}$$

$$\text{but } \begin{matrix} x = r\cos\phi \\ y = r\sin\theta \end{matrix} \Rightarrow \begin{matrix} x' = x\cos\theta - y\sin\theta \\ y' = x\sin\theta + y\cos\theta \end{matrix}$$

Rotation – 2D



$x*\cos\theta - y*\sin\theta = x'$

$x*\sin\theta + y*\cos\theta = y'$

$$R \quad * P = \quad P'$$

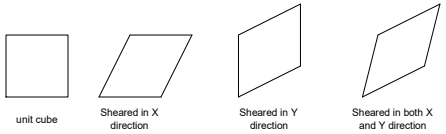
$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} * \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x*\cos\theta - y*\sin\theta \\ x*\sin\theta + y*\cos\theta \end{bmatrix}$$

Homogeneous Form

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Shearing Transformation

$$SH_x = \begin{bmatrix} 1 & a & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad SH_y = \begin{bmatrix} 1 & 0 & 0 \\ b & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad SH_{xy} = \begin{bmatrix} 1 & a & 0 \\ b & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



3D Transformation

- In homogeneous coordinates, 3D transformations are represented by 4×4 matrixes:

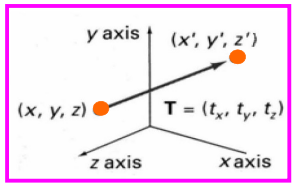
$$\begin{bmatrix} a & b & c & t_x \\ d & e & f & t_y \\ g & h & i & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3D Translation

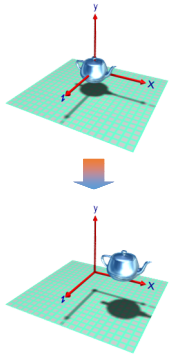
- **P** is **translated** to **P'** by:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$\mathbf{P'} = \mathbf{T} \cdot \mathbf{P}$$



Translation – 3D

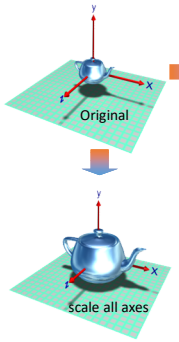


$$x' = x + d_x$$
$$y' = y + d_y$$
$$z' = z + d_z$$

$$\begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x + d_x \\ y + d_y \\ z + d_z \\ 1 \end{bmatrix}$$

$$T(d_x, d_y, d_z) \quad * \quad P = \quad P'$$

Scaling – 3D



$$x' = s_x * x$$
$$y' = s_y * y$$
$$z' = s_z * z$$

$$S(s_x, s_y, s_z) \quad * \quad P = \quad P'$$

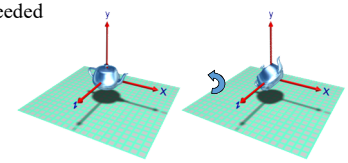
$$\begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x * s_x \\ y * s_y \\ z * s_z \\ 1 \end{bmatrix}$$

Rotation – 3D

For 3D-Rotation 2 parameters are needed

- Angle of rotation
- Axis of rotation

Rotation about z-axis:

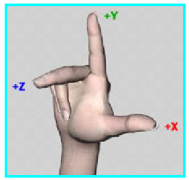


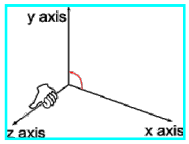
$$R_{\theta,k} \quad * \quad P = \quad P'$$

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x * \cos\theta - y * \sin\theta \\ x * \sin\theta + y * \cos\theta \\ z \\ 1 \end{bmatrix}$$

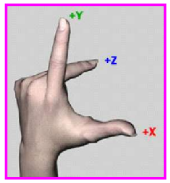
3D Coordinate Systems

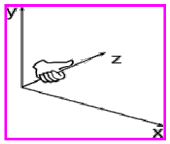
- ***Right Hand*** coordinate system:





- ***Left Hand*** coordinate system:





Rotation about Y-axis & X-axis

About y-axis

$$R_{\theta,j} * P = P'$$

$$\begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x*\cos\theta + z*\sin\theta \\ y \\ -x*\sin\theta + z*\cos\theta \\ 1 \end{bmatrix}$$

About x-axis

$$R_{\theta,i} * P = P'$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y*\cos\theta - z*\sin\theta \\ y*\sin\theta + z*\cos\theta \\ 1 \end{bmatrix}$$

Rotation – 3D

In general, rotations are specified by a rotation axis and an angle. In two-dimensions there is only one choice of a rotation axis that leaves points in the plane.

Rotation Matrix

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$
Identity Matrix

$$\begin{pmatrix} 1 & 0 & 0 & tx \\ 0 & 1 & 0 & ty \\ 0 & 0 & 1 & tz \\ 0 & 0 & 0 & 1 \end{pmatrix}$$
glTranslatef(tx,ty,tz)

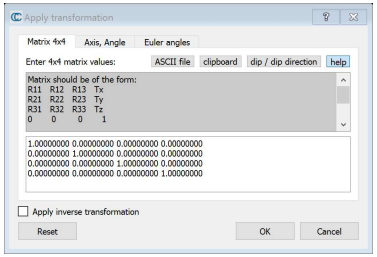
$$\begin{pmatrix} sx & 0 & 0 & 0 \\ 0 & sy & 0 & 0 \\ 0 & 0 & sz & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$
glScalef(sx,sy,sz)

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(d) & -\sin(d) & 0 \\ 0 & \sin(d) & \cos(d) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$
glRotatef(d,1,0,0)

$$\begin{pmatrix} \cos(d) & 0 & \sin(d) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(d) & 0 & \cos(d) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$
glRotatef(d,0,1,0)

$$\begin{pmatrix} \cos(d) & -\sin(d) & 0 & 0 \\ \sin(d) & \cos(d) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$
glRotatef(d,0,0,1)

Generalized 3D rotation matrix:
R=Rx(α)Ry(β)Rz(γ) (multiply together the prior single axis rotations)
Where α, β, γ are roll, pitch, and yaw



http://www.cloudcompare.org/doc/wiki/index.php?title=Alignment_and_Registration

Affine transformations 1

- **Transformation** – is a function that takes a point (or vector) and maps that point (or vector) into another point (or vector).
- A coordinate transformation of the form:

$$x' = a_{xx}x + a_{xy}y + a_{xz}z + b_x,$$
$$y' = a_{yx}x + a_{yy}y + a_{yz}z + b_y,$$
$$z' = a_{zx}x + a_{zy}y + a_{zz}z + b_z,$$

is called a 3D ***affine transformation***.

$$\begin{pmatrix} x' \\ y' \\ z' \\ w \end{pmatrix} = \begin{pmatrix} a_{xx} & a_{xy} & a_{xz} & b_x \\ a_{yx} & a_{yy} & a_{yz} & b_y \\ a_{zx} & a_{zy} & a_{zz} & b_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

- The 4th row for affine transformation is always [0 0 0 1].
- Properties of affine transformation:
  - translation, scaling, shearing, rotation (or any combination of them) are examples affine transformations.
  - Lines and planes are preserved.
  - parallelism of lines and planes are also preserved, but not angles and length.

Affine transformations 2

Generic name for these transformations:
affine transformations

$$x' = a_{xx}x + a_{xy}y + a_{xz}z + b_x$$
$$y' = a_{yx}x + a_{yy}y + a_{yz}z + b_y$$
$$z' = a_{zx}x + a_{zy}y + a_{zz}z + b_z$$

Affine transformations 3

Properties:

1. Transformed coordinates *x',y'* and *z'* are *linearly* dependent on original coordinates *x, y, z;*
2. Parameters a_{ij} en b_k are constant and determine type of transformation;
3. Examples: translation, rotation, scaling, reflection
4. Parallel lines remain parallel
5. Only translation, rotation reflection: angles and lengths are maintained
6. **Only translation and rotation: 6 parameters! (no scaling)**