COMPUTER VISION 2018 - 2019

>SILHOUETTE-BASED VOLUME RECONSTRUCTION

UTRECHT UNIVERSITY

RONALD POPPE

OUTLINE

Depth from images

Silhouette-based volume reconstruction

- Background subtraction
- Volume reconstruction

App of the week

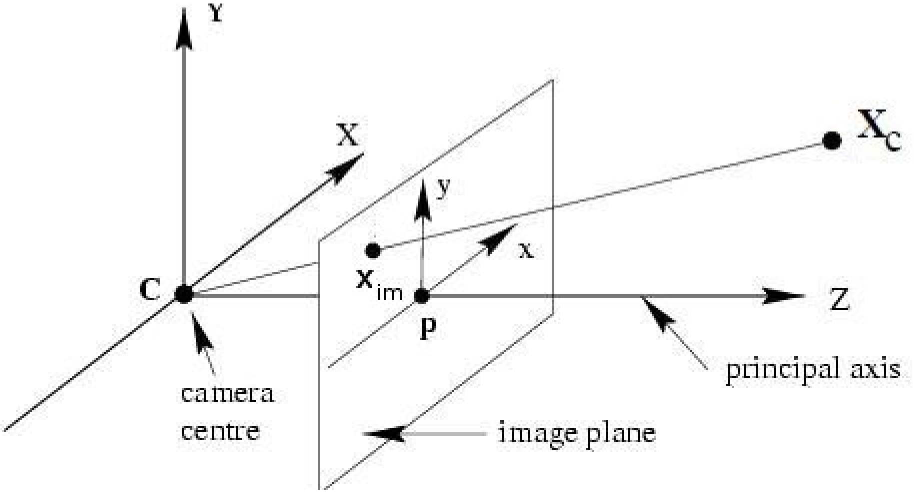
Assignment

DEPTH FROM IMAGES

LAST LECTURE: 3D TO 2D

A 3D point in world coordinates is projected onto a 2D image coordinate (pixel)

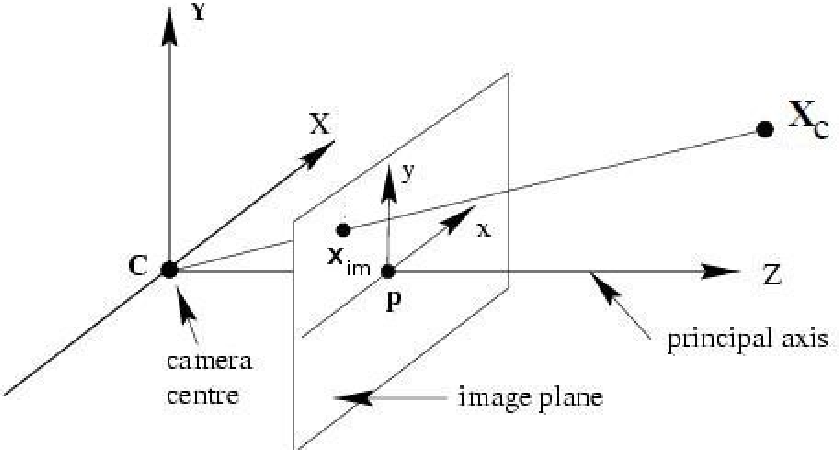
Given proper calibration, we can determine the extrinsic and intrinsic camera parameters and determine the projection



BACK-PROJECTION

We can also reason the other way around:

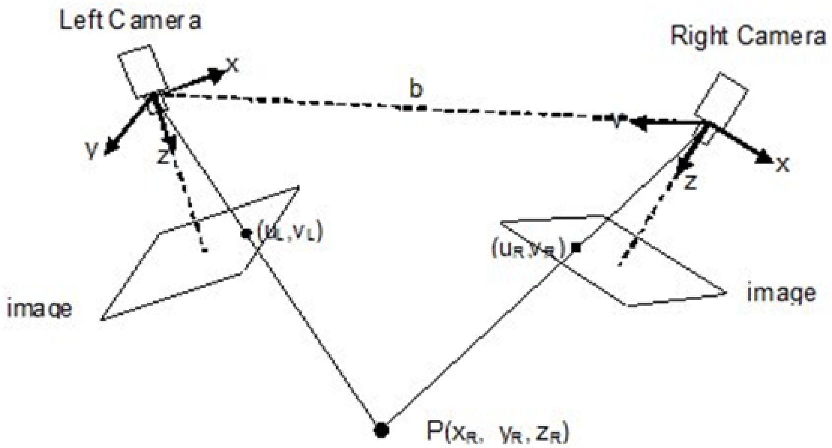
- A 2D image location can correspond to a range of 3D points
- These points are on a line through camera center ***C*** and the projection of $\boldsymbol{X_C}$ on the virtual image plane
- 3D location is known up to a depth-factor



TRIANGULATION

If we identify the 2D projection of a 3D point in two views, we can calculate its postion in 3D

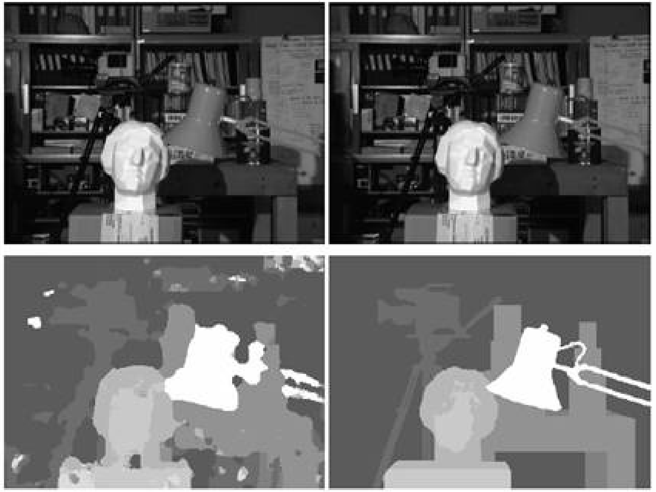
- This is termed triangulation

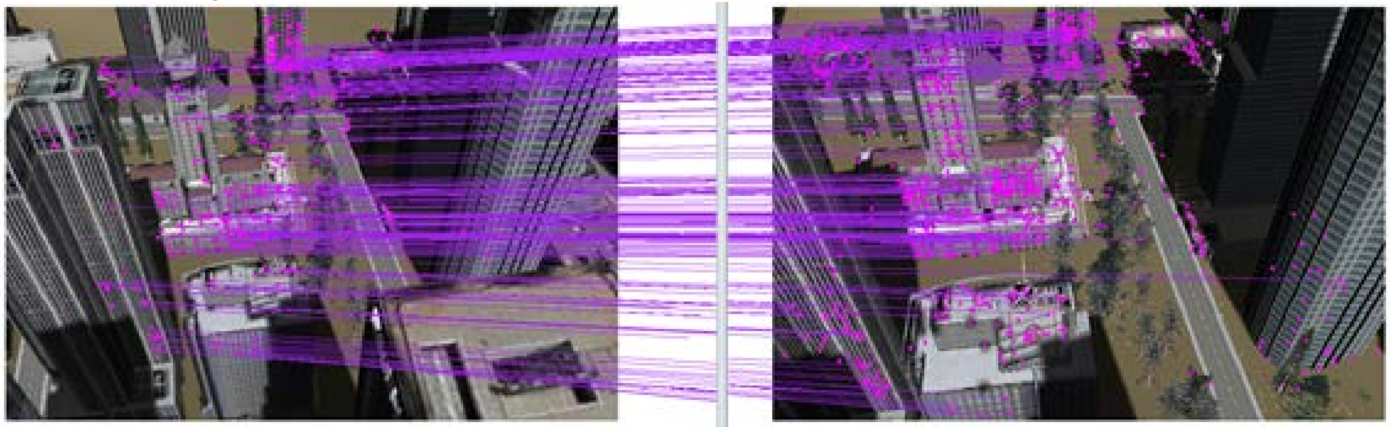


STEREO VISION

Based on triangulation of set of key points, determine the disparity of the whole scene

- Camera parameters are known
- This is termed stereo vision
- Similar to the human eyes





STEREO VISION[2]

Good results are obtained for textured areas

- Easier to match points across views

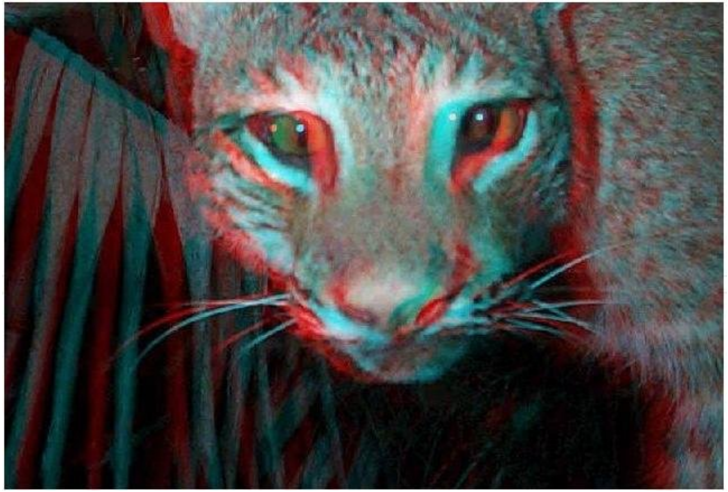
Generally bad results for evenly colored areas

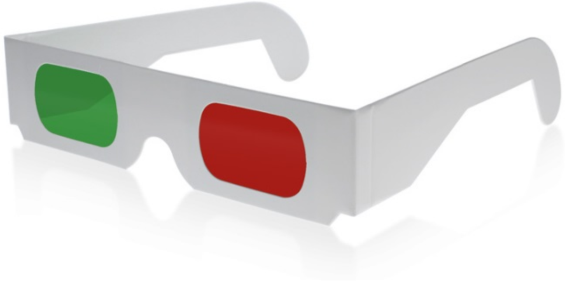
- Lack of texture
- No keypoints will be found

STEREO VISION[3]

Cheap 3D glasses:

- Each eye sees a slightly different image

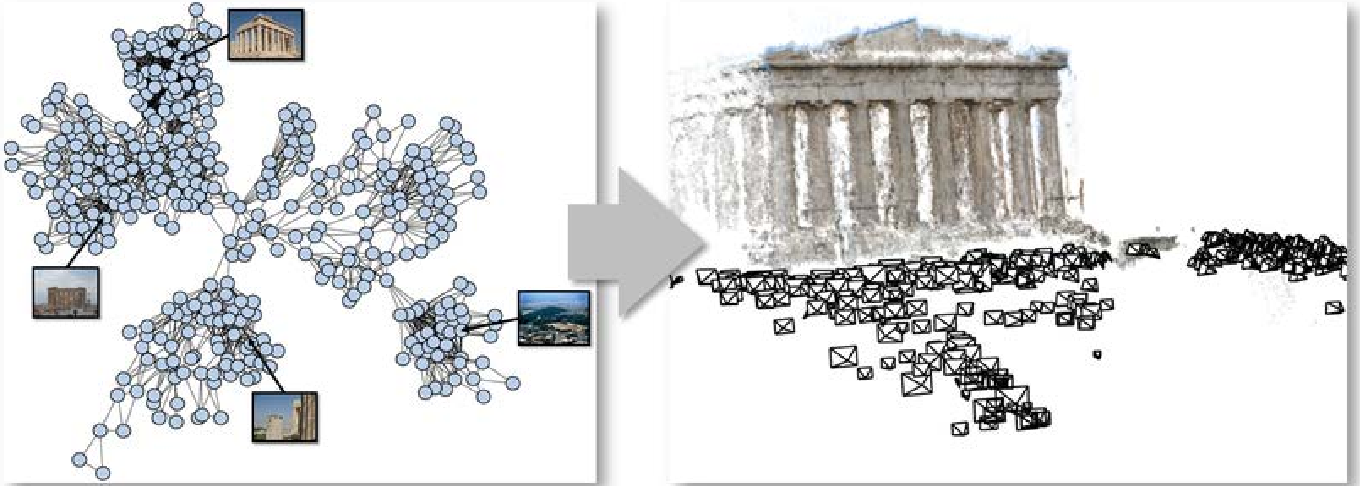




MULTIPLE VIEW STEREO

If multiple views are available, stereo matching for each pair of views can be attempted

Intrinsic camera parameters, extrinsic camera parameters and depth are recovered simultaneously: time-consuming!



https://www.youtube.com/watch?v=s-DqZ8jAmv0

SHAPE FROM MOTION

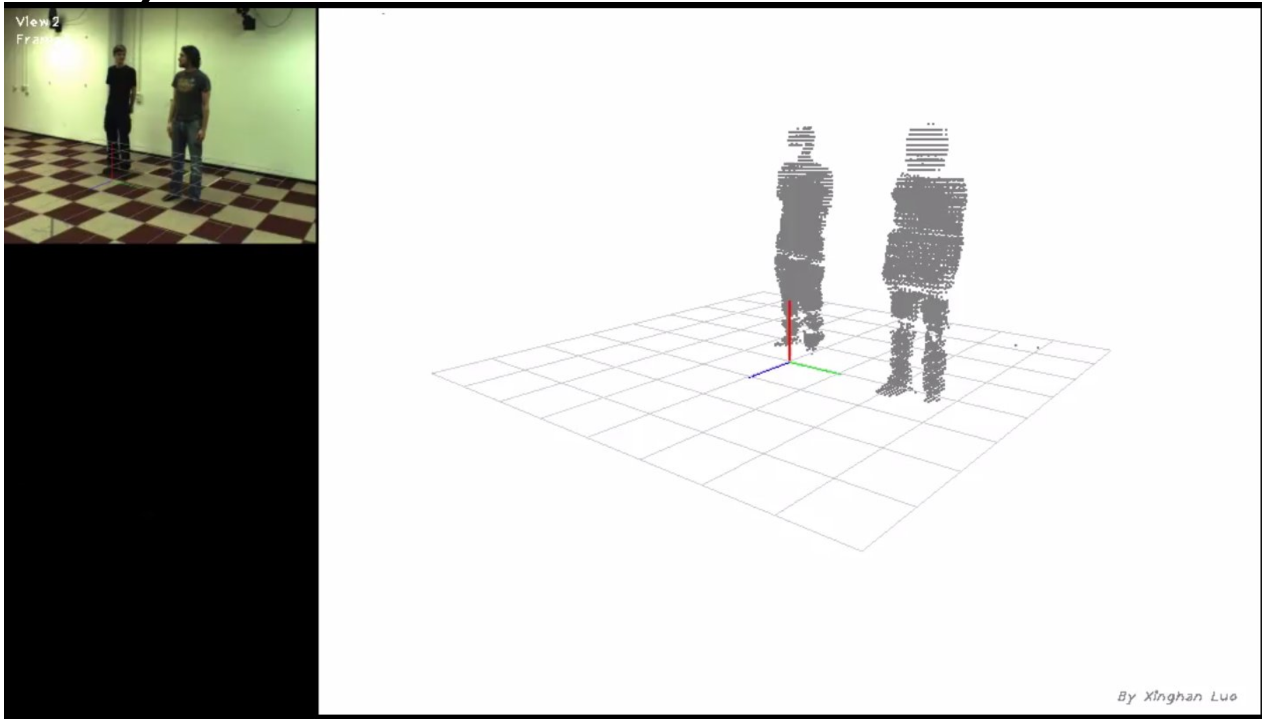
If a video is available, camera intrinsics are the same each frame:

- Subsequent frames in a video can be matched
- Local motion can be determined
- Saves a lot of time evaluating all pairs

SILHOUETTE-BASED VOLUME RECONSTRUCTION

BASIC IDEA

Example (Assignment 2!)



BASIC IDEA²

Use a number of cameras

- Calibrate them

Instead of finding/matching keypoints:

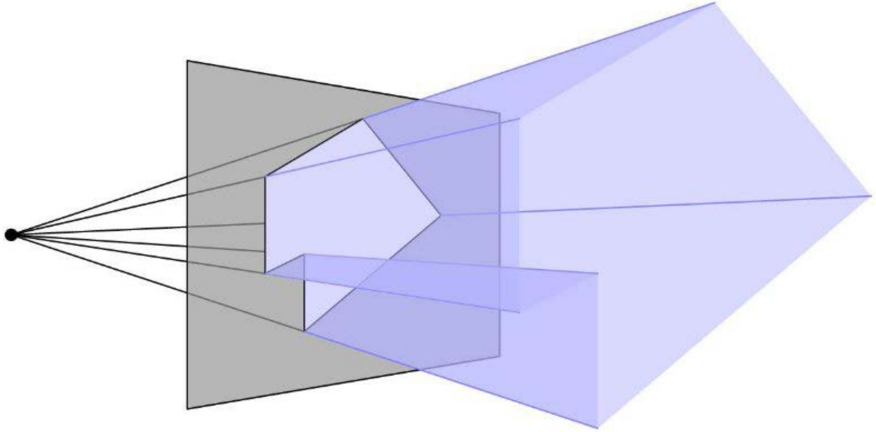
- Determine the foreground
- Perform volume intersection

SILHOUETTE BACK-PROJECTION

A silhouette is a 2D shape

The back-projection of:

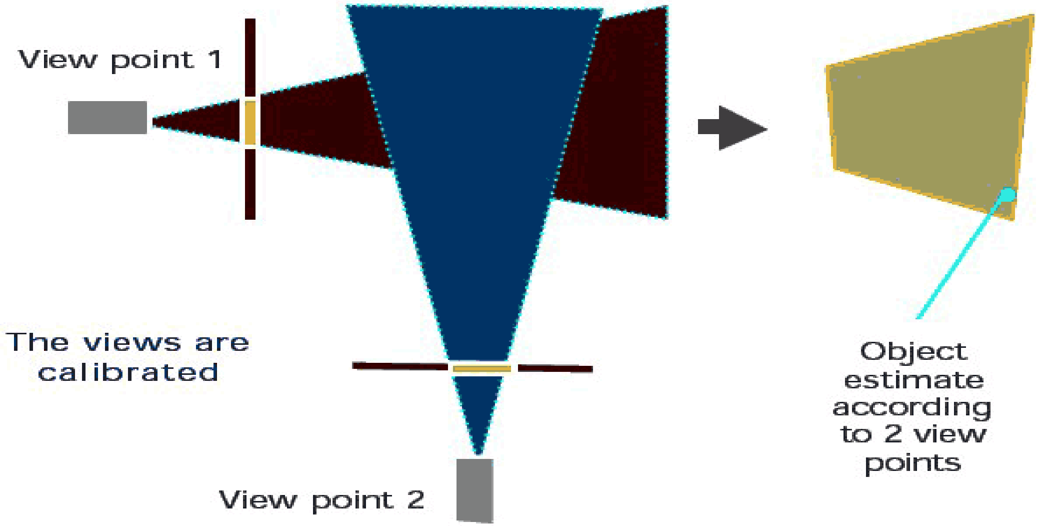
- A 2D point is a line
- A 2D line is a plane
- A 2D plane is a…
- … Volume



VOLUME INTERSECTION

When multiple views are available, we get more volumes

- These can be combined!



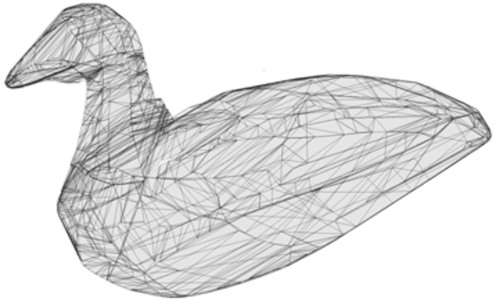
MESH

The intersections carve the shape out as a polygon model

- Typically loads of polygons if there are several cameras

Computationally expensive:

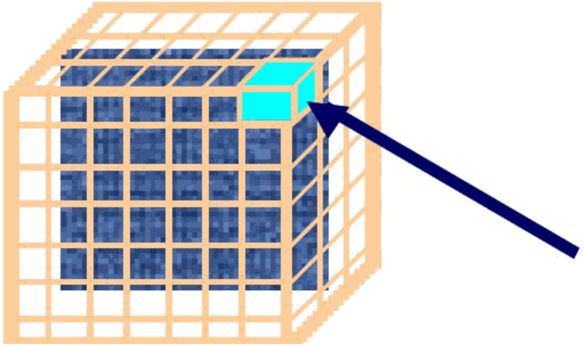
- Lot of storage is needed
- Lot of computation power is needed to determine the novel intersections (floating point operations)



VOLUME DISCRETIZATION

Solution: use a fixed grid of “3D pixels” which are on or off

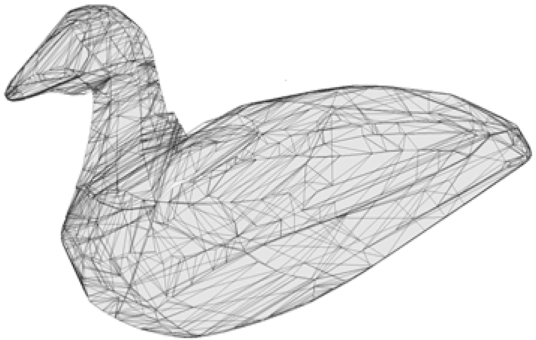
- A 3D pixel is a voxel

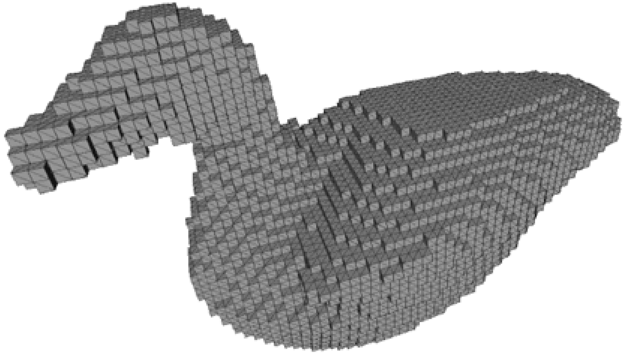


Typically many voxels

- 64 per row: 256k for volume

Still, detail is lost





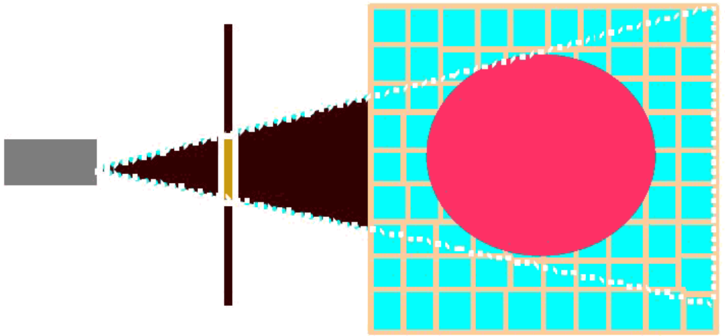
VOLUME DISCRETIZATION[2]

More positive:

- Storage can be limited (when using octrees)
- Calculation can be done efficiently

Look-up table can be constructed for each voxel

- Determines to which pixels (in each view) a voxel projects



LIMITATIONS

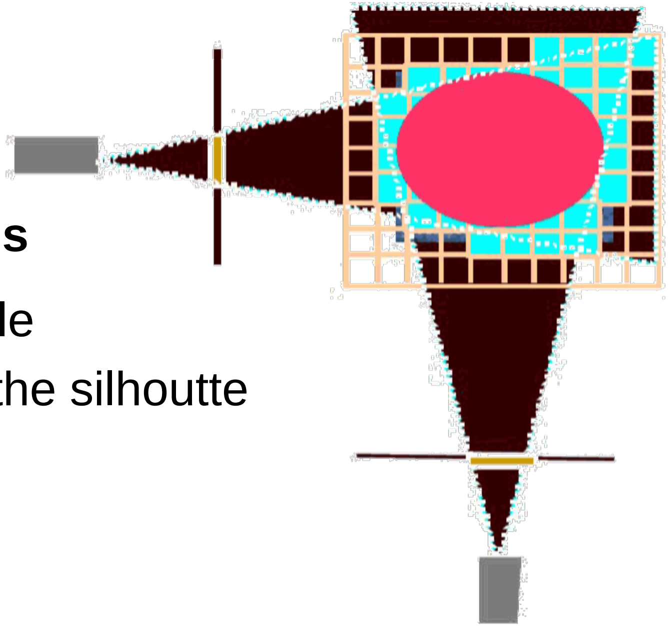
The use of silhouettes, and that of voxels, has a number of limitations:

- Multiple views needed
- Holes (concavities) cannot be modelled
- Depends on good silhouettes

LIMITATIONS²

The more views are used, the more precise the shape estimation

- Estimation is typically conservative: estimated shape is bigger than actual shape



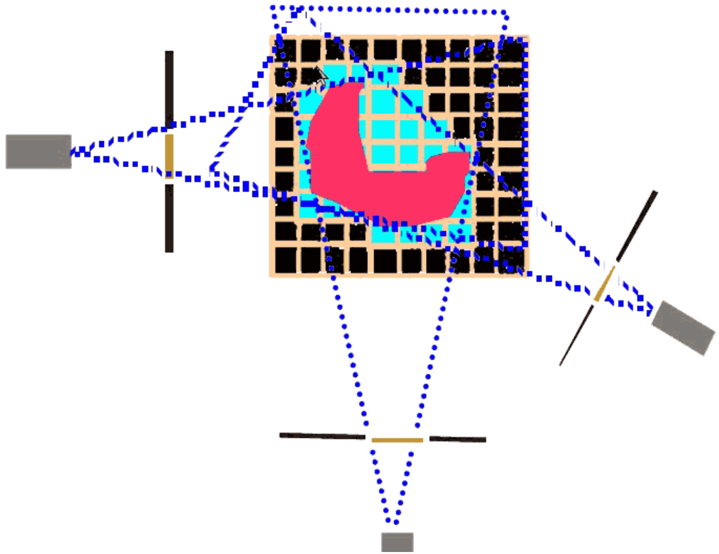
Views should be placed at "suitable" locations

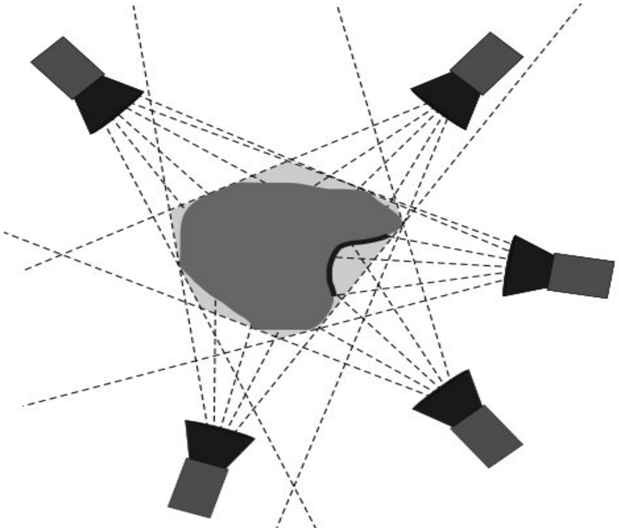
- Ideally, all parts of the object should be visible
- But shapes are determined at the **edges** of the silhoutte

LIMITATIONS[3]

“Holes” in a shape cannot be estimated: only convex objects

- Not a single view “sees” the hole
- Adding cameras has no effect



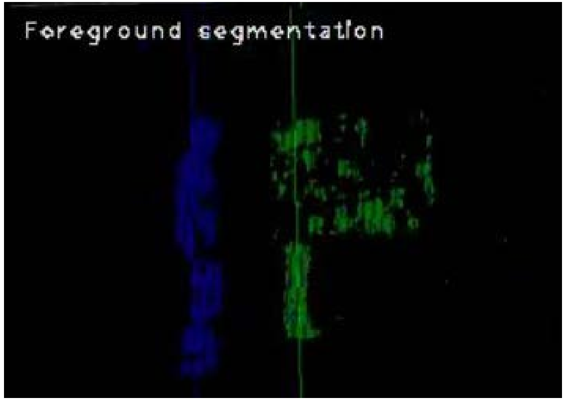


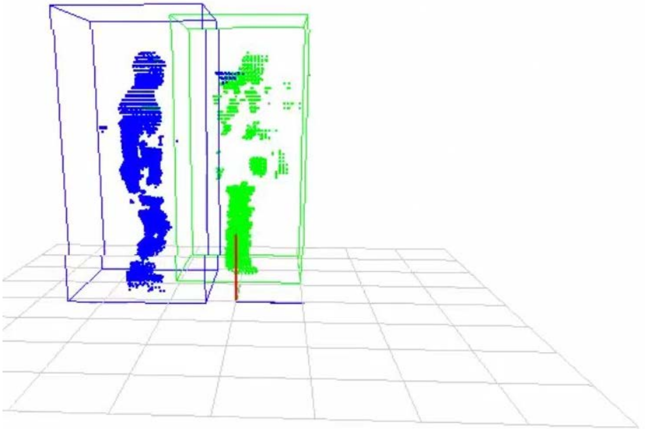
LIMITATIONS[4]

Incorrect background subtraction leads to noisy silhouettes

- Missing parts of the object
- Additional noise

In turn, the voxel model can be incorrect





ALGORITHM RECIPE

1. **Calibrate all cameras (extrinsic and intrinsic parameters) →** **Previous lecture**

2. **Extract silhouettes in each view (background subtraction)**

3. **Determine the voxels that represent the volume of the target objects, by projecting the voxels onto each view.**
   - If they overlap with the silhouette, we retain the voxels; otherwise discard them

BACKGROUND SUBTRACTION

CONCEPT

The idea of background subtraction is to determine which parts of an image are the foreground, and which are background

Common assumptions:

- Background colors are different from those in the foreground
- The background scene is static

CONCEPT^2

By checking every pixel in an image, it is determined whether it is part of foreground or background.

The assumptions are typically not met perfectly

- Additional processing can be applied to improve result

CHROMA-KEYING

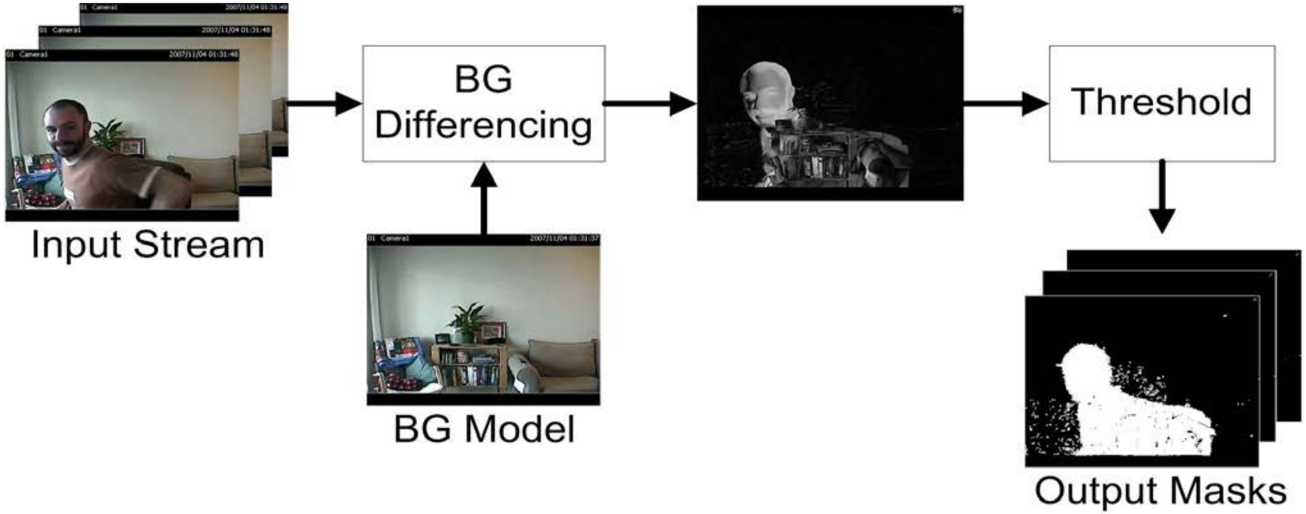
Chroma-keying or “the green-screen technique”

- Each pixel is compared to reference color (green)
- Green least resembles skin color

BACKGROUND SUBTRACTION

Instead of using a background of a single color, a snapshot of the background can be used as a reference model

- Each pixel compared to background model
- No need for dedicated background



CHALLENGES

- Color variation and overlap
- Shadows
- Movement in the background
- Aliasing

CHALLENGES[2]

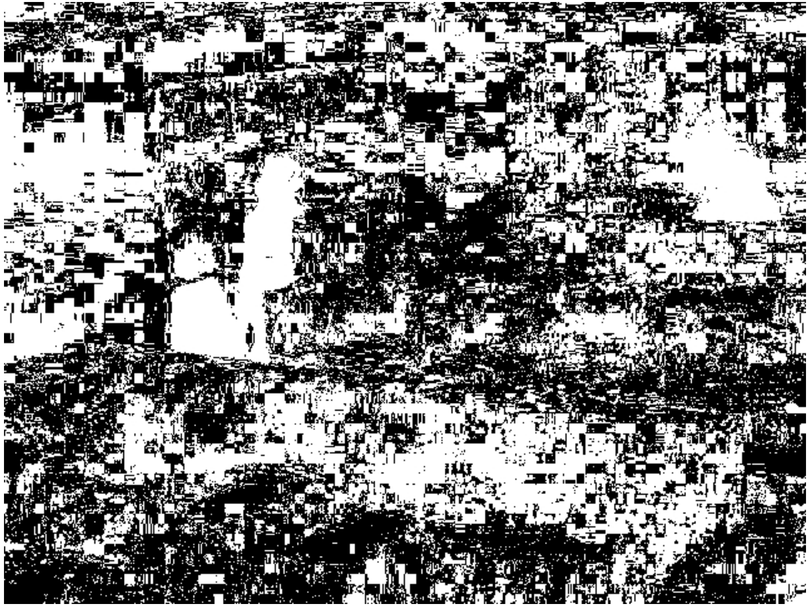
Variations and overlap in color:

- Colors in foreground are similar to those in background
- Colors of background affected by foreground

CHALLENGES[3]

Colors in background can change over time

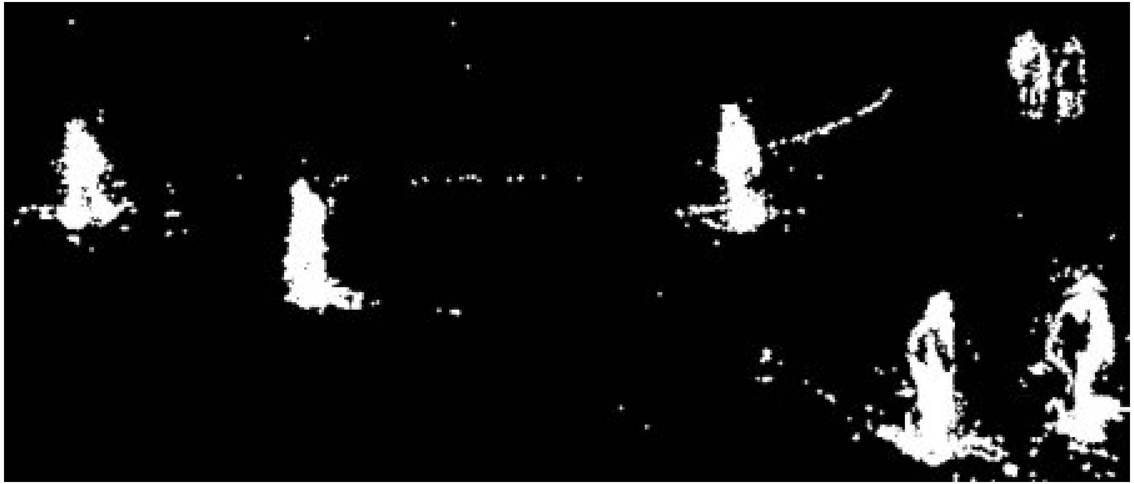
- Light (day/night)



CHALLENGES[4]

Shadows in the background:

- Often “attached” to foreground objects
- Make background darker, without affecting the color too much



CHALLENGES[5]

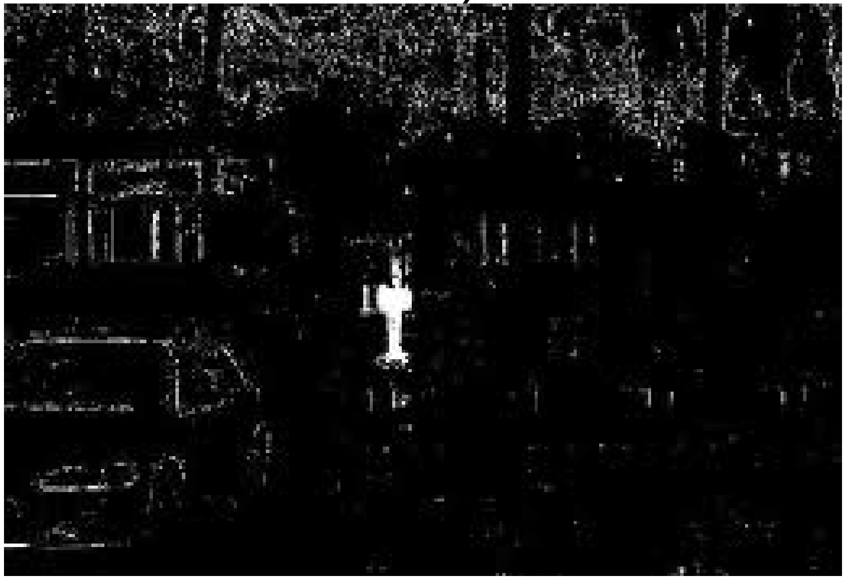
Movement in the surrounding background:

- Violates “static” assumption

CHALLENGES[6]

Aliasing is the process of averaging pixel values:

- Can cause problems at light-dark intersections
- Pixel values fluctuate (also due to small camera movements)



MODELING BACKGROUNDS

Easiest: take a picture B with exactly the same extrinsic and intrinsic camera parameters but without foreground objects

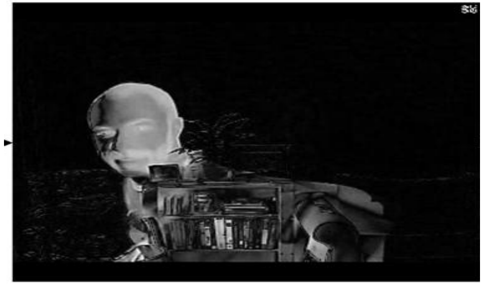
Compare each pixel in a new image *I* to the background *B*

- Pixels with a difference above a certain threshold δ are foreground
- $D = |I - B|$
- Foreground: $D > \delta$

Can also be done per color channel: allows for different thresholds

MODELING BACKGROUNDS[2]

So, $D = |I - B|$ becomes:



And $D > \delta$:

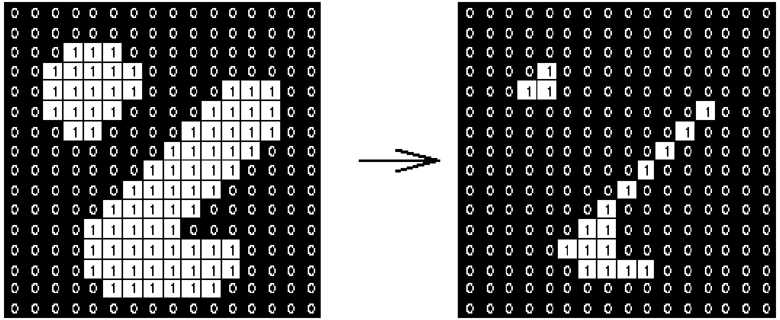
NOISE REDUCTION

To remove noise, we can apply morphological operations

- Binary filters that change a pixel depending on the neighbors

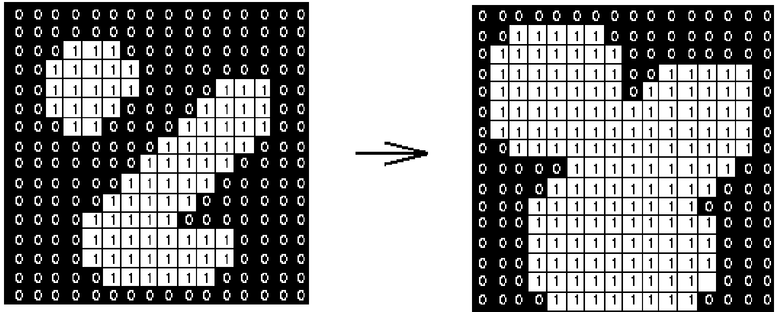
Erosion:

- Remove outliers



Dilation:

- Fill holes



GAUSSIAN MODELS

Issues:

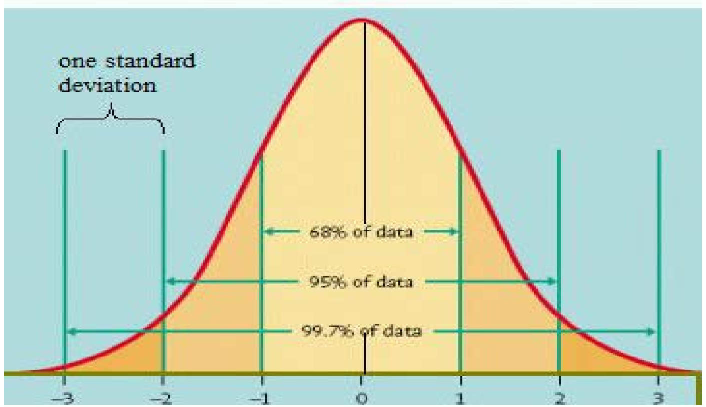
- Differences with lighter colors are typically larger
- Natural variation in the color of a background pixel

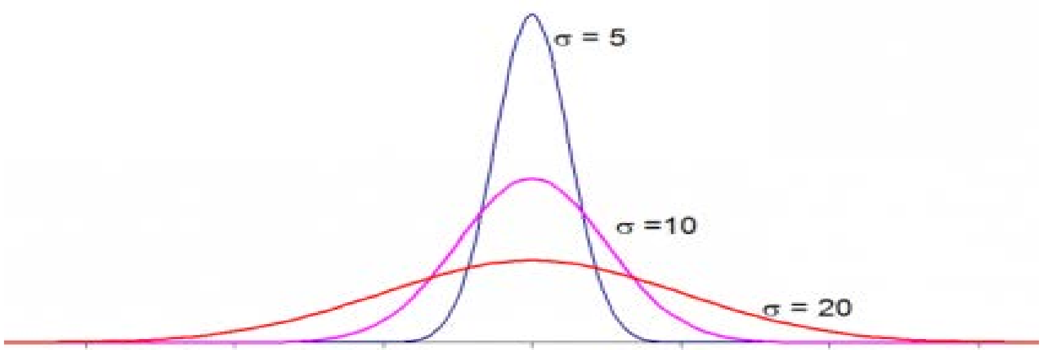
Solution:

- Set threshold per pixel depending on color and variation of background
- Typically modeled as a normal distribution

GAUSSIAN MODELS[2]

Normal distribution:





Two parameters per "Gaussian":

- Mean value
- Standard deviation

GAUSSIAN MODELS[3]

When modeling a pixel value with a Gaussian:

- Mean corresponds to mean pixel value
- Standard deviation is larger for pixels that vary more

In practice:

- Brighter pixels will have a larger standard deviation
- Pixels close to edges will have a larger standard deviation

GAUSSIAN MODELS[4]

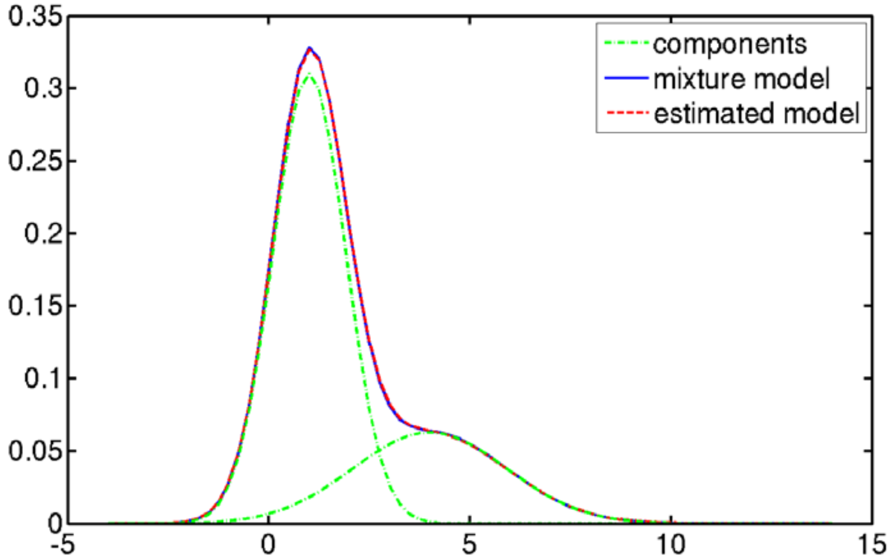
When doing background subtraction, we can have a threshold that determines how many standard deviations (instead of pixel values) a pixel’s value is from the mean:

- For larger standard deviations, pixel values should be more different
- This corresponds with our intuition

GAUSSIAN MIXTURE MODELS[5]

Sometimes, there are more "sources" of pixel values (shadows, reflection, traffic lights, etc.)

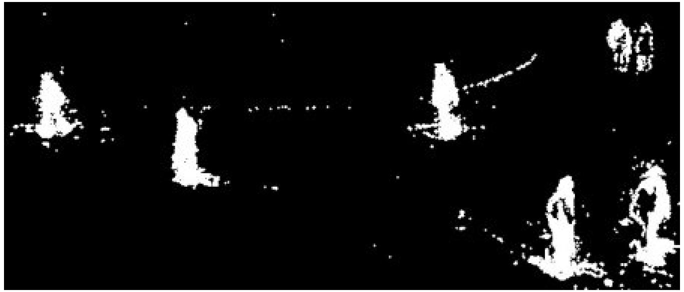
- Instead of a single normal distribution, use a mixture



GAUSSIAN MIXTURE MODELS[6]

One of the mixture components might correspond to shadows

Pixel value should be at least a certain number of standard deviations from each component to be considered as foreground

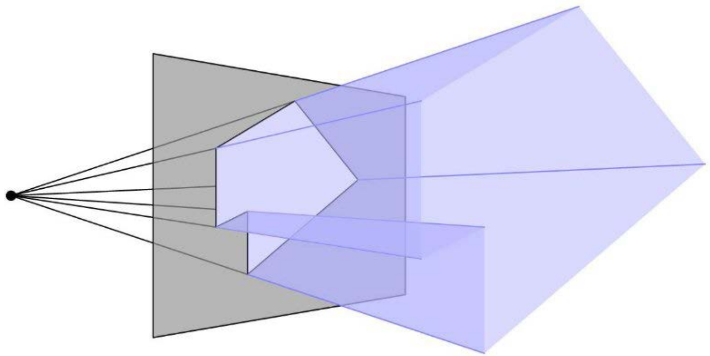


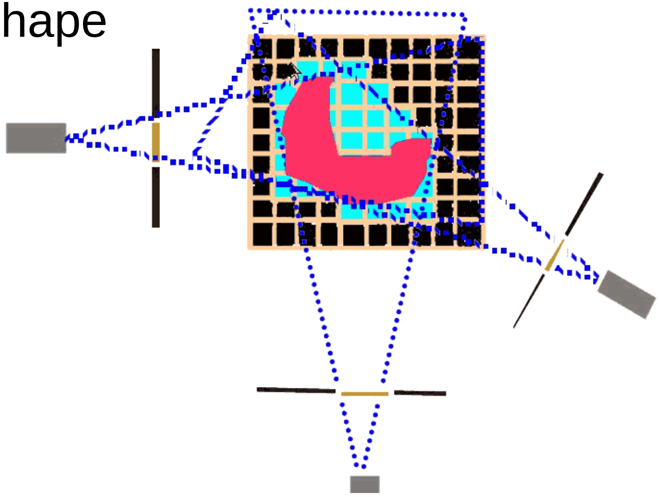
VOLUME RECONSTRUCTION

VOLUME RECONSTRUCTION

The back-projection of a silhouette in 2D is a 3D volume

- Intersection of the 3D volumes from multiple views “carves out” the estimated 3D shape of the object
- Holes cannot be estimated
- Reconstructed shape is overestimation of true shape

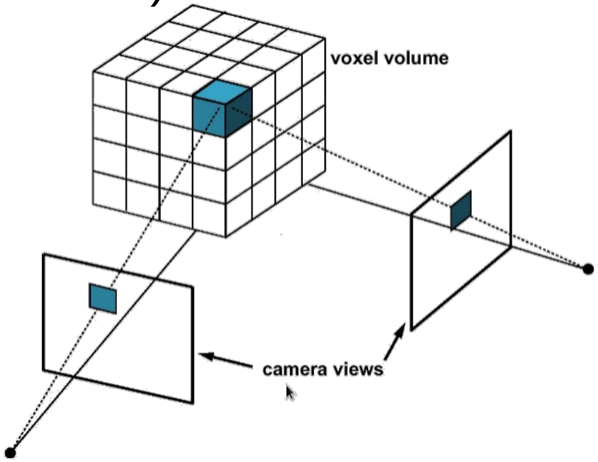




VOLUME RECONSTRUCTION[3]

Calculating volume intersections using voxel grids can be a computationally effective solution:

- Grid of voxels used to represent the object
- Placement of voxel grid can be anywhere
- Grids can have any size (but powers of 2 are convenient)



LOOK-UP TABLE

For each voxel, we can determine per view if and where in the image it is visible

- Camera intrinsic and extrinsic parameters used for projection
- Voxel does not have to be visible in every view

Typically only a single pixel coordinate (center) used

- Silhouette extraction should be robust
- Area of pixels can also be used (corners), use average or threshold

LOOK-UP TABLE^2

A voxel is a 3D pixel, but we usually work with just the center

Algorithm for construction of look-up table:

For every voxel $\{X_V, Y_V, Z_V\}$ **in the voxel volume**

For every view c

Project $\{X_V, Y_V, Z_V\}$ **onto the image plane of** $\boldsymbol{c}$**:** $\{x_{im}, y_{im}\}$

Store $\{X_V, Y_V, Z_V\}$, c **and** $\{x_{im}, y_{im}\}$ **in the look-up table**

So look-up table has three types of information!

VOXEL-BASED RECONSTRUCTION

Once the look-up table has been made, we can start the voxel-based reconstruction

Idea: a voxel should be on if the projection in each view indicates that it is part of the foreground

We need to store the corresponding pixel value (foreground/background) for each:

- Voxel
- View

VOXEL-BASED RECONSTRUCTION[2]

Two options:

- Iterate over voxels
- Iterate over the image locations per view (from look-up table)

Iteration over voxels:

- +: no need to store intermediary results for each view
- -: has to be re-calculated when views are added or silhouettes change

Iteration over image locations per view:

- +: no need to inspect other views when there is a change in one view
- -: all values should be stored

VOXEL-BASED RECONSTRUCTION[3]

Algorithm (iterating over pixels):

For every view c**:**

For every pixel $\{x_{im}, y_{im}\}$**:**

If $\{x_{im}, y_{im}\}$ **is foreground:**

For each voxel corresponding to this pixel:

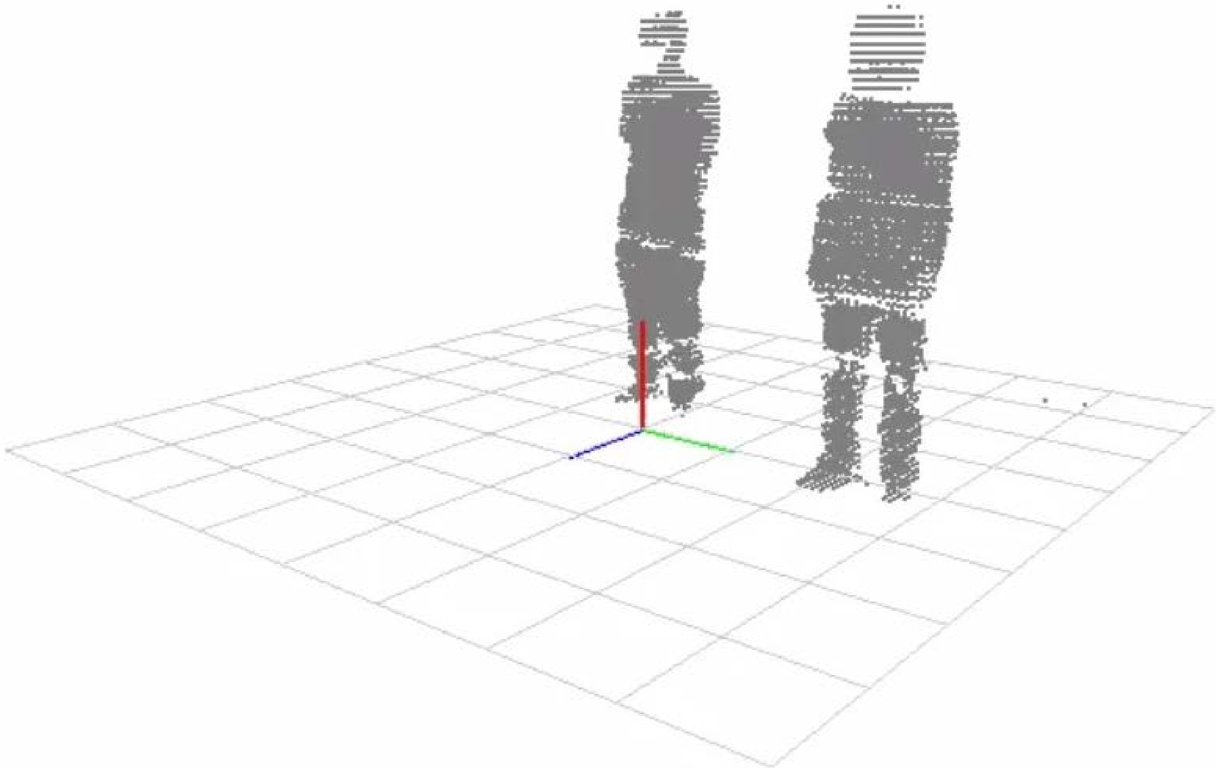
Mark the voxel visible for view c

For each voxel:

Mark the voxel visible if it is visible in all views in the table

VOXEL-BASED RECONSTRUCTION[4]

Expected result



VOXEL-BASED RECONSTRUCTION[5]

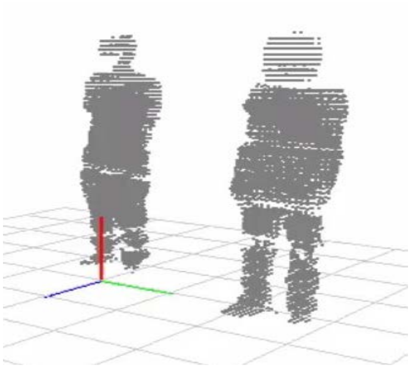
In practice:

- If there's a hole in a silhouette of one view, a complete "line" of voxels will not be visible
- On the other hand, if there are spurious (extra) pixels in one view, chances are that these will not be "on" voxels

So playing around with thresholds for background subtraction is a good idea:

- A little bit progressive (slightly more noise) might work well

VOXEL-BASED RECONSTRUCTION[6]



Once the voxel model is obtained, there can be noise

- Missing voxels
- Spurious voxels

We can apply erosion and dilation also in 3D:

- Erosion: a voxel is removed if at least *x* neighbors are “off”
- Dilation: a voxel is added if at least *x* neighbors are “on”
- *x* depends on quality of voxel model and on the shape

VOXEL-BASED RECONSTRUCTION[7]

Typical for this assignment:

Shadows will be present:

- Have a stricter threshold around the feet?
- Be less strict when pixels become darker
- Typically easier in HSV color space

People are standing:

- When there is a voxel “on” above and below, chances are that this voxel is also on → “1D dilation”

VOXEL-BASED RECONSTRUCTION[8]

We have assumed binary voxels

- We can also color them

Since we know the projection in each view, we can use the colors of the pixels in each view

- Will be slightly different due to lighting issues

Important: we need to take into account occlusions (or order)

- Pre-determine z-order of voxels per view
- We only need to do this once as the camera does not move

EFFICIENCY: SPEED

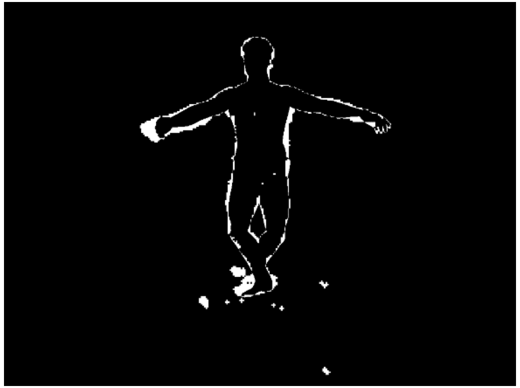
With video, changes from frame to frame are often small

- No need to recalculate all partial visibility values
- Only check the pixels that have changed

Change in image can be determined with XOR operation:

- Binary difference

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0



EFFICIENCY: STORAGE

Voxel models increase in the 3rd power of their length

- A 64 x 64 x 64 voxel model contains 256k voxels
- A 256 x 256 x 256 voxel model contains 16M voxels

Voxel models are binary and can be described efficiently

- Neighboring voxels often have the same value
- Sizes of a power of 2 allow for a coarse-to-fine description

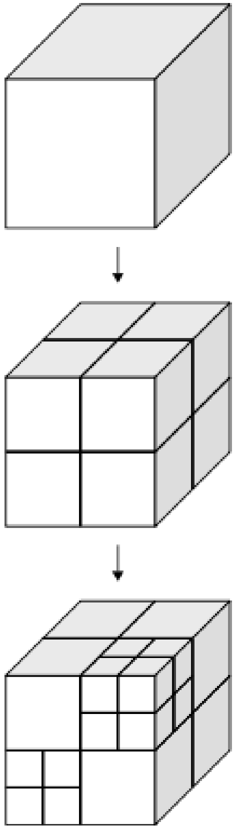
EFFICIENCY: STORAGE[2]

Octrees describe a volume as a string of 8 values

- Each corresponds to an octant

Values can be on, off or mixed

- Mixed values are recursively processed
- First level considers the whole voxel grid
- Final level considers individual voxels

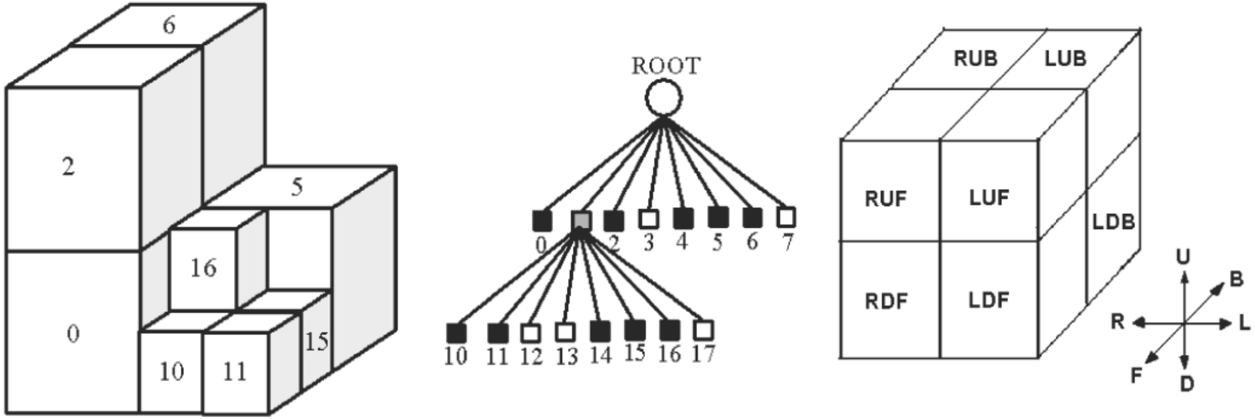


EFFICIENCY: STORAGE[3]

Storing the trees saves a lot of space

- Changing a single or set of voxel values is less efficient

Octrees are also used in rendering



FROM VOXELS TO POLYGONS

Voxel models can be extracted efficiently but have drawbacks

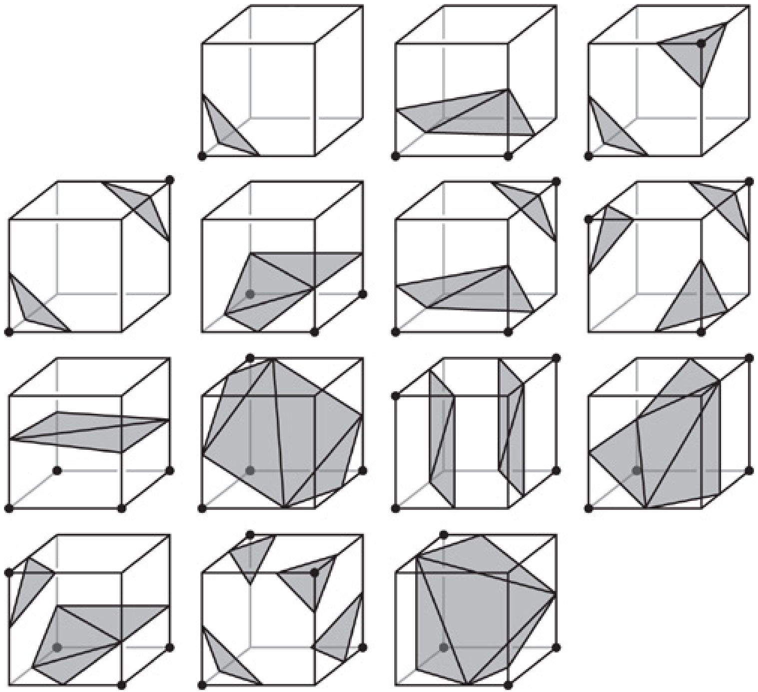
- Often low resolution
- Shape built up in layers
- No smooth surface

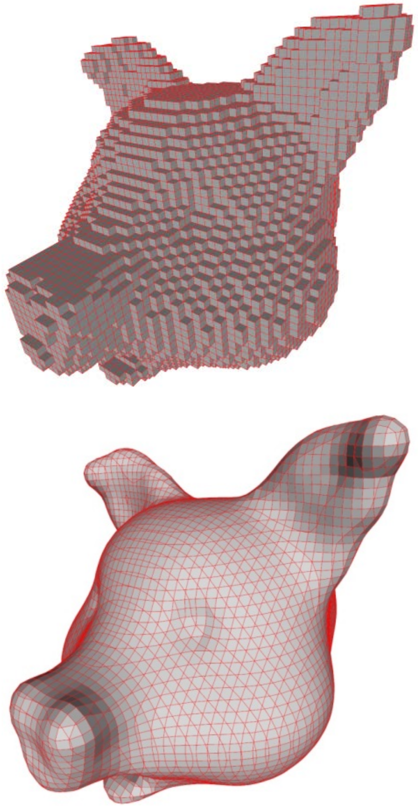
Voxel models can be converted to polygon models using the Marching Cubes algorithm

- Considers a local patch of 2 x 2 x 2 voxels
- Polygon model can subsequently be smoothed

FROM VOXELS TO POLYGONS[2]

15 different possibilities for voxel patch





APP OF THE WEEK

HANDHELD OBJECT SCANNER

Scenario:

- It’s Valentine’s day, you don’t have a gift for your partner/crush and find yourself in the Intratuin. Now what? Buy a plant? Call a friend?

No! Use your phone for something else! Remember that she loves garden gnomes and that you are technology-savvy!

HANDHELD OBJECT SCANNER^2

You've just learned how to make volumetric models from multiple cams

- Might work as well for a single cam at multiple locations

Intrinsic and extrinsic parameters should be determined:

- Intrinsic can be done beforehand
- Extrinsic can't, as the camera position will differ
- Add a marker (e.g. a small chessboard) to the scene!

HANDHELD OBJECT SCANNER[3]

The recipe:

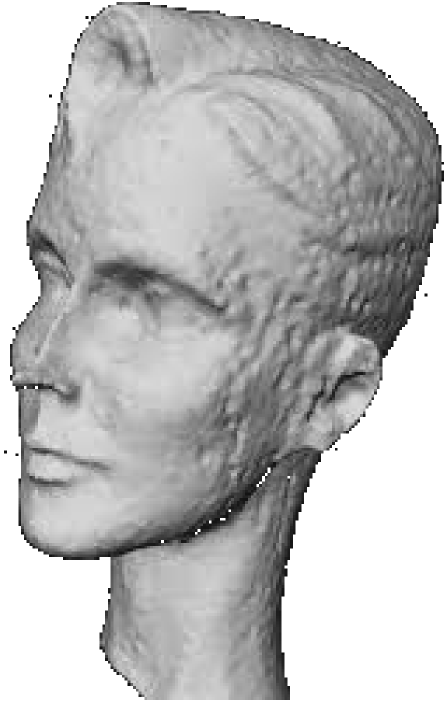
- Take a green screen and put a marker in the scene
- Shoot the hell out of a gnome
- Calibrate and make the 3D model
- Send it to a 3D printer
- Deliver at doorstep of partner/crush…
- Done

And it's cheaper than

https://www.youtube.com/watch?v=AYq5n7jwe40

HANDHELD OBJECT SCANNER[4]

Of course, you can also scan your face and make a personalized garden gnome



Please send me the pictures if you're done!

ASSIGNMENT

ASSIGNMENT

Assignment 1 due Sunday February 17, 23:00

- If you haven't started… Now is a good idea!
- If you don't have a partner: Let me know straight away!

If you get stuck:

- Ask Breixo: b.solinofernandez@students.uu.nl
- Join Slack: https://join.slack.com/t/infomcv2019/signup

ASSIGNMENT 2

Assignment 2 covers part of this lecture

- Deadline is Sunday February 24, 23:00

NEXT LECTURE

NEXT LECTURE

Voxel-based clustering

- Clustering of voxels
- Techniques: histograms, Gaussian mixture models, K-means
- Basis for Assignment 3

Next Thursday 11:00-12:45, RUPPERT-042