COMPUTER VISION 2022 - 2023

>03. SILHOUETTE-BASED VOLUME RECONSTRUCTION

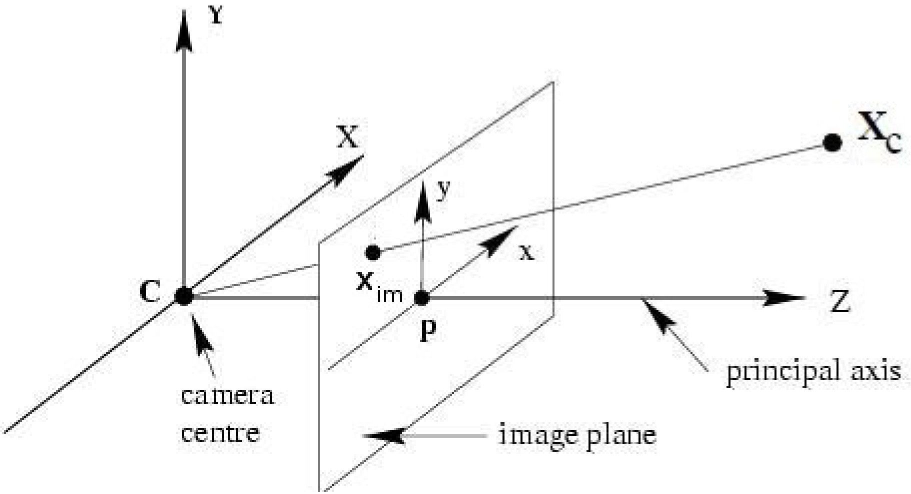
UTRECHT UNIVERSITY

RONALD POPPE

RECAP: 3D TO 2D

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

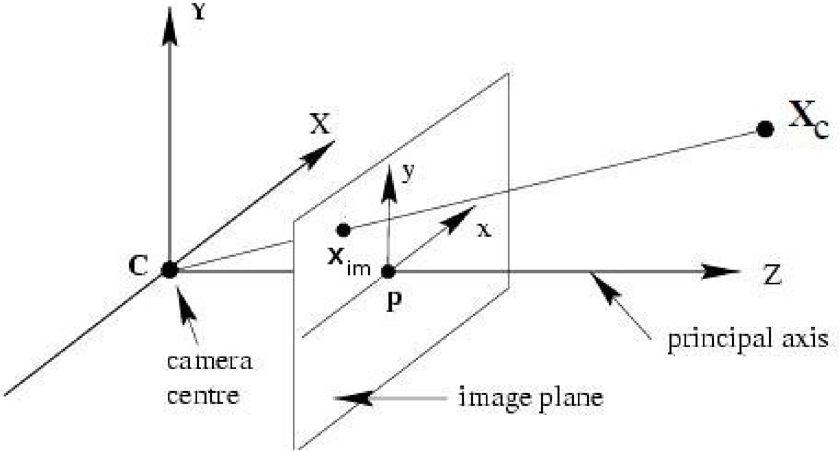
Given proper calibration, we can determine the intrinsic and extrinsic camera parameters and determine the projection



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 ($\boldsymbol{x}_{im\_x}$, $\boldsymbol{x}_{im\_y}$, f)
- 3D location is known up to a depth-factor

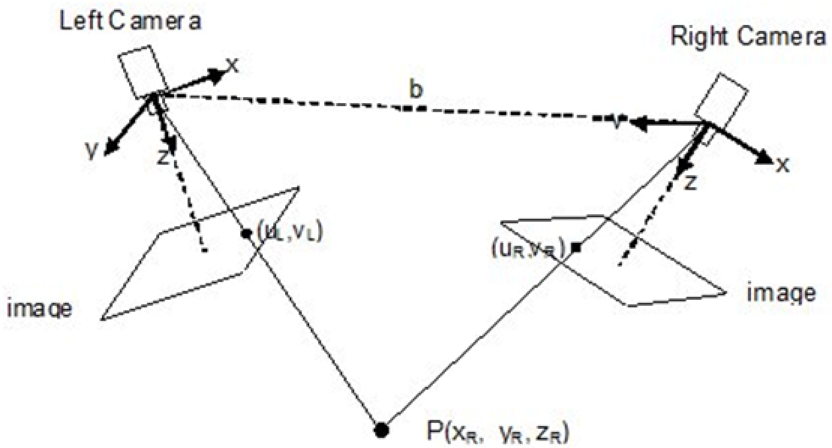


DEPTH FROM IMAGES

TRIANGULATION

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

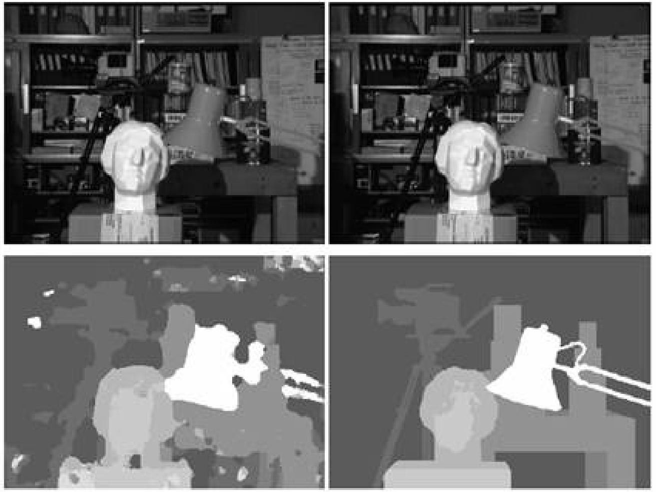
- Requires calibrated cameras

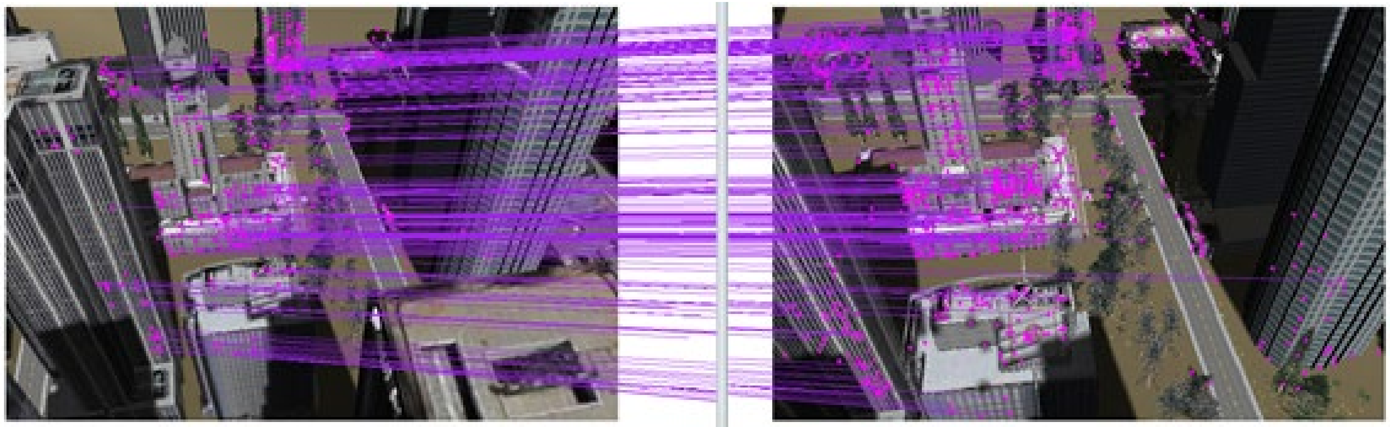


STEREO VISION

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

- Requires matching points across views
- Similar to how the human eyes work





STEREO VISION²

Good results are obtained for textured areas

- Easier to match points across views

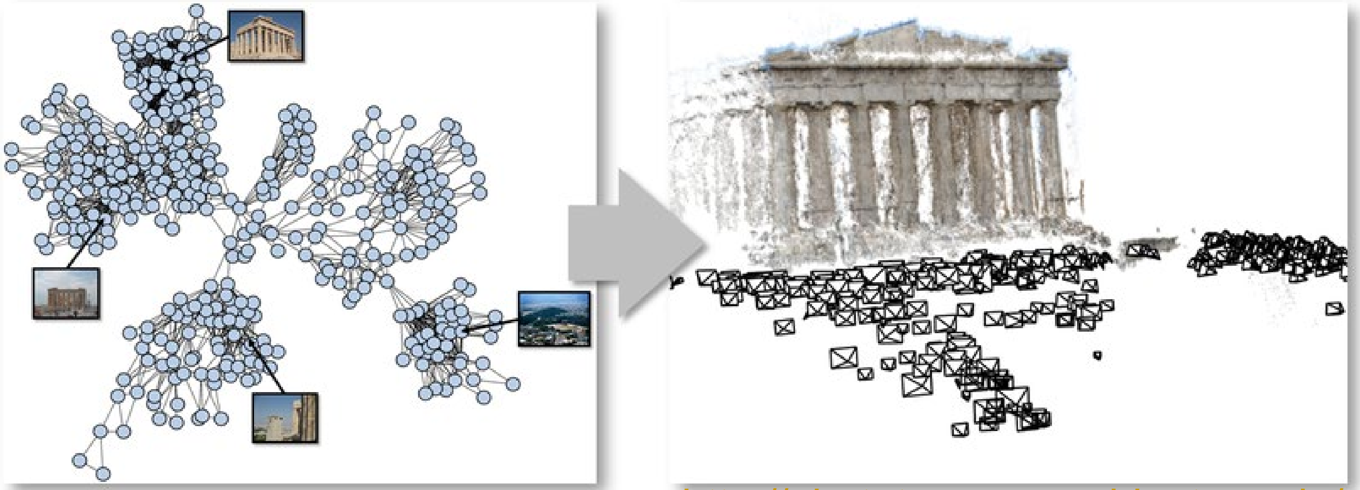
Generally bad results for evenly colored areas

- Lack of texture → No keypoints will be found

MULTIPLE VIEW STEREO

If more than two views are available, stereo matching for each pair of views can be attempted

Intrinsic camera parameters, extrinsic camera parameters and depth need to be estimated simultaneously: time-consuming!

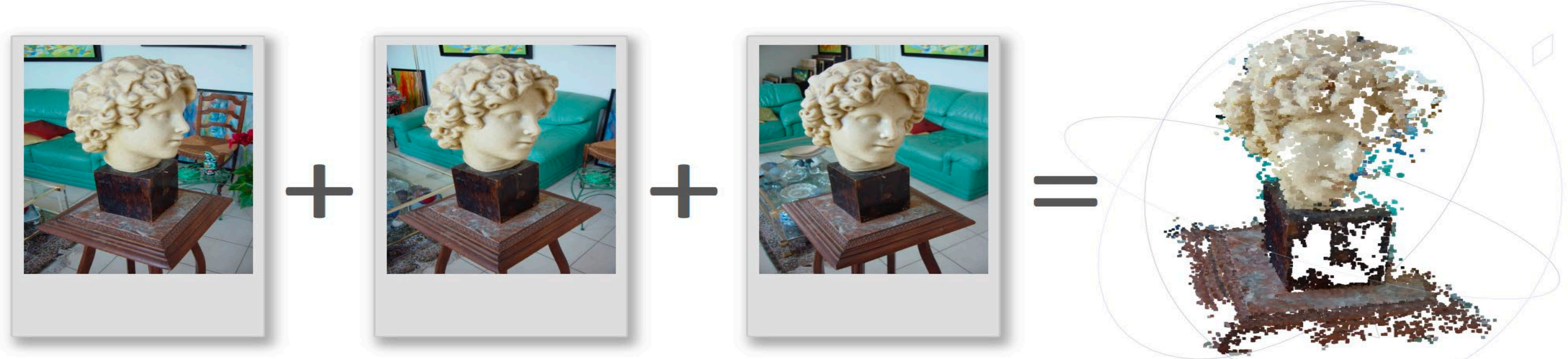


http://phototour.cs.washington.edu/

SHAPE FROM MOTION

In a video without zooming, 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



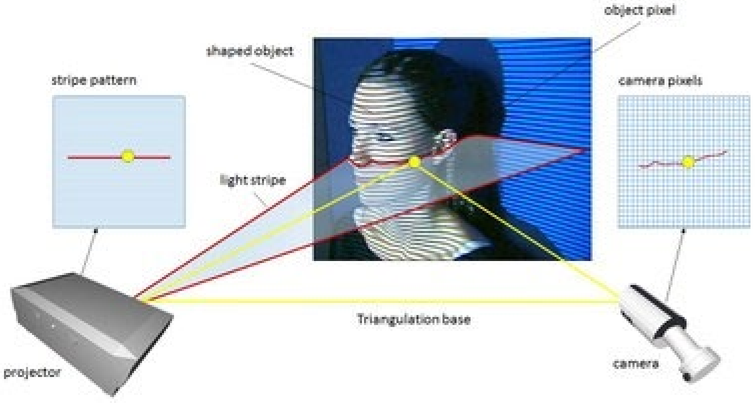
SHAPE FROM SHADING

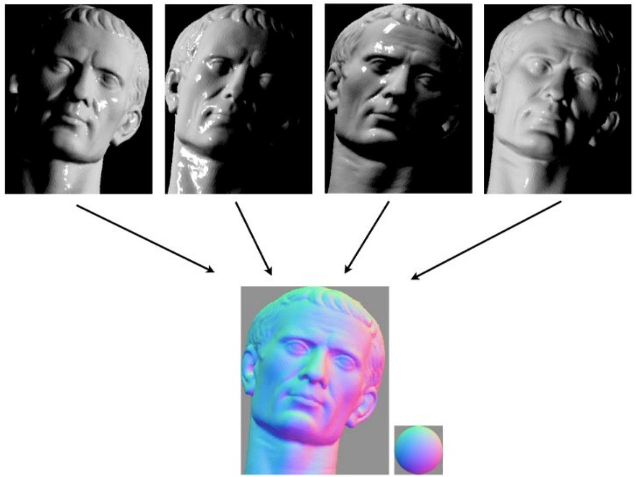
Estimate surface normals from inputs with varying lighting conditions

- Photometric stereo

Structured light provides systemic variation

- 3D shape can be covered

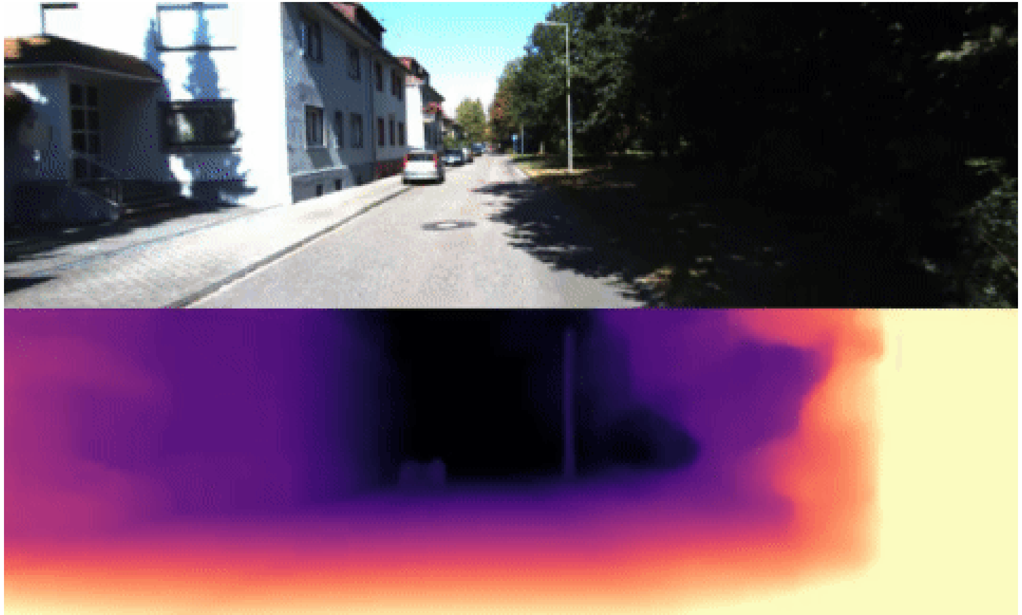




SHAPE FROM A SINGLE IMAGE

Learn a mapping from 2D to 3D using data

- Typically domain-specific
- Using deep learning

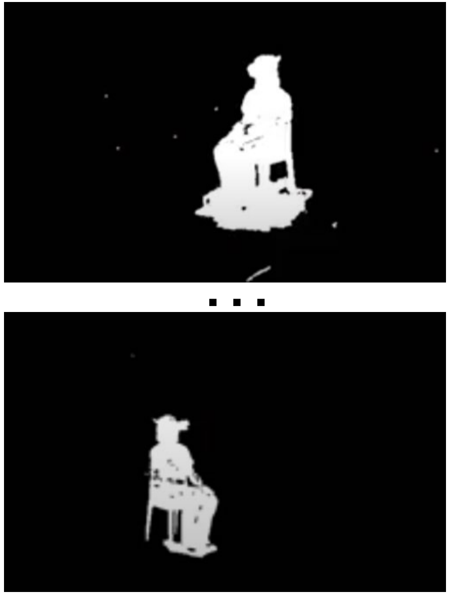


Godard et al. (2019) Digging into Self-Supervised Monocular Depth Prediction. ICCV.

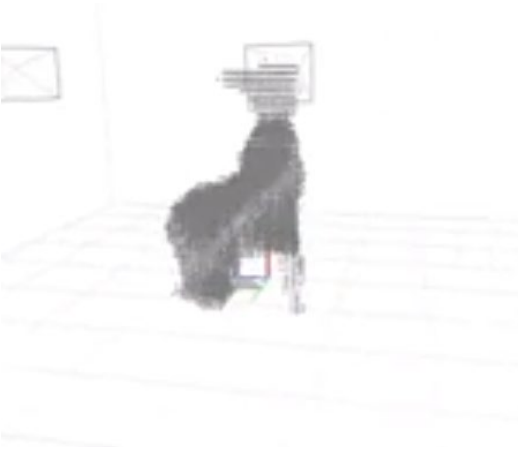
SILHOUETTE-BASED VOLUME RECONSTRUCTION

BASIC IDEA

1



2



3

BASIC IDEA^2

Use a number of cameras

- Calibrate them

For each view

- Separate the foreground from the background
- Perform volume intersection silhouette back-projection

BACKGROUND SUBTRACTION

Split image into foreground and background

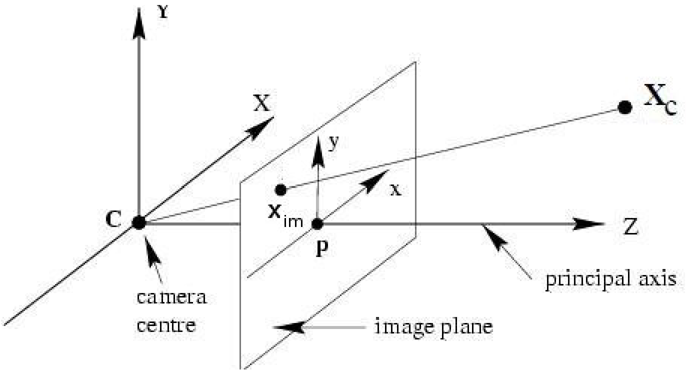
- Each pixel belongs to one of the two

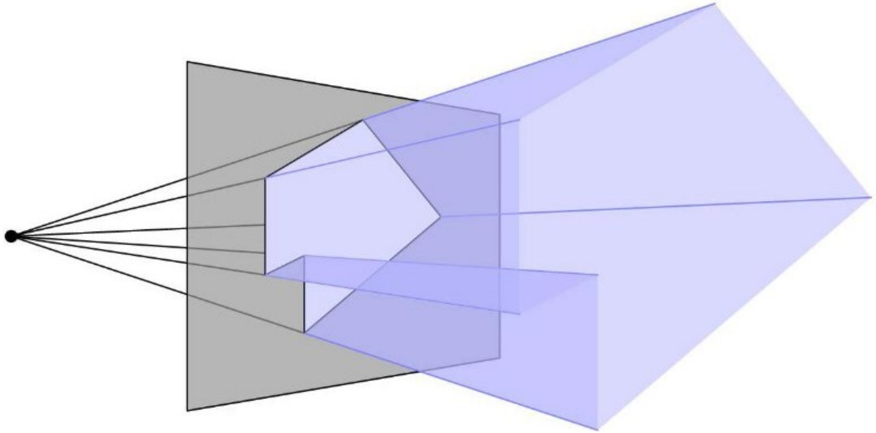
SILHOUETTE BACK-PROJECTION

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

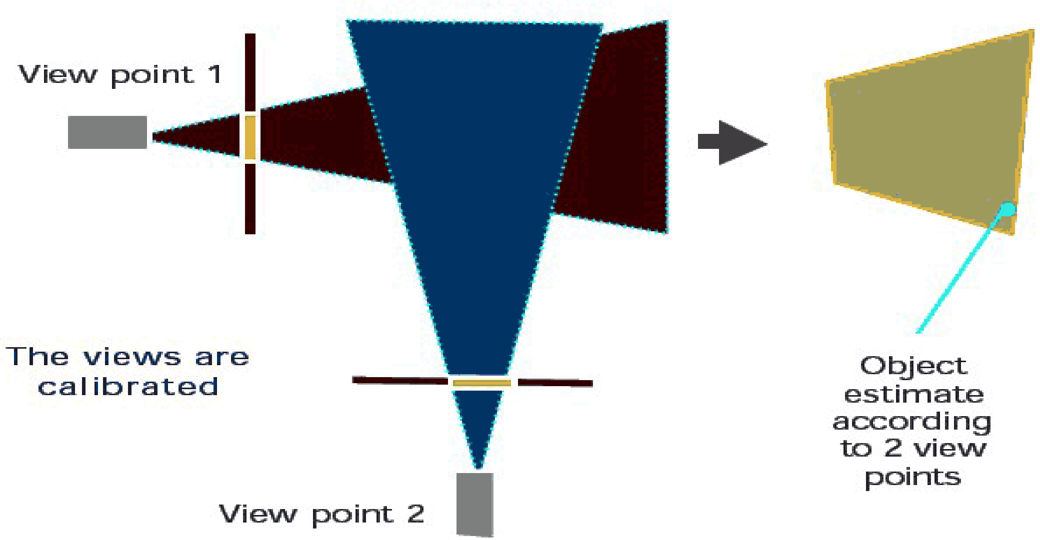




VOLUME INTERSECTION

For the silhouette of each view, we obtain a back-projected volume

- These can be combined by keeping only the overlapping part



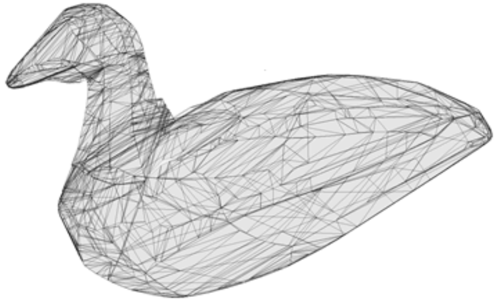
MESH

The intersections carve the shape out as a polygon model

- Typically loads of polygons if there are several cameras

Computationally expensive:

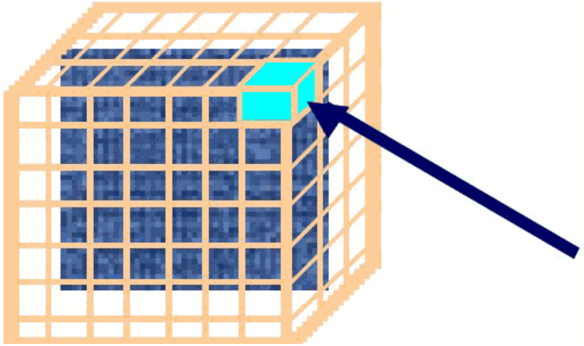
- Lot of storage is needed
- Lot of computation power is needed to determine the novel intersections (floating point operations)



VOLUME DISCRETIZATION

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

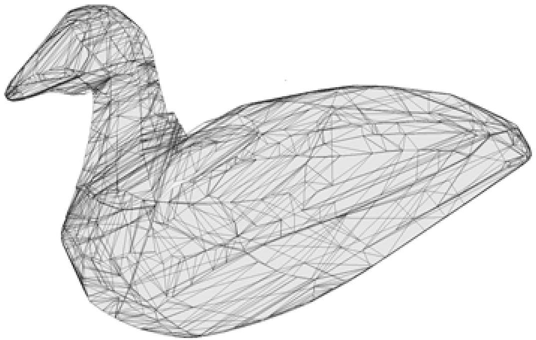
- A 3D pixel is a voxel

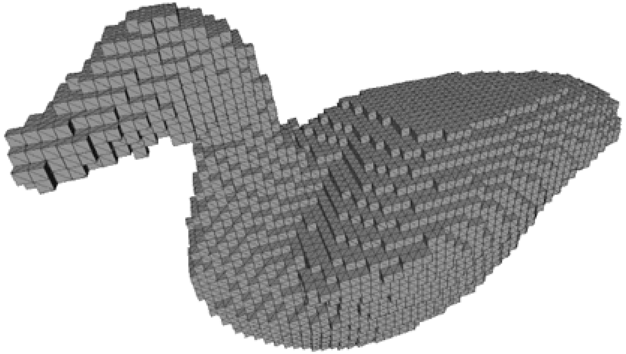


Typically many voxels

- 64 per row: 256k for volume

Still, detail is lost





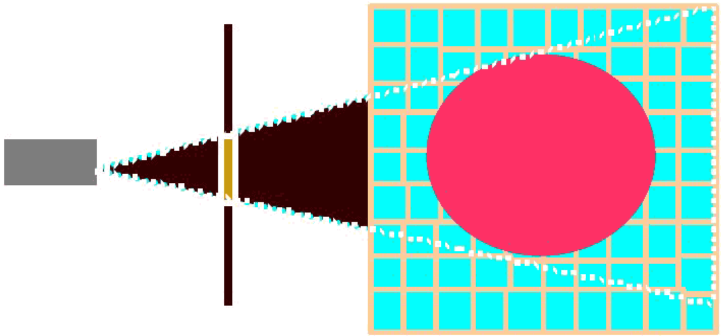
VOLUME DISCRETIZATION²

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



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

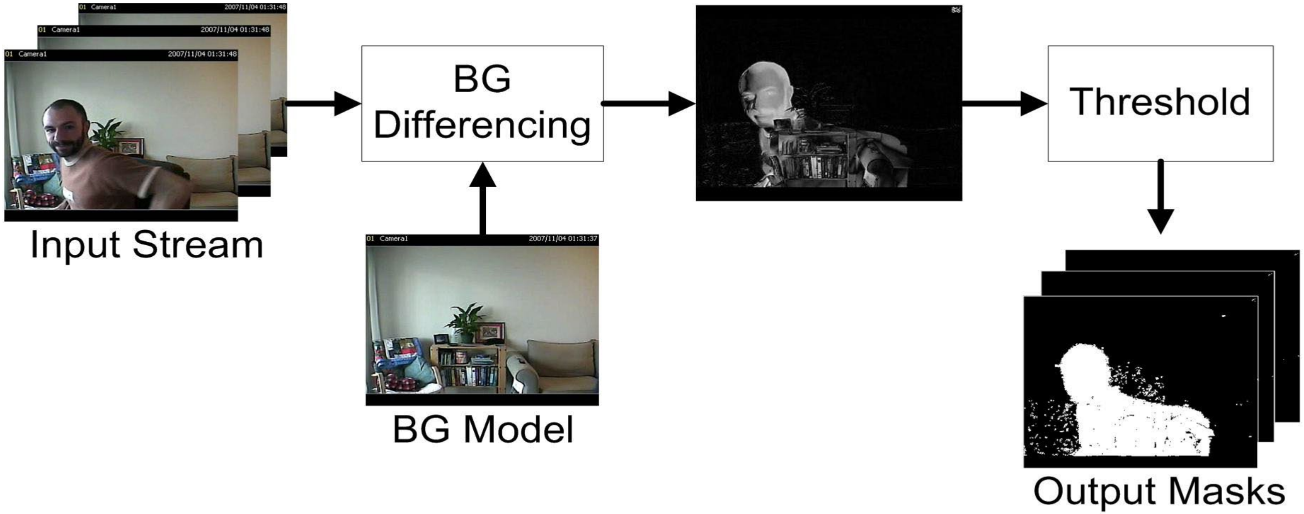
Chroma-keying or “the green-screen technique”

- 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 (green) background



CHALLENGES

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

CHALLENGES[2]

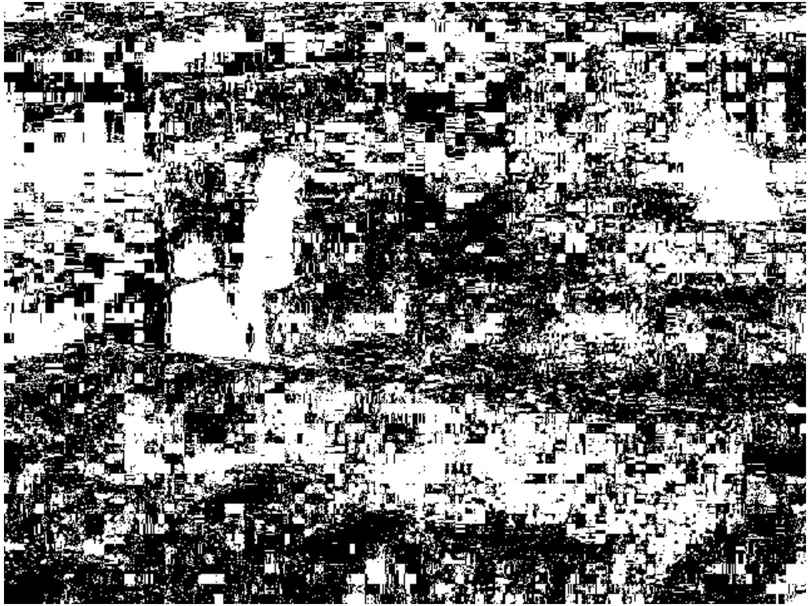
Variations and overlap in color:

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

CHALLENGES³

Colors in background can change over time

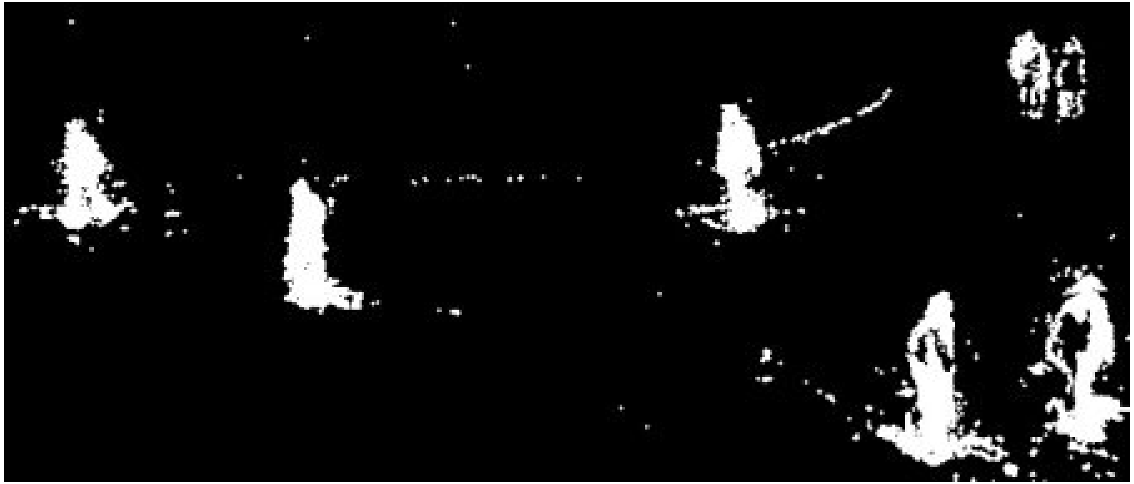
- Light (day/night)



CHALLENGES[4]

Shadows in the background:

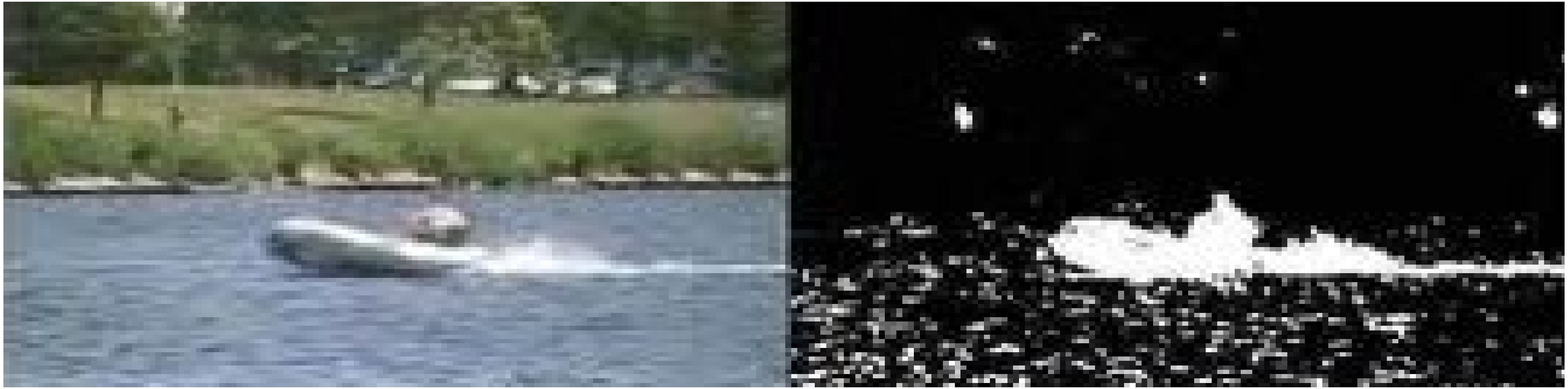
- Often “attached” to foreground objects
- Make background darker, without affecting the color too much



CHALLENGES[5]

Movement in the surrounding background:

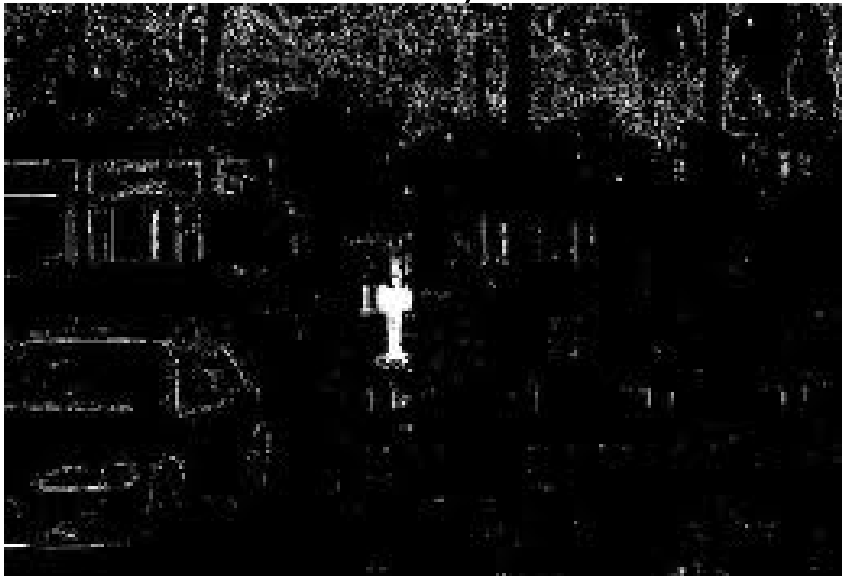
- Violates “static” assumption



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 reference picture *R* with exactly the same extrinsic and intrinsic camera parameters but without foreground objects

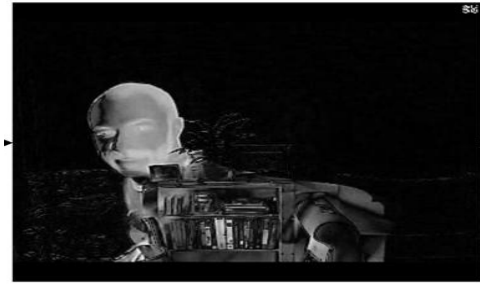
Compare each pixel in a new image *I* to the reference image *R*

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

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

MODELING BACKGROUNDS^2

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



And $D > \delta$:

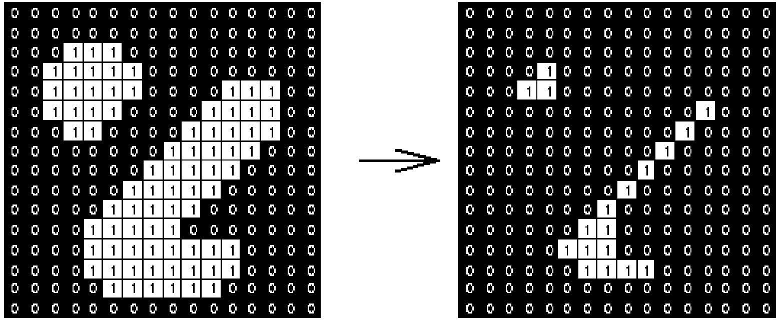
NOISE REDUCTION

To remove noise, we can apply morphological operations

- Binary filters that change a pixel depending on the neighbors

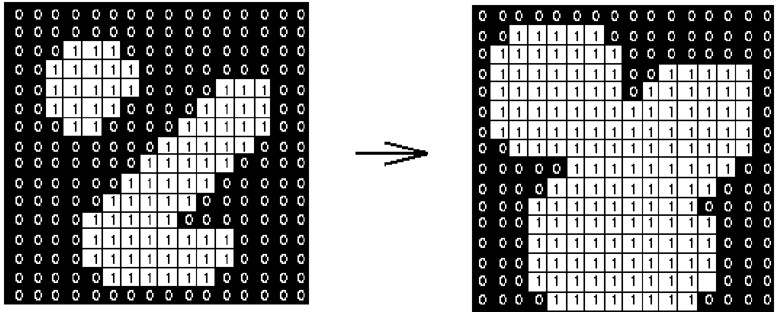
Erosion:

- Remove outliers



Dilation:

- Fill holes



GAUSSIAN MODELS

Issues:

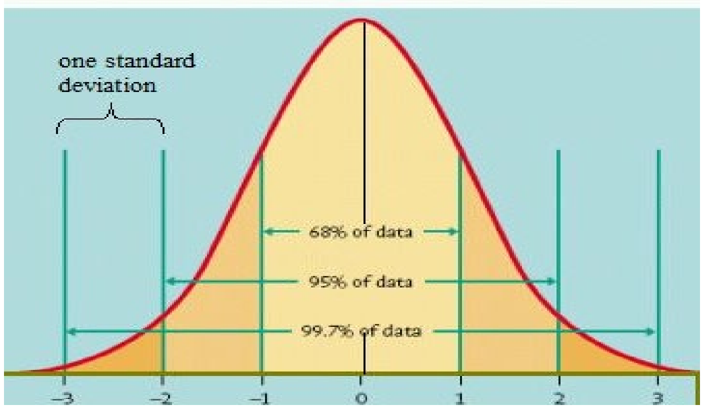
- Differences with lighter colors are typically larger
- Natural variation in the color of a background pixel

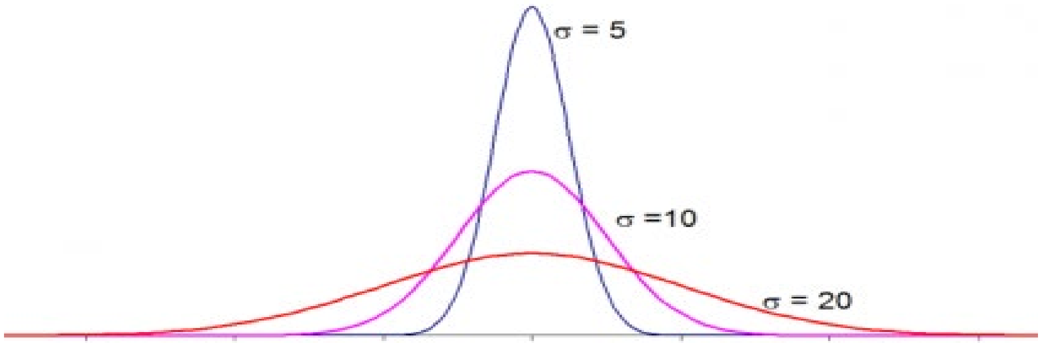
Solution:

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

GAUSSIAN MODELS²

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]

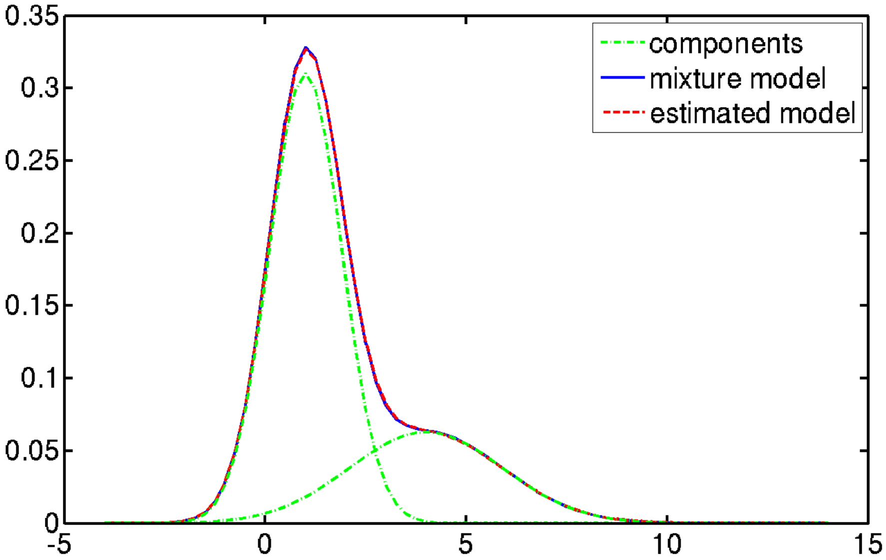
When doing background subtraction, we can have a threshold that determines how many standard deviations (instead of pixel values) a pixel's value is from the mean:

- 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.)

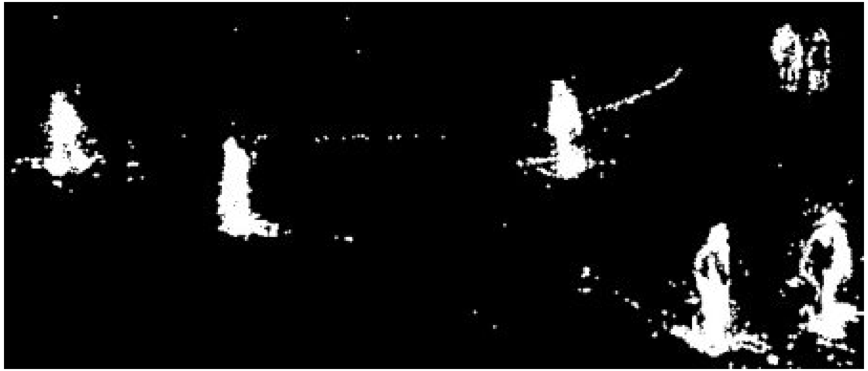
- Instead of a single normal distribution, use a mixture



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



BREAK

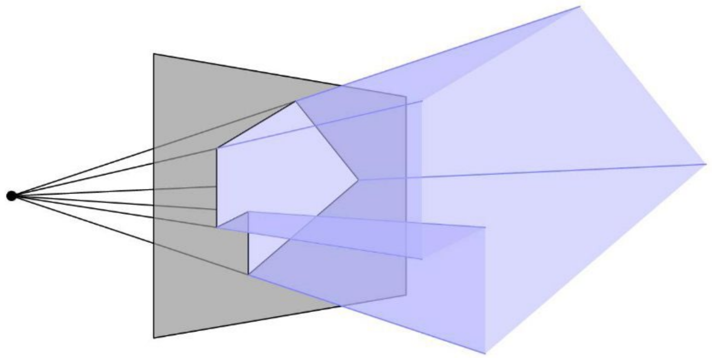
If you are interested in joining the intervision group to provide feedback, let me know!

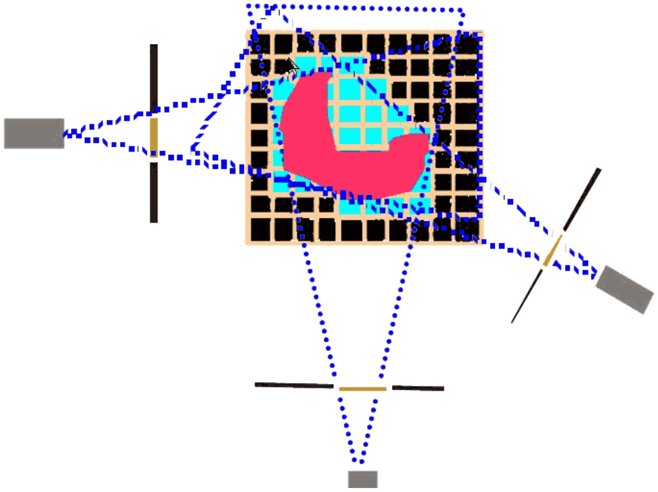
VOLUME RECONSTRUCTION

VOLUME RECONSTRUCTION

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

- Intersection of the 3D volumes from multiple views “carves out” the estimated 3D shape of the object
- Reconstructed shape is overestimation of true shape

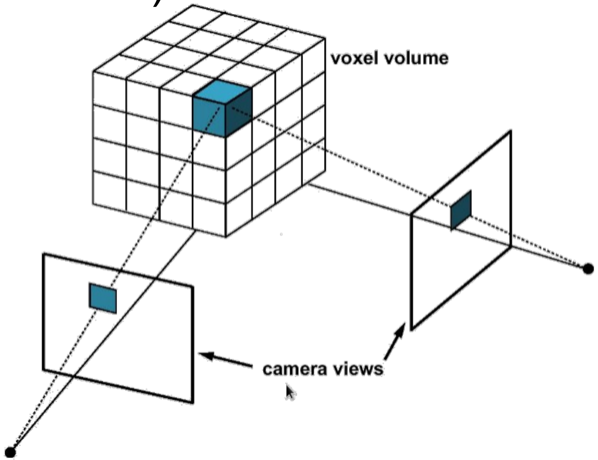




VOLUME RECONSTRUCTION[2]

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)



VOLUME RECONSTRUCTION[3]

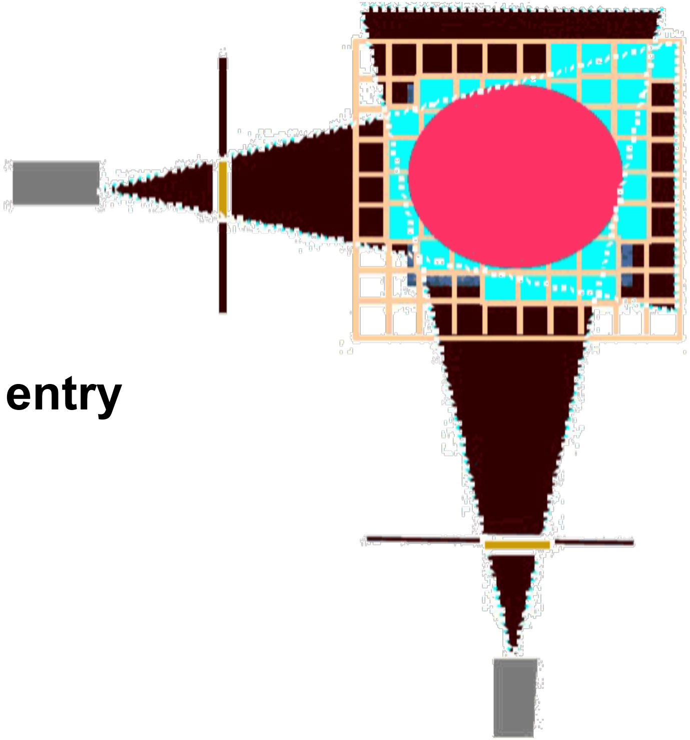
Voxels are either on or off

- Value depends on projection in each view

All "on" voxels together form the 3D shape

For each voxel, we can create a look-up table entry

- Voxel → pixel location in each view
- If pixel in all views is foreground → voxel on



LOOK-UP TABLE

For each voxel, we can determine per view where in the image it projects to

- Camera intrinsic and extrinsic parameters used for projection

A voxel is a 3D pixel with 8 corners, but we usually work with just the center

- Typically only a single pixel coordinate (center) used
- Silhouette extraction should be robust
- Area of pixels can also be used

LOOK-UP TABLE²

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 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 (on/off) 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 pixels (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 in the look-up table):

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:

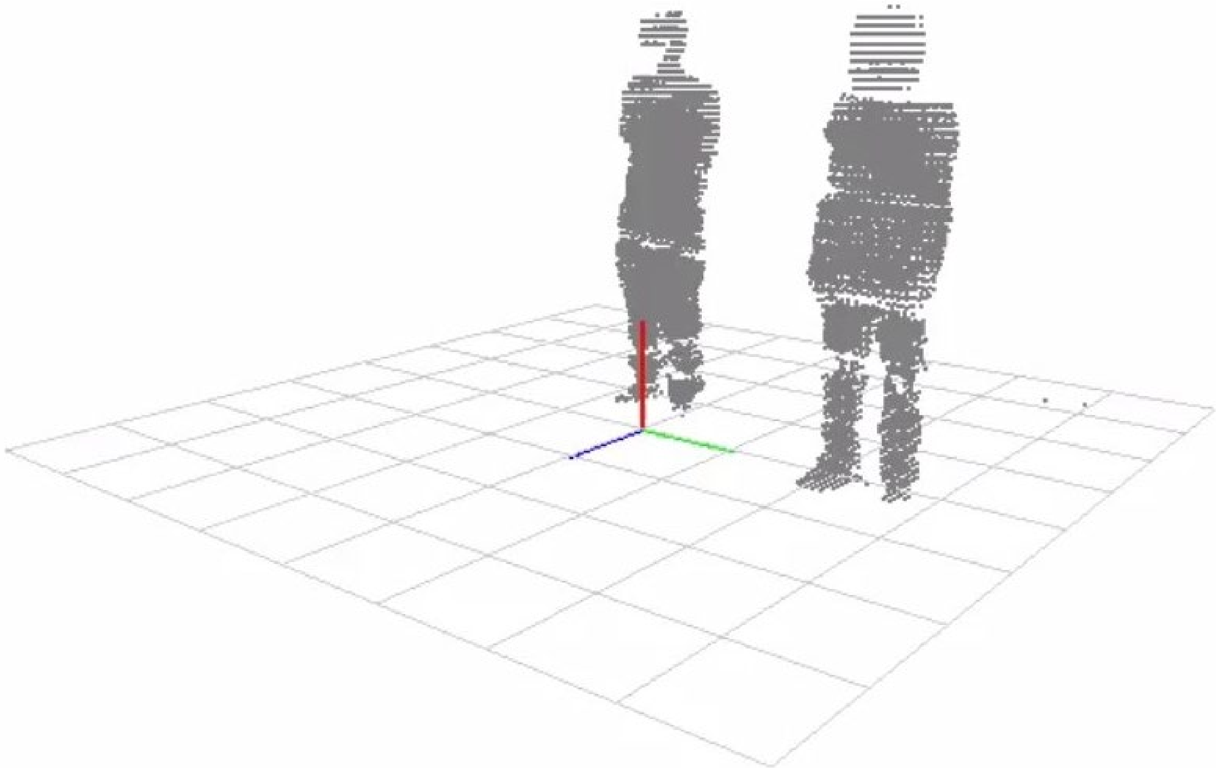
        Mark the voxel visible for view c

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, the voxel model is conservative:

- 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

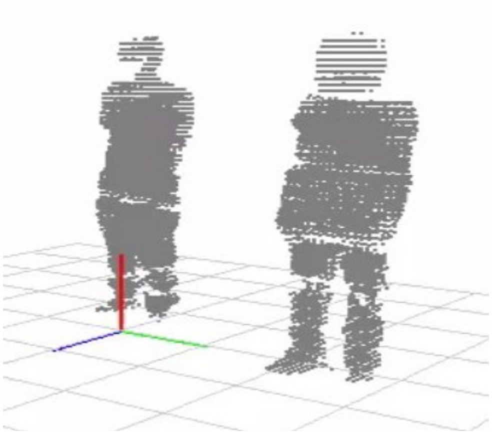
So play around with thresholds for background subtraction:

- 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



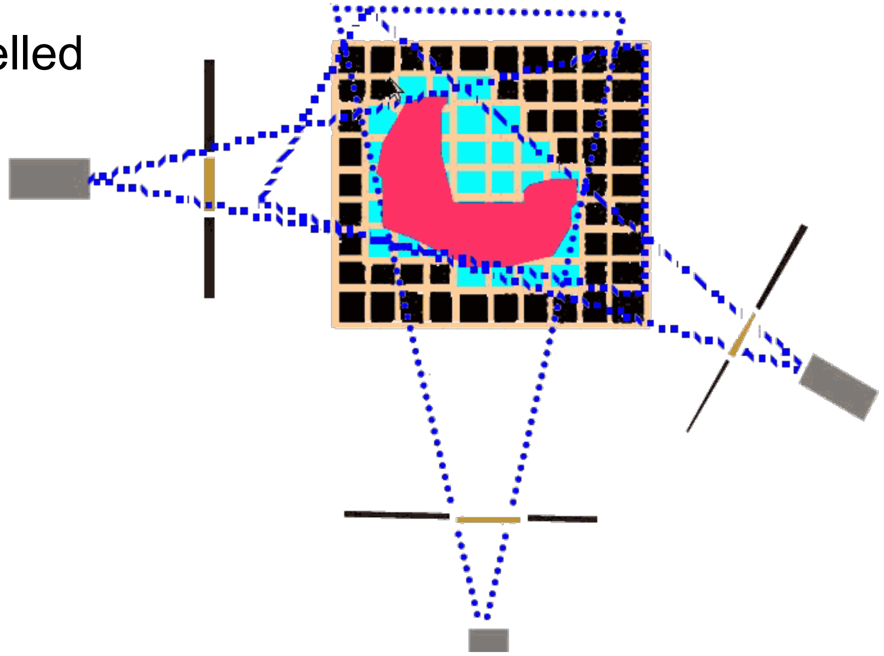
We can apply erosion and dilation also in 3D:

- 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

LIMITATIONS

Silhouette-based volume reconstruction has some of limitations:

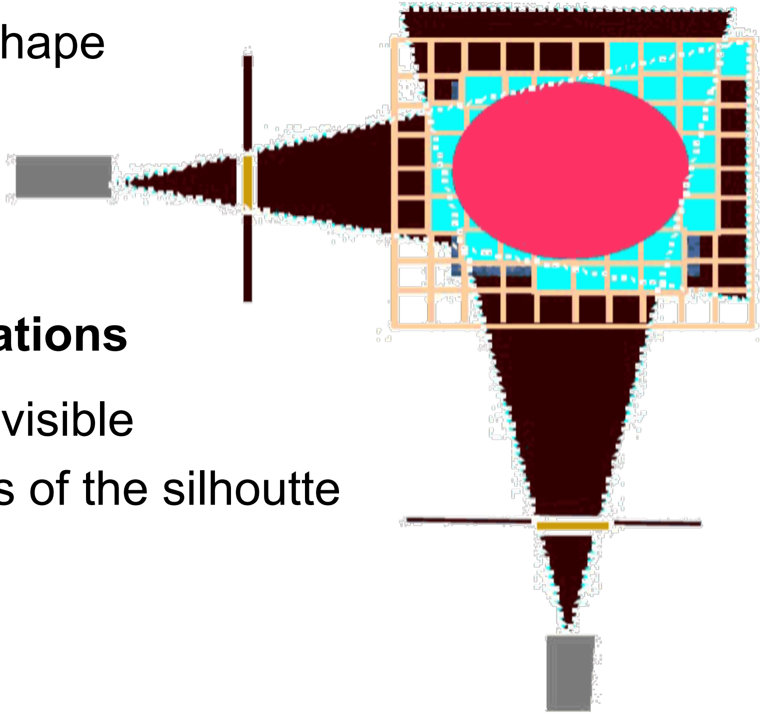
- Multiple views needed
- Concavities ("dents") cannot be modelled
- Depends on good silhouettes



LIMITATIONS²

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

- Estimated shape is bigger than actual shape
- Missing foreground pixels cause holes



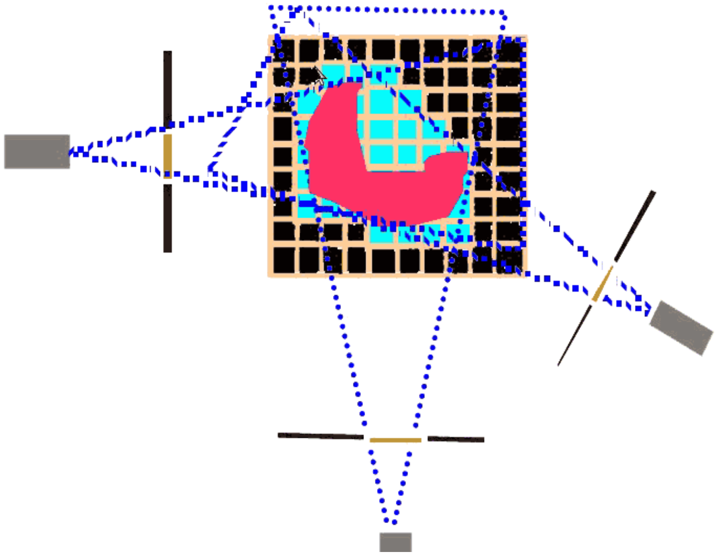
Views should be placed at “suitable” locations

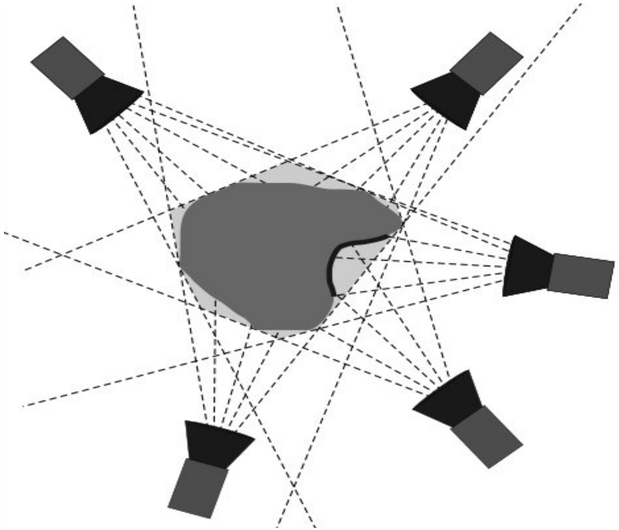
- Ideally, all parts of the object should be visible
- But shapes are determined at the edges of the silhoutte

LIMITATIONS[3]

Concavities cannot be estimated: only convex objects

- Not a single view “sees” the concavity, it cannot be carved out
- Adding cameras has no effect



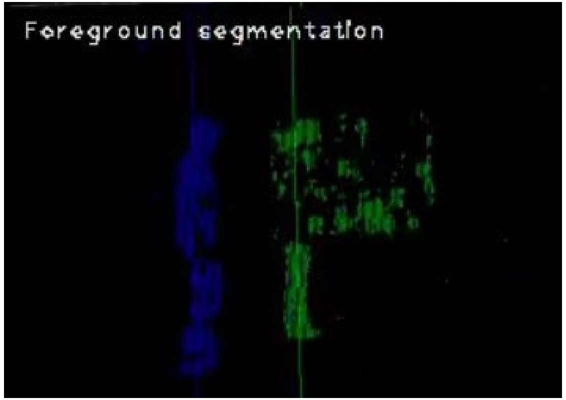


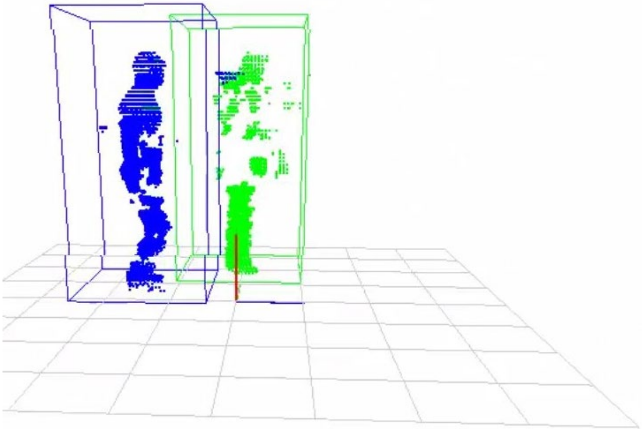
LIMITATIONS[4]

Incorrect background subtraction leads to noisy silhouettes

- Missing parts of the object
- Additional noise

In turn, the voxel model can be incorrect





EFFICIENCY: SPEED

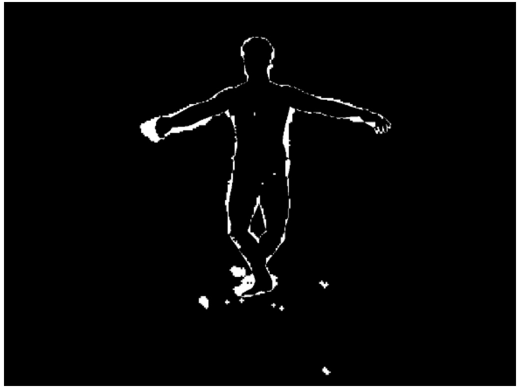
With video, changes from frame to frame are often small

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

Change in silhouette 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

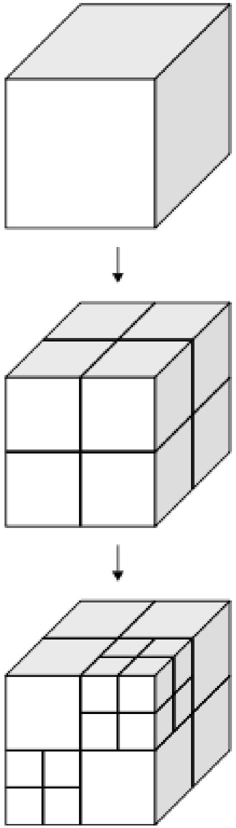
EFFICIENCY: STORAGE[2]

Octrees describe a volume as a string of 8 values

- Each corresponds to an octant

Values can be on, off or mixed

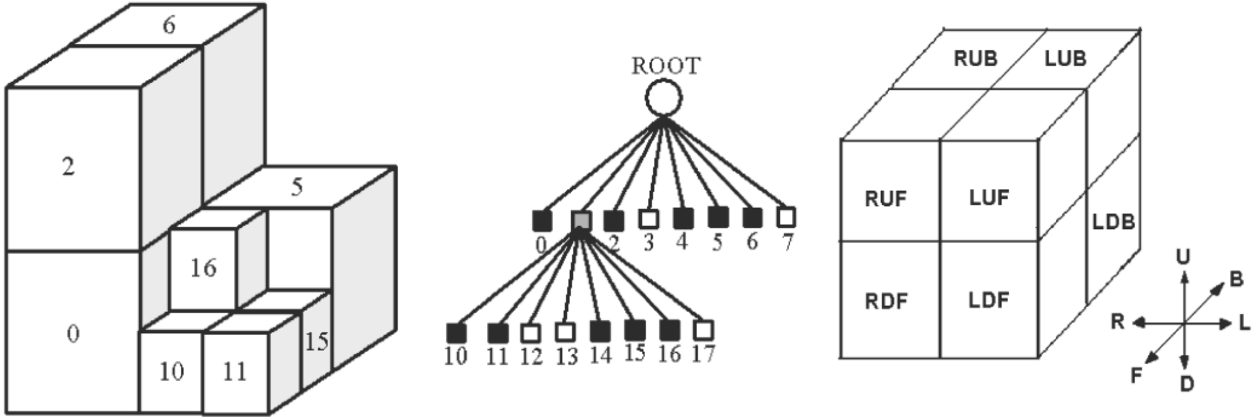
- Mixed values are recursively processed
- First level considers the whole voxel grid
- Final level considers individual voxels



EFFICIENCY: STORAGE[3]

Storing the tree saves a lot of space

- Changing a one or more voxels is less efficient



FROM VOXELS TO POLYGONS

Voxel models can be extracted efficiently but have drawbacks

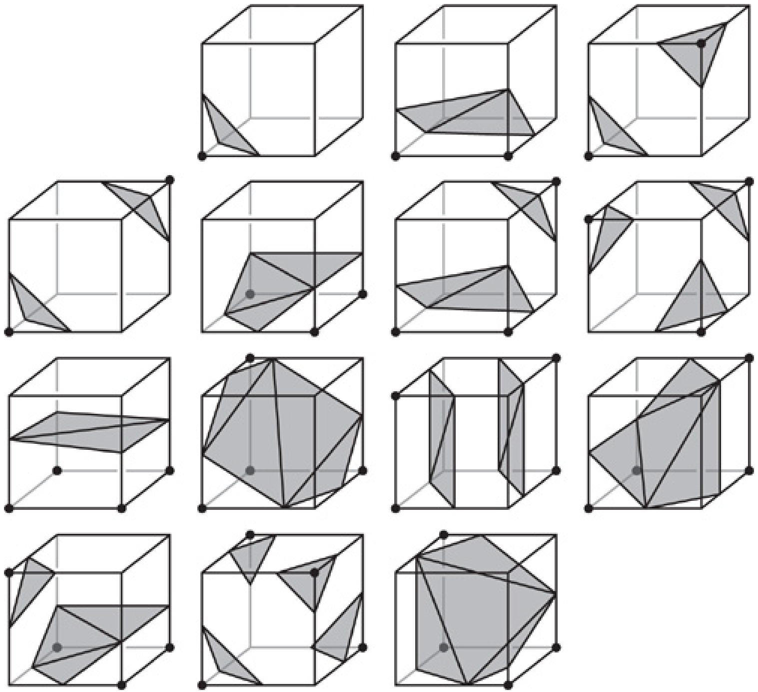
- Often low resolution
- Shape built up in layers
- No smooth surface

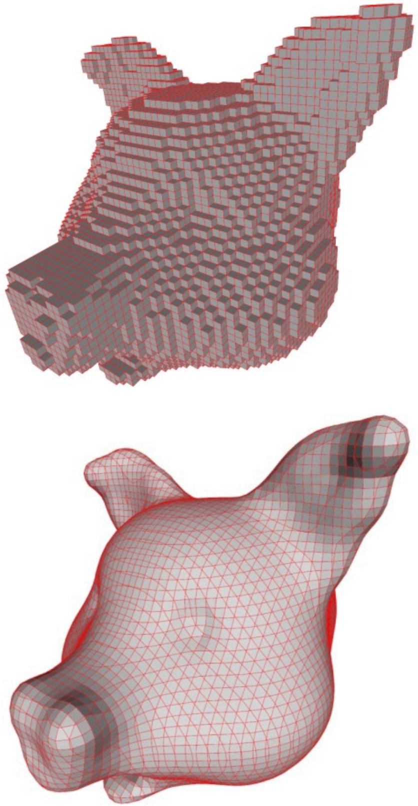
Voxel models can be converted to polygon models using the Marching Cubes algorithm

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

FROM VOXELS TO POLYGONS²

15 different possibilities for voxel patch





ASSIGNMENT 2 WALK-THROUGH

ALGORITHM RECIPE

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

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

3. **Determine the voxels that represent the volume of the target objects, by projecting each voxel onto each view**
   - If it overlaps with the silhouette, retain the voxel; otherwise discard it

ASSIGNMENT 2 WALK-THROUGH

Implementation of the silhouette-based volume reconstruction algorithm

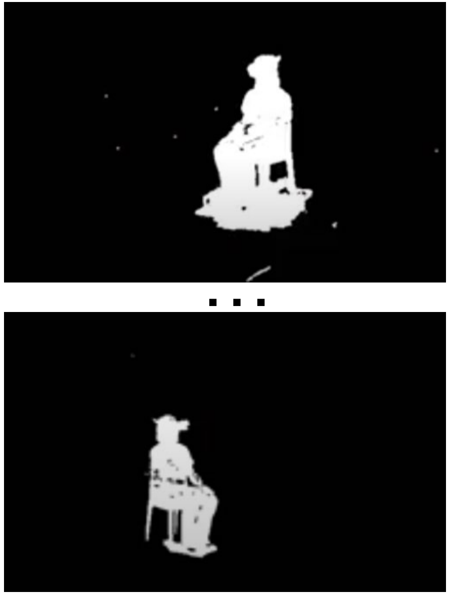
- Follow-up for Assignment 1
- Understanding challenges (blur, shadows, failing assumptions)
- Understanding multi-view geometry

Counts for 10% of total grade

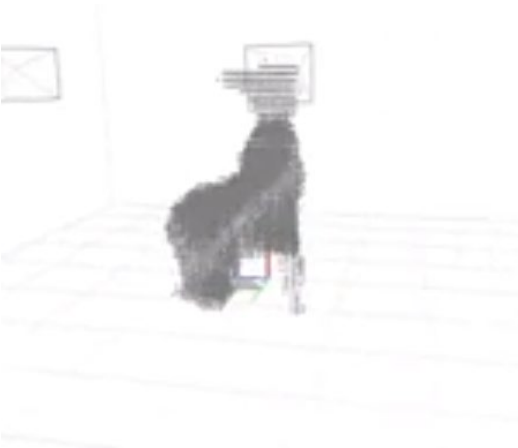
- 70 fixed points, 30 points in choice tasks

BASIC IDEA

1



2



3

1: CAMERA CALIBRATION

Calibrate the cameras

- Cameras will not move, so calibration only needs to be right once
- Make sure you store the camera intrinsics and extrinsics
- The cameras are of the same type but might differ in intrinsics
- Establish the relation between 2D (image) and 3D (world) to populate the look-up table. Check for function **projectPoints**.

Use your code from Assignment 1 for calibration and manual detection

1: CAMERA CALIBRATION[2]

The OpenCV findChessboardCorners will not work

- Use the function that you made in Assignment 1
- Of course: more accurate annotations → more accurate calibration
- Make sure the same corner across views is always annotated first (requires that you understand how cameras are relatively positioned)

1: CAMERA CALIBRATION[3]

You will be graded on the quality of the intrinsics and extrinsics (20 points)

- Put the latter in your report
- Check if the camera locations make sense, use the visualization code

2: BACKGROUND SUBTRACTION

Split image into foreground and background

- Distance between reference image and frame
- More complex options give more points

Post-processing

- Dilation/erosion

Check cvFindContours

- Tree of (nested) foreground areas
- Select “blobs” with minimum size
- Select largest blob

2: BACKGROUND SUBTRACTION^2

Reference image can be:

1. A single frame from the background video
2. An average of several frames from the background video
3. Per pixel the average and standard deviation over several frames

Average/SD per color channel

Memory efficient solution exist:

- https://en.wikipedia.org/wiki/Moving_average
- https://en.wikipedia.org/wiki/Algorithms_for_calculating_variance

2: BACKGROUND SUBTRACTION[6]

Shadows will be present:

- Have a stricter threshold around the feet?
- Be less strict when pixels become darker
- Easier in HSV color space, and using a Gaussians distribution

People are standing:

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

2: BACKGROUND SUBTRACTION[3]

How to know when your segmentation is optimal?

- So how do we set the optimal parameters/thresholds?

Two options:

- Create a mask manually and compare it to the segmentation result
- Introduce a criterion that measures the quality of the segmentation

In both cases: loop over all parameter settings

- Or use a smarter search algorithm

2: BACKGROUND SUBTRACTION[4]

Option 1:

- Use Paint or any other drawing tool
- Pick a frame from the video and color everything black except for the foreground → ground truth mask
- Compare the segmented segmentation to the mask (XOR?)

2: BACKGROUND SUBTRACTION[5]

Option 2:

- We want coherent segmentations without many holes and without many protrusions
- Use cvFindContour before and after (geodesic) image processing steps such as erosion and dilation
- If the image processing steps have little effect, we typically have a good segmentation

2: BACKGROUND SUBTRACTION[6]

You will be graded on both the sophistication (15 points) of your method, as well as the results (10 points)

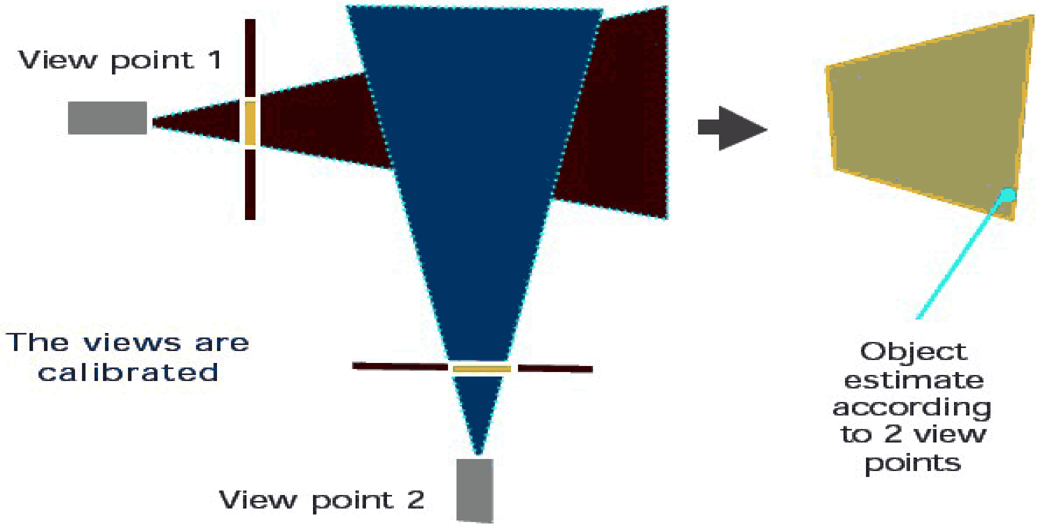
- Of course, more sophisticated methods CAN give better results, but only when applied well
- Look at where improvements can be made. Post-processing or in the background subtraction itself.
- The “hole” in the t-shirt of “the horse dude” is tricky to fix (why?), don’t spend too much time on this.

Quality is determined by lack of noise (e.g., in the ceiling) and lack of holes. Also, make sure legs of the chair are clearly discernible.

3: VOLUME RECONSTRUCTION

For the silhouette of each view, we obtain a back-projected volume

- These can be combined by only keeping the overlapping part
- Overlap can be discretized to a voxel volume



3: VOLUME RECONSTRUCTION[2]

Voxels are either on or off

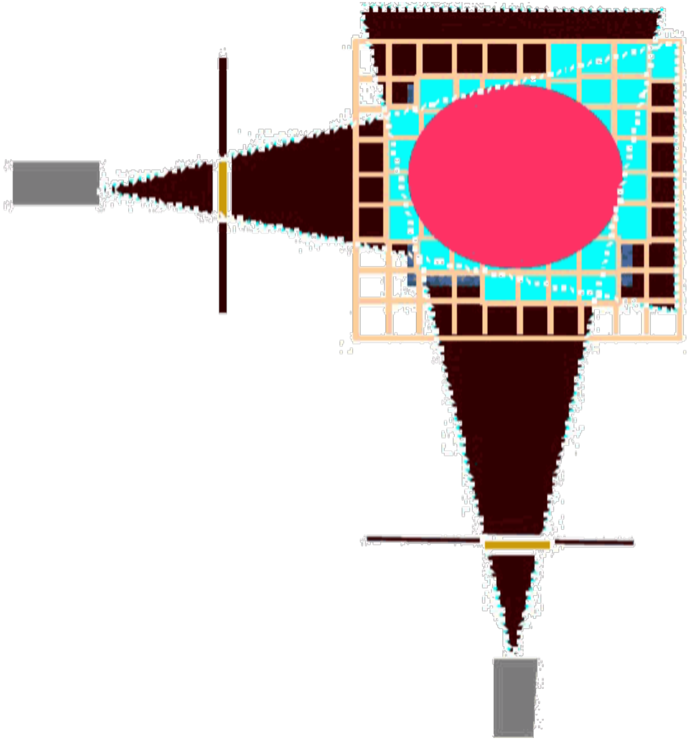
- Value depends on projection in each view

We can create a 3D look-up table once

- Voxel → pixel location in each view

We use the look-up table in every frame

- If pixel in all views is foreground → voxel on
- More computationally efficient



CHOICE TASKS

Find four chessboard corners automatically

- Probably requires image processing tricks, such as Hough transform
- Using the (Windows) magnification tool does NOT grant you points

Surface mesh

- Only exception to using third-party code
- Marching cubes

CHOICE TASKS²

Optimizing the construction of the voxel model

- Check the slides for a computational efficiency trick
- Will save you a lot of time while working on the assignment

Coloring the voxel model

- Think explicitly about the depth ordering to reason is a voxel is visible from a certain view
- Colors per view will be slightly different due to lighting issues
- Think how to deal with visibility from multiple views
- Distance to camera doesn't change. Store in look-up table?

CHOICE TASKS³

There are simple ways to speed up the processing

- Reduce the size of the voxel space during development (no points)

For calculating the look-up table

- Use parallelization: https://bisqwit.iki.fi/story/howto/openmp/
- Eligible for 5 choice points

REPORTING

Submit a 2-page report with:

1. An explanation of your methods.
2. The extrinsic camera parameters (rotation matrix and translation) for each of the four cameras.
3. The thresholds for your background subtraction (or the description of other parameters in your approach), and how it is determined if a pixel is foreground or background.
4. A brief summary of your choice tasks.
5. A link to a video (Youtube, Vimeo, Wetransfer, etc.) clearly showing the 3D reconstruction of the input videos. Make sure the link is accessible.

ASSIGNMENT

ASSIGNMENT

Assignment 1 due Sunday February 18, 23:00

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

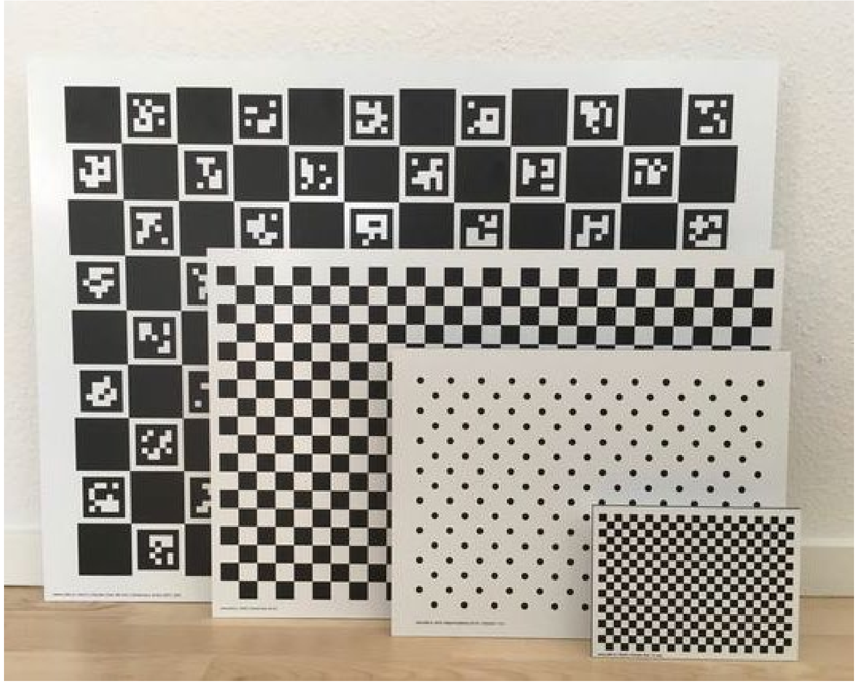
If you get stuck:

- Ask me now
- Ask on Teams

ASSIGNMENT^2

Calibration patterns explained

- OpenCV supports many methods



https://calib.io/blogs/knowledge-base/calibration-patterns-explained

ASSIGNMENT[3]

Assignment 2 deadline is Wednesday February 28, 23:00

- Framework available

NEXT LECTURE

NEXT LECTURE

Voxel-based clustering

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

Monday 13:15-15:00, BOL 1.206

QUESTIONS?

MATERIALS

Computer Vision: Algorithms and Applications (Szeliski)

- Chapter 12.7-12.8
- Chapter 13.1-13.2

Background materials:

- Real-time 3D Reconstruction
- A Theory of Shape by Space Carving
- Foreground detection
- Mesh Animation from Multi-view Silhouettes + Dataset
- Marching Cubes on Wikipedia