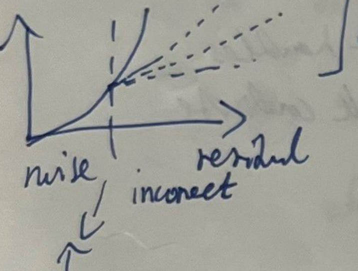
11/4/2022

3D computer Vision

.) Solve for the line and data [problem being 8 lemmas of really added data]

1) Add epipolar lines where point correspondences are

1) Inlier is on plot of data [on-line) vice versa



(redescending Algorithm
cuz of zero gradient problem)

-] RANSAC we can handle more than 50% outliers "supposedly"
requires "τ", designed for pose estimation

.) The larger the support the better the result] key intuition

1) The Algo [repeat N times]
1) Randomly select points, determine inliers
2) distance threshold + number of trials ↑ higher cost

) Our version can take into account method without "44"

$N = k(e, p)$

.) Image Stitching using SIFT + RANSAC !! [Joining planar scenes using epipolar geometry to stitch]

$\forall i \in [0, N-1], q_i = f(p_i)$

→ Solving 2 sparse liner matrix eq (5) (7)

i) precomputation in rest mesh

i)

for handles | $v_i' = c_i \quad (i \in C)$ C set of handles
c_i = handle coordinate

ring(i)

↓

i: 5,4,3,2,1

p

15/05/2022

3D Computer Vision

Self Calibration from projective Recon

1) Camera intrinsics when unknown ?

Homogeneous Coordinates
Finding H matrix

$$\underbrace{P}_{3\times 4} \sim \underbrace{K}_{3\times 3} \, [\underbrace{R}_{3\times 3} \;\; \underbrace{T}_{3\times 1}]$$

intrinsic, extrinsic/pose

$$\begin{bmatrix} \cdot & \tau & x_0 \\ 0 & \cdot & y_0 \\ 0 & \cdots & 1 \end{bmatrix}$$ upper triangle

$x_0, y_0 \rightarrow$ projection on retinal plane

~~focal length~~ All in pixels/length

α = aspect ratio $\rightarrow$ for camera pixels

τ = axis of the sensor / skew

assumption $\frac{x_0}{2}, \frac{y_0}{2}$

$\tau \leftarrow 0$

$\alpha \leftarrow 1$

- "Typically Self Calibration is finding focal length"

$P_1, \ldots, P_n$

$\downarrow$

$K_1, \ldots, K_n$ $\quad \boxed{K_i = K \;\; \forall_i}$

5/4/3 assumptions

Photogrammetry being diff cam of varying al positions

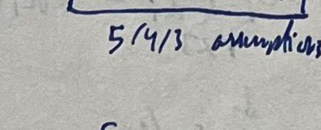
Self Calib ambiguity

$f_1, \ldots, f_n$

n unknowns

Euclidean projection from something else

.) Projective Recon



w_{ij} $\quad i \in [1, m]$

$j \in [1, n]$

$$w_{ij} \sim \begin{bmatrix} x_{ij} \\ y_{ij} \\ 1 \end{bmatrix} \qquad w_{ij} \sim P_i Q_j$$

$v_{ij} \sim P_i Q_j$

$H_{4\times4}$ det $H \neq 0$

Lifter space/dimension homogenous

$P_i H H^{-1} Q_j \Rightarrow \boxed{v_{ij} \sim P_i' Q_j'}$

[Change in coordinate frame]

2 sort proj basis/Euclidean

Self calib = is find 'H'

Proj recons is solving this

Brings points at infinity to plane

$\left[P_1, \ldots P_n \mid Q_1, \ldots Q_m \right.$ st $v_{ij} \sim P_i Q_j \ \forall_i \ \forall_j$

1) Scene constraints are a big problem
∴ assume accuracy to scene

Euclidean only if $P_i' \ldots \to K_i'$

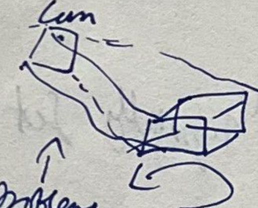
∴ Self calib is finding H back to euclidean

1) Absolute Conic assumption / Affine cube
different types to include diff
cones / infinites / going back each
time [statistical approach]

Plane at infinity is a set of directions

e.g. Use of known eq for not using projective projection

1) Constant or varying parameters [scenarios]



1)4 changing projective basis

1) 8 unknowns remain 'H' out of 15.

problems
– Bundle adjustment for finding distinct coeff

1) Finding 8 var in 'H' the eq becomes in 'Z'

$\exists K$ st $\forall i$ $P_i Z \sim [\underbrace{K R_i}_{3\times3} \ -K R_i t_i]$

↓
up to scale

revision it

$P_i Z_i \begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \end{bmatrix} \sim R_i K$

$\hookrightarrow P_i Z \begin{bmatrix} I \end{bmatrix} Z^T R_i^T \sim K K^T$

X is symmetric

$X \sim Z \begin{bmatrix} I \end{bmatrix} Z^T$

$\left. \begin{aligned} P_i X P_i^T &\sim K K^T \\ P_i X P_i^T &\sim K K^T \end{aligned} \right] P_k X P_k^T \sim P_e X P_e^T$

Varying focal length Calib

$K^{-1} P_i X P_i^T K^{-T} \sim \begin{bmatrix} f & & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \simeq 0$

reduces to a linear eq which is easy to solve

Presentation discussion on differentiable rendering

1) How raster makes light:
   1) Save BRDF where light will be reflected, ~~that~~
   2) Pixel knows where each triangle came from
   3) Shadow maps for shading
   4) Shader produces all lights, colors etc

·) Reynolds transport theorem:
   1) generalisation of leibniz rule

1) slide ~~14~~ 14:
   L_{ij} p: point ~~at~~ used to sample the pixel space with rays
   · u_k : random rays

1) Slide 11

Reynolds transport theorem [This is the general form] S^2 could be H^2 ↓ Hemisphere

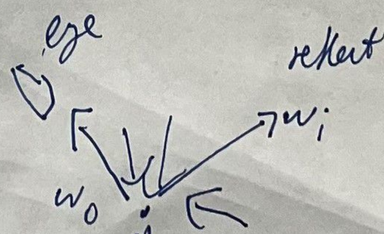
1) We calculate the differential irradiance

1) given the incident direction "w" the integrant fe of the original integral is independent of π ∴ interior $= 0$

·) Boundary integral is not zero! This integral is over the discontinuity boundary of fe which are directions pointing towards the edges of area light

·) $V_{\partial H^2}(w)$ = how fast boundary moves wrt π.
∴ each point "w" ~~at~~ on the boundary it is the scaler change rate velocity along the normal direction. To compute this we parameterise the boundary curve, ~~Any~~ Any parameterisation is ok since scaler normal velocity is parameterisation independent.

1) General term now has BSDF



1) Slide 15:

θ = parameters w = direction	Angular support: Integrands vanish outside a small neighbourhood.

Slide 15: θ_0 used cuz

transformation 'T' should be designed to yield the identity map $\theta = \theta_0$.

θ_0 = concrete para value for which gsch are computed.

so transformation does not affect the computation of $\underline{I}$.

-) radiance : amt of light ~~[illegible]~~ incoming to a point from a single direction

irridance : amount of light incomming to a point from all direction

1) Intrinsic image decomposition: The problem of ~~[illegible]~~ decomposing an image in 2 layers: 1) reflectance / albedo inversed color of material and 2) shading, product by the interaction between light and geometry

landmark free statistical shape modelling via Neural flow deformations

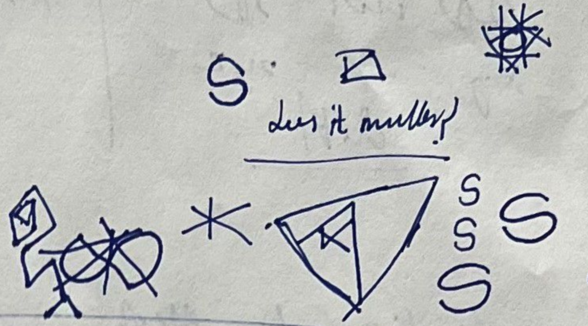
12/12/2022

1

1) Sim field vector space = distance that it will move
2) generate mesh from partial data
3) Implicit or real representation
4) Chamfer dist for inference + optim
5) study Neural ODEs

S

does it matter?



1) Train domain specific Mesh deformation Latent space
2) propagate grads to trained latent diff space to better deform based on image differential from renderer.
3) Perform comparison in CNN feature space for better grad understanding

If not this then heavy reliance on computed loss.

pix to face [7][3][C] = face on mesh

12 1

11 same

3x

12

6x6 6x6

4x4 4x4

8x4

4E

4U $\times$ 4E

1x1

1x1

or out

4U

6 6

3 3

2 2

4x4 4U

4 out

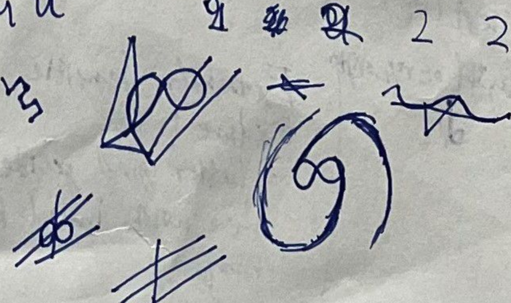
2 2

pix to face = 'f' given the coords
zbuf = z-coords of nearest faces at each pixel
bary coords = [w0, w1, w2] = bary[n, x, y, z]
dists = [N, x, y, fP]

$(\vec{p}-\vec{c})\cdot(\vec{p}-\vec{c}) = r^2$

$\vec{r}(t) = \vec{o} + t\vec{d}$

$\forall \vec{p} \in \mathbb{R}^3 : \vec{p} \in S \leftrightarrow (\vec{p}-\vec{c})\cdot(\vec{p}-\vec{c}) - r^2 = 0$



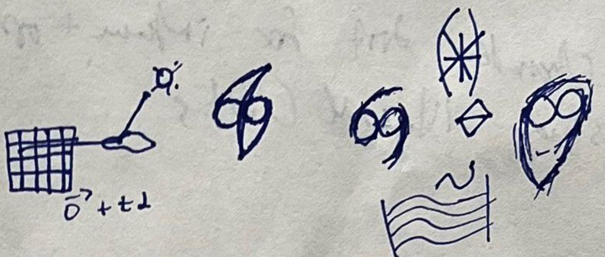
1) Projected vertices : | Binary Classification of vert diff

1) Registration pyzes | 1) Model in space to decrease error in primitives.

1) 2D and 3D | 1) Test with primitives

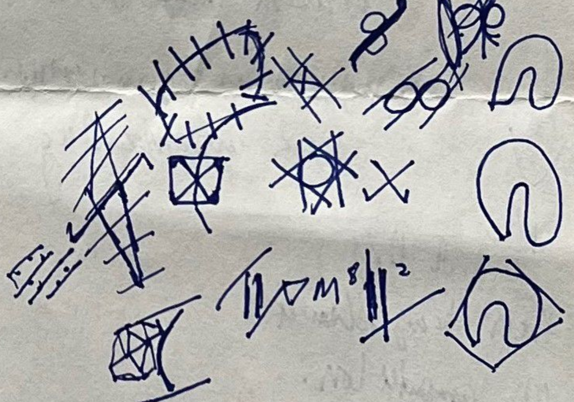
xy - xyz

diff

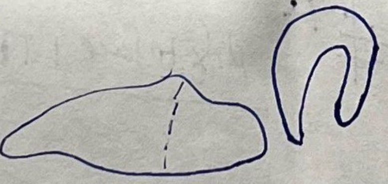


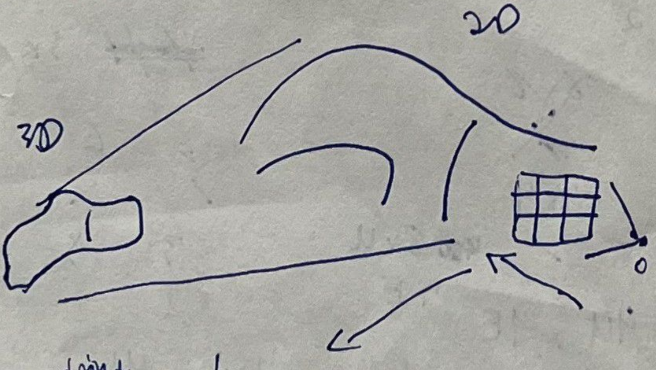
∴ Initialisation of a Mesh in the form of surface extraction from style image

1) Code given but doesn't provide instructions on how to run with own images
- Hobler
- defocus blur
- changing lighting cond



ray intersection in world coordinates should return xyz considering the same scale etc





$$\cancel{\partial I} \quad \frac{\partial I}{\partial V_i} = \left[\frac{\partial I}{\partial V_x}, \frac{\partial I}{\partial V_y}, \frac{\partial I}{\partial V_z} \right]_N$$

ray = origin + world coordinates

ray_intersect(ray)

ray = mi.Ray3f(o=ray_origin, d=

Extremely sensitive to texture!

∴ Better grad guide can be done with initial texture map

would the gradients be enough to guide the mesh to loosely register with target?

Nope!

Deep CPD, Fast DR prove otherwise

∴ implement seperate MLP for deformation based on grads from $\frac{\partial I}{\partial V_i}$

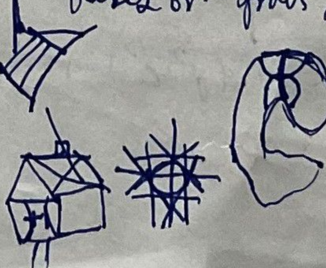


Table for renderers

~~Input~~, output, method

- Better initial guess

Little step test
and
Big step test

Full transform No clip
~~-) Mesh thrown out~~
-) Zoomed in middle

Top silhouette | Problem might be rendered reference image

Check normal rendered image
vs
Mask silhouette image

Works good

-) Divide 256 Check results

$\boxed{\text{better}}$

Normalise grads

Works good

Transform function overview? ✓	clip-ranges ✓

Scheduled L.R ??

Imp ✓

74, 111,

Later Imp

most-render 37? still a prob ✓

error = 0.088 - Low L.R

error = 0.048 x10 L.R | 0.0025

[Multiple solution problem based on L.R]

[So does initial condition matter?
Yes in the Last experiment
∴ Not fully robust to initial environment]

[findings]

1) Normalising Ref Img veg Imp

1) L.R ~~causes~~ choice causes reaching solution with multiple solutions

CLIP Values ∴ L.R + Clip values really imp to choose

understand clip vals ✗

[why we can do it]?

Have to Limit vals
can't let them have wide ranges

Best Res [L.R 0.05
trem −0.8, 0.5

x −0.5, 0.5
y −0.5, 0.5
z −0.5, 0.9] Img 50% + 100%
with
render_38.png
render_77_BW

0.09 L.R

- Use quaternions
- redo exp with partial view

-) If pose is perfect ∴ [illegible]

1) Exp observation! [discontinuities in grad causing Jumps]

Exp show that no limits work for easy cases but not for complex ones

[Render matching is important]

13, 15, 18, 138