Render - Register - Deform

1) Deep CPD [Category level 3D Non rigid . . .]

-> Imp concept [deformation fields] for visible parts
+ shape [latent] space for inferring deformation of occluded part

-> flow net 2 [modified with 2RGB, 2 mask]
↓ out
[3 channel for x y z for each point]

-> Coherent point drift [CPD]

2 sets D-dimensional X, Y

CPD ⟶ (output) deformation field mapping Y to next X

- So we consider Y points as centroids of Gaussian Mixture Model
- X points are considered as samples drawn from these GMMs
- equal isotropic covariances σ^2 and equal membership probabilities
$P(m) = \frac{1}{M}$ for all GMM component
- CPD outputs ↑ a deformed point set as Maximisation of the probability of X being drawn from GMM, by moving centroids Y
- in a coherent matter [Expectation Maximisation process]

$\therefore T \Rightarrow T(Y, v) = Y + v(Y)$
↓ initial pos ↘ Displacement function

coherent point drift for any D-dimensional set of points $Z \in \mathbb{R}^{N \times D}$, v is defined as

$$v(Z) = G(Y, Z)\, W$$

$W \in \mathbb{R}^{N \times D}$ weight matrix
D dimensional deformation vector

$G(Y, Z)$ element wise $G(y_i, z_j) = \exp\left(-\frac{1}{2\beta^2}\|y_i - z_j\|^2\right)$

↓ estimated in M step of EM algo

- method

$C \in \mathbb{R}^{n \times 3}$ point cloud
$C_m \in \mathbb{R}^{m \times 3}$ mesh

$$C'_m = C_m + \underbrace{G(C_m, C)}\ \underbrace{W(C, D)}, \quad W(C, D) \in \mathbb{R}^{n \times 3}$$

↓ deformed mesh
↳ offsets multiplied to mesh

offsets added to C
0 = observed instance or deformed towards

$W(G,0) = G_1(C,C)\, W(C,0)$ offset Matrix represented as a 3 channel image x,y,z

-deformation on occluded parts

[shape space] for a known object category.

$\hat{T}_i(C, w_i) = C + G_1(C,C) w_i$

↳ lower dimensional space spanned by principle components of all deformation fields W_i, $w_i \in \mathbb{R}^{3n}$

∴ we can apply PCA Expectation Maximization [PCA-EM] to find latent space of category.

∴ mat $L \in \mathbb{R}^{L \times 3n}$ containing L principle components is determined and point $x \in \mathbb{R}^L$ in latent space can be mapped into feature vector $\hat{w} \in \mathbb{R}^{3n}$ by $\hat{w} = Lx + \bar{w}$ ↳ mean of all feature vectors

∴ canonical mat C with L represent Human model for object cat.

So find $\hat{w}$ for all points of C occluded

∴ $\hat{\delta} = \bar{G}(C,C)(Lx + \bar{w})$

↓ $\mathbb{R}^{3n \times 3n}$ same as $G_1(C,C)$ but with added zeros to match space

Main loss is ∴

$$L(\delta_{vis}, \hat{\delta}) = \sum_{v \in i_{vis}} \| \delta_{vis}(v,\cdot) - \hat{\delta}(v,\cdot) \|^2$$

↑ v^{th} row of δ

↓ visible indices

for training inverse deformation is used based on interpolation

2) Iterative 3D Det reg

-> Comparison in Image loss

- using U-net features Feat-Loss

- ~~Pose~~ 6D Pose estimation

3) Deformable Reg [Kow et al]

$\Pi: \mathbb{R}^3 \to \mathbb{R}^2$ pin hole camera

$M_0 \to CT$

$M^* \to$ Lap Image Volume

Register $(M_0) \to M$ ↳ evolved model

using isotropic Neo Hookean elastic Model

ϕ_{det} per tetrahedral

contour constraint $\phi_{contour}(M) = dist(C, C^*)$ $C \in \mathbb{R}^2$

$\therefore$ $C_{silhouette} \subset \Pi(M)$ etc

$C_{ridge} \subset \Pi(K(M))$ gaussian curvature operator K auto or selected by User

utilising all

$\phi_{point}(M) = dist(n, L^*)$ $n \in M$ end point of contour seg C

L^* sight line passing thro this points image pos on $C^*_{ridge-segment}$

- Uses shading to better adapt the Model 'M' to Image 'I'
- visible surface emerges as a triangle mesh in the tessellated 'M'

$\therefore$ $\phi_{shading}$ for each triangle [compare grey level of Image pix with projected triangle

$\phi_{shading}(M) = p(n, n^*)$

- optimisation

1) Contour dist optim first
2) then shading optim

$\phi: S \to \mathbb{R}$

$\mathcal{L}$ solution set $\Pi: S \to \mathcal{L}$

~~$M \in \underset{M}{\operatorname{argmin}} \sum_{i=1}^{m} dist(M, \mathcal{L}_i)$~~

Basically search 'M' to all constraint solution sets

See fig 1 for optim pipeline

-) Constraint mappings are linearized version of ϕ_{det}

4) Learning Deformable Tet Mesh for 3D recon [DefTet]

~~) Color and visibility gradient back propagated~~

.) 3D spatial graph that's tetrahedralized

.) Each tet is a polyhedron with 4 triangle faces, 6 edges and 4 vertices. Neighboring tet share a face, 3 edges, 3 verts

.) So how to form a 3D object:

1) Predict offsets for each vertices of the initial tet
2) Predict occupancy of each tet

.) $v_i = [x_i, y_i, z_i]^T$ vertex of tet.

$f_s = [v_{a_s}, v_{b_s}, v_{c_s}]$ Triangle face of tet

$T_k = [v_{a_k}, v_{b_k}, v_{c_k}, v_{d_k}]$ Tet

.) Binary occupancy of tet T_k as O_k

$O_k = O_k(e(T_k))$ Func 'e' defines different occupancy functions eg occupy wrt tet's centroid

.) $P_s(v) = O_{s_1}(v_{s_1})(1 - O_{s_2}(v_{s_2})) + (1 - O_{s_1}(v_{s_1}))O_{s_2}(v_{s_2})$

— probability that a triangle face f_s defines a surface of an object

* T_{s_1}, T_{s_2} neighboring tets that share face f_s

— Prb that neighboring tet to be occupied and the other not

.) $[[\Delta x_i, \Delta y_i, \Delta z_i]_{i=1}^N, [O_k]_{k=1}^K] = h(V, I; \theta)$

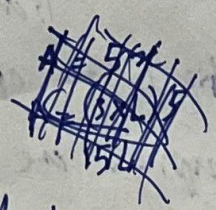
I = image or point cloud

h = neural net → to predict ~~deformation~~ offset of vertex and occupancy of tet

θ = parameters N.N h

∴ deformed verts are

$v_i' = [x_i + \Delta x_i, y_i + \Delta y_i, z_i + \Delta z_i]^T$



.) Inference is of the N.N is deformed tets and occupancy when given I.

·) Total loss :

$$L_{all}(\theta) = \lambda_{recon} L_{recon} + \lambda_{vol} L_{vol} + \lambda_{lap} L_{lap} + \lambda_{sm} L_{sm} + \lambda_{del} L_{del} + \lambda_{arap} L_{arap}$$

L_{recon} = reconstruction loss that aligns surface faces with ground truth faces depending on 2D or 3D.

L_{Lap} = Laplacian loss to penalize change in relative positions of neighboring vertex

L_{Delta} = penalizes large deformations

L_{vol} = equi volume loss penalizes difference in volume for each tet

L_{arap} = penalizes distortions of each tet

L_{sm} = smoothness loss penalizes difference in normals for neighboring faces when 3D G.T available

·) Recon loss with 3D G.T

$$L_{occ} = \sum_{k=1}^{K} \hat{O}_k \log(O_k) + (1-\hat{O}_k) \log(1-O_k),$$

- Supervise occupancy prediction from 'h' with Binary Cross Entropy
- $\hat{O}_k$ ground truth occupancy for Tet T_k
- If winding number of centroid of a tet is positive consider it ~~positive~~ occupied

·) Deformation :

obtain G.T occupancy whose $P_s(v)$ equals 1, occupied faces as F. Deform F to align with target objects surface.

$$L_{surf}(\theta) = \sum_{p \in S} \min_{f \in F} dist_f(p,f) + \sum_{q \in S_f} \min_{p \in S} dist_p(p,q)$$

$dist_f(p,f)$ = L2 dist from point p to f

$dist_p(p,q)$ = L2 dist point p to q

$$L_{recon3D}(\theta) = L_{occ} + \lambda_{surf} L_{surf}$$

1) Recon via diff rendering

- Recon when 2D available.
- Algo to project deformable tet to a 2D image using the given camera.
- Consider initial Mesh with 'V' and 'f'.
- each vertex v_i with an RGB attribute c_i and visibility D_i
- shoot ray from ~~pixel I_j~~ camera origin to pixel I_j to a face 'f' in the Mesh. $[f_{i_1}, \ldots, f_{i_L}]$ L total number of rays and the faces are ordered in increasing depth.
- Render all faces and softly combine via visibility.
- $f_{ik} = (v_a, v_b, v_c)$, attribute in pixel I_j is obtained via barycentric face

$$[w_a, w_b, w_c] = \text{Barycentric}(v_a, v_b, v_c, I_j)$$

$$D_j^k = w_a D_a + w_b D_b + w_c D_c \quad , \quad C_j^k = w_a C_a + w_b C_b + w_c C_c$$

$$m_k = \prod_{i=1}^{k-1} (1 - D_j^i) D_j^k$$

visibility m_k of face f_{ik} in pixel I_j depends on its own visibility and all the faces in front of it.

$$M_j = \sum_{k=1}^{L} m_k \quad , \quad R_j = \sum_{k=1}^{L} m_k C_j^k$$

for pixel I_j we use a weighted sum to obtain color R_j and visibility M_j

- Based on chain rule gradients from M_j and R_j can be naturally backpropagated to vertex positions
- They found that per vertex occupancy and choosing the closest works well

$$\therefore O_k = \max(D_{a_k}, D_{b_k}, D_{c_k}, D_{d_k})$$

2D Loss:

$$L_{recon\,2D} = \sum \left| R_j^{GT} - R_j \right|_1 + \lambda_{mask} \left| M_j^{GT} - M_j \right|_1$$

(Image, Render)

M^{GT} is the mask for object in Image

1) Network architecture from DVR [Differentiable volumetric rendering ...]
training with 3D or 2D GT

- Here the output of DVR the last layer is changed to accomodate tetrahedron shape. Initial set of 40x40x40

t_θ = texture/color Network

f_θ = occupancy Network

$$L(\hat{I}, I) = \sum_u \| \hat{I}_u - I_u \| \qquad I_u = \text{RGB val of } I_{ray}$$

$$\frac{\partial L}{\partial \theta} = \sum_u \frac{\partial L}{\partial \hat{I}_u} \cdot \frac{\partial \hat{I}_u}{\partial \theta}$$

since $\hat{I}_u = t_\theta(\hat{p})$ ray intersection point $(\hat{p})$

$$\therefore \frac{\partial \hat{I}_u}{\partial \theta} = \frac{\partial t_\theta(\hat{p})}{\partial \theta} + \frac{\partial t_\theta(\hat{p})}{\partial \hat{p}} \cdot \frac{\partial \hat{p}}{\partial \theta}$$

$$\frac{\partial \hat{p}}{\partial \theta} = \frac{\partial r(\hat{d})}{\partial \theta} = w \frac{\partial \hat{d}}{\partial \theta} \qquad r(d) = r_0 + dw \rightarrow \text{ray}$$

$\hat{d}$ = surface depth

$$\frac{\partial \hat{d}}{\partial \theta} = -\left(\frac{\partial f_\theta(\hat{p})}{\partial \hat{p}} \cdot w\right)^{-1} \frac{\partial f_\theta(\hat{p})}{\partial \theta}$$

5) Fast gradient descent for Surface Capture via diff rend

·) Non stochastic GS:D use

·) Analogous to flexicubes

·) Extract 3D mesh

·) Volumetric RGBA like rest
color opacity

·) Input : In → multiview Calib images
Bg → Background images [captured before object is placed in scene]

·) Use diff volume recon with appropriate regularization terms

·) opacity parameterization could be voxel, ~~SH~~, FCNN

·) RGB , S.H, NN etc.

All optim ~~based on~~ of volumetric diff rending is based on color similarity loss

$$L_{photo} = \sum_{r \in rays} \| C(r) - C_{gt}(r) \|$$

$C(r)$ ↓ sampled $C_{gt}(r)$ ↓ gt color

$a_i \in [0,1]$ transparency
$c_i \in [0,1]^3$ color

·) sparse grid representation where RGBA obtained by interpolation as in plenoxels.

$$T_K = \prod_{j<K} a_j \quad \text{partial transmittance}$$

K is a value that indexes the regular intervals

$$C_K = \sum_{j<K} (1-a_j) c_j T_j \quad \text{partial color}$$

$$C = \sum_{K<\delta} (1-a_K) c_K T_K + T_\infty C_{bg}$$

·) color and transparency of a sample obtained by linear interpolate of 8 closest voxels $a = lerp(\bar{a})$, $c = lerp(\bar{c})$ — $\bar{a}$ bar indicates voxel val

view dependent effects obtained by S.H → bands

[diffuse specular input]

$$\bar{c}(v) = \sum_{l=0}^{B} \sum_{m=-l}^{l} Y_{l,m}(v) \beta_{l,m}$$

$Y_{l,m}(x,y,z)$ func are polynomials

color control: $\beta_{l,m} \in [-1,1]^3$

v = unit dir of ray

·) losses

1) Robust photometric loss

$$\|x\|_H = x^2 \text{ if } x < \epsilon \text{ else } \epsilon(|x| - \frac{\epsilon}{2}) \quad \text{Huber norm}$$

$$\therefore L_{photo} = \sum_{r \in rays} \| C(r) - C_{gt}(r) \|_H$$

·) Spatial Reg

$$L_{smooth\ a} = \sum_{voxels} \| \nabla \bar{a} \|_2^2 \quad \text{Total variation regularization}$$

·) Ray color consistency

$$L_{sample} = \sum_{sample} \| c_k - C \|_H T_k (1 - a_k) \quad \text{favor voxel color consistency}$$

·) S.H

$$L_{S.H} = \sum_{voxels} \| \beta_{l,m} \|_H$$

·) Ballony occupancy bias

$$L_{ballony} = \sum_{voxels} a^{-2}$$

for difficult regions with no texture or hard to reconstruct, this promotes more opaque voxels

·) Color normalisation

$$c_{gt} = g\, C_{uncorrected\ gt} + b$$
$$c_{bg} = g\, C_{uncorrected\ bg} + b$$

} Camera gain and bias for different exposure

·) Grad descent

we need analytical grads here

$$\frac{\partial L_{photo}}{\partial a_k} = \frac{\partial L_{photo}}{\partial C} \frac{\partial C}{\partial a_k}$$

$$\frac{\partial L_{photo}}{\partial C} = \| C - C_{gt} \|'_H$$

$$\frac{\partial C}{\partial a_k} = -c_k T_k + \frac{C - c_k}{a_k}$$

·) March 1 ray to build T and C then second ray to build $\frac{\partial L_{photo}}{\partial \alpha_k}$

$$\therefore \frac{\partial L_{photo}}{\partial \bar{\alpha}} = \sum_{r \in rays} \sum_{k \in S(r)} \frac{\partial L_{photo}}{\partial \alpha_k} \frac{\partial \alpha_k}{\partial \bar{\alpha}}$$

$S(r)$ = set of samples on ray 'r' to which voxel of interest contributes

·) We don't take all vals in $S(r)$ ∴ we interpolate per sample gradients

$$\frac{\partial L_{photo}}{\partial \bar{\alpha}} \approx \sum_{corners} lerp\left(\frac{\partial L_{photo}}{\partial \alpha}\right)$$

check 'fig 4'

·) Surface extraction done using transparency vals $\bar{\alpha}$ using marching cubes, color is computed per pixel at rasterization time.

~~6) Learning Deformable Tetrahedral Meshes for 3D recon~~

6) Deep Marching Tetrahedra : A hybrid representation for high Res 3D shape synth ~~[Def Tet]~~ [DMTET]

~~1) Optimises for vertex placement and occupancy~~

~~is differentiable wrt. 3D recon Losses~~

1) DMTet directly optimises for the reconstructed surfaces, which allows better detail formation.

2)

2) Contribs:
   1) Using Marching Tet as a differentiable iso-surface layer allows topological change for the shape shown by an implicit field.
   2) Incorporate MT in a DL framework, combining implicit + explicit fields.
   3) Coarse to fine optimisation to scale DMTet to higher Res

1) Our Network learns grid deformations and subdivision jointly to better represent surface! Not relying on pre computed hierarchy.

1) Deep implicit field [DIF] represent a 3D shape as a zero level set of a continous func parameterised by a NN.

1) Optimise voxelised 3D input to a mesh that incorporates G.T data [coarse voxel]

1) Extracting triangular 3D Models .... [Nvdiffrec]

1) $\arg\min_{\phi} E_c[L(I_\phi(c), I_{ref}(c))]$

Empirical Risk

ϕ = para [SDF, vert offset, etc]

$I_\phi(c)$ = Image from renderer 'c' pose

$I_{ref}(c)$ = ref image

L = loss

2) Transparency / translucency not supported

3) Usually works with 50+ views assuming 360 coverage requiring pose etc just like ref

4) Main contrib is the inverse differentiable Lighting