Journal Article Summaries

Jeffrey Liang

June 5, 2024

Summary 1: MeshSDF: Differentiable Iso-Surface Extraction

Citation: Remelli *et al.*, 2020
Summary: Using a neural network to model a signed distance function, we sample points from the zero-level set of the watertight shape and then use Marching Cubes (MC) to create a mesh between points. Rather than differentiating with respect to the MC mesh, we differentiate directly with the neural network since the loss function only cares about zero-level set. There do exist differentiable MC variants but they can be complex/apply only in certain conditions.
Key Points:

- Vertices are sampled from zero level set usually, so if the loss function only cares about the zero level set then you can directly differentiate with respect to the neural net. When the loss function cares about the facets, then we need a differentiable Marching Cubes (but the points are still sampled from the zero level set).
- When a point s undergoes an infinitesimal perturbation Δs, the local surface is perturbed in the opposite direction of the surface normal: $\frac{\partial v}{\partial s} v = -n(v) = -\nabla s(v)$.
- a negative Δs causes points with slightly positive distance values move closer to the zero set (as you subtract), that is, the surface is inflated.
- Marching Cubes (MC) converts implicit functions to 3D surface meshes $\mathcal{M} = (V, F)$. It samples the network on a discrete 3D grid, detecting zero-crossing of the field along grid edges and builds a surface mesh with a lookup table. The position of vertices on grid edges involves linear interpolation, which is modelled by a function discontinuous at $s_i = s_j$, so cannot allow topology changes through backpropagation.
- This method is used for tasks where we deform a 3D mesh $\mathcal{M} = (V, F)$, where $V = \{\boldsymbol{v}_1, \boldsymbol{v}_2, ...\}$ denotes vertices in $\mathbb{R}^3$ and F facets to minimise a task-specific loss function $\mathcal{L}_{\text{task}}(\mathcal{M})$.
- Signed distance function (SDF): $s : \mathbb{R}^3 \to \mathbb{R}$. Neural network f_θ is tarined on watertight surfaces $\mathcal{S}$ to approximate SDF s by minimising

$$\mathcal{L}_{\text{sdf}}\left(\{\boldsymbol{z}_S\}_{S\in\mathcal{S}}, \theta\right) = \sum_{S\in\mathcal{S}} \frac{1}{|X_S|} \sum_{\boldsymbol{x}\in X_S} |f_\theta(\boldsymbol{x}, \boldsymbol{z}_S) - s(\boldsymbol{x})| + \lambda_{\text{reg}} \sum_{S\in\mathcal{S}} \|\boldsymbol{z}_S\|_2^2$$

  where $\boldsymbol{z}_S \in \mathbb{R}^Z$ is a Z-dimensional encoding of surface S, θ denotes network parameters, X_S represents 3D point samples we use to train our netowrk, and λ_{reg} affects regularisation.

  When given a mesh, we want to be able to evaluate:

$$\frac{\partial \mathcal{L}_{\text{task}}}{\partial \boldsymbol{z}} = \sum_{\boldsymbol{v}\in V} \frac{\partial \mathcal{L}_{\text{task}}}{\partial \boldsymbol{v}} \frac{\partial \boldsymbol{v}}{\partial f_\theta} \frac{\partial f_\theta}{\partial \boldsymbol{z}}.$$

  As f_θ approximates an SDF, we can replace $\frac{\partial \boldsymbol{v}}{\partial f_\theta}$ with $-\nabla f_\theta(\boldsymbol{v}, \boldsymbol{z})$.

- Two loss functions are considered between surfaces S and T:

$$\mathcal{L}_{\text{task1}} = \min_{s \in S} d(s, T) + \min_{t \in T} d(S, t), \qquad \text{surface-to-surface distance}$$
$$\mathcal{L}_{\text{task2}} = \|\text{DR}(S) - \text{DR}(T)\|_1 \qquad \text{image-to-image distance}$$

- For shape optimisation, we can use the following loss function (aerodynamics):

$$\mathcal{L}_{\text{task}}(\mathcal{M}) = \iint_{\mathcal{M}} g_\beta \boldsymbol{n}_x \, \mathrm{d}\mathcal{M} + \mathcal{L}_{\text{constraint}}(\mathcal{M}).$$

Summary 2: Structure-from-Motion Revisited

Citation: Schönberger and Frahm, 2016
Summary: A new (incremental) SfM technique which looks to address image registration and triangulation more efficiently and robustly. It augments scene graph using geometric verification strategy. Next best view selection for incremental reconstruction process. Robust triangulation to produce more complete scene structure with reduced computational cost. An iterative BA, re-triangulation, and outlier filtering strategy to improve completeness. More efficient BA parametrisation for dense photo collections.
Key Points:

- SfM starts with correspondence search given a set of images $\mathcal{I} = \{I_i \mid i = 1, ..., N_I\}$:
  1. Feature extraction: $\mathcal{F}_i = \{(x_j, \boldsymbol{f}_j) \mid j = 1, ..., N_{F_i}\}$ at location $\boldsymbol{x}_j \in \mathbb{R}^2$
  2. Matching ($\mathcal{O}(N_I^2 N_{F_i}^2)$): find most similar feature in the other image through brute force to give set of potentially matching image pairs $\mathcal{C} = \{\{I_a, I_b\} \mid I_a, I_b \in \mathcal{I}, a < b\}$ and associated feature correspondences $\mathcal{M}_{ab} \in \mathcal{F}_a \times \mathcal{F}_b$.
  3. Geometric verification: estimate projection transformation to map between feature points (use homography or epipolar geometry) + RANSAC. Outputs a scene-graph, with images as nodes and verified pairs of images as edges.
- Incremental reconstruction: outputs pose estimates $\mathcal{P} = \{\boldsymbol{P}_c \in \boldsymbol{SE}(3) \mid c = 1, ..., N_P\}$ for registered images and the reconstructed scene structure as a set of points $\mathcal{X} = \{\boldsymbol{X}_k \in \mathbb{R}^3 \mid k = 1, ..., N_X\}$.
  1. Initialisation: done with carefully selected two-view reconstruction - can be from densse location in the image graph with many overlapping cameras.
  2. Image registration: can register to current model by solving Perspective-n-Point (PnP) problem suing using correspondences to triangulated points already in the image. Each new registered image extends the set $\mathcal{P}$. This paper attempts to improve in this part.
  3. Triangulation: a new scene point can be triangulated and added to $\mathcal{X}$ as soon as one more image also covering the same scene but from a different viewpoint is registered. The redundancy and additional correspondences increases stability. This paper addresses more robust/efficient method.
  4. Bundle adjustment: A joint non-linear refinement of camera parameters $\boldsymbol{P}_c$ and point parameters $\boldsymbol{X}_k$ which minimise the reprojection error $E = \sum_j \rho_j \left(\|\pi(\boldsymbol{P}_c, \boldsymbol{X}_k) - \boldsymbol{x}_j\|_2^2 \right)$, where π projects scene points to the image space and ρ_j is a loss function to down-weight outliers. Levenberg-Marquardt is used to solve BA problems. Can use Schur complement trick. This paper attempts to improve on this part.

- Current challenges for SfM: failure to register images which should be, produce broken models due to mis-registrations or drift. Caused by incomplete scene graphs, or in failures in reconstruction stage due to inaccurate scene structure.

Summary 3: CAPNet: Continuous Approximation Projection for 3D Point Cloud Reconstruction Using 2D Supervision

Citation: Navaneet *et al.*, 2019
Summary: Constructs a 3D point cloud by projecting them into multiple 2D views and comparing with ground-truth 2D masks in each view (weak supervision). A continuous approximation of points in the point cloud produces smooth projections in a differentiable manner. Uses encoder-decoder architecture.
Key Points:

- Point cloud representations are better than volumetric representations since the latter suffers from information sparsity. Surface voxels provide structural information, however internal voxels increase the computational complexity with minimal addition to information.
- *Goal*: From a single image of an object, reconstruct a 3D point cloud representation of the object. If I is an image from the training set and f is a trained network, let $p = f(I)$ be the corresponding 3D point cloud reconstruction. A projection $P(p, v)$ is from an arbitrary view point v is obtained by performing a perspective transformation and projecting the transformed point cloud onto a plane. The transformed point $\hat{p}_n = (\hat{x}_n, \hat{y}_n, \hat{z}_n)$ in camera coordinates is obtained from: $\hat{p}_n = K(R_v p_n + t_v) \quad \forall n \in \{1, ..., N\}$, where K, R_v, t_n are the camera intrinsics and extrinsics. Training uses ground truth 2D masks M to supervise projection $\hat{M} = P(p, v)$.
- The encoder takes in a 2D image and the decoder reconstructs the point cloud. To compute the loss, the predicted point cloud is projected from V different view-points and compared with corresponding ground truth projections.
- The final loss function is $\mathcal{L} = \mathcal{L}_{bce} + \lambda \cdot \mathcal{L}_{\text{aff}}$. Let $\hat{M}^v_{i,j} = \tanh\Big(\sum_{n=1}^{N} \phi(\hat{x}_n - i) \cdot \phi(\hat{y}_n - j)\Big)$, where ϕ is the kernel function $\phi(k) = \exp\Big(\frac{-k^2}{2\sigma^2}\Big)$ (this Gaussian kernel allows smooth, accurate projections with no holes). The loss function is binary cross-entropy loss:

$$\mathcal{L}_{bce} = \sum_{v=1}^{V} -M^v \log\Big(\hat{M}^v\Big) - (1 - M^v) \log\Big(1 - \hat{M}^v\Big)$$

  where M^v and $\hat{M}^v$ are the ground truth and predicted masks, respectively, of dimension (H, W). This loss function results in reconstructions with many outlier points, so a *nearest point affinity loss* is imposed which minimises the nearest neighbour distance between two pixel maps weighted by pixel confidence:

$$\mathcal{L}_{\text{aff}} = \sum_{v=1}^{V} \sum_{i,j}^{H,W} \min_{(k,l) \in M^v_+} ((i-k)^2 + (j-l)^2) \hat{M}^v_{i,j} M^v_{k,l} + \sum_{v=1}^{V} \sum_{i,j}^{H,W} \min_{(k,l) \in \hat{M}^v_+} ((i-k)^2 + (j-l)^2) M^v_{i,j} \hat{M}^v_{k,l}$$

  where M^v_+ and $\hat{M}^v_+$ are sets of pixel coordinates of the ground truth and predicted projections whose values are non-zero.
- The resultant point cloud is evaluated with the ground truth by computing the Chamfer distance:

$$d_{\text{Chamfer}}(\hat{P}, P) = \sum_{x \in \hat{P}} = \min_{y \in P} \|x - y\|_2^2 + \sum_{y \in P} = \min_{x \in \hat{P}} \|y - x\|_2^2.$$

Ground truths were obtained by randomly sampling 16,384 points on the object surface and then performing farthest point sampling to obtain 1024 points.

Summary 4: Occupancy Networks: Learning 3D Reconstruction in Function Space

Citation: Mescheder *et al.*
Summary: Uses a neural network classifier to represent the continuous decision boundary of a 3D surface.
Key Points:

- Existing 3D representations can be broadly categorised into three categories: voxel-based representations, point-based representations, and mesh representations. Point clouds lack connectivity structure of underlying mesh and hence require extra post-processing steps to extract 3D geometry.
- Let the occupancy function $o : \mathbb{R}^3 \to \{0, 1\}$ denote whether that point is occupied by the object. The paper's occupancy function outputs a probability between 0 and 1 for every point $p \in \mathbb{R}^3$. We want to condition the reconstructed 3D object output on the input $x \in \mathcal{X}$. Hence, we train an *occupancy network* $f_\theta : \mathbb{R}^3 \times \mathcal{X} \to [0, 1]$, which takes a pair (p, x) and outputs a real number representing the probability of occupancy.
- We randomly sample points in the 3D bounding volume of the object. The loss function for a batch $\mathcal{B}$ is:

$$\mathcal{L}_{\mathcal{B}}^{\text{gen}} = \frac{1}{|\mathcal{B}|} \sum_{i=1}^{|\mathcal{B}|} \left[\sum_{j=1}^{K} \mathcal{L}(f_\theta(p_{ij}, z_i), o_{ij}) + \text{KL}(q_\psi(z|(p_{ij}, o_{ij})_{j=1:K}) \| \, p_0(z)) \right]$$