Summarization of:

Learning Kinematic Formulas from Multiple View Videos

by:

Nitish Kumar Naineni

University at Buffalo The State University of New York

Abstract

The paper aims to find the kinematics of an rigid bodies from motion trajectory recorded by multiple cameras.

$$\Delta = vt - \frac{1}{2}gt^2$$

v initial speed
Δ displacement
g gravitational acceleration
t time

Achieved by:

- designing a novel framework consisting of a motion network and a differentiable renderer
- Uses Neural Radiance Field (NeRF) since the geometry is implicitly modeled by querying coordinates in the space
- The motion network is composed of a series of blending functions and linear weights in order to derive the kinematic formulas after training

Neural Radiance Fields (NeRF)

- NeRF is used for scene representation, which uses a multi layered perceptron that takes in the 3d location and viewing direction to output the color and density of the sampled points along the rays in the scene.

$$(c, \sigma) = F\Theta(p, d)$$

F : neural field
p : 3d point location
d : viewing direction
c : color
σ : density

- For NeRF the network is overfit to a single data point i.e. the scene to represent it as a neural radiance field stored in the weights of the network.
- In the model a ray is projected for every pixel of the image for the given viewing direction and position. Along this ray the model is queried for the color and density of the scene. The current framework deals only with motion so direction is not used as an input.

Neural Radiance Fields (NeRF)

- Where the final color of the pixel is computed using the following equation which integrates over the color densities of the scene over near and far bounds $u_n \; and \; u_f$

$$C(\mathbf{r}) = \int_{u_n}^{u_f} e^{-\int_{u_n}^{u} \sigma(\mathbf{r}(v))dv} \sigma\big(\mathbf{r}(u)\big)\mathbf{c}\big(\mathbf{r}(u), \mathbf{d}\big)du$$

- But since it isn't possible to sample along every point along the ray, the ray is divided into K bins from which a point is sampled randomly. For which the final integration commutation ends up as following

$$\hat{C}(\mathbf{r}) = \sum_{k=1}^{K} e^{-\sum_{k'=1}^{k-1} \sigma_{k'}\delta_{k'}} \left(1 - e^{-\sigma_k \delta_k}\right)\mathbf{c}_k$$

where

$$\sigma_k = \sigma(\mathbf{r}(u_k))$$
$$\mathbf{c}_k = \mathbf{c}\big(\mathbf{r}(u_k)\big)$$
$$\delta_k = u_{k+1} - u_k$$

Neural Radiance Fields (NeRF)

- Position is input to the model in an encoded form where the encoding function used is for each of the input coordinates x,y,z

$$\gamma(p) = (\sin\left(2^0 \pi p\right), \cos\left(2^0 \pi p\right), \cdots, \sin\left(2^{L-1} \pi p\right), \cos\left(2^{L-1} \pi p\right)).$$

- Finally the image is trained using reconstruction loss between the rendered pixel colors and the true pixel colors of the scene.

$$L_{\text{rec}} = \sum_{\mathbf{r}} \|\hat{C}(\mathbf{r}) - C_{\text{gt}}(\mathbf{r})\|_2^2,$$

Motion Network

- NeRF is used to learn static scenes where as for the current scenario deal with dynamic scenes. To resolve the paper employs weighted blending functions of time that used to represent the motion.
- The whole motion network is linear and can be explicitly written as a kinematic formula composed by the weighted summation of blending functions such as $\{t, t^2, t^3 \ldots, t^n\}$ after the training process is completed.
- The sample points for NeRF are modified using the Motion network where the new point becomes $r(u) + M(t)$ where u is the sample point, r is the input ray and M in the motion network. The color and density of the sampled point are now calculated using the following:

$$\sigma_k = \sigma(\mathbf{r}(u_k) + M(t)),\ \mathbf{c}_k = \mathbf{c}(\mathbf{r}(u_k) + M(t))$$

Motion Network

- Geometry of the scene cannot be decerned only using color since background might also have color. To make the training process more efficient the paper uses silhouette images of the object in order to avoid data from unnecessary points from the background.
- The occupancy along a ray is computed using the following

$$\hat{O}(\mathbf{r}) = \sum_{k=1}^{K} e^{-\sum_{k'=1}^{k-1} \sigma_{k'} \delta_{k'}} (1 - e^{-\sigma_k \delta_k}).$$

- This is in turn used to compute reconstruction loss taking the silhouette of the objects into account that makes sure the density of the background is zero using the following

$$L_{\text{rec}}^{*} = \sum_{\mathbf{r},t} \|\hat{C}(\mathbf{r},t) - C_{\text{gt}}(\mathbf{r},t)\|_2^2 + \alpha \sum_{\mathbf{r},t} \|\hat{O}(\mathbf{r},t) - O_{\text{gt}}(\mathbf{r},t)\|_2^2$$

Regularization for motion network

- Since the motion network doesn’t have non linear activations, it has high interpretability which allows the motion network to be turned into an analytical function.
- Due to high interpretability we can infuse prior knowledge simply by adding regularizations terms to the final loss function.
- Two of such regularization terms used are
  - Sparsity Regularization
  - Regularization from Differential Equations
- Sparsity Regularization: Training the motion network for a particular experiment may lead to many possible solutions. But often this leads to complex kinematic formulas that fit to the motion of the object. To avoid this the paper enforces sparsity on the weights of the motion network that in turn leads to a simpler kinematic formula.

Regularization for motion network

- Regularization from Differential Equations: Generally there doesn't exist a closed form expression for the physical dynamics of the object such as thermodynamics controlled by heat equation, molecular dynamics controlled by Poisson-Boltzmann equation.
- In such a case we can set the blending function to the basis used in spectral methods that would be helpful in modeling the motion of the object.
- The paper has used different blending functions based on the experiment. Like Cosine basis function in the case of large angle pendulum and sine in the case of the explosion experiment.

Experiments

- To validate the proposed framework in the paper, three experiments were conducted, namely:
  - Kinematics of Free Fall
  - Kinematics of Large Angle Pendulum
  - Kinematics of Explosion
- The first experiment is to validate the sparsity regularization. The aim of this experiment is to find the kinematic equation of a free falling object. Given a short clip of an object there are many possible solutions, but using sparsity regulation we arrive a simpler approximation that converges to a global optima faster compared to without sparsity regularization.

Experiments

- The second experiment is to validate the regularization from the differential equation. The aim of the experiment is to find the kinematic equation of an angle pendulum. Since the motion of the a pendulum can be approximated using t as a basis function , the paper uses cosine instead since it oscillates and would help in modeling the motion of the object better.
- The third experiment is to validate the framework on motion generated by a black box model. In this case the motion was generated by blender, for which the framework doesn’t have any knowledge of its internal animation functions.