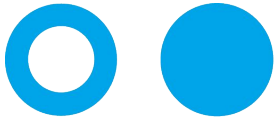
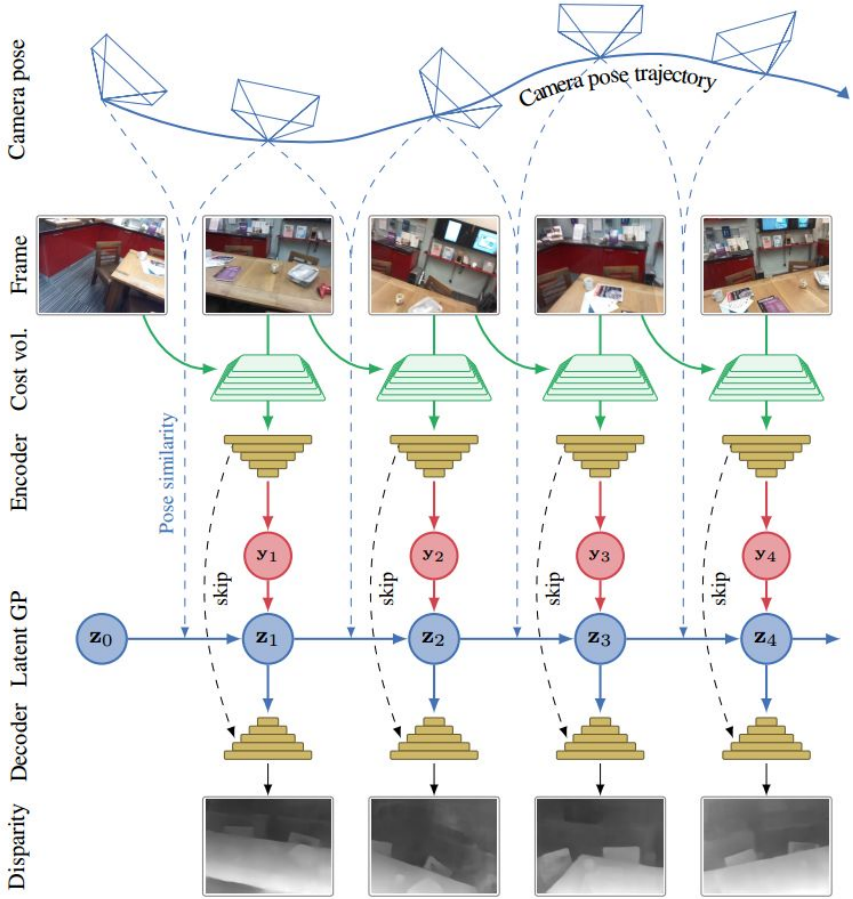


Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

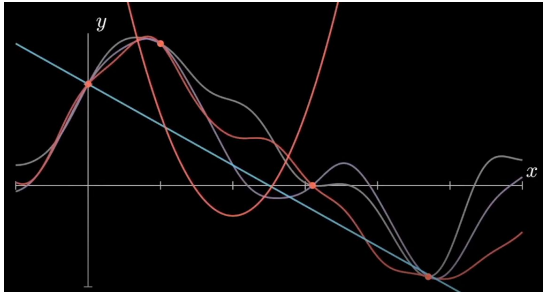
b-it Bonn-Aachen International Center for Information Technology

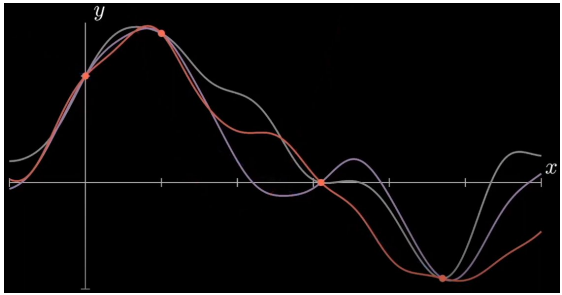
Multi-View Stereo by Temporal Nonparametric Fusion

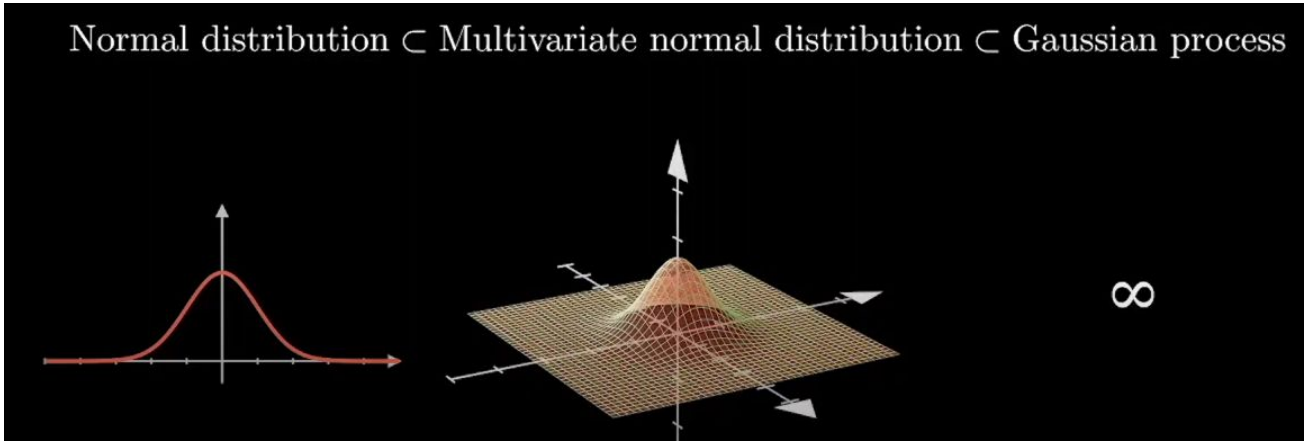

Camera pose
Camera pose trajectory
Frame
Cost vol.
Encoder
Pose similarity
skip
Latent GP
Decoder
Disparity

Gaussian process

- For a set of given inputs consider all possible function with a probability to each of the function.
- And we believe which of them are more likely than other.
- Gaussian process generate probability distribution over function. It means that the functions are described by the gaussian distribution, i.e whenever we generate a function there is some Probability associated with them (From probability distribution over functions).

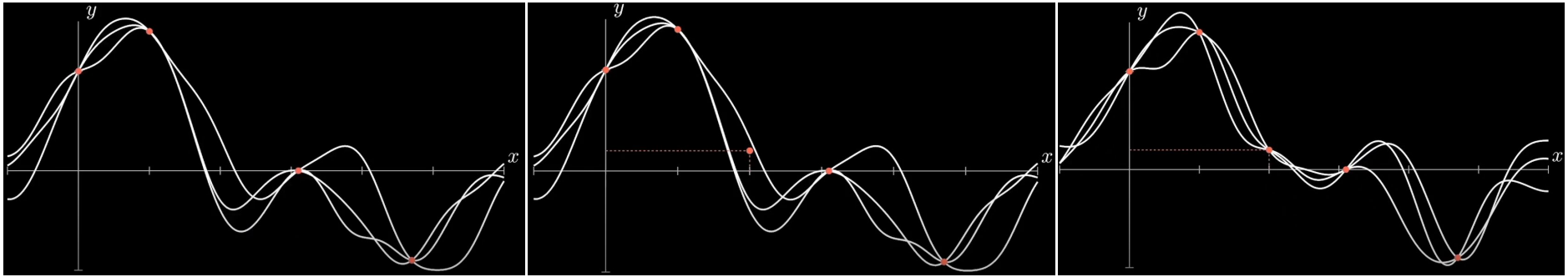






Gaussian process

- We have a different input points and corresponding output (dataset). From this we generate different functions with probability. (Probability distribution over function) (Prior probability distribution of functions)

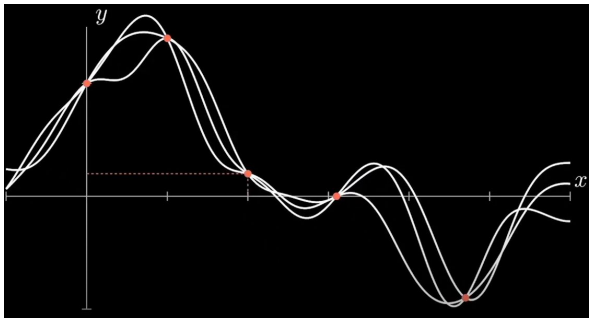


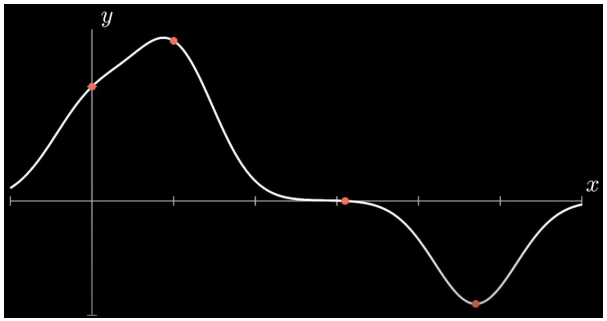
- Let’s say new datapoint comes in then what will be the new function that is being adapted.
- It is calculated using bayes rule. I.e we need to find f given new datapoint y

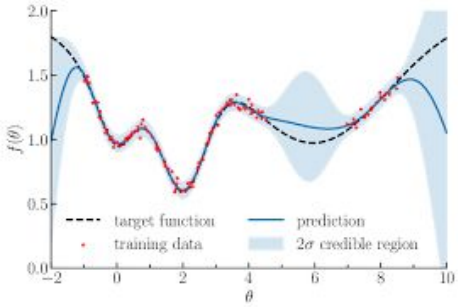
$$p(f|y_i) = \frac{p(f, y_i)}{p(y_i)}$$

Gaussian process

- So once new data comes in, bayes rules is used to update to the new functions as shown below where there is probability associated with each of the function. (Posterior distribution of functions)







- Now we need to choose a single function out of all of these functions that being generated.
- The best guess is the mean function based on the probability measure. But there are infinitely many functions, then how do we get the mean function? Luckily the mean function is the part of the gaussian process and can be calculated.
- At the end we have not only single mean function but range of functions represented by a envelope.

Gaussian Process

Idea behind the Pose-kernel Gaussian process prior

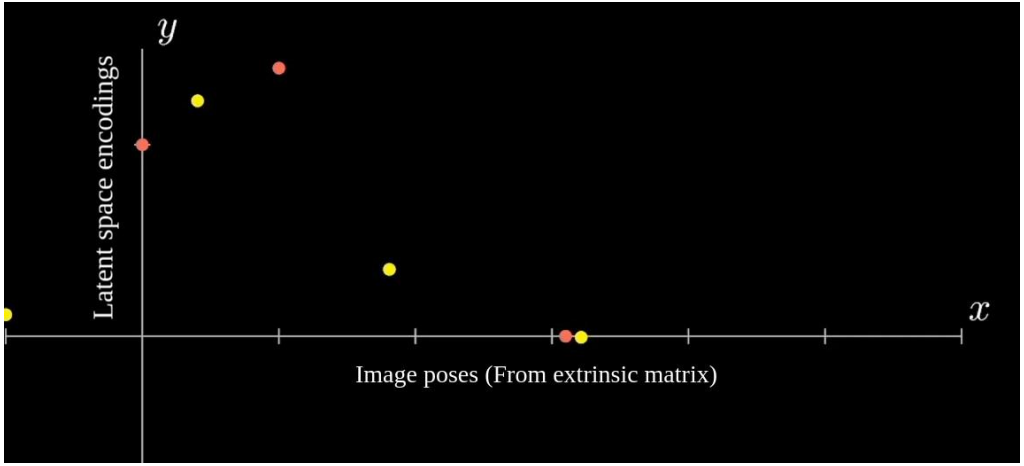
How the disparity map estimation works?

- Compute the cost volume for the pair of images
- Feed it to the encoder
- Get the latent space encodings
- Generate new encodings by passing the encoder output to gaussian process and get the new latent space encodings
- Pass it to the decoder and get the disparity map

Why we need to generate new latent encodings by gaussian process?

- “Define a probabilistic prior on the latent space that would account for a priori knowledge that poses with close or overlapping field of view should produce more similar latent space encodings than poses far away from each other or poses with the camera pointing in opposite directions.”
- Disparity map shows the position of the object present in a frame by gathering information from multiple frames taken around that object.
- Decoder generate output disparity map based on the latent input.
- Let’s consider overlapping or close field of view poses.
- The overlapping frame have object at equal distance from the camera. The overlapping frame or closeness of frames is defined by the camera poses.

- By performing the gaussian process on this encodings it forces the input to the decoder to have similar latent space that are having overlapping or close field of view. Thereby overlapping frame generate similar disparity maps. We can combine these
- Gaussian process outputs latent encodings by taking the camera poses as input.



$$\mathbf{Z} = (\mathbf{z}_1\, \mathbf{z}_2\, \ldots\, \mathbf{z}_N)^\top$$

$$\mathbf{Y} = (\mathbf{y}_1\, \mathbf{y}_2\, \ldots\, \mathbf{y}_N)^\top$$

$$\mathbf{C}_{i,j} = \kappa(P_i, P_j)$$

$$\mathbb{E}[\mathbf{z}_i \mid \{(P_i, \mathbf{y}_i)\}_{i=1}^N]$$

$$D[P_i, P_j] = \sqrt{\|\mathbf{t}_i - \mathbf{t}_j\|^2 + \frac{2}{3}\operatorname{tr}(\mathbf{I} - \mathbf{R}_i^\top \mathbf{R}_j)},$$

$$\kappa(P, P') = \gamma^2 \left(1 + \frac{\sqrt{3}\, D[P, P']}{\ell}\right) \exp\left(-\frac{\sqrt{3}\, D[P, P']}{\ell}\right)$$

$$\mathbb{E}[\mathbf{Z} \mid \{(P_i, \mathbf{y}_i)\}_{i=1}^N] = \mathbf{C}\,(\mathbf{C} + \sigma^2\,\mathbf{I})^{-1}\,\mathbf{Y},$$

$$\mathbb{V}[\mathbf{Z} \mid \{(P_i, \mathbf{y}_i)\}_{i=1}^N] = \operatorname{diag}(\mathbf{C} - \mathbf{C}\,(\mathbf{C} + \sigma^2\,\mathbf{I})^{-1}\,\mathbf{C}),$$

Pose-kernel Gaussian process prior

- Similar latent space encoding for images taken with similar poses.
- Dissimilar latent space encoding for images that are taken far away from each other.
- This information is encoded in covariance function and for this we need to define a distance measure or closeness in pose-space.
- The pose distance measure between the camera poses is defined as below,

$$D[P_i, P_j] = \sqrt{\|\mathbf{t}_i - \mathbf{t}_j\|^2 + \frac{2}{3}\,\mathrm{tr}(\mathbf{I} - \mathbf{R}_i^\top \mathbf{R}_j)},$$

- Where “tr” is the trace of the matrix, **I** is the identity matrix

Pose-kernel Gaussian process prior

- Covariance function structure from Matern class

$$\kappa(P, P') = \gamma^2 \left(1 + \frac{\sqrt{3}\, D[P, P']}{\ell}\right) \exp\left(-\frac{\sqrt{3}\, D[P, P']}{\ell}\right)$$

- Where γ is the characteristics magnitude and ℓ is the length scale of the processes
- In order to share the temporal information between frames in the sequence, we assign independent GP priors to all values in zi, and consider the encoder outputs yi to be noise-corrupted versions of the 'ideal' latent space encodings.

$$z_j(t) \sim \mathrm{GP}(0, \kappa(P[t], P[t'])),$$
$$y_{j,i} = z_j(t_i) + \varepsilon_{j,i}, \quad \varepsilon_{j,i} \sim \mathrm{N}(0, \sigma^2),$$

- where zj(t), j = 1, 2, . . . , (512×8×10), are the values of the latent function z at time t.

Batch solution for solving the inference problem

- Unordered set of image pose pair
- where Z = (z1 z2 . . . zN)⊤ are stacked latent space encodings, Y = (y1 y2 . . . yN)⊤ are outputs from the encoder, and the covariance matrix Ci,j = κ(Pi, Pj).
- The posterior mean E[zi | {(Pi, yi)}N i=1] is then passed through the decoder to output the predicted disparity map.
- In statistical terms, the posterior probability is the probability of event A occurring given that event B has occurred. In our case given the pose P and output of encoder Y we need to calculate Z.

$$\mathbb{E}[\mathbf{Z} \mid \{(P_i, \mathbf{y}_i)\}_{i=1}^{N}] = \mathbf{C}\,(\mathbf{C} + \sigma^2\,\mathbf{I})^{-1}\,\mathbf{Y},$$
$$\mathbb{V}[\mathbf{Z} \mid \{(P_i, \mathbf{y}_i)\}_{i=1}^{N}] = \mathrm{diag}(\mathbf{C} - \mathbf{C}\,(\mathbf{C} + \sigma^2\,\mathbf{I})^{-1}\,\mathbf{C}),$$

- This batch scheme considers all the interconnected poses in the sequence.
- The downside is that the matrix C grows with the number of input frames/poses, N, and the inference requires inverting the matrix—which scales cubically in the matrix size.

Online estimation

- Natural order to image pose pair
- GP inference problem can be solved in state space (SS) form with constant computation and memory complexity
- Covariance function converted to dynamical model
- The initial state (prior) of SS is setup according to Matern covariance function

$$\mathbf{z}_0 \sim \mathrm{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0) \qquad \text{where } \boldsymbol{\mu}_0 = \mathbf{0} \text{ and } \boldsymbol{\Sigma}_0 = \mathrm{diag}(\gamma^2, 3\gamma^2/\ell^2)$$

- Jointly infer the posterior of all the independent GPs, such that the mean μi is a matrix of size 2×(512·8·10).
- A Evolution operator is defined

$$\boldsymbol{\Phi}_i = \exp\left[\begin{pmatrix} 0 & 1 \\ -3/\ell^2 & -2\sqrt{3}/\ell \end{pmatrix} \Delta P_i\right] \qquad \Delta P_i = D[P_i, P_{i-1}]$$

- This gives us the predictive latent space values $\mathbf{z}_i \,|\, \mathbf{y}_{1:i-1} \sim \mathrm{N}(\bar{\boldsymbol{\mu}}_i, \bar{\boldsymbol{\Sigma}}_i)$
- Mean and covariance are propagated by $\bar{\boldsymbol{\mu}}_i = \boldsymbol{\Phi}_i \, \boldsymbol{\mu}_{i-1}, \qquad \bar{\boldsymbol{\Sigma}}_i = \boldsymbol{\Phi}_i \, \boldsymbol{\Sigma}_{i-1} \, \boldsymbol{\Phi}_i^\top + \mathbf{Q}_i$

  where $\mathbf{Q}_i = \boldsymbol{\Sigma}_0 - \boldsymbol{\Phi}_i \, \boldsymbol{\Sigma}_0 \, \boldsymbol{\Phi}_i^\top$

Online estimation

- The posterior mean and covariance is then given by conditioning on the encoder output yi of the current step:

$$\boldsymbol{\mu}_i = \bar{\boldsymbol{\mu}}_i + \mathbf{k}_i\,(\mathbf{y}_i^\top - \mathbf{h}^\top \bar{\boldsymbol{\mu}}_i), \quad \boldsymbol{\Sigma}_i = \bar{\boldsymbol{\Sigma}}_i - \mathbf{k}_i\,\mathbf{h}^\top \bar{\boldsymbol{\Sigma}}_i,$$

$$\text{where } \mathbf{k}_i = \bar{\boldsymbol{\Sigma}}_i\,\mathbf{h}/(\mathbf{h}^\top \bar{\boldsymbol{\Sigma}}_i\,\mathbf{h} + \sigma^2) \qquad \mathbf{h} = (1 \;\; 0)^\top$$

$$z_i \big| y_{1:i} \sim N(\mu_\mathbf{i}, \boldsymbol{\Sigma}_\mathbf{i})$$

$\mathbf{h}^\top \boldsymbol{\mu}_i \Longrightarrow$ Passed to the decoder

Dataset

- Testing dataset - 66 sample images
- Ground truth - 66 depth maps
- Intrinsic matrix
- Extrinsic matrix

GPlayer inputs and outputs

- Input - D, Y
- D - [1, 65, 65] Distribution matrix
- Y - [1, 65, 512, 8, 10] - [b l c h w]
- K - [1, 65, 65]
- I - [1, 65, 65]
- X - [1, 65, 40960]
- Z - [1, 65, 40960]

$$D[P_i, P_j] = \sqrt{\|\mathbf{t}_i - \mathbf{t}_j\|^2 + \frac{2}{3}\mathrm{tr}(\mathbf{I} - \mathbf{R}_i^\top \mathbf{R}_j)},$$

Gaussian process

```
b,l,c,h,w = Y.size()
Y = Y.view(b,l,-1).cpu().float()
D = D.float()
K = torch.exp(self.gamma2) * (1 + math.sqrt(3) * D / torch.exp(self.ell)) * torch.exp(-math.sqrt(3) * D / torch.exp(self.ell))
I = torch.eye(l).expand(b, l, l).float()
X,_ = torch.solve(Y, K+torch.exp(self.sigma2)*I)
Z = K.bmm(X)
Z = F.relu(Z)
```

Y - Latent output of encoder
D - Distribution matrix
K - Kernel
I - Identity matrix
torch.solve(B,A) for AX = B
B == Y
A == K(Includes distribution matrix)+sigma2*I