CS5340 Uncertainty Modeling in AI - Project Semester 1 19/20

September 9, 2019

This project has a total of 120 marks, which constitutes **40% of your final grade** in CS5340. You are allowed to work in a maximum of 2 students (3 is allowed with written approval) per group. Note that **plagiarism will not be condoned**! You may discuss among different groups and check the internet for references, but you MUST NOT submit codes/reports that is copied directly from other sources! Please indicate clearly the names and matriculation numbers of all the group members in the report.

Submit your project by **17 Nov 2019, 2359hrs** via NUS Luminus. 25% of the total score will be deducted for each day of late submission.

Submission Guideline: For each of the three projects, you need to submit the source code, required outputs, and a report. Put them in the respective folder and name them as `proj1, proj2, proj3`. Zip all three folders to a single zip file and name it with your group memebers' NUSNET ID, e.g. `E0208999.zip, E0307788_E0506655.zip`

If you have any question, feel free to ask TAs for help.

- Li Chen: e0267904@u.nus.edu
- Xie Yaqi: yaqixie@comp.nus.edu.sg

1 Vanish Points Estimation based on EM algorithm (40 marks)

1.1 Objective

In this project, you will use the Expectation-Maximization (EM) algorithm proposed in [5] to estimate the location of vanishing points in an image, and assign each pixel to one of the vanishing points at the same time. Figure 1 gives an example of vanishing points estimation given an image which satisfies the Manhattan world assumption [1]. The top figure shows the locations of the vanishing points in the image coordinates (note that the vanishing point on the vertical direction is not shown here because it is at infinity), and the bottom figure shows the assignment of each pixel to one of the three vanishing points.

1.2 Preliminaries

- Manhattan world assumption: It assumes that the scene has a natural cartesian x, y, z coordinate system and the image statistics, such as the direction of edges, will be determined by the alignment of the viewer (or the camera rotation) with respect to this system.

- Vanishing point: A vanishing point is a point on the image plane where parallel lines in three-dimensional space appear to converge because of perspective projections. Given the vanishing point locations of an image, we can estimate the camera rotation and vice versa. For example, given the camera intrinsics $\mathbf{K}$ and rotation matrix $\mathbf{R}$, the vanishing points can be computed as :

$$\mathbf{v} = \mathbf{K} * \mathbf{R} * \mathbf{V} \tag{1}$$

  where $\mathbf{V}$ is the vanishing point direction represented in homogeneous coordinate.

1.3 Vanishing Points Detection based on EM

Vanishing points estimation and assignment is a chicken and egg problem. We can assign each pixel to one of the vanishing points if we know the vanishing point locations (or camera rotation parameters). On the other hand, we can also

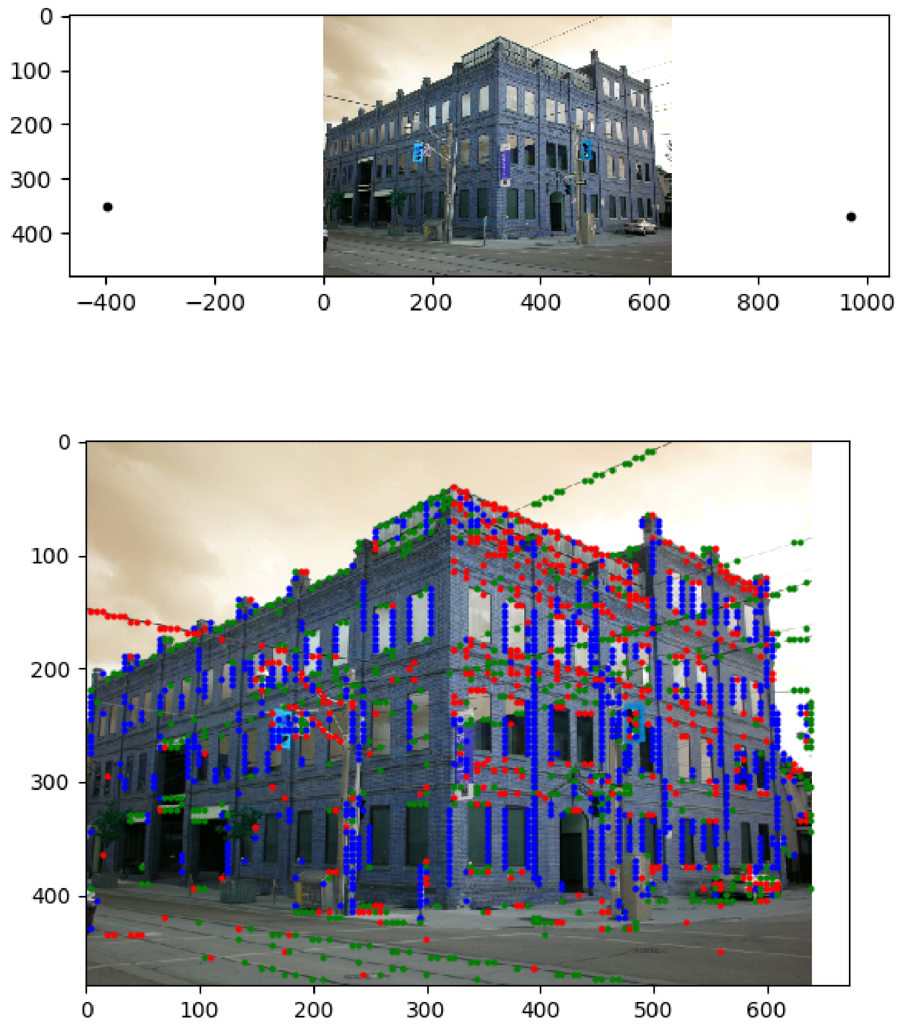


Figure 1: An example of vanishing point estimation and pixel assignment

find a vanishing point in an image if we know all the pixels that belongs to this vanishing point. In this project, we use the EM algorithm for the scenario where we know neither the vanishing point locations (or camera rotation parameters) nor the assignment of each pixel. There are two main steps in the EM algorithm: E-step and M-step. In the E-step, we solve the assignment problem using the camera parameters from the previous step. We compute a posterior $P(M \mid G, \Psi^{old})$ over different edge models M :

$$P(M \mid G, \Psi^{old}) \propto P(G \mid M, \Psi^{old})P(M). \tag{2}$$

Ψ^{old} is the estimated camera parameters from the previous step. M represents a set of models, *ie.* the three vanishing points and others (Note that there are points which do not belong to any of the vanishing points). Intuitively, we calculating the probability that a pixel belongs to each model based on the camera parameters and the observed image gradients G. In the M-step, the objective is to maximize the expected log-posterior $Q(\Psi, \Psi^{old})$ with respect to camera parameters Ψ to obtain a new set of parameters Ψ^{new}:

$$\begin{aligned} Q(\Psi, \Psi^{old}) &\triangleq \langle \log P(G \mid M, f(\Psi)) \rangle + \log P(\Psi) \\ \Psi_{new} &= \underset{\Psi}{\operatorname{argmax}}\, Q(\Psi; \Psi^{old}), \end{aligned} \tag{3}$$

where $\langle . \rangle$ denotes expectation which is taken with respect to the distribution $P(M \mid G, \Psi^{old})$ we obtain from the E-step. The E-step and M-step are iterated until converge.

1.4 Implementation

- **Input Data**: The input data is a RGB image where the scene fulfills the Manhattan assumption. The camera intrinsics which include focal length, principle point and size of the sensor pixels are provided in the 'cameraParameters.mat'. We provide the function to compute the intrinsic matrix K from those intrinsic parameters.

- **Steps**:

  1) Compute the gradient magnitude and direction of an image. You can use cv2.Sobel in python to compute the gradient over x and y direction respectively. For computational efficiency, we will sample the original image every 5 pixels and only extract the 2000 pixels with the largest gradient

magnitude. Note that you have to turn the image into gray scale before computing the gradient by using cv2.cvtColor.

2) The EM algorithm is sensitive to the initial estimate for camera parameters. We will use the results of the bayesian approach in [1] for initialization, and we also use the same search strategy as [1]. We firstly search for the rotation angle around the y axis β from $-45°$ to $45°$ with increments of $4°$. Then we search over the rotation angle around the z axis α and the x axis γ. Please refer to page 9 of [1] for more details. (Note that our notations are slightly different from the original paper because the camera is pointing horizontal direction in their case). We will provide the function of how to convert those angles to a rotation matrix $\mathbf{R}$ such that you can compute the vanishing points by using Eq 1.

3)Then you can use the EM algorithm to find the optimal camera parameters. At the same time you can also get the assignment of each pixel from the E-step. You will have to use the 'least_squares' in scipy for the optimization in Eq 3. Note that we have to convert the rotation matrix to the Cayley-Gibbs-Rodrigu [4] representation such that the orthogonal constraint can be satisfied during the optimization.

- **Help functions**:

  1) Angle2matrix: Convert the rotation angles α, β and γ into rotation matrix, which will be used in Eq 1 to compute the location of vanishing points.

  2) Matrix2vector: Convert rotation matrix $\mathbf{R}$ into the Cayley-Gibbs-Rodrigu representation $\mathbf{S}$, which will be used during the optimization in Eq 3.

  3) Vector2matrix: Covert the Cayley-Gibbs-Rodrigu representation back into a rotation matrix. This function will be used after you get the optimal S to generate the results.

  4) Vp2dir: Compute the predicted edge orientation at each pixel by taking cross product of the coordinates of vanishing point $\mathbf{u}$ and pixel $\mathbf{v_m}$:

$$\mathbf{v_m} \times \mathbf{u} = [x, y, 1]^T. \tag{4}$$

  Note that both coordinates should be represented in the homogeneous coordinate. Then the normal direction of the predicted edges can be computed as $\theta = \arctan(y/x)$, which will be compared with the gradient direction of pixel ϕ_u to compute the angle difference $err = \theta - \phi_u$ in the M-step.

5) Remove_polarity: Add $\pm\pi$ to err and return the minimal value among $abs(err+\pi, err-\pi, err)$ such that the error does not depend on the polarity of the edge. This is just a function to remind you to remove the polarity, you can also implement by yourself.

6) cam_intrinsics: Get the camera intrinsic matrix $\mathbf{K}$ from the .mat file.

7) down_sample: Sample from the original image every 5 pixels and only extract pixels with large gradient magnitude for computational efficiency.

We provide those help functions just for your reference, you can also implement by yourself.

- Some empirical values:

  1) Vp_dir : Vanishing point directions in 3D world represented in homogeneous coordinate.

  2) P_m_prior: Prior distribution over the edge models, namely $P(m_p)$ in [5].

  3) mu, sig : Mean and standard deviation of the Gaussian distribution used to fit the angle difference err.

- **Results**:

  1) Plot vanishing points in each image. You can use the plt.scatter from matplotlib.pyplot.

  2) Visualize the assignment of each pixel, namely the vanishing point each pixel belongs to. Note that different assignment should be denoted with different colors.

  You can refer to Figure 1 as an example.

- **Submission**:

  1) Source code together with the generated results.

  2) A report which includes how you apply the EM algorithm to vanishing point estimation, and a brief description of your results.

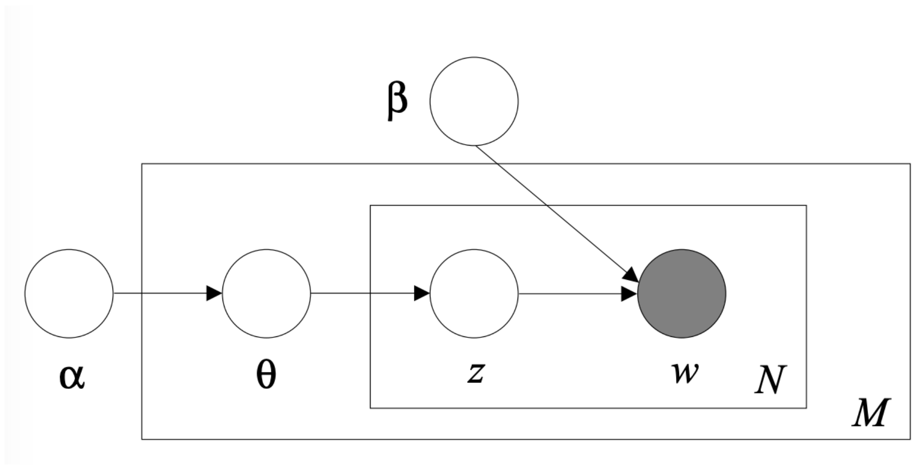


Figure 2: Graphical model representation of LDA.

2 Variational Inference on the Latent Dirichlet Allocation Model (40 marks)

2.1 Objective

In this project, we will perform variational inference on Latent Dirichlet Allocation model to discover human ancestry.

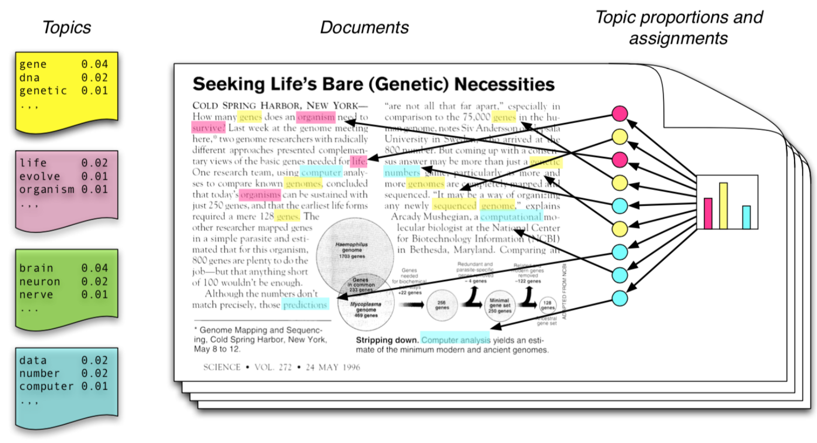
2.2 Preliminaries

2.2.1 Topic models - Latent Dirichlet Allocation Model

latent Dirichlet allocation (LDA) [3] is a generative probabilistic model for collections of discrete data such as text corpora. LDA is a three-level hierarchical Bayesian model, in which each item of a collection is modeled as a finite mixture over an underlying set of topics. Each topic is, in turn, modeled as an infinite mixture over an underlying set of topic probabilities. The LDA model is represented as a probabilistic graphical model in Figure 2. The most popular use for LDA is in modeling a document collection by topics (Figure 3).

LDA assume the following generative process for each document $\mathbf{w}$ in a corpus D:

1. Choose $N \sim Poisson(\xi)$



- Each **topic** is a distribution over words
- Each **document** is a mixture of corpus-wide topics
- Each **word** is drawn from one of those topics

Figure 3: Topic modeling example.

2. Choose $\theta \sim Dir(\alpha)$
3. For each of the N words w_n:
   (a) Choose a topic $z_n \sim Multinomial(\theta)$
   (b) Choose a word w_n from $p(w_n|z_n, \beta)$, a multinomial probability conditioned on the topic z_n

You can refer to the paper [3] for more details.

2.2.2 Mean field Variational Inference

Inference and learning on LDA are both intractable. We use variational inference with mean field assumption, which will be covered later, to perform approximate inference. Variational inference turns inference into optimization problem and use variational distribution q to approximate the true distribution p. The objective is to minimize the KL divergence $KL(q||p) = -\int q(Z)ln\{\frac{p(Z|X)}{q(Z)}\}dZ$. Howver this is hard since posterior $p(Z|X)$ is intractable. It's equivalent to maximize the lower bound of the log likelihood $L(q) = \int q(Z)ln\{\frac{p(X,Z)}{q(Z)}\}dZ$ since $log\ p(x) = KL(q||p) + L(q)$.

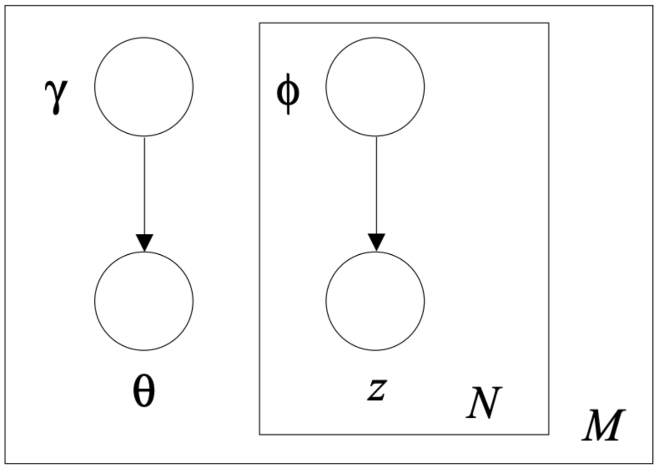


Figure 4: Graphical model representation of the variational distribution used to approximate the posterior in LDA.

The mean field assumption is that we can break the dependency using fully factorized distributions; the coupling between θ, β is removed. That is, we use

$$q(\theta, z|\gamma, \phi) = q(\theta|\gamma) \prod_{n=1}^{N_i} q(z_n|\phi_n) \tag{5}$$

to approximate true posteiror

$$p(\theta, z|w, \alpha, \beta) = \frac{p(\theta, z, w|\alpha, \beta)}{p(w|\alpha, \beta)} \tag{6}$$

The variational distribution is represented as a probabilistic graphical model in Figure 4.

The free variational parameters are Dirichelet parameter γ and multinomial parameters $(\phi_1, \phi_2..., \phi_{N_i})$.

The desideratum of finding a tight lower bound on the log likelihood translates directly into the following optimization problem:

$$(\gamma*, \phi*) = argmin_{(\gamma,\phi)} D(q(\theta, z|\gamma, \phi)||p(\theta, z|w, \alpha, \beta)) \tag{7}$$

Then by computing the derivatives of the KL divergence and setting them equal

```
(1)    initialize $\phi^0_{ni} := 1/k$ for all $i$ and $n$
(2)    initialize $\gamma_i := \alpha_i + N/k$ for all $i$
(3)    **repeat**
(4)        **for** $n = 1$ **to** $N$
(5)            **for** $i = 1$ **to** $k$
(6)                $\phi^{t+1}_{ni} := \beta_{iw_n} \exp(\Psi(\gamma^t_i))$
(7)            normalize $\phi^{t+1}_n$ to sum to 1.
(8)        $\gamma^{t+1} := \alpha + \sum_{n=1}^N \phi^{t+1}_n$
(9)    **until** convergence
```

Figure 5: A variational inference algorithm for LDA

to zero, we obtain the following pair of update equations:

$$\phi_{ni} \propto \beta_{iw_n} exp\{E_q log(\theta_i)|\gamma\} \propto \beta_{iw_n} exp\{\Phi(\gamma_i) - \Phi(\sum_{j=1}^{K} \gamma_j)\} \tag{8}$$

$$\gamma_i = \alpha_i + \sum_{n=1}^{N} \phi_{ni} \tag{9}$$

where Φ is the first derivative of $log\,\Gamma$ function (hint: check out scipy.special.digamma function).

The variational inference procedure is summarized in Figure 5

2.3 Dataset

In applications of population genetics, it is often useful to classify individuals in a sample into populations. An underlying assumption is that there are K ancestor populations, and each individual is an admixture of the ancestor populations. For example, in studies of human evolution, the population is often considered to be the unit of interest, and a great deal of work has focused on learning about the evolutionary relationships of modern populations.

For each individual, we measure some genetic data about them, called genotype data. Each genotype is a locus that can take a discrete count value, individuals with similar genotypes are expected to belong to the same ancestor populations. We can derive the admixture coefficients (θ) for each individual by running an LDA model, where the documents are the individuals, and the words are the genotype.

In this question, we will implement variational inference to infer the population mixture (θ) and the genotype ancestry (topic) assignments (z) for any individual. The variational distribution used to approximate the posterior (for a given individual i) is $q(\theta, z|Y, \phi) = q(\theta|\gamma) \prod_{n=1}^{N_i} q(z_n|\phi_n)$, where the Dirichlet parameter γ and the multinomial parameters $(\phi_1, ..., \phi_{N_i})$ are the free variational parameters (N_i is the number of non-zero genotype loci for this individual).

The data matrix in data.mat provides data about $M = 100$ individuals, each represented by a vocabulary of $N = 200$ genotype loci. This data has been reprocessed into a count matrix D of size MN. D_ij is the number of occurrences of genotype j in individual i, and $\sum_j D_ij$ is the number of genotype loci in an individual.

We learnt the LDA topic model over $K = 4$ ancestor populations, and the inferred β matrix of size NK has been stored in beta_matrix in data.mat. The value of α is 0.1.

2.4 Implementation

1. Allowed packages: `numpy, scipy, pandas, datetime`
2. Function to load matlab data: `data = scipy.io.loadmat(data_path)`
3. You can use `scipy.special.digamma` function to calculate $\Phi(\cdot)$

2.5 Submission

1. Report variational inference update equation for γ, ϕ
2. For individual 1, run LDA inference to find ϕ for each genotype locus, store it as a matrix of size $n_1 \times K$ (where $n_1 := \sum_{1,j} I(D_{1j} \neq 0)$), $I(\cdot)$ being the indicator function, is the number of non-zero genotypes represent in individual 1. Save it as `phi1.out`
3. Construct a matrix Θ of size $M \times K$ to represent the ancestor assignments for all individuals in the population. For each individual i, run LDA inference to find γ, and store it as row of Φ, i.e. $\Theta_i = \gamma$. (hint: update probabilities in log-space to avoid overflow and underflow issues). Save it as `Theta.out`

4. Report the number of iterations and time taken to get to convergence for running inference on all M individuals. (convergence criteria: absolute change in each value of γ, ϕ is less than $e = 1e^{-3}$)

5. Repeat the experiments for $\alpha = 0.01, \alpha = 1, \alpha = 10$, and discuss the change in the ancestor population assignments to the individuals Θ, and iterations required for convergence change as α change

3 Solve the Seam Carving problem by Viterbi algorithm and HMM formulation (40 marks)

3.1 Objective

In this project, we aim to formulate Seam Carving [2], which is an operator for content-aware image resizing, as Hidden Markov model and use Viterbi algorithm to solve it.

3.2 Preliminaries

3.2.1 Hidden Markov Model and Viterbi algorithm

Hidden Markov Model (HMM) is a State space model with Markovian assumptions (conditional independence), i.e. $z_{n-1} \perp z_{n+1} | z_n$. The graphical model representation is shown in Figure 6. Latent variables must be discrete (K possible states), observed variable can be either discrete or continuous. Details of HMM will be covered later. They are especially known for their application in reinforcement learning and temporal pattern recognition such as speech, handwriting, gesture recognition, part-of-speech tagging, musical score following, partial discharges and bioinformatics.

It consists of two basic parameters:

- Transition probability: $p(z_n|z_{n-1})$
- Emission probability: $p(x_n|z_n)$

The joint probability is

$$p(X, Z) = \prod_{n=1}^{N} p(z_n|z_{n-1})p(x_n|z_n) \tag{10}$$

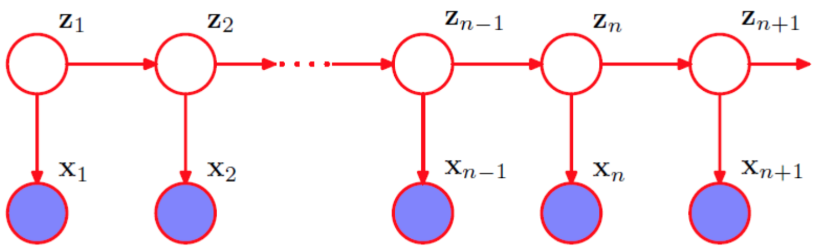


Figure 6: Graphical model representation of the HMM.

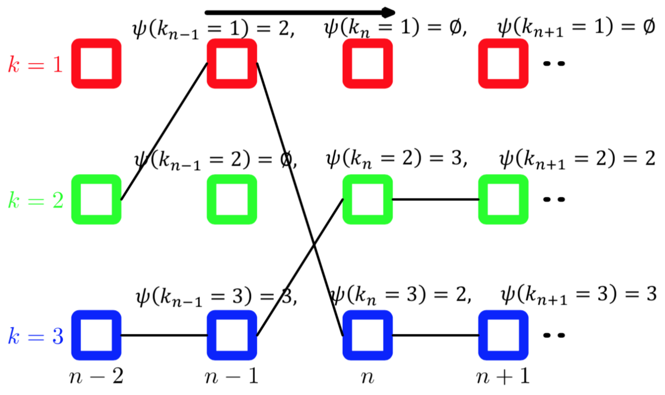


Figure 7: Viterbi algorithm

note that we can add dummy start node z_0.

In this project, we are interested in the task that assume probabilities are known, find the most probable sequence of hidden states for a given observation sequence, which can be solved using Viterbi algorithm.

Viterbi is a dynamic programming algorithm. It compute recursively (work with log probabilities)

$$w(z_{n+1}) = ln\, p(x_{n+1}|z_{n+1}) + max_{z_n}\{ln\, p(z_{n+1}|z_n) + w(z_n)\} \qquad (11)$$

It also keep a record of the values of z_n that correspond to the maxima for each value of the K values of z_{n+1}, denoted by $\phi(k_n)$ where k = 1,...,K. We get $\phi(k_N)$ when we reach the end of the chain z_N. The sequence of latent variable values that corresponds to the maximal probability can be obtained by backtracking the chain recursively: $k_n^{max} = \phi(k_{n-1}^{max})$, which is illustrated in Figure 7.

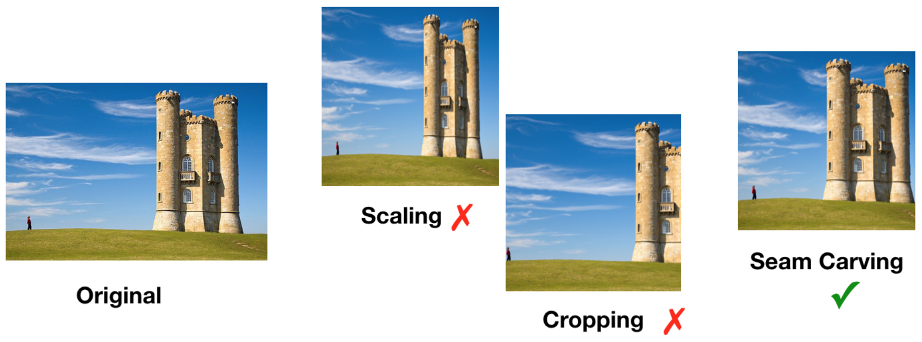


Figure 8: Seam Carving

3.2.2 Seam Carving

Seam carving is an algorithm for content-aware image resizing, i.e. a change in size of the image by modifying the least noticeable pixels in an image. The purpose of the algorithm is image retargeting, which is the problem of displaying images without distortion on media of various sizes. The motivation is illustrated in Figure 8.

A vertical (horizontal) seam is a path of pixels connected from top (left) to bottom (right) in an image with one pixel in each row (column). We want to find the seam with minimal energy (sum of the energy of the pixels on the seam). First of all, we need to assign each pixel energy. To compute the energy of a single pixel, we look at the pixels to the left and right of that pixel. We find the squared component-wise distance between them, that is compute the squared difference between the red components, the squared difference between the green components and the squared difference between blue components, then add them up. We do the same for the pixels above and below the center pixel. Finally, we add up the horizontal and vertical distances.

$$|\Delta x|^2 = (\Delta r_x)^2 + (\Delta g_x)^2 + (\Delta b_x)^2 \tag{12}$$
$$|\Delta y|^2 = (\Delta r_y)^2 + (\Delta g_y)^2 + (\Delta b_y)^2 \tag{13}$$
$$e(x, y) = |\Delta x|^2 + |\Delta y|^2 \tag{14}$$

3.3 Seam Carving as HMM

For example, if we want to find vertical seams, i.e. find each pixel per row. Then we treat number of columns as number of possible states for hidden variable K.

Transition probability (ensure continuity):

- Transition probability (ensure continuity):

$$p(z_n = k' | z_{n-1} = k) = \begin{cases} 1 & \text{if } |k - k'| \leq 1 \\ \infty & \text{else} \end{cases}$$

- Emission probability: $p(x_n | z_n = k) = e(n, k)$

Use Viterbi algorithm and replace max with min, i.e. find the sequence of latent variables corresponds to the minimum energy.

3.4 Implementation

1. Allowed packages: `numpy, cv2`
2. Functions for reading, writing images, calculating pixel energy, removing seams are already implemented for you. You only need to fill in the Viterbi algorithm part.

3.5 Submission

1. Source code
2. Report
3. Take in an image `image_input.jpg` and output a context aware resized image. Save it as `image_output.jpg`

References

[1] James Coughlan Alan Yuille. Manhattan world. http://www.cs.jhu.edu/~ayuille/pubs/ucla/A179_jcoughlan_NC2003.pdf/.

[2] Shai Avidan and Ariel Shamir. Seam carving for content-aware image resizing. *ACM Trans. Graph.*, 26(3):10, 2007.

[3] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003.

[4] Faraz M. Mirzaei and Stergios I. Roumeliotis. Optimal estimation of vanishing points in a manhattan world. In *Proceedings of the 2011 International Conference on Computer Vision*, ICCV '11, pages 2454–2461, Washington, DC, USA, 2011. IEEE Computer Society.

[5] Grant Schindler and Frank Dellaert. Atlanta world: an expectation maximization framework for simultaneous low-level edge grouping and camera calibration in complex man-made environments. In *CVPR 2004*, 2004.