Detection, classification and estimation of individual shapes in 2D and 3D point clouds

J. Su *, A. Srivastava, F.W. Huffer

Department of Statistics, Florida State University, Tallahassee, FL 32306, United States

A R T I C L E   I N F O

Article history:
Received 12 March 2012
Received in revised form 12 September 2012
Accepted 14 September 2012
Available online 25 September 2012

Keywords:
Statistical shape detection
Shape estimation
Point cloud
Poisson process

A B S T R A C T

The problems of detecting, classifying, and estimating shapes in point cloud data are important due to their general applicability in image analysis, computer vision, and graphics. They are challenging because the data is typically noisy, cluttered, and unordered. We study these problems using a fully statistical model where the data is modeled using a Poisson process on the object's boundary (curves or surfaces), corrupted by additive noise and a clutter process. Using likelihood functions dictated by the model, we develop a generalized likelihood ratio test for detecting a shape in a point cloud. This ratio test is based on optimizing over some unknown parameters, including the pose and scale associated with hypothesized objects, and an empirical evaluation of the log-likelihood ratio distribution. Additionally, we develop a procedure for estimating most likely shapes in observed point clouds under given shape hypotheses. We demonstrate this framework using examples of 2D and 3D shape detection and estimation in both real and simulated data, and a usage of this framework in shape retrieval from a 3D shape database.

© 2012 Elsevier B.V. All rights reserved.

1. Introduction

An important feature for characterizing objects in images is their shapes and, as a consequence, shape analysis has become an integral part of object classification. One way to use shape analysis is to estimate the boundaries of the objects (in images) and to analyze the shapes of those boundaries. While analyses of curves and surfaces are important parts of a shape theory, the practical situations mostly involve heavily under-sampled, noisy, and cluttered discrete data, often because the process of estimating boundaries uses low-level techniques that extract a set of primitives (points, edges, arcs, etc.) in the image domain. Therefore, the problem of detecting and estimating shapes in point clouds is an important problem. In some cases one can obtain labeled points where each point carries a binary label assigning it to either the object or the background. Sometimes these labels can also be associated with specific parts of objects. However, in general, the points are unlabeled and do not carry any information about which part of the scene they belong to. Since point clouds are so general, as they contain no interpretation of the data, they are broadly applicable in different scientific domains. Furthermore, the acquisition and processing of digital 3D point clouds has received increasing attention over the last few years. While visualization of very detailed and complex point clouds has become possible, our ability to draw inferences at a semantic level are still very limited. Even tasks as basic as selecting all windows in a scan of a house currently require a disproportional amount of user interaction. This is due to the fact that the acquired raw data does not provide any structure let alone semantic information. Therefore, the extraction of structured shapes from 2D and 3D point clouds is an important topic for a wide field of applications.

What makes the problem of shape detection in point clouds difficult? Here are some of the issues: (1) Unknown Pose and Scale: A shape can be present in the data at an arbitrary pose and scale, as shown in Fig. 1, and one does not know these
variables a priori. The key is to be able to search over the corresponding parameter spaces in an exhaustive fashion and reach global solutions. (2) Noise and Clutter: There is invariably some observation noise associated with shape measurements. Also, the data is commonly corrupted by the presence of points that belong to either the background or other objects, termed as clutter in this paper, as shown in Fig. 1. In this setting it is natural to develop statistical models, and seek efficient global solutions for estimating unknown variables. Note that this problem is different from the problem of comparing unlabeled point patterns which has been addressed by a number of landmark papers, including Rangarajan et al. (1997), Walker (1999), Taylor et al. (2003), Kent et al. (2004), Green and Mardia (2006) and Dryden et al. (2007). In our problem context, one of the two objects being compared is a well-defined shape while the other is a point cloud. In contrast, in the earlier papers both the objects can be point clouds. Also, while most previous works focus on registration or comparison of two point clouds, our problem is actually to detect and estimate a shape.

1.1. Past work

There has been a large body of work detecting objects in 2D and 3D point clouds. The general theory of statistical analysis of spatial data, especially that involving point patterns, can be found in Illian et al. (2008) and Cressie and Wikle (2011). The work on object detection in point clouds can be broadly divided into the following categories.

The first category of papers detects shapes from 2D point clouds. de Souza et al. (1999) presents a statistical approach for identification of objects in digital images where a point distribution model is fitted using Procrustes analysis to a set of training images and used as a prior distribution for the shape of a deformable template. A recent paper by Srivastava and Jermyn (2009) develops a Bayesian approach for shape classification in 2D point clouds. The authors estimate the posterior probability of a given shape by integrating over unknown variables such as pose, scale, and point labels using a Monte Carlo method. The second category of papers works on surface reconstruction from 3D point clouds including Hoppe et al. (1992), Alexa et al. (2001), Kolluri et al. (2004), Mederos et al. (2005) and Dey and Goswami (2006). Most methods call for some kind of connectivity information and are not well equipped to deal with a large amount of outliers. A region growing approach has also been used to detect planes in 3D point clouds by Verma et al. (2006). This approach often delivers a superior segmentation but still suffers from the problems of noisy data. The point clouds have also received attention in the computer graphics community. For example, Cohen-Steiner et al. (2004) have proposed a general variational framework for approximation of surfaces by planes, which is extended to more elaborate approximations by Wu and Kobbelt (2005).

The iterative closest point (ICP) algorithm by Besl and McKay (1992) is a method that uses the nearest-neighbor relationship to assign a binary correspondence at each step. This estimate of the correspondence is then used to refine the transformation, and vice versa. Halma et al. (2010) finds the locations of target objects using single spin image matching and then retrieves the orientation and quality of the match using the ICP algorithm. Chui and Rangarajan (2003) proposes a point matching algorithm for non-rigid registration. They develop an algorithm with the thin-plate splines as the parameterization of the non-rigid spatial mapping and estimate correspondence between points. Blanz et al. (2004) presents a technique that uses a vector space representation of shape (3D Morphable Model) to infer missing vertex coordinates. Another approach that models 3D objects using mixtures of point patterns and draws inferences using MCMC algorithm in a Bayesian setup is described by Micheas et al. (2012).
1.2. Our approach

We present a fully statistical framework for detecting pre-determined shapes in point clouds. An important goal is to provide a likelihood, and thus a confidence, of finding a shape in a given data. We develop a model-based approach where the data is modeled using a Poisson process on the object’s boundary, corrupted by an additive noise and a clutter process. The clutter process itself is modeled using an independent Poisson process on the image domain. Using analytical likelihood functions dictated by the model, we develop a generalized likelihood ratio test for detecting a shape, as described in Section 2.1. The classification threshold is based on an empirical distribution of the log-likelihood ratio estimated using Monte Carlo method under the null distribution; this is presented in Section 2.2. The ratio test requires optimizing the pose and scale associated with hypothesized shapes which, in turn, is performed using a combination of grid search and gradient descent. We use analytical expressions for the gradients of the log-likelihood function. This is illustrated for the 2D problem in Section 3.1 and the 3D problem in Section 3.2. The classification of shapes in 2D and 3D point clouds, based on comparing their log-likelihood ratios, is described in Section 4, while the problem of estimating shapes in the given clouds using selected points is studied in Section 5. The paper ends with a short discussion and conclusion in Section 6.

2. Problem formulation

We consider the following problem: We are given a point cloud \( Y = \{ y_i \in \mathbb{R}^n, i = 1, 2, \ldots, m \} \) in a domain \( U \subset \mathbb{R}^n \) and we want to develop a statistical framework for deciding if there is a pre-determined shape contained in this set. Only the shape is known but its location, orientation, and scale in the scene is unknown. One can extend this idea to detection of full shape classes, i.e., a set of shapes belonging to the same population, using statistical shape models (Srivastava et al., 2005) but that idea is not explored in this paper.

To illustrate the problem, some examples of 2D and 3D point clouds are shown in Fig. 1. In the top row we are interested in finding the likelihood that the shape of a “runner” is present in these clouds. A quick inspection ascertains the presence of a runner in the first two panels, even though the second one is more cluttered than the first one, but the situation is not so clear for the last case. The bottom row shows a similar problem for 3D point clouds. We would like to develop a framework to perform this detection automatically.

We will treat this as a problem of binary hypothesis testing — the null hypothesis is that \( Y \) is simply clutter, i.e., the shape of interest is NOT present in \( Y \), and the alternate hypothesis is that \( Y \) is generated from that shape, i.e., a shape is present in \( Y \).

\[
H_0 : \text{Shape is absent, Likelihood } P(Y|C) \\
H_1 : \text{Shape is present, Likelihood } P(Y|S).
\]

Here, \( C \) denotes the clutter and \( S \) denotes the shape of interest. The challenge, of course, is to develop appropriate probability models that will enable us to evaluate the two likelihoods.

2.1. Model-based shape detection

We take a model-based approach where we evaluate the likelihood of a point cloud containing a shape, and compare that with the likelihood of the cloud being pure clutter. The points present in a given point cloud can be one of two types: (i) points belonging to a shape and (ii) points associated with the background clutter. We will propose an observation model for each of them separately.

Shape is a characteristic that is invariant to similarity transformations, but when a shape occurs in a scene, it has a specific scale, position, and orientation. From the perspective of shape detection, these variables are considered nuisance variables that have to be either estimated or integrated out. In order to better explain the model description and a detection solution, we will start with a simpler problem where we seek a specific object, i.e. known shape, position, orientation, and scale. Let \( \beta : D \to \mathbb{R}^n \) be a parameterized object, where \( D \) is a domain for the parameterization. For finding shapes of curves in 2D images we will have \( D = [0, 1] \) and \( n = 2 \), while for finding surfaces in 3D images we will have \( D = [0, 1]^2, n = 3 \). We will assume that the curves are parameterized by a constant speed parameter in the 2D case, i.e. \( \beta(s) \in \mathbb{R}^2 \) such that \( |\beta'(s)| = \text{constant} \). Since no such canonical parameterization exists for surfaces, we will work with arbitrarily parameterized surfaces, i.e. \( \beta(s) \in \mathbb{R}^3 \) where \( s \in [0, 1]^2 \) parameterizes the surface \( \beta \). In both the cases, we are going to restrict to those \( \beta \)'s that are absolutely continuous on \( D \). To develop the data model, we make the following assumptions:

1. Points belonging to \( \beta \): We assume that these points are realizations of a Poisson process on the parameterized object \( \beta \).

Let \( \gamma : D \to \mathbb{R}_{\geq 0} \) be the intensity function of the Poisson process along \( \beta \). The number of points generated from any part of the object is a Poisson random variable with mean being the integral of \( \gamma \) on that part. In particular, \( k \), the total number of points belonging to the object, is a Poisson random variable with mean \( K = \int_D \gamma(s)ds \in \mathbb{R}_{\geq 0} \). Let the points sampled from \( \beta \) be denoted by \( \mathbf{x} = [x_1, x_2, \ldots, x_i], x_j \in \mathbb{R}^n \). The actual observations \( y_i \) are assumed to be noisy versions of \( x_j \).
For given \( x_1, x_2, \ldots, x_n \), the \( y_i \)'s are assumed to be independent of each other with the identical density \( f(y|x) \). Under this model, the two hypotheses can be re-written as:

\[
\begin{align*}
H_0 : \gamma &= 0, \quad \text{Likelihood} \; P(\mathbf{Y}|C) \\
H_1 : \gamma &> 0, \quad \text{Likelihood} \; P(\mathbf{Y}|S).
\end{align*}
\]

2. **Points associated with clutter:** This subset of observations, independent of the first subset, comes from the clutter and we model them as realizations of a Poisson process with the intensity \( \lambda : U(\subset \mathbb{R}^n) \rightarrow \mathbb{R}_{>0} \), where \( U \) is the region containing observed points, e.g. \( U = [a, b]^2 \) for 2D and \( U = [a, b]^3 \) for 3D point clouds. Let \( \Lambda = \int_U \lambda(y) \, dy \in \mathbb{R}_{>0} \).

The full observation \( \mathbf{Y} \) can now be modeled as a Poisson process with the intensity function: \( \xi(y) = \lambda(y) + \int_D f(y|\beta(s)) \gamma(s) \, ds \). The probability density function of \( \mathbf{Y} \), given \( \beta, \gamma, \lambda \), and for a fixed \( m \), is given by: \( P_m(\mathbf{Y}|\beta, \gamma, \lambda) = \prod_{i=1}^m \xi(y_i) e^{-\Lambda-\lambda}, \) where \( m \) is the total number of points in the data. The null hypothesis is that all the points belong to the Poisson clutter. In that case, the likelihood function is given by: \( Q_m(\mathbf{Y}|\lambda) = e^{-\Lambda} \prod_{i=1}^m \lambda(y_i) \). The likelihoods for both the cases, \( H_0 \) and \( H_1 \), involve certain parameters that are generally not known beforehand. Thus, taking a simple likelihood ratio is not possible and we resort to the **generalized likelihood ratio test** (GLRT). This is based on maximum likelihood estimates (MLEs) of parameters, under the respective hypotheses, and uses the MLEs for evaluating the likelihood ratio. The generalized likelihood ratio is given by:

\[
\frac{Q_m(\mathbf{Y}|C)}{P_m(\mathbf{Y}|S)} = \frac{\max_{\lambda} Q_m(\mathbf{Y}|\lambda)}{\max_{\lambda, \gamma} P_m(\mathbf{Y}|\beta, \gamma, \lambda)} = \frac{\max_{\lambda} \left( e^{-\Lambda} \prod_{i=1}^m \lambda(y_i) \right)}{\max_{\lambda, \gamma} \left( e^{-\Lambda}\sum_{i=1}^m \xi(y_i) \right)}. \tag{1}
\]

So far the unknown parameters are full functions and that involves tremendous computational complexity. We will simplify the evaluation of GLR in Eq. (1) by making the following additional assumptions:

1. The noise added to the points sampled from \( \beta \) is i.i.d. Gaussian with mean zero and variance \( \sigma^2 I_{n \times n} \). Therefore, the conditional density \( f(y|x) \) takes the form \( \frac{1}{(2\pi)^{n/2}\sigma^n} e^{-\frac{1}{2\sigma^2} \|y-x\|^2} \) for \( y, x \in \mathbb{R}^n \).

2. Both the Poisson intensities are constant, i.e., \( \lambda(y) = \lambda \) and \( \gamma(s) = \gamma \) and we get \( \Lambda = \lambda \int_U dy \) and \( \Gamma = \gamma \int_D ds \). To simplify the discussion, we scale both the integrals to be one such that \( \Lambda = \lambda \) and \( \Gamma = \gamma \).

With these assumptions, the likelihood ratio simplifies to:

\[
\frac{Q_m(\mathbf{Y}|C)}{P_m(\mathbf{Y}|S)} = \frac{\max_{\lambda} \left( e^{-\lambda} \prod_{i=1}^m \lambda \right)}{\max_{\lambda, \gamma} \left( e^{-\lambda - \gamma} \prod_{i=1}^m (\lambda + \gamma \alpha_\sigma(y_i)) \right)}.
\]

The numerator on maximization becomes \( e^{-m\lambda} \). The quantity \( \alpha_\sigma(y) = \frac{1}{(2\pi)^{n/2}\sigma^n} \int_D e^{-\frac{1}{2\sigma^2} \|y - \beta(s)\|^2} ds \). Notice that \( \alpha_\sigma(y) \) is high if a point \( y \) is close to the object \( \beta \), with the closeness being measured relative to the scale \( \sigma \). Some illustrations of \( \alpha_\sigma \) in \( \mathbb{R}^2 \) are shown in Fig. 2. The top row shows the case for different \( \sigma \)'s (from left to right: \( \sigma = 0.01, 0.02, 0.03 \)) but a fixed curve. As \( \sigma \) increases, the region of high likelihood spreads further away from the curve. The bottom row shows \( \alpha_\sigma \) for different curves but a fixed \( \sigma = 0.02 \).

Define a function \( H \) to be the logarithm of \( P_m(\mathbf{Y}|\beta, \gamma, \lambda) \). Let \( \theta = [\gamma, \lambda, \sigma] \in \mathbb{R}^3 \) denote three unknown parameters associated with the shape. Then, the function \( H : \mathbb{R}^3 \rightarrow \mathbb{R}_+ \) is given by \( H(\theta) = -\gamma - \lambda + \sum_{i=1}^m \log(\lambda + \gamma \alpha_\sigma(y_i)) \), and let \( \hat{\theta} = \arg \max_{\theta} H(\theta) \) be maximizer (actually, \( \hat{\theta} \) is the MLE of \( \theta \) under the hypothesis \( H_1 \)). We solve for the MLE of \( \theta \) using a gradient approach. Of the three components of \( \theta \), we search exhaustively for the parameter \( \sigma \) and use a gradient-based approach to search over the remaining two \( \gamma \) and \( \lambda \). For each value of \( \sigma \) in a certain range, say \( [\sigma_1, \sigma_2] \), we maximize \( H \) over the pair \( (\gamma, \lambda) \).

For a fixed \( \sigma \), the function \( H : \mathbb{R}^2 \rightarrow \mathbb{R} \), given by \( H_\sigma(\lambda, \gamma) = -\gamma - \lambda + \sum_{i=1}^m \log(\lambda + \gamma \alpha_\sigma(y_i)) \) has the following properties:

1. Its derivatives with respect to \( \lambda \) and \( \gamma \) are given by:

\[
\frac{\partial H_\sigma}{\partial \lambda} = -1 + \sum_{i=1}^m \frac{1}{\lambda + \gamma \alpha_\sigma(y_i)}, \quad \frac{\partial H_\sigma}{\partial \gamma} = -1 + \sum_{i=1}^m \frac{\alpha_\sigma(y_i)}{\lambda + \gamma \alpha_\sigma(y_i)}. \tag{2}
\]
Fig. 2. Likelihood maps $\alpha_{s}$ for some curves. The top row shows the case for the same curve but different $\sigma$’s and the bottom row shows different curves but with the same $\sigma$.

2. Its Hessian matrix is given by:

$$
\begin{pmatrix}
\sum_{i=1}^{m} -\left(\alpha_{\sigma}(y_{i})\right)^{2} / \left(\lambda + \gamma \alpha_{\sigma}(y_{i})\right)^{2} & \sum_{i=1}^{m} \frac{-\alpha_{\sigma}(y_{i})}{\lambda + \gamma \alpha_{\sigma}(y_{i})} \\
\sum_{i=1}^{m} \frac{-\alpha_{\sigma}(y_{i})}{\lambda + \gamma \alpha_{\sigma}(y_{i})} & \sum_{i=1}^{m} \frac{-1}{\lambda + \gamma \alpha_{\sigma}(y_{i})}
\end{pmatrix}
$$

It is easy to show that the two eigenvalues of the Hessian matrix are non-positive so that $H_{\sigma}$ is a concave function in $\lambda$ and $\gamma$.

Therefore, one can use the gradient search over $\lambda$ and $\gamma$, and reach a global optimizer. For $\sigma$ the situation is different and we use an exhaustive grid search over allowable values of $\sigma$ to reach a global maximizer. This combined gradient and grid search algorithm is summarized below:

**Algorithm 1 (MLE of $\theta$).**

- For each $\sigma \in [\sigma_{l}, \sigma_{u}]$ perform the following:
  1. Set $t = 0$ and initialize the pair $[\gamma_{t}, \lambda_{t}]$ with random values in the range $[0, m]$.
  2. Update the estimates using: $[\gamma_{t+1}, \lambda_{t+1}] = [\gamma_{t}, \lambda_{t}] + \delta \left( \frac{\partial H_{\sigma}}{\partial \gamma}(\gamma_{t}, \lambda_{t}), \frac{\partial H_{\sigma}}{\partial \lambda}(\gamma_{t}, \lambda_{t}) \right)$, for a small $\delta > 0$.
  3. If the norm of the gradient vector is small, then stop the loop. Else, set $t = t + 1$ and return to step 2.
- Set the current values to be $(\hat{\gamma}(\sigma), \hat{\lambda}(\sigma))$.
- Define the MLE $\hat{\theta}$ to be $(\hat{\gamma}(\hat{\sigma}), \hat{\lambda}(\hat{\sigma}), \hat{\sigma})$ where $\hat{\sigma} = \arg\max_{\sigma \in [\sigma_{l}, \sigma_{u}]} H(\hat{\gamma}(\sigma), \hat{\lambda}(\sigma), \sigma)$.

With the estimated parameters, the log-likelihood ratio (LLR) becomes

$$
R(Y) = \log \frac{Q_{m}(Y|C)}{P_{m}(Y|S)} = -m + m \log(m) - H(\hat{\theta}).
$$

The generalized likelihood ratio test is given by: $R(Y) \geq \mu$. Next, we will focus on how to choose the threshold $\mu$.

**2.2. Empirical determination of threshold**

In the binary test, the LLR $R(Y)$ is to be compared with a threshold $\mu$ to decide if a shape is present in the data or not. Ideally, this threshold is dictated by the probability distributions of $R(Y)$ under the null hypothesis. In practical situations,
where it is difficult to ascertain these distributions, one uses either the asymptotic theory or an empirical approach to reach an optimal value of $\mu$. Taking an empirical approach, we estimate the probability density function of $R(Y)$ under the null hypothesis. We generate 1000 realizations of $Y$, each using $\gamma = 0$ (null hypothesis = clutter) and a fixed $m$ equal to the observed number of points in our data, and compute a histogram of $R(Y)$ values. Using this estimated density function, we can decide the threshold $\mu$ for a specific Type I error rate, denoted by $\alpha$. One can repeat this for different values of $m$ to catalog distributions of $R(Y)$ for different $m$’s.

The main advantage of a numerical evaluation of the threshold is that we need not assume any specific form for the underlying density, nor do we need to invoke any asymptotics. The disadvantage, however, is that we need to do this for every shape we are interested in, since we do not have an analytical expression. We point out that this computation is offline and can be performed for each of the shapes beforehand. It takes approximately 16 min to generate an empirical distribution of $R(Y)$, for a fixed shape, in the 2D case (with $m = 100$ points) and two hours in the 3D case (with $m = 500$ points).

3. Detection of shapes in point clouds

We will describe the problem of shape detection in 2D and 3D point clouds separately in the next two sections.

3.1. Shape detection in 2D point clouds

In this case the domain of parameterization is $D = [0, 1]$ and the observation space is $\mathbb{R}^2$, so that $\beta : [0, 1] \rightarrow \mathbb{R}^2$ is a closed, continuous contour parameterized at a constant speed. For our experiments, we have taken closed curves from the MPEG4 shape database. The points belonging to the curve are assumed to be realizations of a 1D Poisson process on the parameterized curve $\beta$. The intensity function of the Poisson process is $\gamma \in \mathbb{R}_{>0}$. And points associated with clutter are modeled as realizations of a 2D Poisson process with the intensity $\lambda \in \mathbb{R}_{>0}$. At first we will assume that the position, orientation, and scale of the curve $\beta$ are fully known but later we will generalize to curves with unknown transformations.

3.1.1. Detection of a known curve

In this section we focus on detection of a shape formed by a curve $\beta$ in a square domain $U$. Now the scalar map $\alpha_\gamma : \mathbb{R}^2 \rightarrow \mathbb{R}_+$ is given by $\alpha_\gamma(y_i) = \frac{1}{2\pi\sigma^2} \int_0^1 e^{-\frac{1}{2\sigma^2} |y_i - \beta(s)|^2} ds$. With this setup, one can use Algorithm 1 to find MLE of $\theta$ and use the resulting LLR $R(Y)$ to detect the presence of $\beta$ in $Y$.

We demonstrate MLE of $\theta$ using Algorithm 1 on some simulated datasets where we simulate the point cloud data $Y$ as follows. First, we sample the curve $\beta$ randomly (uniformly) with $k$ points, with $k$ being a Poisson random variable with a certain mean. Then, we add an independent Gaussian perturbation (mean zero, fixed variance) to their positions. Additionally, we generate a clutter sample from a two-dimensional homogeneous Poisson process with certain mean intensity in the region $U$. Fig. 3 presents an example of maximizing the function $H$ using the gradient method. The first two panels show the curve and the simulated point cloud using the parameters $\gamma = 30, \lambda = 25, \sigma = 0.030$. The next panel shows the evolution of $H_\sigma$ over $(\gamma, \lambda)$ versus iteration index for a fixed $\sigma = 0.060$, while the rightmost panel shows the plot of $H(\hat{\gamma}(\sigma), \hat{\lambda}(\sigma), \sigma)$ versus $\sigma$. The MLE of $\theta$ is $\hat{\gamma} = 32.45$, $\hat{\lambda} = 25.74$, $\hat{\sigma} = 0.032$. The maximum value of $H$ lies close to the true $\sigma = 0.03$.

In addition, for each point $y_i$ of an observation, we calculate the corresponding value $\alpha_\hat{\gamma}(y_i)$. For the same point cloud, shown in the left column of Fig. 4, we apply Algorithm 1 for two curves — a runner and a wineglass and display the results in the two rows respectively. The third panel of each row shows the estimated $\alpha_\hat{\gamma}$ map for that curve and the rightmost panel shows the value of $\alpha_\hat{\gamma}(y_i)$ for each of the data points using its thickness. Recall that $\alpha_\hat{\gamma}(y_i)$ is large if the point $y_i$ is close to the curve $\beta$. Since the observation $Y$ here was generated from the runner curve, $\alpha_\hat{\gamma}(y_i)$’s have larger values for that shape and lower values for the wineglass shape.
3.1.2. Detection of a curve with unknown pose and scale

So far we have assumed a fixed curve $\beta$ but a contour can be present in an image at an arbitrary position, orientation, and scale. Therefore, $\beta$ can have variable shape, position, rotation, and scale. Let $O \in SO(2)$ denote the orientation, $T \in \mathbb{R}^2$ denote its translation, and $\rho \in \mathbb{R}^+$ denote its scale. First let $\beta_0$ be a standardized curve, i.e., it has a fixed shape, its centroid is at the origin, its length is one, and its major axes are aligned with the canonical axes. Then, define $\beta(t) = \rho O \beta_0(t) + T$ to be a transformed version of that curve. Here, $O$ and $T$ denote the orientation and translation, respectively. $\rho$ and $\beta_0$ denote its scale. Thereupon, $\beta$ can have variable shape, position, rotation, and scale variables: $\theta, \beta, \rho, \sigma$. Therefore, $\beta$ can take a general form:

$$\beta(\tau) = \begin{bmatrix} \cos(\tau) & -\sin(\tau) \\ \sin(\tau) & \cos(\tau) \end{bmatrix}, \tau \in [0, 2\pi]$$

and $\beta$ is parametrized by $\theta, \beta, \rho, \sigma$. The function $H$ keeps the same form, except the scalar map $\alpha$, now also depends on these other variables, i.e., $\alpha_\gamma(y|\omega) = \frac{-1}{2\pi} \int_0^1 e^{-\frac{1}{2\sigma^2} \|y - \beta(s)\|^2} (y_1 - \beta(s), g(s)) \, ds$.

where (i) $g(s) = O\beta_0(s)$ for $\frac{\partial H}{\partial \theta}$, (ii) $g(s) = \rho \dot{O}(\tau) \beta_0(s)$ for $\frac{\partial H}{\partial \rho}$, and (iii) $g(s) = e_j$ for $\frac{\partial H}{\partial e_j}$. Here $\dot{O}(\tau) = \begin{bmatrix} -\sin(\tau) & -\cos(\tau) \\ \cos(\tau) & -\sin(\tau) \end{bmatrix}, e_1 = (1, 0)^T$, and $e_2 = (0, 1)^T$. We will use $\frac{\partial H}{\partial \omega}$ to denote the gradient with respect to the position, rotation, and scale variables:

$$\frac{\partial H}{\partial \omega} = \begin{bmatrix} \frac{\partial H}{\partial \omega_1} & \frac{\partial H}{\partial \omega_2} & \frac{\partial H}{\partial \omega_3} \end{bmatrix}.$$ 

The algorithm for jointly estimating $\theta$ and $\omega$ is summarized below:

**Algorithm 2 (Joint MLE of $\theta$ and $\omega$ for 2D).** To start, we first translate the center of $Y$ to the origin and align the major axes with canonical axes using Procrustes rotation.

- For each $\sigma \in [\sigma_1, \sigma_2]$ perform the following:
  1. Set $t = 0$ and initialize the pair $(y_t, \lambda_t)$ with random values in the range $[0, m]$ and initialize $\omega_t = [1, \lambda_{2i+1}, [0, 0]^T]$. 
  2. Update the estimates using:

$$\begin{bmatrix} y_{t+1} \\ \lambda_{t+1} \\ \omega_{t+1} \end{bmatrix} = \begin{bmatrix} y_t \\ \lambda_t \\ \omega_t \end{bmatrix} + \delta 
\begin{bmatrix} \frac{\partial H}{\partial \gamma} (y_t, \lambda_t, \omega_t) \\ \frac{\partial H}{\partial \lambda} (y_t, \lambda_t, \omega_t) \\ \frac{\partial H}{\partial \omega} (y_t, \lambda_t, \omega_t) \end{bmatrix},$$

where $\delta$ is a positive, diagonal matrix. Each diagonal component of $\delta$ corresponds to a small step size for each variable in $\theta$ and $\omega$.

3. If the norm of the gradient vector is small, then stop. Else, set $t = t + 1$ and return to step 2.

- Set the current values to be $(\hat{\gamma}(\sigma), \hat{\lambda}(\sigma), \hat{\omega}(\sigma))$.
- Define the MLE $(\hat{\theta}, \hat{\omega})$ to be $(\hat{\gamma}(\hat{\sigma}), \hat{\lambda}(\hat{\sigma}), \hat{\omega}(\hat{\sigma}), \hat{\sigma})$ where \( \hat{\sigma} = \arg\max_{\sigma \in [\sigma_1, \sigma_2]} H(\hat{\gamma}(\sigma), \hat{\lambda}(\sigma), \hat{\omega}(\sigma), \sigma). \)

Shown in Fig. 5 is an illustration of this gradient process. For the same point cloud, we estimate the rotation $O$ (left), translation $T$ (middle), and scale $\rho$ (right) in the top panel with the evolution of $H$ versus iteration in the bottom panel. Each plot in the bottom displays a maximization over just one of them with the other two held fixed.

Next we provide a set of detection results involving simulated point clouds. For a fixed $m$, say $m = 50$, we find $k$ (number of points from the curve) and $n$ (number of clutter points) such that $m = n + k$. Then, we transform a curve with a random
rotation, translation and scale, and select $k$ random points on it uniformly, and finally add Gaussian noise to the selected points. Next we generate $n$ clutter points uniformly on the domain $U$ and mix the two sets to obtain $Y$. For the resulting $Y$, we compute LLR under $S$ and perform the binary detection by comparing it with $\mu$. The left side of Fig. 6 shows two examples of this setup. In both cases, the data is simulated from the runner shape, and the resulting large negative values of $R(Y)$, $R(Y) = -90.4$ (left) and $-96.0$ (right), support rejecting $H_0$ where $\mu = -3.8$ for $\alpha = 0.05$. We also plot the original curve $\beta_0$ under estimated position, orientation, and scale on top of the cloud data.

Beyond the experiments on individual point clouds, we are interested in evaluating the average performance of our method on a large dataset. To study the detection performance systematically, we will study the variability in the probability of Type II error by changing the model parameters. The probability of Type I error is kept fixed at $\alpha = 0.05$ in these experiments. The probability of Type II error is estimated using the equation: $P(\text{Type II Error}) \approx \frac{\text{No. of times } (R(Y) > \mu)}{N}$, where $N = 1000$. As the number of sampled points on the curve decreases, or as the noise or clutter increases, the detection performance suffers. We have evaluated the shape detection performance as a function of $r = n (\# \text{ of points on the curve})/k (\# \text{ of clutter points})$, with $m = n + k$, and noise std dev. $\sigma$, and have plotted the estimated Type II error probability versus these variables for the runner shape in Fig. 6 (right side). The first of those results shows the probability of Type II error versus the noise level $\sigma$, for different values of the ratio $r$. In the case where $r = 0$, there is obviously no dependence on $\sigma$ and the probability of Type II error is simply $1 - \alpha = 0.95$ as expected. In other cases, where $r$ is strictly positive, there is a steady increase in this error probability as the noise level increases. The rightmost plot shows the estimated error probability versus the ratio $r$ while $\sigma$ is kept fixed at 0.1. For a fixed noise level, the error probability decreases with an increase in $r$; this is intuitively clear as more points on the curve allow for a better detection performance. We also compare the error rates of detection in two situations: curve with known rotation and curve with unknown rotation. As shown in the plot, the error increases for the latter case, quantifying the penalty paid for in estimating this additional unknown.
3.2. Shape detection in 3D point clouds

In this section, we focus on detection of a shape formed by a parameterized surface \( \beta \) in a cubical domain \( U \subset \mathbb{R}^3 \). We set \( U \) to be a cube that contains all of \( Y \). Now, the given point cloud is \( Y = \{y_i \in \mathbb{R}^3, i = 1, 2, \ldots, m\} \). Let \( \beta : [0, 1]^2 \rightarrow \mathbb{R}^3 \) be a parameterized surface. Using the same Poisson models, points belonging to the surface are assumed to be realizations of a 2D Poisson process on the parameterized surface \( \beta \) with a fixed intensity \( \gamma \in \mathbb{R}_{>0} \). Points associated with clutter are modeled as realizations of a 3D Poisson process with a fixed intensity \( \lambda \in \mathbb{R}_{>0} \). Similar to the 2D shape detection, we will introduce the detection of a fixed surface in 3D point clouds first and then extend it to detection of surface under unknown transformations.

3.2.1. Detection of a known surface

Starting with the problem of detecting the shape formed by a fixed surface \( \beta \), we notice that the expressions for gradients of \( H \) with respect to the parameters \( \lambda \) and \( \gamma \) are the same as in Eq. (2) except \( \alpha_{\sigma} \) becomes \( \alpha_{\sigma} : \mathbb{R}^3 \rightarrow \mathbb{R}_+ \), given by

\[
\alpha_{\sigma}(y_i) = (\frac{1}{\sqrt{2}\pi}\sigma)^3 \int_{[0,1]^2} e^{-\frac{1}{2\sigma^2}||y_i - \beta(\sigma)||^2} ds. 
\]

Thus Algorithm 1 directly applies and we can estimate the unknown parameters and consequently evaluate \( R(Y) \) using that algorithm. We demonstrate this gradient method on a simple example. For a facial surface shown in the left panel of Fig. 7, we simulate a point cloud using the parameters \( \gamma = 250, \lambda = 150, \sigma = 0.01 \), shown in the second panel of Fig. 7. The face surface \( \beta \) and the point cloud \( Y \) are plotted together in the third panel where the red points are from the surface and the blue ones are from the clutter. The right panel shows the evolution of \( H(\hat{\gamma}(\sigma), \hat{\lambda}(\sigma), \sigma) \) versus \( \sigma \) under Algorithm 1. As shown there, the function \( H \) achieves the maximum at \( \hat{\sigma} = 0.01 \). The MLE of \( \theta \) is found to be \( \hat{\gamma} = 262.62, \hat{\lambda} = 152.53, \hat{\sigma} = 0.01 \) and the LLR is \( R(Y) = -716.66 \).

3.2.2. Detection of a surface with unknown pose and scale

So far we have assumed a fixed surface \( \beta \), but a surface can be present in \( Y \) at an arbitrary position, orientation, and scale. Therefore, \( \beta \) can have variable position, rotation, and scale. Let \( O \in SO(3) \) denote the orientation, \( T \in \mathbb{R}^3 \) denote its translation, and \( \rho \in \mathbb{R}_+ \) denote its scale. Let \( \beta_0 \) to be a standardized surface (center of mass is at the origin, and major axes are aligned with canonical axes) with a fixed shape and define \( \beta = \rho O \beta_0 + T \) to be a transformed version of that surface. When we allow unknown transformations, denoted by \( \omega = (\rho, O, T) \in \mathbb{R}_+ \times SO(3) \times \mathbb{R}^3 \), the cost function \( H \) keeps the same form, except the function \( \alpha_{\sigma} \) now depends on these transformation variables, i.e.,

\[
\alpha_{\sigma}(y_i|\omega) = (\frac{1}{\sqrt{2}\pi}\sigma)^3 \int_{[0,1]^2} e^{-\frac{1}{2\sigma^2}||y_i - \rho O \beta_0(t) - T||^2} ds. 
\]

These additional variables are also estimated, along with \( \theta \), using maximum-likelihood estimation. The gradients of \( H \) with respect to \( \theta \) remain same as earlier, and its gradient with respect to the translation is given by:

\[
\frac{\partial H}{\partial T_j} = (2\pi)^{-\frac{3}{2}} \sigma^{-5} \sum_{i=1}^{n} \frac{\gamma}{\lambda + \gamma \alpha_{\sigma}(y_i|\omega)} \int_{[0,1]^2} e^{-\frac{1}{2\sigma^2}||y_i - \beta(s)||^2} \langle y_i - \beta(s), e_j \rangle ds, \quad j = 1, 2, 3, 
\]

where \( e_1 = (1, 0, 0)^T, e_2 = (0, 1, 0)^T, e_3 = (0, 0, 1)^T \). To define its gradient with respect to \( O \in SO(3) \), we first define the directional derivative \( \eta_j \) of \( H \) in the direction of \( E_j, j = 1, 2, 3 \), where

\[
E_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} 0 & 0 & -1 \\ 1 & 0 & 0 \end{bmatrix}, \quad E_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \quad E_3 = \frac{1}{\sqrt{2}} \begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}. 
\]

This directional derivative is given by:

\[
\eta_j = (2\pi)^{-\frac{3}{2}} \sigma^{-5} \sum_{i=1}^{m} \frac{\gamma}{\lambda + \gamma \alpha_{\sigma}(y_i|\omega)} \int_{[0,1]^2} e^{-\frac{1}{2\sigma^2}||y_i - \beta(s)||^2} \langle y_i - \beta(s), \rho OE_j \beta_0(s) \rangle ds. 
\]
Then, we get the full gradient of $H$ with respect to $O$ in the form of a matrix $M = \eta_1 E_1 + \eta_2 E_2 + \eta_4 E_3$.

As opposed to the 2D case, the function $H$ in the 3D case may have several local maximums in the transformation $\omega$ space, especially in the scale space. So, we try to search globally using a grid search in the scale space. The MLE procedure is summarized below:

**Algorithm 3 (MLE of $\theta$ and $\omega$ for 3D).** Translate the center of $Y$ to the origin and align the major axes with canonical axes using Procrustes rotation.

- For each pair of $\sigma \in \{\sigma_i, \sigma_u\}$ and $\rho \in \{\rho_i, \rho_u\}$, perform the following:
  1. Set $t = 0$ and initialize the pair $[y_t, \lambda_t]$ with random values in the range $[0, m]$ and initialize $\omega_t = [\rho, l_{3x3}, [0, 0, 0]^T]$.
  2. Compute the gradient terms $\frac{\partial H}{\partial \sigma}, \frac{\partial H}{\partial \rho}, \frac{\partial H}{\partial \omega}$, and $M_i$.
  3. Update the estimates using: $$\begin{bmatrix} y_{t+1} \\ \lambda_{t+1} \\ \rho_{t+1} \end{bmatrix} = \begin{bmatrix} y_t \\ \lambda_t \\ \rho_t \end{bmatrix} + \delta \begin{bmatrix} \frac{\partial H}{\partial \sigma}(y_t, \lambda_t, \omega_t) \\ \frac{\partial H}{\partial \rho}(y_t, \lambda_t, \omega_t) \\ \frac{\partial H}{\partial \omega}(y_t, \lambda_t, \omega_t) \end{bmatrix},$$ where $\delta$ is a diagonal matrix of step sizes.
  4. The update for the rotation matrix is performed by $O_{t+1} \rightarrow O_t e^{\delta_t M_i},$ where $\delta_t$ is the step size and $M_i$ is the gradient matrix for orientation.
  5. If the norm of the gradient vector is small, then stop. Else, set $t = t + 1$ and return to step 2.
- Record the limiting $(y, \lambda, T, O)$ for each pair of $\sigma$ and $\rho$ and select the solution that results in the largest $H$ value.

An example of this algorithm is shown in Fig. 8. Starting with a parameterized surface with the shape of a horse, shown in the left panel of Fig. 8, we transform it with random rotation, translation and scale, and simulate a point cloud using the parameters $\gamma = 80$, $\lambda = 20$, $\sigma = 0.010$, $\rho = 0.717$, shown in the second panel of Fig. 8. Using Algorithm 3 on this point cloud, the estimated values are $\hat{\gamma} = 83.29$, $\hat{\lambda} = 28.49$, $\hat{\sigma} = 0.016$, $\hat{\rho} = 0.712$ and the maximum log-likelihood ratio is $R(Y) = -471.12$. The third panel shows the original surface drawn at the estimated position, rotation, and scale along with the original $Y$ and the fourth panel shows the evolution of $H$ under Algorithm 3 for the optimal $\sigma$ and $\rho$.

**4. Shape classification**

The procedure of detecting shapes can be easily extended to classification of shapes by computing the LLR for a representative for each candidate shape class and then selecting the class with the smallest LLR. Let $R(Y|S_i)$ denote the LLR for the $i$th shape $S_i$. The estimated shape class is given by: $i = \text{argmin}_i R(Y|S_i)$. We will demonstrate this idea for both 2D and 3D shape classification.

**4.1. 2D classification**

We will illustrate the problem of shape classification in 2D point clouds using both simulated and real point clouds. The use of simulated data allows one to measure exhaustive classification performance versus noise, clutter, and other variables, while real data is useful to study individual cases.

**Simulated data:** In this experiment we take curves representing 10 shapes shown in the left of Fig. 9, generate point clouds according to the data model and perform classification, i.e., we compute LLR for each of the 10 classes and select the one with the smallest LLR. To estimate probability of correct classification, we have used 1000 runs (simulations of $Y$) for each value of $k$ (the number of points from the shape) and $\sigma$ at randomly generated transformations. The number of points from clutter is chosen to be the half of points from the curve. For example, $\gamma = 20$, $\sigma = 0.03$ implies that $Y$ is simulated by sampling 20 points on the curve and 10 clutter points on average. For these simulations, the underlying shape is picked from the set of ten randomly with equal probability. The results are shown in the right of Fig. 9 where the probability of
Fig. 9. 2D shape classification. Left: ten representing shapes. Right: comparison of 2D classification versus $k$ for different $\sigma$.

Fig. 10. Pre-processing of 2D images to obtain point clouds.

Fig. 11. Shape classification in 2D point clouds from real images. In each example, the first column is the real image, the second column is the generated point cloud and the third column is the LLRs for 10 classes.

correct classification is plotted versus $k$, for three different Gaussian noise levels. This plot suggests that, in case of low noise, the sampling of shapes by $k = 10$ results in approximately 90% classification rate.

Experiment 2: Real images: Now we present some classification results using point clouds extracted from real images as follows. Starting with an image, we use a median filter on the image to remove noise and calculate image gradients with an auto-adaptive threshold. We then merge rectangular regions that roughly contain high gradients and find the largest such rectangular region. Then, we obtain a binary image for the rectangular region. Next, we apply an image thinning algorithm to the rectangular region and find the contour of the object. Finally, we sample random points on the contour to obtain $Y$. Some steps of this process are shown in Fig. 10 using an example image.

Using this technique, we extract point clouds $Y$ from a number of test images: the images and the corresponding point clouds are shown in the left two panels of Fig. 11 in each example. For these images, we try to classify them according to the ten shapes $\beta$ shown in the left of Fig. 9. That is, in this experiment $Y$ comes from images and $\beta$ comes from the shapes shown in Fig. 9. The results, in form of the LLRs for each of the 10 classes, are shown in the right panels for each case in Fig. 11. In most cases we obtain a very low LLR for the correct shape (and sometimes a similar shape). It must be noted that the test
and training data came from different sources, the shapes present in the test images are slightly different from the training shapes, and that causes some of the misclassifications in these experiments.

4.2. 3D shape classification

To investigate shape estimation and classification in 3D, we construct a database of 20 classes from Princeton shape benchmark in Shilane et al. (2004), where each class contains 5 examples, shown in Fig. 12. The test set consists of 20 examples, generated by choosing one example from each class, and the training set contains the remaining 80 shapes. To generate cloud data, we take a surface from the test set, transform it with random transformation, and simulate \( Y \) by sampling \( k \) points from the transformed surface with a Gaussian noise and half number of clutter points. For each \( Y \), we compute the LLRs for each of the 80 training shapes using Algorithm 3 and select the class containing the shape with the smallest LLR. One such example is shown in the left panel of Fig. 13 where the two shapes with smallest LLRs are found to be from the same class as the point cloud. The performance curve versus number of sampling points \( k \) is plotted in the right of Fig. 13. In order to compare our performance with some current methods, we have studied the performance of classification using the Gromov–Hausdorff distance (Gromov, 1999; Mémoli and Sapiro, 2005) and the iterative closest point (ICP) algorithm (Besl and McKay, 1992) based on 1-nearest neighbor classifier. This plot suggests that the sampling of shapes by \( k = 500 \) points can result in approximately 90% classification rate using our method and the Gromov–Hausdorff distance. It also shows that our method and Gromov–Hausdorff distance are better than the ICP algorithm. It is intuitive that the classification performance varies from class to class. That is, we expect better performance for identifying classes that have small variations within the class and are distinct from other classes, compared to classes that have large variations and are similar to other classes. For example, for a fixed \( k = 250 \), the classification rate of nearest neighbor classifier is 78% for the class of “bottle” rather than 69% for the class of “couch”. Under the same setup, the rates are 70.4% for chair, 72.6% for airplane, 79.5% for shoe, and 78.4% for duck.

5. Shape estimation

Beyond the problems of shape detection and classification, there is an interesting problem of estimating the shape itself from the data. By shape estimation we mean that we select a relevant subset of points in the given cloud and join them in an appropriate order to form our best estimate of the underlying shape. This brings up two important questions: How
should we select the relevant points and how should we determine the ordering? For the first question, the issue can be handled using marked point process models, where each point is characterized by a binary mark: 0 for clutter and 1 for object boundary. Then, one can estimate these marks or labels according to a chosen model. While it is possible to impose a prior distribution on the parameters and use MCMC-based Bayesian estimation of labels, we have simply used the estimated parameters to select a subset of $Y$ as follows. We set $n = \hat{\gamma}$ (after rounding) as the number of points to be selected, and then pick $n$ data points with the largest values of $\hat{\alpha}_\sigma(y_i)$. For the second question, the one relating to the ordering of points, we use the ordering borrowed from the fitted hypothesized shape. That is, for each selected point, we find the nearest point on the object $\beta$ (under estimated position, orientation and scale parameters) and inherit the ordering from those corresponding points. We illustrate this process using examples from 2D and 3D domains.

5.1. Shape estimation in 2D

For a given point cloud and a hypothesis shape $\beta_0$, we first detect if the shape is present in the data. If yes, using the estimates $\hat{\gamma}$, $\hat{\sigma}$, and $\hat{\omega}$, we select $\hat{\gamma}$ number of points in $Y$ with the largest values of $\alpha_{\hat{\sigma}}$. Then, we connect them in the same order as their nearest neighbors on $\beta_0$. Examples of this process are shown in Fig. 14.

5.2. Shape estimation in 3D

Given a 3D point cloud $Y$ we first test the presence of given shape hypothesis $\beta_0$. If the shape is detected then, as earlier, we select $\hat{\gamma}$ points with the largest $\alpha_{\hat{\sigma}}$ values. Next we use the triangle information of the matched shape surface, i.e., $\beta_0$ at the estimated transformation, to reconstruct the shape from the point cloud. A basic vertex contraction algorithm taken from Garland (1999) is used here. Some examples are shown in Fig. 15. The point cloud in each example is simulated from the surface at randomly generated position, orientation, and scale with some added clutter. Each pair of panels shows a point cloud and estimated shape from the cloud. Fig. 16 shows that the method can also estimate part of shape correctly. Here each row shows a point cloud simulated from part of the shape template, estimated template under the estimated transformation in the cloud, and the shape estimated from the cloud.

Computational cost: We summarize the computational cost of our procedure. This cost is computed using MATLAB on a single-processor desktop PC. For computing LLR for a 2D shape involving $m = 100$ data points, the program takes approximately one second, and for computing LLR for a 3D shape involving $m = 500$ data points, it takes about eight seconds.

6. Conclusion

We have presented a fully statistical framework for detecting, classifying and estimating shapes in cluttered point cloud and have demonstrated it using interesting examples of both real and simulated data. This framework is based on a composite Poisson process: one for points generated from the shape and another for points belonging to the background clutter. This model allows computation of a log-likelihood ratio for each class against clutter and this ratio leads to a formal procedure for detection and classification of shapes. Furthermore, we can also estimate the shape from the cloud based on this framework and have applied it to both 2D and 3D cases.
Although this paper is motivated by problems in computer vision and image understanding, the framework is also generally applicable in other scientific disciplines, where one comes across point cloud data, such as bioinformatics, chemistry and biology. The framework may, of course, need adjustments according to the situation. For instance, one may need to model and estimate inhomogeneous Poisson processes for either the object or the clutter, or change the model for additive noise from Gaussian to a different family, or represent shapes using solid objects (with interior points) rather
than just their boundaries. The estimation of unknown parameters and mechanism for drawing inferences will change accordingly.

References


