Efficient and Noise Tolerant Action Recognition Using Negative Space Action Descriptors

Shah Atiqur Rahman¹,* , Insu Song¹, M. K. H. Leung², Ickjai Lee¹, and Joanne Lee¹

¹School of Business and IT, James Cook University Australia
²FICT, Universiti Tunku Abdul Rahman (Kampar), Malaysia
{atiqur.rahman, insu.song, ickjai.lee, joanne.lee}@jcu.edu.au, asmkleung@gmail.com

*Corresponding author

ABSTRACT

Due to the number of potential applications and their inherent complexity, automatic capture and analysis of actions have become an active research area. In this paper, an implicit method for recognizing actions in a video is proposed. Existing implicit methods work on the regions of subjects, but our proposed system works on the surrounding regions, called negative spaces, of the subjects. Extracting features from negative spaces facilitates the system to extract simple, yet effective features for describing actions. These negative-space based features are robust to deformed actions, such as complex boundary variations, partial occlusions, non-rigid deformations and small shadows. Unlike other implicit methods, our method does not require dimensionality reduction, thereby significantly improving the processing time. Further, we propose a new method to detect cycles of different actions automatically. In the proposed system, first, the input image sequence is background segmented and shadows are eliminated from the segmented images. Next, motion based features are computed for the sequence. Then, the negative space based description of each pose is obtained and the action descriptor is formed by combining the pose descriptors. Nearest Neighbor classifier is applied to recognize the action of the input sequence. The proposed system was evaluated on both publically available action datasets and a new fish action dataset for comparison, and showed improvement in both its accuracy and processing time. Moreover, the proposed system showed very good accuracy for corrupted image sequences, particularly in the case of noisy segmentation, and lower frame rate.

Keywords- Action recognition; Negative space action descriptors; Silhouette; Fuzzy membership; Implicit method; Cycle length, fish actions.
1. Introduction

One of the general goals of artificial intelligence is to design machines which act intelligently and in a human-like manner. For a machine to be truly intelligent and useful, it must have the ability to perceive the environment in which it is embedded. It needs to be able to extract information from its environment independently, rather than relying on the information supplied by users externally. The visual analysis of human motion attempts to detect, track and identify people, and more generally, to interpret behaviors of subjects from sequences of images, in which humans perform certain actions. Such technologies will allow us to communicate with machines more easily than before, by allowing more advanced input modalities, such as gestures. For example, people extend their hands to robots to do a handshake. In reply, the robots may do the same to make the handshake happen by recognizing the action of the people from their gestures, thereby avoiding the necessity of input from users and eventually improving the communication mechanism with the robots.

Difficulties in action recognition occur due to cluttered background, camera motion, occlusion, viewpoint changes and geometric and photometric variances of objects. Application areas for the human action recognition include virtual reality, games, video indexing, teleconferencing, advanced user interfaces and video surveillance.

1.1 Motivation and Proposed work

Much work has been done regarding action recognition (Aggarwal and Ryoo, 2011; Moeslund et al., 2006; Poppe, 2010; Weinland et al., 2011). Some methods build a model first and then recognize actions by tracking the changes of different limbs in either 2D (Ferrari et al., 2009) or 3D (Horaud et al., 2009) spaces. These approaches are complex, due to the large variability of shapes and articulations of human bodies, fast motions, self occlusion, changes in appearance, etc. Other approaches employ gradient or intensity based features, e.g., the bag of words methods (Wei et al., 2012) to recognize actions. The performance of the gradient and intensity based methods depends on the detection of a sufficient number of stable interest points. However, detecting interest points could be difficult due to complex backgrounds. Silhouette-based methods are becoming popular due to their robustness to noise and because it is easier to extract regions of interest (ROI) than the space-time interest points (Wang and Suter, 2007a). Based on the dynamics of the actions, silhouette based
methods can fall into two categories: explicit and implicit. In the explicit-model based methods, actions are treated as a composition of a sequence of poses (Wang and Suter, 2007a). First, each pose is described individually, and then the dynamics of poses are represented by, for example, Hidden Markov Model (HMM) and Dynamic Time Warping (DTW) to recognize actions. However, most of the explicit-model based techniques need to model each action individually, and usually explicit methods take a longer time in action classification than implicit methods. Implicit methods extract features from the combination of all the poses of an action sequence and generate action descriptors. The implicit modeling approaches have the advantage of simple and efficient action recognition, i.e., faster processing time, and can work with a small number of training data. However, most of the implicit methods require an additional step of dimensionality reduction by using different techniques, e.g., Principle Component Analysis (PCA) and Linear Discriminant Analysis (LDA). Additionally, some of the systems require manual identification of cycle lengths (Kumar et al., 2011) or use sliding windows of fixed sizes (Gorelick et al., 2007) to construct the action template.

In this paper, we propose an implicit-model based method that uses Negative Space Action Descriptors (NSAD) which form descriptions of actions using surrounding regions of poses. Since no individual features, such as hands and head, need to be identified and tracked, NSAD constructs much more efficiently than previous methods and is robust to noisy segmentation. In this paper, we show that the method can be applied to recognition of human actions, as well as activities of other subjects, e.g., fishes. The proposed system extracts features from the negative space, while other implicit methods extract features from the positive space. For action recognition, the silhouette of a subject is termed positive space, and the surrounding regions of the silhouette inside its bounding box are referred to as the negative space of the subject, as shown in Fig. 1. Basically, the negative space of a subject is the inverse set of the positive space (silhouette) of the subject. Now, the question is why negative space based action description is efficient? The answer is that negative space can describe poses by simple shapes which are naturally formed inside the bounding box, avoiding sophisticated and complex limb tracking or high dimensional polygonal approximation, which are used to describe poses in positive space based methods (Ikizler and Forsyth, 2008; Sigal and Black, 2006). For example, positive space based methods, such as feature tracking methods, first identify different limbs
of the human body hierarchically by using some reference points and then track the limbs to recognize actions, which is a complex process. On the other hand, the negative space based methods only need to identify the bounding box of the human silhouette, which is much simpler than identifying and tracking different limbs, and the negative space regions formed inside the bounding box can be described using simple shapes. In Fig. 1(b), each of the four negative space regions (1, 2, 3, 4) is represented by either triangles or quadrangles which are formed naturally by the bounding box and the silhouette. A pose can then be represented by using the simple shapes only, as shown in Fig. 1(c).

![Figure 1](image)

**Fig. 1:** (a) bounding box of a human silhouette. White regions inside the box are negative space, and the human silhouette is the positive space; (b) Negative space regions can be viewed as triangles or quadrangles; (c) Poses are represented y using triangles or quadrangles only.

Another advantage of our system is that the method is relatively robust to partial occlusion and noisy segmentation, since negative space based pose description is less affected by partial occlusion and noisy segmentation (Rahman et al., 2012a). Specially, in the case of noisy segmentation, negative space based methods are particularly better than other state of the art methods. Furthermore, since each negative space region can be described by simple shapes, e.g., triangles, quadrangles, our system describes poses by low dimensional and computationally efficient feature vectors. These low dimensional feature vectors are later used to form negative space action descriptors (NSADs) for describing actions. Hence, each NSAD describes an action with low dimensional features. This implies that NSAD does not require dimensionality reduction, whereas most of the other implicit methods require a significant amount of processing time for dimensionality reduction (Zhang et al., 2010). Additionally, in our system, we propose a method based on signal
processing technique to determine the cycle length of each action automatically, without any prior knowledge of the action.

The feature extraction method from poses proposed here is the same as our previous method (Rahman et al., 2011), but the action representation technique proposed here is novel. The main contribution of this paper is the novel method of constructing action descriptors that generate features from negative space based pose descriptions which can efficiently describe poses using simple shapes anchored to the bounding box, whereas positive space based methods need to identify suitable reference points to perform complex limb tracking or high dimensional polygonal approximation. The new action descriptors can be used to recognize actions more efficiently with comparable accuracy than the state of the art methods. Furthermore, the proposed method has shown very good accuracy in the case of corrupted image sequences, especially for corrupted images due to noisy segmentation. Another contribution is to compute the number of frames in a cycle (cycle length) of an action automatically without any prior knowledge about the action. A shorter version of this work can be found in (Rahman et al., 2012b).

1.2 Paper Organization

The remainder of this paper is organized as follows. In Section 2, previous work is reviewed. In Section 3 we describe our system in detail, as to how it computes moving speeds, partitions negative space, extracts features, forms action descriptors, and classifies descriptors. Section 4 shows experimental results of our system and compares system performance with other methods. Section 5 concludes the paper.

2. Previous work

Many criteria could be used to classify previous work, for example, low level feature cues, type of models and dimensionality of the tracking space. We have chosen the first criterion, since we mainly focus on proposing simple and efficient features to describe actions. Existing work can be broadly divided into three major categories based on the low level feature cues. They are feature tracking based methods, intensity and gradient based methods and silhouette based methods. Some of
the related work is discussed in this section. Fuller reviews can be found in (Aggarwal and Ryoo, 2011; Moeslund et al., 2006; Poppe, 2010; Weinland et al., 2011).

Feature tracking based methods: Many methods recognize actions based on tracking features in either 2D or 3D space. Hence, body segments can be approximated as 2D ribbons and 3D volumes accordingly. In 2D space, these models are represented by stick figure (Guo et al., 1994), ribbons (Leung and Yang, 1995), blobs (Wren et al., 1997) and rectangular patches (Ju et al., 1996). Sometimes, tree structure model representation is used for pose estimation (Felzenszwalb and Huttenlocher, 2005). In these models, body parts are represented by nodes and edges of graphs where nodes represent limbs and edges represent kinematics constraints between connected limbs. (Mikolajczyk et al., 2004) introduced robust AdaBoost body part detectors to locate different human parts. These parts model the joint likelihood of a body parts configuration. (Micilotta et al., 2005) extended this approach by using RANSAC to assemble a body parts configuration with pose constraints which are learned previously. Since the descriptions of body limbs are local, these methods can fail to track different limbs with the same image data, e.g., two legs of a person). (Sigal and Black, 2006) used a graphical model where each node of the graph represents different limbs and edges represent kinematics constraints between different limbs. Their method recovers 2D poses even in the presence of occlusion, but the method needs manual initialization. Later, (Ferrari et al., 2009) proposed a very similar method to (Sigal and Black, 2006), where the initialization is done automatically.

2D feature tracking approaches are relatively computationally simple, usually work on the positive space, i.e., the silhouette, and provide a semantic level description of poses. However, these methods are sensitive to the details of human shape variations, noise and segmentation errors.

Besides 2D feature tracking, some methods use 3D information about body parts to classify actions. (Hofmann and Gavrila, 2009) estimated upper body pose from three cameras where candidate poses were generated from one camera image and verified by another two camera images. However, this method cannot handle occlusion. (Ikizler and Forsyth, 2008) proposed a tracker based method where complex queries are searched by Finite State Automata. However, if the tracker fails to track different limbs correctly, which can occur during self-occlusion, the system performance is degraded.
(Lee and Nevatia, 2009) proposed a three-stage approach which enables a hierarchical estimation of 3D poses. They addressed various issues, including automatic initialization, self and inter-occlusion. Conversely, the computation speed of their system is high. (Horaud et al., 2009) proposed a method for tracking human motion based on fitting an articulated implicit surface to 3D points and normals. However, their system performance decreases for improper initialization of the algorithm. 3D feature tracking techniques are accurate and invariant under viewing angle changes, but the computational cost is high. Multiple cameras may be required for reconstruction of the 3D body configuration. Hence, it is difficult to apply to human action recognition on video from real world scenes.

In general, the feature tracking approach is complex, due to the large variability of shape and articulation of human body, fast motions, self occlusion, changes of appearance, etc.

**Intensity and gradient based methods:** These methods employ gradient or intensity based features to recognize actions. In these methods, first the space-time interest points are detected from the input pose, and then actions are recognized by modelling the interest points. ‘Bag-of-words’ approaches that originated from text retrieval research are being adapted for action recognition. These studies are mostly based on the idea of forming codebooks of ‘spatio-temporal’ features. (Laptev and Lindeberg, 2003) first introduced ‘space-time interest points’ features and used SVM to classify actions. By using linear filters, (Dollar et al., 2005) extracted cubes from poses and recognized actions by forming histograms from these cubes, whereas (Niebles et al., 2008) used pLSA on the extracted cubes for recognition. (Fathi and Mori, 2008) developed a method based on mid-level features which were extracted from low level optical flow features. AdaBoost classifier is then used to recognize actions from these mid-level features. However, this method needs some parameters to be manually initialized.

Scale Invariant Feature Transform (SIFT), which was introduced by (Lowe, 2004), has been used in a variety of applications including object classification. (Uemura et al., 2008) employed SIFT features with a motion model based on optical flow in order to accurately classify multiple actions. Their system was able to successfully recognize actions in a sequence with large motion and scale changes. Partly inspired by (Lowe, 2004), the Speeded Up Robust Features (SURF) (Bay et al., 2008) utilizes second order Gaussian filters and the Hessian matrix in order to obtain fast scale and rotation
invariant interest point detectors and descriptors. Generally, the high dimensionality of the descriptors is a drawback of SIFT descriptors.

Performance of intensity and gradients based methods depends on the reliable and sufficient detection of stable interest points, which can be difficult to derive in cases of noisy and dynamic backgrounds.

Silhouette based methods: Silhouette based methods are becoming popular, due to their robustness to noise and easy extraction of regions of interest (Wang and Suter, 2007a). Silhouette based methods first obtain foreground silhouettes by background segmentation and then extract features from the segmented images to recognize actions. Proposed methods on silhouette based action recognition differ in their ways of modeling poses and pose sequences. Based on the dynamics of the actions, proposed methods can fall into two categories: explicit and implicit.

In explicit models, actions are treated as a composition of a sequence of poses. These models consist of two components: pose description and modeling of the dynamics of poses. In these models, sometimes poses are described by histograms of oriented rectangles (Ikizler and Duygulu, 2009), combinations of motion, color and texture (Yamato et al., 1992), projections of human silhouette (Bissacco et al., 2007; Lao et al., 2009), moments (Davis and Tyagi, 2006), un-weighted graphs (Lv and Nevatia, 2007), weighted graphs (Li et al., 2008), curvatures in different area of silhouette (Ali and Aggarwal, 2001; Mazzaro et al., 2002), etc. After describing each pose, actions are modeled by HMM (Davis and Tyagi, 2006; Lao et al., 2009; Yamato et al., 1992) , GMM (Li et al., 2008), DTW (Ikizler and Forsyth, 2008; Rahman et al., 2012b), etc. However, most of the explicit-model based techniques need to model each action individually, and action classification time is higher than for implicit methods.

In the case of implicit-model based methods, features are extracted from combinations of all the poses of an action sequence. Motion History Image (MHI) and Motion Energy Image (MEI) are the methods of describing actions as temporal templates (Bobick and Davis, 2001; Meng et al., 2007), where matching is done by Hu moments (Bobick and Davis, 2001) or a SVM classifier (Meng et al., 2007). A 3D extension of temporal templates was proposed by (Weinland et al., 2006), where Motion History Volume (MHV) was constructed by using images from multiple cameras, and classification
was done by Fourier analysis in cylindrical co-ordinate. Gait Energy Image (GEI) and Motion Intensity Image (MII) recognize actions based on Eigen space methods and dimensionality reduction by either PCA (Han and Bhanu, 2005) or LDA (Diaf et al., 2010). (Zhang et al., 2010) proposed active energy image (AEI) which has the advantage of retaining the dynamic characteristics of gait for recognition. (Kumar et al., 2011) proposed a LDA tree based classifier to recognize actions. They represent an action using a thinning human silhouette of each frame and then form a histogram for each grid of the bounded ellipse of the thinned silhouette. However, their system is not scale invariant and is not robust to partial occlusion. (Gorelick et al., 2007) represent actions as space-time shapes and extracted space-time features for action recognition, such as local space-time saliency, action dynamics, shape structures and orientation. The implicit modelling approaches have the advantage of simple action recognition, i.e., faster processing time, and can work with a small quantity of training data. However, most of the implicit methods require an additional step of dimensionality reduction, for example, by using PCA and LDA. Furthermore, some of the implicit methods require manual estimation of cycle length (Kumar et al., 2011) or a sliding window (Gorelick et al., 2007) to form the action templates.

In this paper, we propose an implicit method called Negative Space Action Descriptors (NSAD) which forms action descriptors by extracting features from the surrounding regions of a silhouette, term “negative space,” by art theory. Extracting features from negative space provides the advantage of describing actions with a few computationally simple features, i.e., dimensionality reduction is not required, which are relatively robust to noise and partial occlusion. Unlike other implicit methods, our system automatically computes the cycle length of actions without any prior knowledge about the actions.
3. Proposed System

A diagram of our proposed system is shown in Fig. 2. The training data and input (test) data are processed separately, and processes share some common steps. The input to our system is an image sequence containing an action performed by a single person. Multi-person activity recognition is left to the future. In the pre-processing step, background segmentation and shadow elimination are performed on the input image sequence. The speed of the human body is calculated from the pre-processed images and is used to create (motion-based) feature values of the input image sequence. This speed feature is computed based on the displacement of silhouettes of poses from frame to frame.

During the training phase, fuzzy membership functions are generated automatically using the speeds of training sequences. These fuzzy membership functions are then used for calculating speed scores, i.e., motion-based features, as indicated in Fig. 2. Along with the motion based features, shape based features are extracted from each pre-processed image by performing the following processes: i) capturing of the negative space of a pose by cutting a bounding box containing a human silhouette...
from the image, ii) partitioning of complex negative space regions into simple regions and iii) extraction of region-based and positional features from negative space regions and forming the feature vectors of the poses. After forming the feature vectors for all the images, i.e., poses, the number of frames in a cycle of each action, i.e., cycle length, is calculated automatically based on the feature vectors of the poses of the actions. The cycle length of the actions is then used along with the shape-based feature vectors of poses to form the action feature vectors, called NSADs, for the input image sequence. Next, the speed scores are appended to the NSADs, and, lastly, the action type of the input sequence is recognized by employing a Nearest Neighbor classifier.

3.1 Preprocessing

In the preprocessing step, two tasks are performed: background segmentation and shadow elimination. We employ an existing background segmentation algorithm to segment the background. Many background segmentation approaches have been proposed. Among them, background subtraction is widely used (Benezeth et al., 2008). We employ one of the background subtraction algorithms (Li et al., 2004) in our work. Two versions of this algorithm are available: one with shadow detection and the other without shadow detection. In this work, the without-shadow-detection version is employed. After the segmentation, binary images are obtained, with each containing one silhouette of the subject.

Segmented binary images may contain shadows if the input images contain shadows. Negative space based methods can recognize actions effectively even in the presence of small shadows (Rahman et al., 2011), but they are less effective with long shadows. Hence, we discard long shadows from segmented binary images by using a histogram-based technique as proposed in (Rahman et al., 2011), where shadows are detected based on the distribution of foreground pixels, and then discard it.

3.2 Speed calculation

The speed of an object is an important cue for the recognition of actions. For example, the speed of a running action of a human is greater than the speed of a walking action. The speed $hor_{sp}$ is
calculated by taking the displacement of the centroid of a silhouette along the X-axis of consecutive frames divided by the time required for that displacement.

\[ \text{hor}_x = \frac{ds}{dt} = |x_i - x_{i-1}| \times \text{fr rate} \]  \\
(1)

where \( x_i \) is the X-axis coordinate of the centroid of the silhouette of the \( i^{th} \) frame and \( \text{fr rate} \) is the frame rate of the sequence.

Then, this speed is normalized to the scale variations

\[ \text{hor}_n = \frac{t_{\_ \text{hor disp}} \times \text{fr rate}}{n-1 \times \text{t height}} = \frac{t_{\_ \text{hor disp}} \times \text{fr rate}}{t_{\_ \text{height}}} \]

where \( t_{\_ \text{hor disp}} = \sum_{i=2}^{n} |x_i - x_{i-1}| \), \( n \) is the total number of frames in the sequence and \( t_{\_ \text{height}} \) is the sum of the heights of all the bounding boxes of silhouettes excluding the first frame.

Viewpoint changes can affect the accuracy of the speed calculation method (2). This can be compensated for by performing viewpoint normalization: \( t_{\_ \text{hor disp}} = t_{\_ \text{obs disp}} / \cos \delta \), which is illustrated in Fig. 3 where \( \delta \) is calculated as

\[ \delta = \frac{1}{2} \tan^{-1} \left[ \frac{2\mu_{11}}{\mu_{20} - \mu_{02}} \right] \]  \\
(3)

Here, \( \mu_{ij} \) is the \( i, j^{th} \) order centralized moment of the centroids (the black dots in Fig. 3) of the human silhouette in the sequence. After viewpoint normalization, the speed becomes

\[ \text{ac hor}_n = \frac{t_{\_ \text{obs disp}} / \cos \delta \times \text{fr rate}}{t_{\_ \text{height}}} = \frac{\text{hor}_n}{\cos \delta} \]  \\
(4)

Note that (4) is applied only to those actions having significant movement, e.g., walking, running, which can be determined by the value of \( \text{hor}_x \). For non-moving actions, e.g., waving, clapping, (4) is not applied, since viewpoint change does not affect the speed calculation for these actions.
3.2.1 Automatic generation of Fuzzy membership functions

During the speed calculation step, in the training phase, for each type of action, fuzzy membership functions are generated which are then employed to calculate speed scores. The Gaussian membership function is chosen for the shape of the membership functions, since it requires only two parameters, i.e., mean and standard deviation. These two parameters can be calculated from the speeds of each action type of the training data. First, the speed of each training sequence is calculated. Next, to remove noisy data, the top and bottom 5% of the data in terms of speed are truncated from each action type, and then membership functions are generated by calculating their mean and standard deviation from the truncated data. To avoid over-partitioning of the feature space, membership functions are merged depending on the overlapped area of the membership functions. If one membership function overlaps in more than 60% of its area with another membership function, both membership functions are replaced by a new membership function. The parameters of the new membership function are determined by combining both of the overlapping membership functions. This merging process goes on until no new membership function is generated. One example is shown in Fig. 4, where membership functions of ‘skip’, ‘side’ and ‘jump’ actions (shown in Fig. 4(a)) are merged into one membership function in Fig. 4(b). The same merging occurs for ‘bend’, ‘jack’, ‘jump’, ‘wave1’ and ‘wave2’ membership functions.

To determine the overlap amount of two membership functions, the intersection area of two membership functions has to be determined. The intersection area of two curves can be found by integrating the difference between the curves up to the initial and final limits of the intersection points.
between the two curves. For example, let \( f(x) \) and \( g(x) \) be two curves, and \( x = a \) and \( x = b \) be the initial and final intersection points of the two curves, respectively; so, the area between the curves is

\[
\text{ins}_\text{area} = \int_a^b (f(x) - g(x)) \, dx
\]

(5)

Now, for two Gaussian curves with means \( \mu_1 \) and \( \mu_2 \), and standard deviations \( \sigma_1 \) and \( \sigma_2 \), the intersection area will be

\[
\text{area}_{\mu_1, \mu_2} = \int_a^b \left( \frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-\frac{(x-\mu_1)^2}{2\sigma_1^2}} - \frac{1}{\sqrt{2\pi\sigma_2^2}} e^{-\frac{(x-\mu_2)^2}{2\sigma_2^2}} \right) \, dx
\]

(6)

At this moment, the task is to determine the value of two limits \( a \) and \( b \), since other parameters of the equation are known. Usually, Gaussian distribution ranges from \(-\infty\) to \(+\infty\), but to reduce calculation, the three-sigma rule is employed. The three-sigma rule or empirical rule states that approximately 68% of values of a normal distribution lie within one standard deviation (\( \sigma \)) away from the mean; approximately 95% of the values lie within two standard deviations and approximately 99.7% are within three standard deviations. Suppose two overlapped Gaussian membership functions \( g_1 \) and \( g_2 \) are given, and their parameters are \( \mu_1, \sigma_1 \) and \( \mu_2, \sigma_2 \) respectively. Hence, the lower limit and upper limit of (6) will be

\[
a = \max(\mu_1 - 3\sigma_1, \mu_2 - 3\sigma_2)
\]

(7)

\[
b = \min(\mu_1 + 3\sigma_1, \mu_2 + 3\sigma_2)
\]

(8)
One example of overlapping Gaussian membership functions is shown in Fig. 5, where the lower limit of (6) will be $a=\mu_2 - 3\sigma_2$ and the upper limit will be $b=\mu_1 + 3\sigma_1$. After calculating the area, if the ratio $\frac{\text{area}_{\text{ins}}}{\min(\text{area}_{g_1}, \text{area}_{g_2})}$ is greater than or equal to 0.6, i.e., at least 60% overlap, the two membership functions will be merged.

3.3 Extracting Features from Negative Space

To describe each pose, pose features are extracted from the negative space of the poses in our system. For this purpose, first, the negative space of a pose is captured. The complex negative space regions are then partitioned into simple negative space regions to describe the regions using simple shape based features. After the region partitioning process, location and shape based information from each negative space region of the poses are extracted to describe each pose. Below, we describe these processes in more detail.

3.3.1 Capturing negative space

From the definition of negative space, it is known that the space between an object and the canvas containing the object is negative space (Rahman et al., 2012a). In the case of human actions from segmented binary images, the object is a human silhouette. To capture the negative space, we need to define the canvas for the human silhouette. In our system, we define the canvas as the bounding box of the human silhouette. In this scenario, the regions inside the bounding box other than the human silhouette form the negative space. For example, in Fig. 6, the white regions inside the
bounding box are the negative space regions. Features are extracted from these white regions only to describe the poses. Negative space regions will contain the same information as the positive space regions, in the sense that the negative space is the complementary set of the positive space, but negative space regions can describe the poses more efficiently using simple shapes anchored to the bounding box, whereas positive space based methods need to find suitable reference points to perform complex limb tracking or high dimensional polygonal approximation.

3.3.2 Region partitioning

Due to continuous movement of objects, the number of the negative space regions of the same poses of the same person captured in different times may be different as shown in Fig. 6, where poses 6(a) and 6(b) are taken from the same pose group, but the number of the negative space regions is not the same. To cope with this situation and to simplify the matching process, region partitioning is applied. We employ the region partitioning technique of (Rahman et al., 2012a), where for each negative space region of the poses, the peninsula is identified by the line growing process, which is illustrated in Fig. 6. A peninsula for partitioning a region is identified by the protrusive measure of three distances (Fig. 6(c)). The region is partitioned into two by the tip of the peninsula if the protrusive measure is greater than a certain value (Fig. 6(d), Fig. 6(e)).

![Fig. 6: Region partitioning scenarios. (a) No partition is needed, (b) partition is desired (c) partitioning measures taken for region ‘x’ of (b), (d) partition output of region ‘x’. (e) Final partition output of (b)](image)

3.3.3 Positional feature

To extract the positional features of a negative space region, the bounding box of the pose is labeled with 14 anchoring points (Fig. 7). For each region, a mid-point on the side of the bounding box is computed (‘*’ in Fig. 7), and the region is assigned a positional label with respect to the nearest
anchoring point from that mid-point. For example, anchoring points for regions A, B, C, D and E are points 1, 12, 9, 5 and 7, respectively.

![Positional features](image)

**Fig. 7:** Positional features. Numbers represent the anchoring points and letters represent the regions, ‘*’ represents the mid-point for each region

### 3.3.4 Region based features

Simple region based features are extracted to describe the shape of each negative space region. Shape-based features are chosen to approximate the regions using triangles or quadrangles, since negative space regions can be approximated by these two simple shapes (Rahman et al., 2012a). Our shape based features are area, orientation, eccentricity, rectangularity, horizontal and vertical side lengths of the bounding boxes included in a region. We selected these six features to describe a region, because these features are effective in approximating triangles and quadrangles (Rahman et al., 2012a). Most of the features are extracted by statistical moments. All the feature values are normalized to the range [0, 1].

### 3.4 Generating Action Descriptors

In this section, we describe the process of constructing action descriptors. First, we form feature vectors for each pose from the extracted features described in the previous section. The cycle length is required during the formation of the action descriptors. Cycle length is computed automatically from the feature vectors of the poses of a sequence. Next, the action descriptors are formed based on the feature vectors of the poses of an action cycle.
3.4.1 Forming feature vector of a pose

Two types of features, i.e., positional and region based features, are combined to generate feature vectors. To do this, the feature vector of each pose is divided into 14 blocks representing the 14 anchoring points. Each of these blocks is further divided into six cells to represent the six shape based features. For a negative space region, all six feature values are assigned to the block corresponding to the anchor point of that region. For instance, in Fig. 7 the anchor point of region 'A' is 1, hence the six feature values of region 'A' are assigned to block 1 (Fig. 8). The gray scale color in each cell in Fig. 8 represents the value of a feature (0 is black and 1 is white). Hence, a fully black block corresponds to an anchoring point not assigned to any negative space region. The dimensionality of the feature space of the proposed system is 6×14=84, whereas other implicit methods' dimensionality is the same as the resolution of the input segmented image or the size of the bounding box (Chen et al., 2009; Han and Bhanu, 2005; Zhang et al., 2010), which is much higher than ours.

![Fig. 8: Feature vector of Fig. 7 represented as gray scale image](image)

3.4.2 Computing cycle length

To construct the NSAD, the cycle length of the video sequence has to be determined. In this case, we are denoting the minimum number of frames required to complete the action as the cycle length. For this purpose, we calculate the distance between the feature vector of the first frame in the sequence and the feature vectors of the rest of the frames in the sequence. Some implicit methods use fixed sized windows (Gorelick et al., 2007) or manually identify the cycle length (Kumar et al., 2011) to form the action descriptors. Some other methods compute the cycle lengths based on the change of pixels of the lower part of human body (Han and Bhanu, 2005; Zhang et al., 2010), but those methods will fail to determine cycle lengths for those actions which do not have any leg movement (e.g. two-hand waving). Fig. 9 illustrates the cycle length calculation technique proposed in our system.
Finding the cycle length of an action is similar to finding the fundamental period of a wave signal. Fundamental period can be computed by computing fundamental frequency, which is the reciprocal of the fundamental period. There are two types of fundamental frequency tracking algorithms: tracking in time domain and tracking in frequency domain. Time domain processes are straightforward and less expensive. Among the time domain processes, we choose a combination of auto-correlation function (ACF) and average magnitude difference (AMDF) method, since the combined method performs well in cases of noisy data (Abdullah-Al-Mamun et al., 2009). ACF over AMDF is employed in our system via the following equations.

\[
C_{Len} = \arg \max_\tau \left( \frac{acf(\tau)}{amdf(\tau)} \right)
\]  

(10)

\[
acf(\tau) = \sum_{i=0}^{n/2} s_i s_{i+\tau}
\]  

(11)

**Fig. 9:** Example of cycle length calculation. (a) Distance between 1st frame and all the frames of a sequence. (b) computed ACF over AMDF coefficients and (c) some example frames of the sequence.
Here $s_i$ is the distance between the 1st and $i^{th}$ frames, $\tau$ is the time lag in terms of frames and $n$ is the number of frames in $s$. Note that we calculate ACF over AMDF only for half of the sequence, to avoid the tapering effect. One example of computing the cycle length is shown in Fig. 9 where a sequence of the ‘walk’ action type is shown. Distances from the 1st frame to all other frames in the sequence are shown in Fig. 9 (a), and the ACF over AMDF coefficients are shown in Fig. 9 (b). For this example, the cycle length is 16, as the maximum coefficient occurs at that point. Some example frames are shown in Fig. 9 (c). Observe that frame 1 and frame 16 are approximately the same pose, which indicates that the cycle length of this sequence is 16 indeed.

3.4.3 Forming descriptor of an action

To form the NSADs, different statistics, i.e., mean, standard deviation, delta and delta-delta, are computed from the pose feature vectors of an action cycle. Next, one or more of these statistics are combined together to construct the NSADs. We form three types of NSADs, which are described below:

**NSAD-type1:** In this type, we form NSAD by taking only the means of the feature vectors of an action cycle, as in the following equation:

$$NSAD_j = \frac{1}{M} \sum_{i=1}^{M} f_{i,j}$$  \hspace{1cm} (9)

where $j=1,2,...,84, f_{i,j}$ is the $j^{th}$ feature of $i^{th}$ pose feature vector, $M$ is the number of poses in a cycle of an action, i.e., cycle length, which is automatically calculated by the method described in the previous section. In this type, the length of each NSAD is 84.

**NSAD-type2:** In this type, we construct NSAD by appending the standard deviation to the mean of the feature vectors. Hence, NSAD-type2 can be expressed by the following equation:

$$NSAD_j = \text{concatenate} \left[ \left( \frac{1}{M} \sum_{i=1}^{M} f_{i,j} \right), \left( \frac{1}{M} \sum_{i=1}^{M} \left( f_{i,j} - \bar{f}_j \right)^2 \right)^{\frac{1}{2}} \right]$$  \hspace{1cm} (10)

$$\bar{f}_j = \frac{1}{M} \sum_{i=1}^{M} f_{i,j}$$  \hspace{1cm} (11)
Length of each NSAD-type 2 is 168.

**NSAD-type3:** We add two more statistics, i.e., Delta and Delta-Delta, of the feature vectors to form the type 3 NSAD. Delta and Delta-Delta coefficients capture the temporal information of the features among the neighboring poses. The Delta coefficients are computed using the following linear regression formula:

$$d_t = \frac{\sum_{\theta=1}^{\Theta} \theta (c_{t+\theta} - c_{t-\theta})}{2 \sum_{\theta=1}^{\Theta} \theta^2}$$  \hspace{0.5cm} (12)

where $d_t$ is the delta coefficient at time $t$, $c_t$ is the feature coefficient and $\Theta$ is the window size which is 5 in our system. $\text{delta}(c)$ is the mean of all $d_t$ for the coefficient $c$. The Delta-Delta coefficients are computed using linear regression of Delta features. The type 3 NSAD is defined as:

$$\text{NSAD}_t = \text{concatenate}\left(\frac{1}{M} \sum_{i=1}^{M} f_{i,t}, \left(\frac{1}{M} \sum_{i=1}^{M} (f_{i,t} - \bar{f})^2\right)^{1/2}, \text{delta}\left(f_{\tau,t}\right), \text{delta}\left(\text{delta}\left(f_{\tau,t}\right)\right)\right)$$  \hspace{0.5cm} (13)

Length of NSAD-type 3 is 336.

Some NSADs of type 1 are shown in Fig. 10, where three action sequence descriptors are shown for each action type. Notice that the descriptors from the same action type form similar NSADs, and the descriptors from different action types form dissimilar NSADs (e.g. wave and jog) in terms of
position of negative space regions and their feature values. This indicates that NSADs can contain distinguishing information for different types of actions.

3.4.4 Appending speed score to NSAD

After forming different types of NSADs, we append the speed scores of the input sequence to the NSADs. For instance, in case of Weizmann datasets, four speed scores are appended to each descriptor, since there are four membership functions in the dataset (Fig. 4 (b)). For an action sequence, the speed scores are calculated for each membership function based on the actual speed of the subject. Before being appending to the NSADs, the speed scores are subtracted from 1 to convert them from similarity measures to dissimilarity measures, since we want to minimize the distance between two image sequences from the same action type.

3.5 Classification

Given an input action sequence, we form the NSADs for the sequences and find the Euclidean distance with NSADs of the training sets which are generated during the training phase. Nearest Neighbor classifier is used to recognize the input action. At this moment, we are interested to investigate the description power of NSADs in case of action recognition. Hence, we employ one of the simple distance metrics and the simplest classifier in our system. Use of other distance metrics and classifiers is left to future work.

4. Experimental results

The method was tested on two publically available datasets and one novel fish-action dataset. Some example frames from different datasets are shown in Fig. 11. The fish action dataset is available on the Internet\(^1\) for other researchers to compare their performance results. Parameters of our system are estimated from the training data of the Weizmann dataset and applied on all the datasets.

**Weizmann dataset:** In this dataset, there are nine persons performing ten actions (Gorelick et al., 2007). Background segmented images are provided by the author of the dataset, and we employed those in our system. Fuzzy membership functions employed for this dataset are shown in Fig. 4. We employ a leave-one-out (LOO) testing scheme, as most other systems also use this testing scheme.

\(^1\)(http://kdd.kopo.com)
**KTH dataset**: This dataset is more challenging than the Weizmann dataset, due to the considerable amount of camera movement, long shadows, different clothing and scale and viewpoint variations (Schuldt et al., 2004). There are 25 persons performing six actions in four different scenarios. Each video is sub-divided into four sequences, and there are 2391 sequences in the dataset. To extract the silhouette, we employ the algorithm by (Li et al., 2004). Fuzzy membership functions of this dataset for speed score calculation are shown in Fig. 12 (a); they are generated by the same method as used with the Weizmann dataset. As with the Weizmann data set, we employ LOO testing scheme for this dataset, too.

---

**Fig. 11**: Example frames of various datasets used in the system: (a) Weizmann dataset, (b) KTH dataset and (c) Fish action dataset
Fish action dataset: To test our method on detecting animal actions, we collected underwater video footage of different fishes in Watsons Bay on the northwest side of Lizard Island in Australia. The original video recordings were taken using 720x576 resolution; they were then down sampled to 640x480. Five fish actions were manually identified: (a) fishes turning from left to middle (LM); (b) fishes turning from right to middle (RM) facing the camera; (c) fishes searching for food (FD); (d) fishes swimming from left to right side of the frame (SLR) and (e) fishes swimming from right to left side of the frame (SRL). The video clips containing only these five fish actions were then used to generate the fish action dataset. There are a total of 95 sequences (LM-33, RM-33, FD-11, SLR-09 and SRL-09). Background segmented images are obtained manually. Fuzzy membership functions for this dataset are shown in Fig. 12 (b). As with the previous two datasets, we employ LOO testing scheme for this dataset. For this dataset, we compute the direction of the fishes by taking the sign of the speed, i.e., if the speed is negative, fishes’ direction is left to right, otherwise right to left, to distinguish between different direction actions.

4.1 Classification result

<table>
<thead>
<tr>
<th>NSAD Type</th>
<th>Dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Weizmann</td>
</tr>
<tr>
<td>Type 1</td>
<td>94.45%</td>
</tr>
<tr>
<td>Type 2</td>
<td>95.56%</td>
</tr>
<tr>
<td>Type 3</td>
<td>95.56%</td>
</tr>
</tbody>
</table>

The accuracy of different types of NSADs for different datasets is shown in Table 1. For all the datasets, NSAD type 2, which is a combination of mean and standard deviation of the features,
gives best result. Generally, NSADs constructed with more statistics from the features should give better results but the construction method also increases the dimensionality of the feature space. NSAD type 3 should give the best or at least same result as NSAD type 2. However, in this system, the test sequences are not aligned with the training sequences, i.e., test and training sequences may not start from the same type of pose, which affects the calculation of delta and delta-delta coefficients.

![Confusion matrix](image)

**Fig. 13:** Confusion matrix for different datasets (a) Weizmann dataset, (b) KTH dataset (c) Fish action dataset. For all dataset left: NSAD-type1, left: NSAD-type2, left: NSAD-type3. For all the confusion matrices, rows correspond to input action and columns correspond to training actions.

Our method achieved a high level of accuracy. Comparisons of our method with other methods are shown in Table 2 and Table 3 for Weizmann and KTH datasets, respectively. Our accuracy is 95.56% and 94.49% for Weizmann and KTH dataset, respectively which is comparable with other state of the art methods. In Table 2, we also show the average feature extraction time of a (180 × 144 × 50) segmented sequence and classification time for Weizmann dataset that are reported
by the corresponding authors. The feature extraction time of our un-optimized Matlab® code is 4.3s for a (180 × 144 × 50) segmented sequence on a Pentium 4, 3.4 GHz machine. This is approximately 12 frames per second (≈50/4.31). The speed can be further improved by using C++.

Our earlier method (Rahman et al., 2011), which was an explicit method, extracted features from negative space. The explicit method achieved better accuracy (100% in case of Weizmann dataset) than the implicit method, but the explicit method's classification time is higher than that of the implicit method. (Fathi and Mori, 2008) and (Lin et al., 2009) also achieved 100% accuracy where both of them extracted features by using optical flow information, which is expensive and difficult to derive due to, for example, aperture problems, smooth surfaces and discontinuities. (Lin et al., 2009) method did not mention the feature extraction time or the classification time for their system but it is understandable from their system that the computation time would be high since they utilized the dynamic time warping algorithm which is computationally expensive (Sakurai et al., 2005). Hence, our proposed system is the fastest method which can achieve accuracy levels comparable with those of other state of the arts methods. For the fish action dataset, the recognition accuracy was 84.4%, which indicate that our method is also effective in recognizing the actions of non-human subjects. Confusion matrices for all the datasets are shown in Fig. 13. As with other methods, most of the confusion by the classifier are made in the similar actions, e.g., between 'jog' and 'run' in KTH dataset.

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy</th>
<th>Feature extraction time</th>
<th>Classification time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed method</td>
<td>95.56%</td>
<td>4.3s</td>
<td>1.1s</td>
</tr>
<tr>
<td>(Rahman et al., 2011)</td>
<td>100%</td>
<td>4.3s</td>
<td>32s</td>
</tr>
<tr>
<td>(Fathi and Mori, 2008)</td>
<td>100%</td>
<td>--</td>
<td>37.5s</td>
</tr>
<tr>
<td>(Lin et al., 2009)</td>
<td>100%</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>(Bregonzio et al., 2012)</td>
<td>96.6%</td>
<td>12.8s</td>
<td>0.35s</td>
</tr>
<tr>
<td>(Liu and Yuen, 2010)</td>
<td>98.3%</td>
<td>7.35s</td>
<td>624s</td>
</tr>
<tr>
<td>(Gorelick et al., 2007)</td>
<td>97.8%</td>
<td>30s</td>
<td>--</td>
</tr>
<tr>
<td>(Batra et al., 2008)</td>
<td>88.2%</td>
<td>6s</td>
<td>--</td>
</tr>
<tr>
<td>(Grundmann et al., 2008)</td>
<td>94.6%</td>
<td>--</td>
<td>28.4s</td>
</tr>
</tbody>
</table>

Table 2: Accuracy of different methods for Weizmann dataset

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed method</td>
<td>94.49%</td>
</tr>
<tr>
<td>(Rahman et al., 2011)</td>
<td>95.49%</td>
</tr>
<tr>
<td>(Lin et al., 2009)</td>
<td>93.43%</td>
</tr>
<tr>
<td>(Fathi and Mori, 2008)</td>
<td>90.5%</td>
</tr>
<tr>
<td>(Chua et al., 2011)</td>
<td>80.9%</td>
</tr>
<tr>
<td>(Bregonzio et al., 2012)</td>
<td>94.33%</td>
</tr>
<tr>
<td>(Grundmann et al., 2008)</td>
<td>93.52%</td>
</tr>
<tr>
<td>(Liu and Yuen, 2010)</td>
<td>81.5%</td>
</tr>
</tbody>
</table>

Table 3: Accuracy of different methods for KTH dataset
4.2 Robustness to noise

To evaluate our system for robustness to noisy segmentation, as was done in earlier research, we explicitly added ‘salt & pepper’ noise (Wang and Suter, 2007b), to the segmented images of Weizmann dataset to simulate noisy segmentation. The amount of added noise was controlled by a parameter called ‘noise density’. In this experiment, the bounding box was extracted only for the biggest blob from the corrupted image. Inside the bounding box, pixels of biggest blob were treated as foreground pixels, and other foreground pixels which were not connected with the biggest blob were removed. Hence, some parts of the human body may have been discarded in some testing image frames (Fig. 14), which may actually happen in the case of noisy segmentation. Some example frames are shown in Fig. 14. As training data, original (uncorrupted) images were employed. Comparisons of our methods with (Wang and Suter, 2007b) and (Rahman et al., 2012a) methods, which performed the same experiment on same dataset, are shown in Fig. 15. The accuracies of the negative space based methods (proposed and (Rahman et al., 2012a)) decrease slightly, as the high noise density increases, whereas the accuracy of the positive space method (Wang and Suter, 2007b) drastically decreases in the context of high noise density. It is evident from this experiment that our system is more robust to noisy segmentation.

Fig. 14: Silhouette images with different levels of noise. Input images are shown in the top row and the bottom row is the extracted bounding box image. In both rows, noise levels are 0.0, 0.1, 0.2, 0.3, 0.4 and 0.5 from left to right, respectively.
4.3 Contributions of different features

Contributions of different types of features in our system are shown in Fig. 16. For instance, without speed features, the accuracy is 90\%, whereas without orientation, it is 92.22\%. From this experiment, it is evident that all the features contribute equally in our system, since without one feature, the accuracy does not differ too much. By combining all the features we get the best result as indicated by the left most bar in Fig. 16.

4.4 Robustness to partial occlusion

In this experiment, robustness of the system was evaluated in cases of high irregularity of actions. In this regard, the \textit{Weizmann robust deformation} dataset was employed (Gorelick et al., 2007). This dataset contains ten video sequences (normal walking, walking in a skirt, carrying a briefcase, a limping man, occluded legs or no legs, knees up, walking with a dog, sleepwalking or
moonwalking, swinging a bag, occluded by a pole, of a person, walking in various complex scenarios in front of different non-uniform backgrounds.

![Image of various actions](image1)

Fig. 17: Examples of image frames used in the robustness experiment. The upper row shows the actual images and the lower row contains the segmented images for each frame. (a) Normal walk, (b) Swinging bag, (c) Carrying briefcase, (d) Walking with a dog, (e) Knees up (f) Limping man, (g) Moonwalking, (h) Occluded legs (i) Walking in a skirt and (j) Occluded by a pole.

Eight of the sequences were classified as walking (accuracy 80%). Knees up is classified as 'jump', and moonwalking is classified as 'run', whereas the walk action is the 2nd and 4th choice respectively for the two misclassified actions. Though not all of the deformed actions are classified correctly, our system can still be considered as robust to partial occlusion and non-rigid deformation, since most of the deformed actions were recognized correctly.

### 4.5 Recognizing action with lower frame rate

![Graphs showing recognition results](image2)

Fig. 18: Recognition result of different datasets for different frame rates. (a) Weizmann, (b) KTH and (c) Fish dataset

In this experiment, we recognized actions by employing fewer frames from an action by reducing the frame rate. Reducing the frame rate lowers the number of frames in an action cycle,
which will result in lowering the processing time, but it also loses some information about the action.

In this experiment, we examined how well our system could perform with less information and how much it improved the processing time. Fig. 18 shows the recognition result for different types of NSADs, and Table 4 shows recognition time for NSAD type 2. From the figure and table, we can see that our system can recognize action effectively, even in the lower frame rate of ten frames per second in real time, which is actually the frame rate for different surveillance cameras. This indicates that our proposed system can be applied in video surveillance to recognize actions.

**Table 4:** Time required (seconds) in different datasets for recognizing actions in the case of NSAD-type 2.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Frame rate</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10</td>
<td>15</td>
<td>20</td>
<td>25</td>
</tr>
<tr>
<td>Weizmann</td>
<td>2.5</td>
<td>3.2</td>
<td>3.9</td>
<td>4.5</td>
</tr>
<tr>
<td>KTH</td>
<td>3.97</td>
<td>4.87</td>
<td>5.67</td>
<td>5.4</td>
</tr>
<tr>
<td>Fish</td>
<td>1.62</td>
<td>1.92</td>
<td>2.12</td>
<td>2.4</td>
</tr>
</tbody>
</table>

## 5. Conclusion

In this article, we propose a negative-space shape based feature extraction method for action recognition. This method is called Negative Space Action Descriptors (NSAD). In this method, shape-based pose features are extracted from the surrounding regions, i.e., negative space, of subjects of interest for pose description. The average of the shape-based features of poses of one cycle length sequence, along with the speed information of the subjects, is then used as the feature vector for each action. We propose a technique to compute the cycle length of an action automatically without any prior knowledge of the action. The recognition accuracy of our system is comparable with state of the art methods. The system was found to be robust to noisy segmentation, partial occlusion and non-rigid deformation of actions. The feature extraction method is computationally efficient, being ten times faster than our previous method and 25 times faster than other methods. Moreover, the feature vector dimension is also small. Hence, our method does not require expensive feature selection and reduction steps. Furthermore, our system can recognize actions effectively in real time from the videos that are captured in lower frame rate, such as videos captured by surveillance camera, which lose much
information about the action. Therefore, our method enables action recognition tasks to be performed online, on mobile applications or on video surveillance.

The test results for the fish-action data are particularly significant. The results show that action recognition systems can be extended to provide behavioral information and real-time detection of visual behavior cues of marine species. The proposed method enables the acquisition of behavioral data at levels of speed and accuracy that far exceed the capabilities of current methodologies, giving researchers unprecedented sampling power in the field and laboratory. In these critical times of global change, detailed knowledge of ecological systems and their requisite species is more important than ever. Currently, productivity in the fields of ecosystem function and biodiversity is limited principally by the rate at which we capture and manage morphological and ecological information. Development of computational image analysis and feature extraction methods is a crucial missing capacity needed to enable scientists and environmental managers to overcome this bottleneck in a meaningful time frame.

In future work, we hope to develop our own segmentation algorithm. We will also explore the use of different distance metrics and classifiers in our system.

Acknowledgements

The project was funded by JCU Singapore research grant JCUS004/2010/IS. We would like to thank Stephen Walker for providing the video footage of butterfly fishes.

References:


