Fuzzy points: algebra and application

R.E. Mercer[a,1], J.L. Barron[a,*,1], A.A. Bruen[b], D. Cheng[a,1]

[a]*Department of Computer Science, The University of Western Ontario, London, Ontario, Canada N6A 5B7*
[b]*Theoretical Biology and Biophysics Group, T–10, Los Alamos National Laboratory, Los Alamos, NM 87545, USA*

Abstract

A fuzzy point is a region representing the uncertain location of a normal Euclidean point. A fuzzy point in the plane is considered to be a closed disk (a circle and its interior). The algebra of fuzzy points (which includes fuzzy vectors and fuzzy angles) is presented. Since fuzzy points are represented as closed disks, the lengths of fuzzy vectors, and the angles between fuzzy vectors can be viewed as properties of circles in the plane. Methods to compute the magnitude of a fuzzy angle are given. An application of fuzzy point algebra to the problem of detecting and tracking storms in Doppler radar image sequences, which motivates this work, is discussed. © 2002 Pattern Recognition Society. Published by Elsevier Science Ltd. All rights reserved.

Keywords: Fuzzy storms; Fuzzy vectors; Fuzzy angles; Fuzzy algebra; Doppler radar imagery; Tracking deformable objects

1. Introduction

Various problems have necessitated the need for a rigorous treatment of objects or concepts that do not have precise realizations. One such problem is the tracking of non-rigid objects that can change shape between successive images. Many tracking algorithms [1–3] make the assumption that the tracked objects are rigid. Two possible modifications to the current rigid object tracking methodology are apparent: create algorithms to deal with the non-rigidity of the tracked objects [4]; or find a modified representation of the objects and use the original (essentially unchanged) rigid object tracking algorithms with this new representation. We have done the latter for a tracking algorithm developed in our storm tracking project [5,6].

Rigid object tracking algorithms represent an object as a set of points designating significant features on the object. The algorithms perform the tracking of rigid bodies by tracking the set of points. The points are traditional Euclidean points. We have developed a new representational concept, the *fuzzy point*,[2] that can be used to represent non-rigid objects. By representing non-rigid objects as fuzzy points, simple variants of Euclidean point tracking algorithms potentially can be used to track non-rigid objects by tracking the fuzzy points.

---

* Corresponding author. Tel.: +1-519-661–2111 Ext 86896; fax: +1-519-661-3515.

E-mail address: barron@csd.uwo.ca (J.L. Barron).

---

[2] Although the term "fuzzy point" contains the word "fuzzy", there is no relationship between our sense of the word and the sense that it is used in "fuzzy sets" or "fuzzy logic". Using the terminology of fuzzy sets, a *fuzzy subset* [7] of a set S is a mapping μ from S into the real interval $[0,1]$. Chaudhuri [8] gives a definition of a fuzzy point which is classically fuzzy: a fuzzy set with non-zero membership only at one point of the support space. Importantly, this membership function is not uniformly 0 or 1 and the support space is $\mathbb{R}^2$. If we were to use the terminology of classical fuzzy sets, the membership function μ maps each point in the region representing our fuzzy point to 1. Thus, our fuzzy points are "crisp".

A fuzzy point is a region (the closure of an open, simply connected set of points) representing the uncertain location of a Euclidean point. In the work described here fuzzy points in the plane are realized as circles and their interiors. We present the algebra of fuzzy points, which includes fuzzy vectors and fuzzy angles, using properties of plane geometry. The lengths of fuzzy vectors, and the magnitudes of angles formed by fuzzy vectors are viewed as properties of circles. We discuss these properties and provide an $O(n)$ method to compute the magnitude of fuzzy angles in a discrete space. We finish with a short presentation of a tracking algorithm [9,10] that tracks fuzzy points, focussing on those aspects of the algorithm that use the fuzzy point algebra.

1.1. Motivation

Motivation for fuzzy points has resulted from the inability of traditional point tracking methods [5,6] to deal with two factors when tracking storms, meteorological phenomena observed in Doppler radar images. First, storms in radar images are deformable 2D objects. Second, the change in shape of storms between successive images can be quite dramatic. The data that we use are gathered at an operational radar installation (Atmospheric Environment Service, King City, Ontario, Canada). Because the data are sampled only every 10 min due to the operational nature of the radar, it may be aliased due to the dynamic nature of storms. Because the objects being tracked can change shape quite dramatically between images, the standard deformable object tracking algorithms cannot be used, since these algorithms tolerate only small amounts of shape change between images.

In our initial attempts to track storms a storm is represented as the center of mass of the storm (a point). This representation transforms the storm tracking problem into a point tracking problem. Unfortunately, some of the features of storm tracking do not transfer to the point tracking problem. What appears as reasonably smooth object motion to a human observer in successive storm domain images can appear as more arbitrary point motion in the point domain and in some cases the points move in unacceptable ways. For instance, a storm can quite easily be seen to be moving in one direction but the point that represents it can be seen to be moving in a radically different direction (sometimes opposite to the direction of the storm).

Traditional point tracking algorithms assume somewhat smooth transitions from one image to the next in a sequence of images. These algorithms do not produce tracks for objects which do not have smooth movement. The notion of smoothness cannot be absolute, so some arbitrariness in direction and speed must be allowed. In the methods used in our storm tracking project [5,6] the arbitrariness in direction and speed is captured as thresholds. The performance of these algorithms are sensitive to the choice of thresholds, three of which are important to this discussion: (1) the threshold to determine the boundary between an object and the background (this threshold effects the calculation of the center of mass of the object, the number of objects being tracked, and the size of the objects being tracked); (2) the threshold to determine acceptable distances between objects in sequential images; and (3) the threshold to determine acceptable angles formed by the object's direction in three successive images. To produce tracks for centers of mass that demonstrate movement which would not be considered smooth, such as that described above, but which represent storms with smooth movement, the three threshold values need to be set at unacceptable levels. Boundary thresholds set to make more precise the location of the center of mass produce more objects to track than what one observes in the data. Large angle and distance thresholds, which are needed in order to make the algorithm generate storm tracks which are obvious to a human observer, allow invalid tracks, as well.

Using the centers of mass of the objects as the points to track has proven difficult because the arbitrariness in the object shape translates into an arbitrariness in the location of the center of mass, which acts against the assumptions of the tracking algorithm. In order to meet these assumptions, the arbitrariness in the location of the center of mass can be hidden from the algorithm by instead representing the centers of mass as fuzzy points, the representational concept developed in this paper, and having the algorithm track fuzzy points.

1.2. Literature survey

One of the basic tasks in computer vision is to determine the correspondence between the representations of the same object in different images. This process is commonly referred to as the *correspondence problem*. Moreover, most tracking techniques assume some reasonable *smoothness of motion*, that is, the motion of an object cannot change abruptly. As well, most correspondence algorithms track Euclidean image points, which usually denote some type of robust image feature that exists over two or more images. A few algorithms track other types of features, such as lines, corners, edgels, grayvalue regions and texture elements [11] and, in our case, fuzzy points.

Barnard and Thompson [1] used relaxation labelling to determine the correspondence between a set of feature points selected from a pair of stereo images. Disparities, represented as vectors, are constructed from the feature points in one image to the corresponding feature points in the second image. A "no match" disparity allows for the possibility that a point has no match. Based on a local smoothness assumption for disparities, the probability of each hypothesized disparity at each feature point

is iteratively refined by a relaxation labelling algorithm that takes into account supporting evidence from neighboring similar disparities. The refinement is stopped once a preset level of convergence is reached. For each feature point in one image, the disparity with the highest certainty is selected as the best match. Dreschler and Nagel [12] modified Barnard and Thompson algorithm to explicitly take into account both supporting and contradicting neighboring disparities in their relaxation.

Sethi and Jain [3] used a path coherence technique to obtain a trajectory of objects represented by individual points. The motion of the object is assumed to be smooth. They utilized a sequence of frames, instead of just two frames at a time, to determine correspondence. Trajectories are initially set up using the nearest neighbor match and the matches are iteratively refined using a greedy exchange algorithm which maximizes the path coherence.

One of the difficulties with the use of multiple frames to establish correspondence is the problem of occlusion. Salari and Sethi [2] proposed a solution using phantom feature points. Incomplete tracks are padded out with these points which are basically used as a computational convenience in a modified greedy exchange algorithm.

The path coherence technique was later generalized to property coherence by Sethi et al. [13]. They applied the property coherence method to track line tokens over multiple frames. They defined the property space for line tokens as a five-dimensional space, consisting of midpoint coordinates of the line token, line length, line orientation, and the directed distance of the line. Correspondence between line tokens was then based on these five attributes. Krezeski et al. [5] uses a temporal relaxation algorithm with property coherence to track storms in a Doppler radar image sequence. Property coherence allows other properties such as storm size and storm shape, in addition to the path of the storm, to be tracked over time.

Hwang [14] uses a heuristic search to maximize smoothness of computed tracks. Further analysis is possible on the most plausible tracks found. Rangarajan and Shah [15] assume that the initial correspondences are given. They use a non-iterative algorithm to maximize *proximal uniformity* of tracks, i.e. it penalizes large accelerations and displacements. The IPAN tracker [16] finds correspondence by a competitive linking process. It detects occlusion events (points that disappear or reappear) as any point that does not link to the previous or next frame in a sequence. Verestóy and Chetverikov [17] present some preliminary comparison results for five of the tracking algorithms [2,3,14–16] discussed above using synthesized point sets.

Lai [18] introduced an algorithm for tracking multiple features in an image sequence. The method uses relaxation labelling to deduce the disparity of each possible match. A constraint-aided exchange scheme was proposed to recover wrong correspondences originating from the occlusion of feature points.

Leymarie and Levine [4] proposed a method for tracking deformable objects, such as living cells. They used an active contour model, commonly called a *snake* model, to represent the deformable shape. A snake, introduced by Kass et al. [19], is an energy-minimizing spline guided by external constraint forces and influenced by image forces that pull it towards features such as lines, edges and subjective contours. User interaction is permitted to initialize the first frame of the tracking sequence. His algorithm provides satisfactory results when the deformation and movement of cells are small in consecutive frames. The storms in our Doppler image sequences do not satisfy these assumptions.

Bjerkaas and Forsyth [20] developed a simple storm tracking system to assist meteorologists. The system used a nearest neighbor algorithm to match storm centers in adjacent images. It achieved a forecast error rate lower than the mean rate achieved in regular operations. Hodges [21] has shown a practical use of the dynamic scene analysis techniques developed by Sethi et al. [2,3] on data generated by general circulation models. His system achieved good results when tracking meteorological activities.

Most of our previous work in automatic storm tracking can be found in two Masters' theses [6,22] and four published papers [5,9,10,23]. Here we highlight a few features of the work that form the basis of our current work.

Based on a *merge and split* algorithm developed by Horowitz and Pavlidis [24], Zhang [6] devised a similar region growing algorithm to identify hypothesized storm masses in a sequence of Doppler radar intensity images. By investigating Barnard and Thompson's spatial relaxation labelling algorithm [1], Zhang developed a temporal relaxation algorithm to track storm centers represented by normal Euclidean points over time. The algorithm is based on the *smoothness assumption* of storm movement. Initially, disparities between the storm centers in adjacent images are constructed and each disparity is associated with a certainty value. The certainty value is then modified by relaxation on distance and angle compatibility, which is an iterative process. In the end, disparities with the highest certainty values are chosen as the best matches. To handle the merging and splitting of storms, Zhang's algorithm allows storms to be matched to several storms in the adjacent images. This approach was not too successful because her tracking algorithm sometimes matched pre-merged storms to a storm that was not the merging storm.

Krezeski et al. [5] has added *property coherence* and *pseudo-storms* to improve Zhang's tracking algorithm. Property coherence [13] allows multiple features of a storm to be tracked over time in addition to the location of the storm center. Five additional storm properties that he selected were *average intensity, storm size, velocity variance, storm shape and orientation*, and *convexity*. Zhang's storm hypothesis method was used. Again

relaxation labelling is used, except now the probability of each disparity at each iteration is determined by how compatible its storm properties are with neighboring storm disparities. Krezeski used the concept of pseudo-storms as proposed by Einfalt et al. [25]. They suggested that storms which are close together can be considered as a single entity for comparison with other storms in adjacent images. While Krezeski's work produced promising experimental results the algorithm proved to be threshold sensitive because incorrect determination of a storm's center led to incorrect storm matches.

The set of fuzzy points, $\mathbb{P}$, closely resembles the set of *circular disks* (or *circular intervals*) as defined in Ref. [26]. We write the set of circular disks as $K(\mathbb{C}) = \{\langle c, r \rangle \mid c \in \mathbb{C}, r \in \mathbb{R}\}$, where $\mathbb{C}$ denotes the set of all complex numbers. The set $\mathbb{P}$ differs from the set $K(\mathbb{C})$ in representing the center of its elements (i.e. circles) as a Euclidean point (x, y) in favor of a complex number $x + yi$. Many properties of $K(\mathbb{C})$ are given in Ref. [26]. Those properties should apply equally to the set $\mathbb{P}$, since there exists the obvious isomorphism between the two sets $f : \mathbb{P} \to K(\mathbb{C})$ as defined by $\langle (x, y), r \rangle \overset{f}{\mapsto} \langle x + yi, r \rangle$.

Although we have stated that our contribution is not to be considered as a classical fuzzy set, it is appropriate to note that classical fuzzy set approaches to geometry have been studied. Fuzzy sets [7] are defined in the following way: a *fuzzy subset* of a set S is a mapping μ from S into the real interval $[0, 1]$, where μ is called the *membership* function. By considering S to be the Euclidean plane, $\mathbb{R}^2$, many of the elementary geometric properties can be generalized to fuzzy subsets [8,27–29].

2. Fuzzy point algebra

2.1. Definitions

Definition 1. A *fuzzy point* is a circular region representing the uncertain location of a normal Euclidean point. No underlying probability distribution representing this uncertainty is assumed. A fuzzy point is denoted as $P = \langle c, r \rangle$, a circle (and its interior) with center $c = (x, y) \in \mathbb{R}^2$ and radius $r \in \mathbb{R}$. The set of points representing a fuzzy point $P = \langle c, r \rangle$ is the subset of the Euclidean plane $\mathbb{R}^2$:

$$P = \langle c, r \rangle = \{p \in \mathbb{R}^2 \mid ||p - c|| \leqslant r\}. \tag{1}$$

Throughout this paper we will use upper-case letters (e.g. P, Q) to represent fuzzy points and lower-case letters (e.g. p, q) to represent normal Euclidean points. We denote the set of all fuzzy points as

$$\mathbb{P} = \{\langle c, r \rangle \mid c \in \mathbb{R}^2, r \in \mathbb{R}\} \tag{2}$$
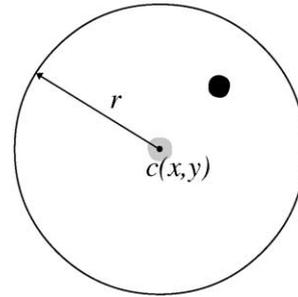

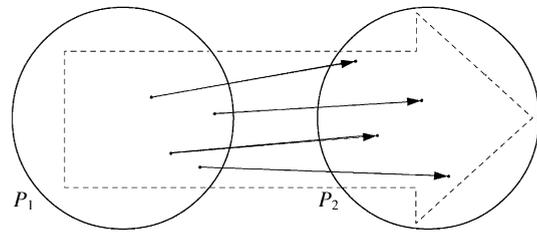Fig. 1. A fuzzy point $\langle c, r \rangle$ represents the uncertain location of a Euclidean point.

Fig. 2. A fuzzy vector $\overrightarrow{P_1 P_2}$ from fuzzy point P_1 to fuzzy point P_2.

The actual location of the targeted point can be anywhere inside or on the circle and not necessarily at its center, as shown in Fig. 1. Two fuzzy points $\langle c_1, r_1 \rangle$ and $\langle c_2, r_2 \rangle$ are equal if and only if $c_1 = c_2$ and $r_1 = r_2$. Note that if $r = 0$ then a fuzzy point $\langle c, 0 \rangle$ will shrink into a single Euclidean point c. The magnitude of r indicates the relative uncertainty of where the actual (Euclidean) point is located.

Definition 2. A *fuzzy vector* from a fuzzy point, P_1, to a fuzzy point, P_2, is the set of all Euclidean vectors from the Euclidean points in the region defined by P_1 to the Euclidean points in the region defined by P_2. A fuzzy vector from P_1 to P_2 is denoted $\overrightarrow{P_1 P_2}$.

Consider two fuzzy points P_1 and $P_2 \in \mathbb{P}$ as shown in Fig. 2. Since a fuzzy point is merely a set of points in the plane, we can construct a displacement vector from any point in P_1 to any point in P_2. We call the infinite set of all such vectors in $\mathbb{R}^2$ the fuzzy vector from P_1 to P_2 and write it as:

$$\overrightarrow{P_1 P_2} = \{\overrightarrow{p_1 p_2} \mid p_1 \in P_1 \text{ and } p_2 \in P_2\}. \tag{3}$$

Definition 3. The *length of a fuzzy vector* $\overrightarrow{P_1 P_2}$ is the set of lengths of all vectors in $\overrightarrow{P_1 P_2}$ and is denoted as $||\overrightarrow{P_1 P_2}||$.
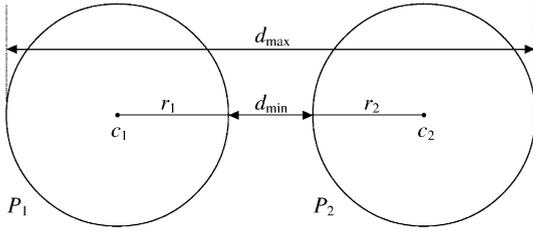
Fig. 3. The length of $\overrightarrow{P_1P_2}$ is the real interval $[d_{\min}, d_{\max}]$.

Fig. 4. The fuzzy angle $\langle \overrightarrow{P_1P_2}, \overrightarrow{P_2P_3} \rangle_\theta$ formed by two non-zero fuzzy vectors.

Let us consider the fuzzy vector $\overrightarrow{P_1P_2}$ connecting the fuzzy points $P_1 = \langle c_1, r_1 \rangle$ and $P_2 = \langle c_2, r_2 \rangle$. Then the length of this fuzzy vector is denoted as

$$||\overrightarrow{P_1P_2}|| = \{||\overrightarrow{p_1p_2}|| \mid \overrightarrow{p_1p_2} \in \overrightarrow{P_1P_2}\}. \tag{4}$$

The set $||\overrightarrow{P_1P_2}||$ can be represented by the real interval $[d_{\min}, d_{\max}]$, where $d_{\min}$ corresponds to the *infimum* of the set, $\inf ||\overrightarrow{P_1P_2}||$; and $d_{\max}$ corresponds to the *supremum* of the set, $\sup ||\overrightarrow{P_1P_2}||$.

From plane geometry, it can be easily verified that $d_{\min}$ and $d_{\max}$ are, respectively, the closest and furthest distance between any two points on the circles defining the fuzzy points P_1 and P_2 (refer to Fig. 3) unless P_1 intersects P_2. In this case, $d_{\min}$ is obviously 0. We consider any fuzzy vector with length of the form $[0, d_{\max}]$ a *zero fuzzy vector*. Note that a zero fuzzy vector is not unique. Note also that the value of $d_{\max}$ is always non-negative and becomes zero if and only if both the fuzzy points degenerate to the same Euclidean point. $d_{\min}$ and $d_{\max}$ can be expressed in terms of c_1, c_2, r_1, and r_2 as

$$d_{\min} = \max\{0, ||c_2 - c_1|| - (r_1 + r_2)\} \tag{5}$$

and

$$d_{\max} = ||c_2 - c_1|| + (r_1 + r_2). \tag{6}$$

The length of a fuzzy vector belongs to $I(\mathbb{R})$, the set of all real intervals. Properties of $I(\mathbb{R})$ such as *interval arithmetic* have been well studied [26,30,31]. Comparing the lengths of two fuzzy vectors is not as trivial as comparing the lengths of two vectors. Since the length of a fuzzy vector is an interval, we can only define a partial ordering on all lengths by

$$[a,b] < [c,d] \quad \text{if and only if } b < c. \tag{7}$$

However, if we so choose (knowing that the comparison may not have a valid mathematical basis), the length of a fuzzy vector has a representation that allows other comparison methods, such as comparing the values of $d_{\min}$, the values of $d_{\max}$, or the values of $(d_{\min} + d_{\max})/2$.

Definition 4. The *fuzzy angle* formed by a non-zero fuzzy vector $\overrightarrow{P_iP_j}$ relative to another non-zero fuzzy
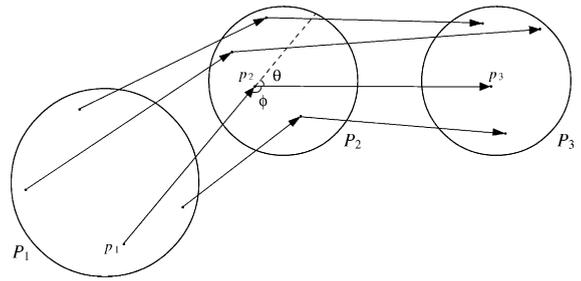
vector $\overrightarrow{P_jP_k}$ is the set of angles formed by all displacement vectors $\overrightarrow{p_{i_l}p_{j_m}} \in \overrightarrow{P_iP_j}$ relative to all displacement vectors $\overrightarrow{p_{j_m}p_{k_n}} \in \overrightarrow{P_jP_k}$. The fuzzy angle formed by the two fuzzy vectors $\overrightarrow{P_iP_j}$ and $\overrightarrow{P_jP_k}$ is denoted as $\langle \overrightarrow{P_iP_j}, \overrightarrow{P_jP_k} \rangle_\theta$.

Let us consider two non-zero fuzzy vectors $\overrightarrow{P_1P_2}$ and $\overrightarrow{P_2P_3}$ connecting three fuzzy points $P_1 = \langle c_1, r_1 \rangle$, $P_2 = \langle c_2, r_2 \rangle$ and $P_3 = \langle c_3, r_3 \rangle$, as shown in Fig. 4. For any point p_2 in P_2, we can find a point p_1 in P_1 and another point p_3 in P_3 to form a pair of displacement vectors $\overrightarrow{p_1p_2}$ and $\overrightarrow{p_2p_3}$. Clearly, $\overrightarrow{p_1p_2}$ is in $\overrightarrow{P_1P_2}$ and $\overrightarrow{p_2p_3}$ is in $\overrightarrow{P_2P_3}$. Now the angle θ formed by $\overrightarrow{p_2p_3}$ relative to $\overrightarrow{p_1p_2}$ can be conveniently determined by the dot-product:

$$\cos\theta = \frac{\overrightarrow{p_1p_2} \cdot \overrightarrow{p_2p_3}}{||\overrightarrow{p_1p_2}|| \, ||\overrightarrow{p_2p_3}||}. \tag{8}$$

The angle θ is a measure of the change in the direction of the vector $\overrightarrow{p_2p_3}$ relative to the vector $\overrightarrow{p_1p_2}$. By convention, counter-clockwise angles are considered positive. We are only interested in the magnitude of the angle. Therefore, we only consider θ that lies in the interval $[0°, 180°]$ or $[0, \pi]$ in radians. The set of all such angles is taken as $\langle \overrightarrow{P_1P_2}, \overrightarrow{P_2P_3} \rangle_\theta$, the fuzzy angle formed by the fuzzy vector $\overrightarrow{P_2P_3}$ relative to the fuzzy vector $\overrightarrow{P_1P_2}$.

Definition 5. The *magnitude of a fuzzy angle* $\langle \overrightarrow{P_1P_2}, \overrightarrow{P_2P_3} \rangle_\theta$ is the set of magnitudes of all angles in $\langle \overrightarrow{P_iP_j}, \overrightarrow{P_jP_k} \rangle_\theta$.

Like the length of a fuzzy vector, the magnitude of a fuzzy angle can also be considered as a real interval $[\theta_{\min}, \theta_{\max}] \in I(\mathbb{R})$, where $\theta_{\min} = \inf \langle \overrightarrow{P_1P_2}, \overrightarrow{P_2P_3} \rangle_\theta$ and $\theta_{\max} = \sup \langle \overrightarrow{P_1P_2}, \overrightarrow{P_2P_3} \rangle_\theta$ (refer to Fig. 5). Unlike computing the length of a fuzzy vector, computing the magnitude of a fuzzy angle is a problem of sufficiently interesting content that we devote the next section to a discussion of this problem.

2.2. Computing the magnitude of a fuzzy angle

Determining the two angles $\theta_{\min}$ and $\theta_{\max}$, the two endpoints of the interval representing the magnitude of the fuzzy angle formed by any two non-zero fuzzy vectors, is equivalent to finding the maximum and minimum angles in three non-intersecting circles. Finding a closed form solution expressing $\theta_{\min}$ and $\theta_{\max}$ explicitly in terms of c_1, c_2, c_3, r_1, r_2, and r_3 is a currently unsolved problem in plane geometry to which we continue to seek a solution. Given that the data in our application has a pixel level resolution, we have modified the angle computation problem by discretizing the plane, essentially working at the pixel level in the image. This modified version still provides sufficiently accurate results for our storm tracking algorithm. With the plane discretized, we use an $O(n)$ search algorithm to approximate the two angles. Details of this search algorithm and related work are developed below. We begin with the brute force $O(m^3)$ algorithm (m is the number of pixels in the largest of the three fuzzy points defining the fuzzy angle) and then develop some properties of the minimum and maximum angles to give an $O(n)$ algorithm (n is the number of pixels on the circle defining the largest fuzzy point).

2.2.1. An $O(m^3)$ algorithm

Because fuzzy points are finite sets of pixels in the discretized version of the problem, and since one representative Euclidean point can be chosen in each pixel, a generate and test approach can approximate the fuzzy angle. An obvious algorithm to compute the fuzzy angle (the minimum and maximum angle interval) is via the direct interpretation of the definition of a fuzzy angle.

Let $\overrightarrow{P_1 P_2}$ and $\overrightarrow{P_2 P_3}$ be two non-zero fuzzy vectors connecting three fuzzy points P_1, P_2 and P_3 in $\mathbb{P}$ as shown in Fig. 4. For each 3-tuple (p_1, p_2, p_3), $p_i \in P_i$, we compute the angle $\theta_{p_1 p_2 p_3}$ formed by the displacement vector $\overrightarrow{p_2 p_3}$ relative to the displacement vector $\overrightarrow{p_1 p_2}$ by the dot-product, denoted by the symbol $\odot$:

$$\theta_{p_1 p_2 p_3} = \odot(p_1, p_2, p_3)$$
$$= \arccos\left(\frac{\overrightarrow{p_1 p_2} \cdot \overrightarrow{p_2 p_3}}{||\overrightarrow{p_1 p_2}|| \, ||\overrightarrow{p_2 p_3}||}\right) \qquad (9)$$

choosing the minimum $\theta_{p_1 p_2 p_3}$ for $\theta_{\min}$ and the maximum for $\theta_{\max}$. Clearly, the algorithm is $O(m^3)$, where m is the number of pixels in the largest circle.

2.2.2. Preliminaries for an $O(n)$ algorithm

To reduce our $O(m^3)$ search algorithm (where m is the number of pixels in the largest fuzzy point) to an $O(n)$ search algorithm (where n is the number of pixels on the circle representing the boundary of the largest fuzzy point), we use some properties of circles, some of which are already known and two of which are stated as
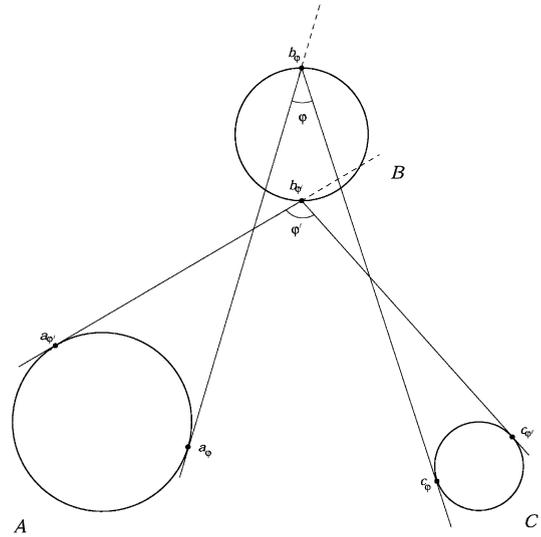

Fig. 5. The magnitude of $\langle \overrightarrow{AB}, \overrightarrow{BC} \rangle_\theta$ is the real interval $[180 - \varphi', 180 - \varphi]$.

Theorems 6 and 7 below. We present the proofs of these two theorems in Appendix A.

In Fig. 4, the complementary angles θ and ϕ (i.e. $\theta + \phi = 180°$) are shown. We use the angle ϕ in the development of the search algorithm below. Let A, B, and C be three non-intersecting circles in the plane. Further, let a_1, b_1, and c_1 be, respectively, a point on each of the three circles or in their interiors and let $\phi = \angle a_1 b_1 c_1$ denote the angle between the two line segments $\overline{a_1 b_1}$ and $\overline{b_1 c_1}$. We write the set of all such angles as

$$S_\phi = \{ \angle a_1 b_1 c_1 \, | \, a_1 \in A, b_1 \in B, \text{ and } c_1 \in C \}. \qquad (10)$$

Theorem 6. *Let $A, B,$ and C be three non-intersecting circles in the plane. Suppose that $\varphi = \min(S_\phi)$. Then there exist three points $a_\varphi \in A$, $b_\varphi \in B$ and $c_\varphi \in C$, such that $\varphi = \angle a_\varphi b_\varphi c_\varphi$.*

Theorem 7. *Let $A, B,$ and C be three non-intersecting circles in the plane. Suppose that $\varphi' = \max(S_\phi)$. Then there exist three points $a_{\varphi'} \in A$, $b_{\varphi'} \in B$ and $c_{\varphi'} \in C$, such that $\varphi' = \angle a_{\varphi'} b_{\varphi'} c_{\varphi'}$.*

Theorems 6 and 7 allow a dramatic reduction in the number of points taken from each fuzzy point P_j by merely considering points that lie on the boundary of each fuzzy point. Another benefit of these results is that we can possibly obtain a more accurate approximation of $\theta_{\min}$ and $\theta_{\max}$. The points used in the calculations are arbitrarily located in the discrete pixels. Now the points can be chosen less arbitrarily: they can be chosen to lie on the true boundary of each fuzzy point. Although $n < m$, the algorithm is still $O(n^3)$.
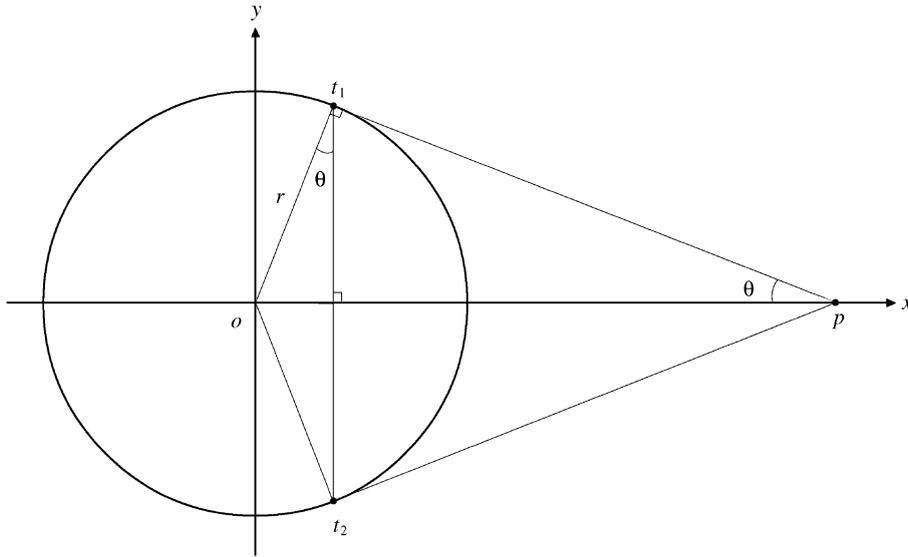
Fig. 6. The home position setup.

2.2.3. An O(n) algorithm

One further improvement to this algorithm is possible. The construction given in the proof of Theorem 6 indicates that points a_2 and c_2 are tangent points, that is, the line segments $\overline{a_2 b_1}$ and $\overline{b_1 c_2}$ are tangent to the circles A and C, respectively. By generating points p_2 on the boundary of the second fuzzy point P_2, we no longer need to generate points from the fuzzy points P_1 and P_3 because the corresponding tangent points on the circles P_1 and P_3 to the point p_2 can be computed using a closed form method which is discussed below. The resulting $O(n)$ algorithm to approximate $\theta_{\min}$ and $\theta_{\max}$, the minimum and maximum angles formed by all displacement vectors from circle P_1 to circle P_2 and all displacement vectors from circle P_2 to circle P_3 whose heads and tails touch in P_2 is presented below. If n points are generated on the boundary of the fuzzy point P_2, then the algorithm will perform $2n$ tangent point computations and $4n$ dot-product operations to determine the values of $\theta_{\min}$ and $\theta_{\max}$.

Initialize $\theta_{\min}$ and $\theta_{\max}$;
For any point p_2 on the boundary of P_2 **do**
      Compute tangent points p_1 and p_1' on P_1 to p_2;
      Compute tangent points p_3 and p_3' on P_3 to p_2;
      Compute $\theta_1 = \odot (p_1, p_2, p_3)$;
      Compute $\theta_2 = \odot (p_1', p_2, p_3)$;
      Compute $\theta_3 = \odot (p_1, p_2, p_3')$;
      Compute $\theta_4 = \odot (p_1', p_2, p_3')$;
      Update $\theta_{\min}$ and $\theta_{\max}$ if necessary;
Endfor

2.2.4. Determination of tangent points

Consider the setup we call the *home position* shown in Fig. 6 where the fuzzy point is $\langle c, r \rangle$, where $c = (0, 0)$ and the exterior point is at $p = (x_0, 0)$, where $x_0 > r > 0$. We can compute the two tangent points for this setup as

$$t_1 = \begin{bmatrix} r \sin \theta \\ r \cos \theta \end{bmatrix} \quad \text{and} \quad t_2 = \begin{bmatrix} r \sin \theta \\ -r \cos \theta \end{bmatrix}, \tag{11}$$

where $\sin \theta$ and $\cos \theta$ can be evaluated by

$$\sin \theta = \frac{r}{x_0} \quad \text{and} \quad \cos \theta = \frac{\sqrt{x_0^2 - r^2}}{x_0}. \tag{12}$$

In the general position setup, shown in Fig. 7, we denote the exterior point as $p = (a, b)$ and the circle center as $c = (e, f)$ with radius r. We compute α, the angle of $\overline{cp}$ with the positive x-axis as

$$\sin \alpha = \frac{(b - f)}{\sqrt{(a - e)^2 + (b - f)^2}}, \tag{13}$$

$$\cos \alpha = \frac{(a - e)}{\sqrt{(a - e)^2 + (b - f)^2}}. \tag{14}$$

Now, we translate/rotate the general position setup (translate by $(-e, -f)$ and rotate by $-\alpha$) to get to the home position setup. We solve for t_1 and t_2 and rotate/translate these points back to the general position setup (rotate by α and translate by (e, f)).
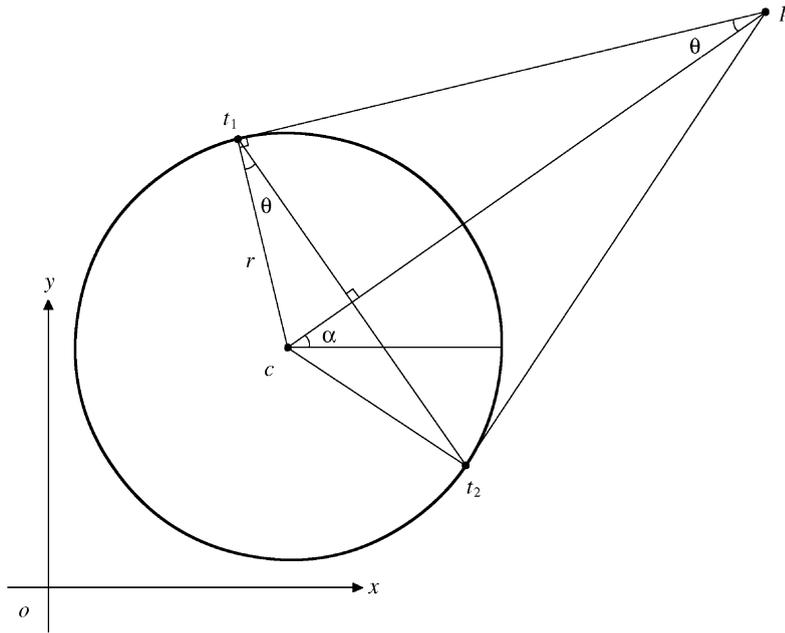
Fig. 7. The tangent from an external point setup.

3. Employing the fuzzy point algebra in a tracking algorithm

The motivation for fuzzy point algebra is to be able to use traditional (Euclidean) point tracking algorithms in situations which do not permit the use of precisely determinable Euclidean points. In our early storm tracking work [5] we developed a temporal relaxation algorithm that tracked storms which were represented as Euclidean points. In theory the algorithm can track points along curves that follow some reasonable assumptions (for instance, the interimage distances that the point moves in an image sequence do not differ by more than a distance threshold, and the angles that are generated by the points in each three-image sequence do not differ by more than an angle threshold). Our attempts to apply the temporal relaxation algorithm on real data were frustrated. In our early storm tracking work a gray-value region that designates a storm in each radar image is represented as a point in each image, the point being the center of mass of the gray-value region. These are the points that are tracked by the temporal relaxation algorithm. The regions designating storms are deformable from image to image. What we discovered was that most deformations permitted the expected tracking, but some moved the centers of mass in ways that gave interimage distances or angles above the respective thresholds, defeating the expected tracking. Raising the thresholds caused the generation of invalid tracks. In most cases small perturbations of the representing point away from the calculated center of

mass once again permitted the expected tracking. With this observation we decided that the arbitrariness of the location of the center of mass should be an aspect of the storm representation and that the algorithm could remain essentially untouched.

In Refs. [9,10] we describe in detail the basic incremental relaxation algorithm used to track Euclidean points (based on Krezeski's temporal relaxation algorithm with property coherence: length of vector, magnitude of angle, and storm size), its implementation, and the results obtained tracking storms in a number of radar data sets. To provide some idea of how the basic point tracking algorithm is changed to utilize the fuzzy point algebra, we highlight below those aspects of the tracking algorithm which use the fuzzy point algebra.

3.1. Locating the hypothesized storms in an image

Radar images are gray-value images representing the reflectivity of water particles in the air. The denser the water particles, the higher the reflectivity. A region of high reflectivity over a sufficiently large area over a reasonable number of images represents a storm. So, a region of high reflectivity in a single image represents an hypothetical storm. Hypothetical storms are generated using a region growing algorithm based on Horowitz and Pavlidis' algorithm [24].

For each set of connected storm image pixels in an hypothesized storm, a weighted average (center of mass) of the x and y storm pixel coordinates (the weights are

the intensities of the storm pixels) is calculated to obtain the center, $c = (\bar{x}, \bar{y})$, of the fuzzy point. (In the basic algorithm this would have been the Euclidean point used to represent the hypothesized storm.) In addition, r, the fuzzy point's radius, is computed,

$$r = k_r \max(s_x, s_y), \tag{15}$$

where, s_x and s_y are the standard deviations of $\bar{x}$ and $\bar{y}$ and k_r is a parameter used to dampen the radius of the fuzzy point (we have used $k_r = 0.5$ in our reported work).

3.2. The construction of disparities

Let S_j and S_{j+1} be hypothesized storms in Doppler images j and $j + 1$. A disparity, represented as a fuzzy vector $\overrightarrow{S_j S_{j+1}}$, is constructed if the infimum of the length of the fuzzy vector [3] is less than the threshold, T_d (with default value 10) and S_j and S_{j+1} have compatible sizes:

$$d_{\min} = \inf ||\overrightarrow{S_j S_{j+1}}|| < T_d \quad \text{and} \quad f_s(S_j, S_{j+1}) > T_{sc}. \tag{16}$$

f_s is a size-compatibility function between two fuzzy storms, $S_1 = \langle c_1, r_1 \rangle$ and $S_2 = \langle c_2, r_2 \rangle$ given in terms of their radii:

$$f_s(S_1, S_2) = \begin{cases} 1 - \frac{|r_1 - r_2|}{\max(r_1, r_2)} & \text{if } r_1 > 0 \text{ and } r_2 > 0, \\ 1 & \text{otherwise.} \end{cases} \tag{17}$$

f_s values are in the interval $[0, 1]$; 0 means no size compatibility while 1 means perfect compatibility. We use a default value of 0.5 for T_{sc}. If the fuzzy vector between S_j and S_{j+1} satisfy the above criteria we call it a fuzzy disparity. The set of all such disparities between images j and $j + 1$ is denoted as L_{j+1}^j.

3.3. Connecting adjacent disparities into a storm track candidate

Consider three adjacent images, j, $j + 1$ and $j + 2$ with disparity sets L_{j+1}^j and L_{j+2}^{j+1}. Let $\overrightarrow{S_j S_{j+1}}$ be a disparity in L_{j+1}^j and $\overrightarrow{S_{j+1} S_{j+2}}$ be a disparity in L_{j+2}^{j+1}. These two fuzzy storms are adjacent because they share a fuzzy storm

---

[3] In our storm tracking algorithm, we are only interested in the values of $d_{\min}$. $d_{\max} = \sup||\overrightarrow{S_j S_{j+1}}||$ is not used in our implementation of the calculation of the length of the fuzzy vector. It was felt to be superfluous and that a threshold for $d_{\max}$ could possibly have negative interactions with other thresholds. We are using a compatible size threshold, T_{sc} with a value of 0.5, so for any size compatible storms, $d_{\min} + 4r \leqslant d_{\max} \leqslant d_{\min} + 6r$ (where r is the smaller of r_1 and r_2). Also, if we chose a threshold for $d_{\max}$, then whenever adjustments are made to the threshold for T_{sc} a similar adjustment to the $d_{\max}$ threshold would have to be made, otherwise it would have conflicting results with the newly adjusted size compatibility threshold.

S_{j+1} in image $j + 1$. We measure the compatibility of two adjacent fuzzy disparities using a weighted sum of *length*, *angle* and *size* compatibility measures.

Let $d_1 = \inf ||\overrightarrow{S_j S_{j+1}}||$ and $d_2 = \inf ||\overrightarrow{S_{j+1} S_{j+2}}||$. Then, the length compatibility function is defined as

$$C_d(\overrightarrow{S_j S_{j+1}}, \overrightarrow{S_{j+1} S_{j+2}}) = \begin{cases} 1 - \frac{|d_1 - d_2|}{\max(d_1, d_2)} & \text{if } d_1, d_2 > 0, \\ 1 & \text{otherwise,} \end{cases} \tag{18}$$

the angle compatibility function is defined as

$$C_\theta(\overrightarrow{S_j S_{j+1}}, \overrightarrow{S_{j+1} S_{j+2}})$$

$$= \begin{cases} \frac{1 + \cos(\sup \langle \overrightarrow{S_j S_{j+1}}, \overrightarrow{S_{j+1} S_{j+2}} \rangle_\theta)}{2} & \text{if } d_1, d_2 > 0, \\ 1 & \text{otherwise,} \end{cases} \tag{19}$$

and the size compatibility function is defined as

$$C_s(\overrightarrow{S_j S_{j+1}}, \overrightarrow{S_{j+1} S_{j+2}}) = \frac{f_s(S_j, S_{j+1}) + f_s(S_{j+1}, S_{j+2})}{2}. \tag{20}$$

The overall compatibility function is defined as

$$C = w_d C_d + w_\theta C_\theta + w_s C_s, \tag{21}$$

where w_d, w_θ and w_s are normalized weights such that $w_d + w_\theta + w_s = 1$. We use $w_d = 0.2$, $w_\theta = 0.2$ and $w_s = 0.6$ in our reported work. The two adjacent disparities are connected together if their compatibility value is greater than a threshold:

$$C(\overrightarrow{S_j S_{j+1}}, \overrightarrow{S_{j+1} S_{j+2}}) > T_c, \tag{22}$$

where we use a value of 0.2 for T_c.

3.4. Modifying the disparity certainty by relaxation

Once all candidate disparity connections have been made, certainty of each connected disparity is refined iteratively by relaxation on the overall compatibility among its adjacent disparities. Consider a disparity $d = \overrightarrow{S_j S_{j+1}}$. The initial certainty of the disparity, denoted as $p_0(d)$, is set to $f_s(S_j, S_{j+1})$. During each iteration, we apply both spatial and temporal consistency constraints to compute the supporting and contradictory evidence of the disparity using the compatibility values. Let E_s and E_c denote, respectively, the supporting and contradictory evidence. Let n_s and n_c denote, respectively, the number of supporting disparities and the number of contradictory disparities. The four quantities are reset to zero at the start of each iteration.

- To apply the temporal consistency constraint, for each adjacent disparity d_t to d of the form $d_t = \overrightarrow{S_{j-1}' S_j}$ or $d_t = \overrightarrow{S_{j+1} S_{j+2}'}$, we compute the compatibility at the kth

iteration ($k > 0$) between the two disparities as

$$C_k(d, d_t) = w_1 \cdot C(d, d_t)$$
$$+ w_2 \cdot \left(\frac{p_{k-1}(d) + p_{k-1}(d_t)}{2} \right) \quad (23)$$

where w_1 and w_2 are normalized weights that sum to 1. We use $w_1 = 0.4$ and $w_2 = 0.6$ in our reported work. If $C_k(d, d_t) > T_k$ (we use $T_k = 0.6$), then we add $p_{k-1}(d)$ to E_s and increment n_s by 1. Otherwise, we add $p_{k-1}(d)$ to E_c and increment n_c by 1.

- To apply the spatial consistency constraint, for each disparity d_s which has the same head storm or tail storm as d, if $p_{k-1}(d) \geqslant p_{k-1}(d_s)$, then we add $p_{k-1}(d)$ to E_s and increment n_s by 1. Otherwise, we add $p_{k-1}(d)$ to E_c and increment n_c by 1.
- The certainty of the disparity d at the kth iteration is modified by

$$p_k(d) = \begin{cases} \frac{1}{2}(1 + \frac{w_s E_s - w_c E_c}{w_s E_s + w_c E_c}) & \text{if } E_s \neq 0 \text{ or } E_c \neq 0, \\ 0 & \text{otherwise,} \end{cases}$$
$$(24)$$

where w_s is the weight of the supporting evidence and is computed as $w_s = n_s/(n_s + n_c)$, and w_c is the weight of the contradictory evidence and is computed as $w_c = n_c/(n_s + n_c)$.

- The iterative process stops at the kth iteration when the certainty of each disparity has converged to the desired level of confidence, say to n decimal places (we use $n = 6$):

$$\varepsilon = |p_k(d) - p_{k-1}(d)| < 10^{-n}$$
$$\text{for each disparity } d, \quad (25)$$

or the maximum number of iterations has been reached: $k \rightarrow T_k$, where T_k is 20 in our reported work.

Once the relaxation process has converged, we construct a set of all longest tracks such that each disparity has a final certainty over a threshold T_p (we use $T_p = 0.85$). We choose a subset of these tracks, with the condition that no storm lies upon more than one chosen track.

Our tracking algorithm is *incremental*. Given n images the hypothesized storm disparities are relaxed. When an $(n+1)$th image is added, the relaxation is restarted using the results for the first n images plus the hypothesized storms of the $(n+1)$th image. We have observed that the result is always the same as if all $n + 1$ images had been initially relaxed. Processing the images and performing the relaxation is completed in less than 1 min. The difference in computation speed between the batch relaxation and the incremental relaxation is insignificant since relaxation converges within ten iterations in either mode
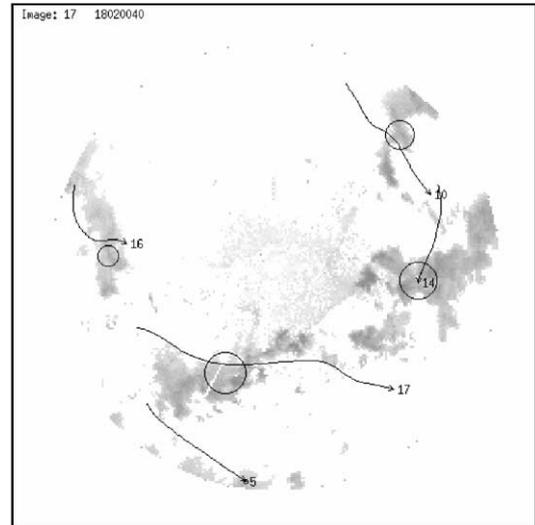

Fig. 8. Image 17 in the 18-series.

but the incremental algorithm allows us to view the current storm tracks as the data become available.

3.5. Comparing some storm tracking results

Fig. 8 shows a radar image (image 17 taken from a 30-image sequence) that has been modified for our presentation here. The gray blotches represent storms. These are the objects that are tracked. We have overlaid the storms with fuzzy points (the circles) representing the centers of mass of the storms that they overlay (see Section 3.1) and labelled lines which represent the storm tracks suggested by the current algorithm [9,10] which tracks fuzzy points representing the storms. Fig. 9 presents all of the storm tracks generated by the current algorithm.

In previous versions of our tracking algorithm [5,6] the center of mass of the storm is a Euclidean point computed as the weighted sum of the pixels representing the storm. The definition of a storm in an image (the pixels that represent a storm) is somewhat ill-defined and the change in storm shape from one image to the next can be somewhat arbitrary. This arbitrariness in the object shape translates into a fuzziness in the location of the center of mass. Using the centers of mass of the objects as the points to track proves difficult because the smooth movement assumption of the tracking algorithms is not always met. Fig. 10 shows the storm tracks [4]

---

[4] The storm tracks are labelled in the following way: Each label "$x-y$" encodes the image number "x" and the storm number "y" in image x.
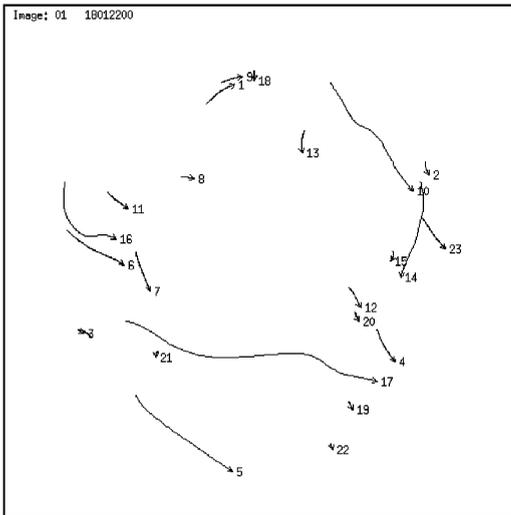
Fig. 9. Storm tracks for the 18-series using fuzzy points to represent the storms.
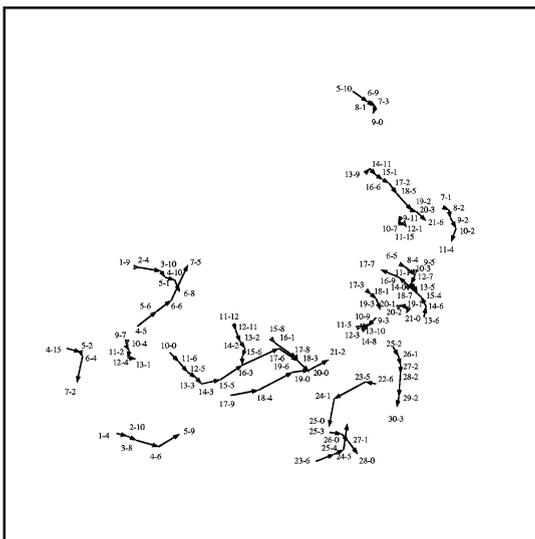

Fig. 10. Storm tracks for the 18-series using Euclidean points to represent the storms.

○ In many situations tracking a large storm fails because the smoothness property described above is not met. To counteract this problem, the gray-value threshold used to decide the storm pixels is set at levels which subdivided objects into smaller objects. It is not obvious how to translate these multiple tracks back to the large object which is represented as a number of small objects.

○ Some tracks are lost. Situations arise in which the large objects obviously track but the smaller objects are more transient and in some cases disappear before a track can be established (we have a minimal length threshold to distinguish between transient objects and real objects that we want to track).

○ The track bending threshold required to allow some desirable tracks to remain is too large, causing other false tracks to be suggested. Some tracks are suggested even though they curve too much (this is meteorologically unrealistic). Track 10-0, 11-6, 12-5, 13-3, 14-3, 15-5, 16-3, 17-6 is an example.

● The tracks in Fig. 10 that move obliquely to the top and right are false tracks. These anomalous tracks are caused by tracking centers of mass that move in this direction but the objects that they represent are actually moving obliquely to the bottom and right. In some cases they are the centers of mass of different objects which are joined by the tracking algorithm because they are closer than the distance threshold. One anomalous track is 4-5, 5-6, 6-6, 6-8, 7-5 in Fig. 10.

In the current algorithm [9,10], we address the above problems by using a fuzzy point, instead of a Euclidean point, to represent a storm center. The fuzzy point, whose size is proportional to the size of the storm to which it corresponds, hides the arbitrariness of the location of the center of mass of the storm from the tracking algorithm. Instead of tracking arbitrarily (albeit precisely) located Euclidean points, the algorithm tracks correctly located fuzzy points. The smoothness criterion does not have to be weakened by inappropriately adjusting the thresholds that implicitly define the notion of smoothness. The movement of a storm (represented as a fuzzy point) in two adjacent images is represented by means of a displacement vector. Fuzzy vectors are used as the displacement vector that connects hypothesized storms in adjacent radar images. To determine whether two fuzzy vectors in adjacent images should form part of a potential storm track, the fuzzy vector length and the magnitude of the fuzzy angle between the two fuzzy vectors must be computed and compared against a preset threshold. The fuzzy point algebra presented in this paper allows us to make these computations and comparisons. A set of appropriately selected fuzzy vectors will form a potential storm track in an image sequence.

generated by a previous algorithm [5,6] which uses normal Euclidean points to represent the center of mass of the objects. When analyzing these results, we reached a number of conclusions:

● To achieve a reasonable amount of correct tracking of the storms when represented by normal Euclidean points, the thresholds need to be set at incorrect levels. Incorrect thresholds have many undesirable consequences.

4. Summary and future work

We have presented our definition of a fuzzy point, and stated some properties of fuzzy points using properties of plane geometry. We have presented the definitions for fuzzy points, fuzzy vectors, the length of a fuzzy vector, and the fuzzy angle formed by two non-zero fuzzy vectors. To approximate the magnitude of a fuzzy angle we developed an $O(n)$ search algorithm. We have motivated our interest in fuzzy points by looking at an object tracking problem in which we track fuzzy points that are used to represent the tracked objects.

We are currently extending our tracking algorithm into 3D using Doppler radar data consisting of 15 elevations of conic Doppler density data captured at 5 and 10 min intervals. Having these elevations of Doppler data changes our 2D tracking problem into a 3D tracking problem. 3D fuzzy storms will simply be a generalization of 2D fuzzy storms (i.e. a generalization from a circle $\langle c = (x, y), r \rangle$ to a sphere $\langle c = (x, y, z), r \rangle$). Our tracking algorithm will only require minor modifications to accommodate lengths of 3D fuzzy vectors and magnitudes of 3D fuzzy angles.

Appendix A. Proofs of the theorems used by the $O(n)$ search algorithm

Lemma A.1. *An inscribed angle is half the central angle in the same arc.*

Proof. Referring to the right circle below, we want to show that $\angle aoc = 2 \angle abc$.
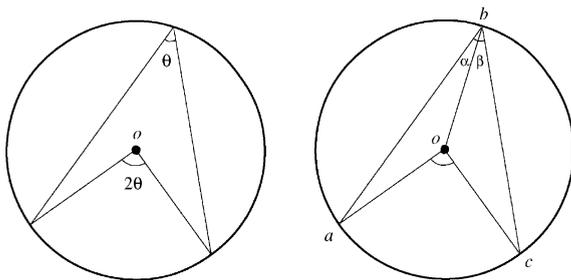
Note that oa, ob and oc are radii of the circle. Let $\angle abo = \alpha$ and $\angle obc = \beta$. Consider $\triangle boa$. Since $oa = ob$, $\triangle boa$ is an isosceles triangle. Thus, $\angle bao = \angle abo = \alpha$ and $\angle boa = 180° - 2\alpha$. Consider $\triangle cob$. Since $oc = ob$, $\triangle cob$ is also an isosceles triangle. Thus, $\angle ocb = \angle obc = \beta$ and $\angle cob = 180° - 2\beta$. Now recall the angle sum of a circle, we have

$$\angle aoc + \angle boa + \angle cob = 360° \tag{A.1}$$

$$\Rightarrow \angle aoc = 360° - \angle boa - \angle cob \tag{A.2}$$

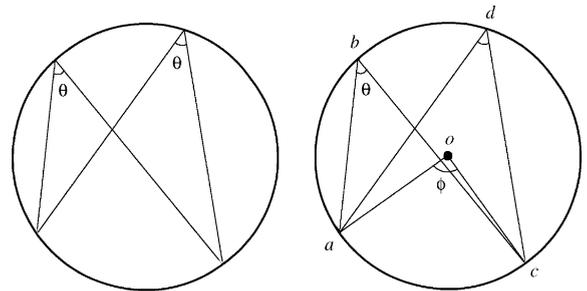$$= 360° - (180° - 2\alpha) - (180° - 2\beta) \tag{A.3}$$

$$= 2(\alpha + \beta) \tag{A.4}$$

$$= 2 \cdot \angle abc. \tag{A.5}$$

$\square$

**Lemma A.2.** *Angles inscribed in the same arc or equal arcs are equal.*

Proof. Consider the right circle below.

Let $\angle abc = \theta$. Then by Lemma A.1, $\angle aoc = 2 \cdot \angle abc = 2\theta$. Again, by Lemma A.1, $\angle adc = \frac{1}{2} \cdot \angle aoc = \theta$. Therefore, we have $\angle abc = \angle adc = \theta$. $\square$

Let A, B, and C be three non-intersecting circles in the plane. Further, let a_1, b_1, and c_1 be, respectively, a point in each of the three circles and let $\phi = \angle a_1 b_1 c_1$ denote the angle between the two line segments $\overline{a_1 b_1}$ and $\overline{b_1 c_1}$ as shown below. We write the set of all such angles as

$$S_\phi = \{ \angle a_1 b_1 c_1 \mid a_1 \in A, b_1 \in B \text{ and } c_1 \in C \}. \tag{A.6}$$
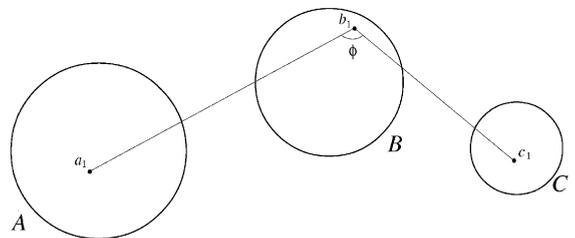

Theorem 6. *Let $A, B,$ and C be three non-intersecting circles in the plane. Suppose that $\varphi = \min(S_\phi)$. Then there exist three points $a_\varphi \in A, b_\varphi \in B$ and $c_\varphi \in C$, such that $\varphi = \angle a_\varphi b_\varphi c_\varphi$.*

Proof. Consider three points $a_1, b_1,$ and c_1 which are on or are in the interior of A, B, and C, respectively. If all three points lie on their respective circles, then the proof

is trivial and is done. Otherwise, fix the point b_1 and consider the situation as shown below.
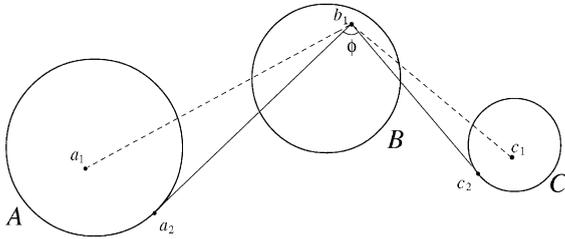

Suppose that $\angle a_1 b_1 c_1$ is the minimum angle. Now suppose that both a_1 and c_1 do not lie on the circles A and C. Then by observation, we can locate two points a_2 and c_2, both lying on their respective circles, such that $\phi = \angle a_2 b_1 c_2$ is smaller than $\angle a_1 b_1 c_1$. This contradicts the fact that $\angle a_1 b_1 c_1$ is minimum. We can use a similar argument when either a_1 or c_1 is not on the circle. Hence, both a_1 and c_1 must lie on their respective circles. Clearly, if b_1 also lies on the circle B, then the proof is done. Otherwise, we have two cases to consider. If a_1, b_1, and c_1 are three collinear points, as shown below,
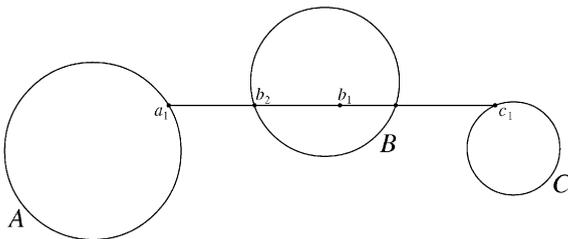
then the line segment $\overline{a_1 b_1 c_1}$ must intersect the circle B at one point, at least, namely b_2, such that $\phi = \angle a_1 b_1 c_1 = \angle a_1 b_2 c_1$.

On the other hand, if the three points are non-collinear, then we can construct a circle D such that a, b, and c are on D, as shown below. Using Lemma A.2, the circle D must intersect the circle B at one point, at least, namely b_2, such that $\phi = \angle a_1 b_1 c_1 = \angle a_1 b_2 c_1$, In either case, we have found three points lying on each of the circles, such that the corresponding angle is minimum.
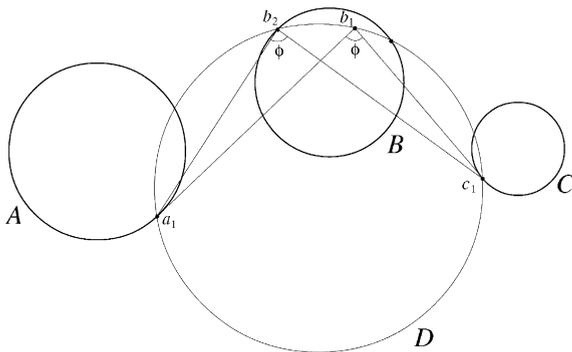
□

Theorem 7. *Let A, B, and C be three non-intersecting circles in the plane. Suppose that $\varphi' = \max(S_\phi)$. Then there exist three points $a_{\varphi'} \in A$, $b_{\varphi'} \in B$ and $c_{\varphi'} \in C$, such that $\varphi' = \angle a_{\varphi'} b_{\varphi'} c_{\varphi'}$.*

Proof. We can use a similar argument as in the proof of Theorem 6. □

References

[1] S.T. Barnard, W.B. Thompson, Disparity analysis of images, IEEE Trans. Pattern Anal. Mach. Intell. 2 (4) (1980) 333–340.

[2] V. Salari, I.K. Sethi, Feature point correspondence in the presence of occlusion, IEEE Trans. Pattern Anal. Mach. Intell. 12 (1) (1990) 87–91.

[3] I.K. Sethi, R. Jain, Finding trajectories of feature points in a monocular image sequence, IEEE Trans. Pattern Anal. Mach. Intell. 9 (1) (1987) 56–73.

[4] F. Leymarie, M.D. Levine, Tracking deformable objects in the plane using an active contour model, IEEE Trans. Pattern Anal. Mach. Intell. 15 (6) (1993) 617–634.

[5] D. Krezeski, R.E. Mercer, J.L. Barron, P. Joe, H. Zhang, Storm tracking in Doppler radar images, Proceedings of the International Conference on Image Processing (ICIP), Vol. III, 1994, pp. 226–230.

[6] H. Zhang, Storm detection in radar images, Master's Thesis, University of Western Ontario, Dept. of Computer Science, 1991.

[7] L.A. Zadeh, Fuzzy sets, Inform. Control 8 (1965) 338–353.

[8] B.B. Chaudhuri, Some shape definitions in fuzzy geometry of space, Pattern Recognition Lett. 12 (1991) 531–535.

[9] D. Cheng, R.E. Mercer, J.L. Barron, P. Joe, Tracking fuzzy storm centers in Doppler radar images, Proceedings of the International Conference on Image Processing (ICIP), Vol. II, 1996, pp. 959–962.

[10] D. Cheng, R.E. Mercer, J.L. Barron, P. Joe, Tracking severe weather storms in Doppler radar images, Int. J. Imaging Systems Technol. 9 (1997) 201–213.

[11] J. Shi, C. Tomasi, Good features to track, In CVPR, IEEE Press, 1994, pp. 593–600.

[12] L.S. Dreschler, H. Nagel, On the frame-to-frame correspondence between grayvalue characteristics in the images of moving objects, Informatik-Fachberichte, Vol. 47, Springer, 1981, pp. 18–29.

[13] I.K. Sethi, Y.K. Chung, J.H. Yoo, Correspondence using property coherence, SPIE Appl. Artif. Intell. X: Mach. Vision Robot. 1708 (1992) 653–662.

[14] V. Hwang, Tracking feature points in time-varying images using an opportunistic selection process, Pattern Recognition 22 (1989) 247–256.

[15] K. Rangarajan, M. Shah, Establishing motion correspondence, CVGIP: Image Understanding 54 (1991) 56–73.

[16] J. Verestóy, D. Chetverikov, Tracking feature points: a new algorithm and comparative performance evaluation, 1998, submitted for publication.

[17] J. Verestóy, D. Chetverikov, Experimental comparative evaluation of feature point tracking algorithms, in:

R. Klett, H.S. Stiehl, M.A. Viergever, K.L. Vincken (Eds.), Performance Characterization in Computer Vision, Kluwer Academic Publishers, Dordrecht, 2000, pp. 167–178.

[18] J. Lai, Tracking multiple features using relaxation, Pattern Recognition 26 (12) (1993) 1827–1837.

[19] M. Kass, A. Witkin, D. Terzopoulos, Snakes: Active contour models, Int. J. Comput. Vision 1 (4) (1988) 321–331.

[20] C.L. Bjerkaas, D.E. Forsyth, Real-time automated tracking of severe thunderstorms using doppler weather radar, Technical Report, Air Force Geophysics Laboratory, Hanscom AFB, MA, 1979.

[21] K.I. Hodges, A general method for tracking analysis and its application to meteorological data, Monthly Weather Rev. 122 (1994) 2573–2586.

[22] D. Cheng, Tracking fuzzy storm centers in doppler radar images, Master's Thesis, University of Western Ontario, Dept. of Computer Science, 1996.

[23] J.L. Barron, R.E. Mercer, D. Cheng, P. Joe, Tracking 'fuzzy' storms in Doppler radar images, in: Bernd Jähne, Horst Haußecker, Peter Geißler (Eds.), Computer Vision and Applications Handbook, Academic Press, Boston, January 1999, pp. 807–820.

[24] S.L. Horowitz, T. Pavlidis, Picture segmentation by a tree traversal algorithm, J. ACM 23 (2) (1976) 368–388.

[25] T. Einfalt, T. Denoeux, G. Jacquet, A radar rainfall forecasting method designed for hydrological purposes, J. Hydrol. 114 (1990) 229–244.

[26] G. Alefeld, J. Herzberger, Introduction to Interval Computations, Academic Press, New York, 1983.

[27] A. Rosenfeld, Fuzzy rectangles, Pattern Recognition Lett. 11 (10) (1990) 677–679.

[28] A. Rosenfeld, Fuzzy geometry: an overview, Proceedings of the First IEEE Conference Fuzzy Systems, San Diego, 1992, pp. 113–117.

[29] A. Rosenfeld, Fuzzy plane geometry: triangles, Pattern Recognition Lett. 15 (12) (1994) 1261–1264.

[30] R.E. Moore, Interval Analysis, Prentice-Hall, Englewood Cliffs, NJ, 1966.

[31] R.E. Moore, Methods and Applications of Interval Analysis, Society for Industrial and Applied Mathematics, Philadelphia, 1979.

About the Author—ROBERT E. MERCER is an associate professor in the Department of Computer Science at The University of Western Ontario. He is Director of the Cognitive Engineering Laboratory and Co-Director of the Centre for Cognitive Science (Artificial Intelligence Group). He received his Ph.D. in Computer Science from The University of British Columbia in 1987. His research interests include developing various knowledge representation frameworks and schemes and their associated reasoning mechanisms, and applying these processes to natural language, computer vision, animation, human–computer interaction, and information retrieval.

About the Author—JOHN L. BARRON graduated from the University of Toronto, Ontario, Canada with an M.Sc. and Ph.D. in Computer Science in 1980 and 1988, respectively. He is currently an associate professor in Computer Science at The University of Western Ontario. His research interests are in Computer Vision and include the measurement and interpretation of both optical flow and range flow and the tracking of (deformable) objects in long image sequences. Applications include measuring plant growth via optical flow and detecting and tracking severe storms in Doppler radar images.

About the Author—AIDEN A. BRUEN read mathematics for his undergraduate and master's degree in Dublin, Ireland. He obtained his doctorate in geometry at the University of Toronto where he worked with his supervisor F.A. Sherk and with H.S.M. Coxeter. Currently, he is working in theoretical biology at the Los Alamos National Laboratory. In July 2001, he takes a new position as Professor of Mathematics and member of the provincially-funded number theory and cryptography research group at the University of Calgary.

His research interests include geometry, error-correcting codes, block designs, statistics, theoretical computer science and cryptography. He serves on the editorial board of Designs, Codes and Cryptography.

About the Author—DAVID CHENG obtained his B.Sc. and M.Sc. degrees in Computer Science at The University of Western Ontario. He is currently a consultant in the IT department of BMO Nesbitt Burns Inc.