Electrical
Computer
Engineering

DEPARTMENT OF ELECTRICAL
AND COMPUTER ENGINEERING

UNIVERSITY OF WISCONSIN-MADISON

From Moving Frames To Summation Invariants: Procedures, Properties and Application

Kerry Widder, Wei-Yang Lin, Nigel Boston, Yu Hen Hu

September 2007 ECE-07-05

ENGINEERING EXPERIMENT STATION
MADISON, WISCONSIN 53706

FROM MOVING FRAMES TO SUMMATION INVARIANTS: PROCEDURES, PROPERTIES AND APPLICATION

Kerry R. Widder[1], Wei-Yang Lin[2], Nigel Boston[1], Yu Hen Hu[1]

[1]University of Wisconsin-Madison, Dept. of Electrical and Computer Eng., 1415 Engineering Drive, Madison, WI, 53706 USA
[2]National Chung Cheng University, Dept. of Computer Science and Information Engineering, EA 310, 168 University Rd., Min-Hsiung, Chia-Yi, Taiwan

ABSTRACT

The method of moving frames, a powerful mathematical tool for deriving geometrically invariant functions, is described. A systematic approach is outlined for the derivation of new members of a family of geometrically invariant features using the moving frame method. This family of features is called summation invariant. An example derivation is given to illustrate the procedure. The current members of this family are summarized and several implementation considerations for these features are investigated. A naming convention is given and a standard test is defined for the purpose of comparing the discrimination ability of these features. This test is used to compare the features derived so far using the application of face recognition and the Face Recognition Grand Challenge (FRGC2.0) dataset.

1. INTRODUCTION

Invariant features are an important tool in the pattern recognition toolbox. Objects to be recognized in images usually are not guaranteed to be in the same location, of the same orientation, of the same size, nor even with the same shape. Thus, having a descriptive feature that is invariant to geometric transformations, like translation, rotation, scale or shear, or that is invariant to all of them, is highly desirable. Summation invariants are one family of geometrically invariant features that was developed recently [1-7].

This paper will briefly review some other existing invariant features and will summarize the summation invariants developed thus far. Then, it will describe the moving frame method and present a systematic approach for deriving summation invariants. Some implementation issues will also be analyzed. A standard experiment will be defined and used to evaluate the discrimination performance of the summation invariants derived so far.

2. INVARIANT FEATURES

2.1. Previous work

One of the early geometric invariant features is the moment invariant [8]. This type of feature is global in nature, using the whole image, and is invariant to rotation, translation, and scale, and in some cases even to illumination changes [9]. The global nature limits the discrimination ability of moment invariants, requiring the use of higher order moments to improve this ability. The higher order moments contain the detail information about the image. The performance of this class of invariant with occlusions is hindered by the global nature of this feature. The higher order moments are more susceptible to being affected by noise, which limits their usefulness, and hence the amount of detail which can be included. Moment invariants have been applied to areas such as airplane recognition [10] and Chinese character representation [11].

Fourier descriptors utilize a set of Fourier transform coefficients to represent a closed curve. The normalization of these descriptors to a transformation group results in them being invariant to transformations within that group. The similarity transform is usually used (rotation, translation and scale), although the affine transformation group has also been used [12]. The global nature of the Fourier transform prevents the extraction of localized feature information. However, unlike moment invariants, the feature space will be inherently large. The higher order detail coefficients will tend to be more affected by noise, so noise immunity can be tuned by dropping some portion of the higher order coefficients. These features have been applied to airplane recognition [12].

Wavelet based invariants offer better localization than Fourier methods and offer multiple resolution levels. Dyadic wavelet features with invariance to affine transformations have been developed [13]. Noise sensitivity is adjustable by changing the number of resolution levels used, i.e., dropping higher resolution coefficients will improve the noise performance at the expense of a loss of detail. Dyadic wavelet invariants have been applied to airplane recognition [13] and Gabor wavelet invariants have been applied to face recognition [14].

Differential invariant features have been widely studied and applied. They are local in nature, but the use of high order derivatives in their calculation makes them sensitive to noise. Methods to minimize the noise sensitivity have been proposed, such as semi-differential invariants [15] and numerical approximations [16]. Differential invariants have a large feature space, which improves discrimination performance, and their local nature allows the accommodation of occlusions. Differential invariants have been applied to medical images [16] and face recognition [17].

Integral invariant features are global in nature, but can be made ‘semi-local’ by changing the limits on the integration [18]. This technique will also increase the size of the feature space and improve the discrimination ability of these features, as well as allow for handling occlusions. The integral invariant features are derived from ‘potentials’ involving integration, instead of derivatives, giving them a decreased sensitivity to noise [19]. Another advantage to this invariant is a systematic method of deriving new features. Integral invariants have been applied to natural images (leaves) [18] and fish and hand shapes [20].

2.2. Summation invariant

Summation invariants share many of the features of integral invariants, including being global in nature, able to be made semi-local to increase the feature space and handle occlusions, decreased sensitivity to noise and a systematic method for deriving new invariants. However, summation invariants are defined with potentials that are summations instead of integrals, so calculating them will not involve numerical approximations. Summation invariants have been applied to fish shapes [1, 6] and face recognition [4-7].

3. SUMMATION INVARIANT

The idea of summation invariant is based on the mathematical method of moving frames. The notion of moving frames was first proposed by Cartan [21], and later formalized into a systematic method by Fels and Olver [22, 23]. To understand and use this method, several concepts need to be defined. In this section, brief definitions and a synopsis of the method in its simplest form are given, followed by extensions of the method using jet spaces. Then, a systematic procedure is given for deriving new families of summation invariants, a naming convention is proposed and a detailed example derivation is given.

3.1. Cartan's Method of Moving Frames

Cartan's method of moving frames provides a framework for deriving geometric invariants for a specific transformation group. Given a manifold, *M*, of dimension, *m*, and a Lie group, *G*, of dimension, *r*, acting smoothly on *M*, invariant functions $I : M \to \Re$ meaning that *I(g∘z) = I(z),* for all $g \in G$ and $z \in M$, are desired. Moving frames provide a mechanism for systematically deriving these invariant functions. Since moving frames are intimately connected to a transformation group, any invariants coming from this method will only be invariant to transformations within that group, or transformation groups that are subsets of the group the invariants were derived under. The concepts discussed here will be illustrated using a simple example consisting of a manifold, *M* = $\Re^2$, and a transformation group, *G* = SO(2), consisting of rotations of angle θ about the origin.

A *parameterized curve* in $\Re^2$ is a curve where some other variable, or parameter, is used to identify points on the curve. Examples of parameters include arc-length, time and order (sample number). A curve describing the motion of an object, like $y = f(x) = ax^2$, could be parameterized by time, as $y(t) = a(x(t))^2$. If samples are taken of a continuous curve, then that curve is parameterized by *n*, where $1 \le n \le N$ is the sample number. If this curve is transformed geometrically, the transformed curve can be described by another set of points parameterized by n, $\{(\bar{x}[n], \bar{y}[n]);\ 1 \le n \le N\}$ such that $(\bar{x}[n], \bar{y}[n])$ is transformed from (x[n], y[n]) for each *n*. The same idea can be extended to higher dimensions, e.g., surfaces in $\Re^3$.

A *potential*, $P_{i,j}$, of order *k* is defined as:

$$P_{i,j} = \sum x^i[n] y^j[n] \quad , \tag{3.1}$$

where k = i + j, and $i, j \ge 0$. The average values of the variables can be recovered from the first order potentials (*k* = 1) by dividing by *N*, the number of samples.

Potentials are similar to moments, but more general. For a discrete function, *f(x),* the moment is defined by:

$$m_p = \sum x^p f(x) \tag{3.2}$$

which can be put in parameterized form as follows:

$$m_p = \sum_n x^p[n] y[n] \quad , \tag{3.3}$$

where y = f(x). In this formulation, it is evident that potential $P_{i,1}$ is the same as m_i, or $P_{i,1} = m_i$. Extending to the case of a surface, $P_{i,j,1} = m_{i,j}$. These observations lead to the following lemma:

Lemma: A geometric moment is equal to a potential with the last index equal to one. (Note: the dimension of the moment is one less than the potential.)

A parameterized 3D surface in $\Re^3$ can be described by a set of points $\{(x[m,n] \quad y[m,n] \quad z[m,n]),\ 1 \le m \le M,\ 1 \le n \le N\}$. In this case, a *potential*, $Q_{i,j,k}$, of order *l* is defined as:

$$Q_{i,j,k} = \sum_{m=1}^{M} \sum_{n=1}^{N} x^i[m,n] y^j[m,n] z^k[m,n], \tag{3.4}$$

where $l = i + j + k$, and $i, j, k \ge 0$.

A *manifold*, *M*, is an object for which every local neighborhood looks like a subset of Euclidean space. A 1D manifold is a smooth curve with no self-intersection. An example of a 2D manifold is a torus.

An *orbit*, $\mathcal{O}_z$, is the set of all transformations of *z* under the action of the given transformation group. In the example *M* = $\Re^2$ and *G* being rotation around the origin, an orbit is the set of all circles centered on the origin.

A *canonical set*, *K*, is a subset of *M* such that *K* intersects each orbit of *z* at exactly one point, *u*. In the example given above, one possible canonical set would be the x-axis from the origin to $+\infty$, since each orbit (circle centered on the origin) would intersect it at only one point.

A *free action* is one where given points *a* and *b* there is at most one transformation that sends point *a* to point *b*. An example of action that is not free is the situation with G = SE(2) (rotations about the origin by angle θ and translations by *a* and *b* in the *x* and *y* directions, respectively) and $M = \Re^2$. For $a = (1,0)$ and $b = (0,1)$, two of the transformations that will take *a* to *b* are $(\theta, a, b) = (\pi/2, 0, 0)$ and $(\theta, a, b) = (0, -1, 1)$, thus the action is not free.

A *transformation group*, *G*, acting on a manifold, *M*, is a group with smooth action that satisfies

$$e \circ z = z,\quad g \circ (h \circ z) = (g \circ h) \circ z, \tag{3.5}$$

for all $z \in M$, $g \in G$, where *e* is an identity element in the group. Examples include Euclidean (rotation and translation) and affine (rotation, translation, scaling and shear).

Smooth action of a transformation group means that the group operation is infinitely differentiable. In the example, the group action of rotation about the origin by angle θ takes a point (x,y) to a point $(\overline{x}, \overline{y})$, where $(\overline{x}, \overline{y}) = ((x \cdot \cos\theta - y \cdot \sin\theta), (x \cdot \sin\theta + y \cdot \cos\theta))$, which is clearly infinitely differentiable.

A *moving frame* is loosely defined as a function of *z* (where *z* is a point on the manifold *M*) that produces the unique transformation, $g \in G$, that sends *z* into *K*. The existence of a moving frame is dependent on the group action on the manifold being free. It is called 'moving' since it is different for each point on the manifold. More formally, it is a smooth map, $\rho: M \rightarrow G$, such that $\rho(g \circ z) = g \circ \rho(z)$ for all $g \in G$ and $z \in M$. In the example, a moving frame for $M = \Re^2$, and *G* being rotation around the origin, is $-\theta$, where θ is the angle of the point *z* with respect to the x-axis, and where *K* is the x-axis from the origin to $+\infty$.

If an invariant function $I(z)$ is given, then any function of *I*, $f(I(z))$, is also invariant for *G* acting on *M*. The invariants, $I_1, \ldots, I_k$, are *fundamental invariants* for *G* acting on *M* if three conditions are satisfied: 1) they are indeed invariant, 2) none of them are redundant, and 3) every invariant can be expressed as a function of them.

The number of fundamental invariants that can be derived using the moving frame method is limited to *k*, where $k = m - r$, $(m > r)$, with *m* the dimension of *M* and *r* the dimension of *G*. Once the moving frame is derived, it can be applied to the remaining *k* dimensions of *M* not fixed in the canonical form to give *k* invariant functions.

To see how this works, let $z = (z_1, \ldots, z_m)$ be a point on the manifold *M*, and let $w(g, z) = (w_1(g, z), \ldots, w_m(g, z))$ be the explicit formulas for the group transformation of *z* (i.e., $\overline{z_1} = w_1(g, z)$). Then the canonical set, *K*, will fix *r* coordinates of *M*, i.e., $K = \{z_1 = c_1, \ldots, z_r = c_r\}$, where the c_i's are constants. The moving frame is found by solving

$$w_1 = c_1, \ldots, w_r = c_r \tag{3.6}$$

for the transformation group parameters. This set of group parameters is the moving frame since it will transform any point *z* in *M* to *K*. The set of equations (3.6) is called the *normalization equations*.

The invariant functions are derived by taking the moving frame and applying it to the remaining explicit formulas not fixed by *K*, i.e., $w_{r+1}, \ldots, w_m$. More formally, the invariant functions are given by

$$I_1(z) = w_{r+1}(\rho(z), z),$$
$$\ldots,$$
$$I_k(z) = w_{r+1}(\rho(z), z) \tag{3.7}$$

and are invariant for any $g \in G$.

To see that these functions are invariant, consider two points in M, z and z′, related by a transformation $h \in G$, i.e., $z' = h \circ z$. These points are by definition in the same orbit. A moving frame $\rho(z)$, derived using (3.6), will take z and z′ and transform them to u and u′, where u and u′ are in K, and $u = (c_1, \ldots, c_r, w_{r+1}(\rho(z), z), \ldots, w_m(\rho(z), z))$ and $u' = (c_1, \ldots, c_r, w_{r+1}(\rho(z'), z'), \ldots, w_m(\rho(z'), z'))$. Since z and z′ are in the same orbit, and K by definition intersects each orbit in only point, then $u = u'$ and hence $w_{r+1}(\rho(z), z) = w_{r+1}(\rho(z'), z')$, etc., and $I_1,\ldots,I_k$ are invariant to all transformations in G.

Applying this procedure to the simple example used in this section, where $M = \Re^2$, and G = SO(2) (rotations of angle θ about the origin), $m = 2$, $r = 1$ and $k = 1$. Thus, one invariant can be derived. Let $K = \{x \mid y = 0, x \geq 0\}$, the set of all points on the positive x-axis plus the origin. The w_i's are given by

$$w_1 = x \cdot \cos\theta - y \cdot \sin\theta$$
$$w_2 = x \cdot \sin\theta + y \cdot \cos\theta \qquad (3.8)$$

and the normalization equation is

$$w_2 = x \cdot \sin\theta + y \cdot \cos\theta = 0 \qquad (3.9)$$

where only one w_i is made constant since $r = 1$.

Solving for θ gives the moving frame, which is

$$\rho(z) = \theta = -\tan^{-1}\left(\frac{y}{x}\right) \qquad (3.10)$$

The invariant is found by applying the moving frame to w_1, the w_i not used in the normalization equations, giving

$$I_1(z) = w_1(\rho(z), x, y) = \sqrt{x^2 + y^2} \qquad (3.11)$$

which is the distance of z from the origin. Intuitively this makes sense, since as a point is rotated about the origin, its distance from the origin will remain constant.

3.2 Extensions using Jet Space

The number of fundamental invariants that can be derived using this method is k, where $k = m - r$, m is the dimension of the manifold and r is the dimension of the group action. In the example of the previous section, with $m = 2$ and $r = 1$, then $k = 2 - 1 = 1$. If $m < r$, then the action is not free and it is necessary to replace the manifold, M, by a larger-dimensional manifold, namely jet space, before the moving frame method can be applied. This expansion can be as large as needed – the more dimensions added, the greater the number of invariant functions that can be generated. In the previous section, the example was a case where the group action was free, hence the moving frame existed and an invariant could be found. In general, this will not be the case, since usually the group action will have a higher dimension than the manifold.

Consider a new example, where $M = \Re^2$ and G = SE(2) (rotations about the origin by angle θ and translations by a and b in the x and y directions, respectively). The group action in this example will map a point $z = (x, y)$ into a transformed point $\bar{z} = (\bar{x}, \bar{y})$, where

$$\bar{x} = x \cdot \cos\theta - y \cdot \sin\theta + a$$
$$\bar{y} = x \cdot \sin\theta + y \cdot \cos\theta + b \qquad (3.12)$$

In this example, $m = 2$, $r = 3$ and $k = -1$ and the group action is not free. This means a moving frame does not exist and the moving frame method cannot be used to find invariants. To overcome this limitation, a manifold must be found with dimension greater than the group action. This is accomplished by generating a jet space with sufficient dimension and using it for deriving the invariants.

Traditionally, a *jet space*, J^n, is a Euclidean space with additional coordinates corresponding to the derivatives of the dependent variables, up to the nth order:

$$(x, u^{(n)}), \qquad (3.13)$$

where x represents all the independent variables and $u^{(n)}$ represents all of the dependent variables and all partial derivatives up to the nth order. This mechanism is used to formally handle derivatives when dealing with the action of transformation groups. The additional coordinates provide a richer description. They also expand the dimensions of the space that applies to the particular problem being addressed, allowing more invariants to be generated. Applying the group action to jet space is called prolonging it into jet space.

The new example used in this section, with $k = -1$, will require two derivative terms in the jet space to achieve $k = 1$ and thus to allow generating one invariant. This jet space is given by

$$J^{(2)} = (x, y, y_x, y_{xx}), \qquad (3.14)$$

where y_x denotes the first derivative of y with respect to x and y_{xx} denotes the second derivative of *y* with respect to *x*. After prolonging the group action into this jet space, the resulting transformed coordinates for a parameterized curve, *z(t) = (x(t), y(t))* are given by (3.12) and

$$\overline{y_x} = \frac{d\overline{y}}{dt}\frac{dt}{d\overline{x}} = \frac{x_t \sin\theta + y_t \cos\theta}{x_t \cos\theta - y_t \sin\theta}$$

$$\overline{y_{xx}} = \frac{d}{d\overline{x}}\frac{d\overline{y}}{d\overline{x}} = \frac{x_t y_{tt} + x_{tt} y_t}{\left(x_t \cos\theta - y_t \sin\theta\right)^3} \quad (3.15)$$

These four formulas are the w_i's for this example. Adding more terms to the jet space would allow the generation of more invariants.

To derive the moving frame for this example, a canonical set, *K* must be defined and the normalization equations arising from *K* solved for the parameters of the moving frame. A suitable *K* is

$$w_1 = \overline{x} = 0, \quad w_2 = \overline{y} = 0, \quad w_3 = \overline{y_t} = 0 \quad (3.16)$$

Solving for *θ*, *a* and *b* yields the moving frame

$$\theta = \tan^{-1}\left(\frac{y_t}{x_t}\right)$$

$$a = \frac{xx_t + yy_t}{\sqrt{x_t^2 + y_t^2}}, \quad b = \frac{xy_t + yx_t}{\sqrt{x_t^2 + y_t^2}} \quad (3.17)$$

Substitution of this moving frame into the formula for $\overline{y_{xx}}$ (3.15) gives the invariant

$$I(z) = \frac{x_t y_{tt} + x_{tt} y_t}{\left(x_t^2 + y_t^2\right)^{3/2}} = \kappa \quad (3.18)$$

which is the curvature.

The derivative jet space provides the needed dimensionality for applying the moving frame method and generating invariants. However, the reliance on higher order derivatives makes the resulting invariants sensitive to noise. A different approach that addresses this noise sensitivity uses potentials, which are based on integrals instead of derivatives, to define the jet space. This eliminates the high order derivatives and hence the noise sensitivity problem.

The integral potential jet space, J_p^n, is defined as a Euclidean space with coordinates

$$(x, y, x_0, y_0, V_{(n)}), \quad (3.19)$$

where (x_0, y_0) is the initial conditions and $V_{(n)}$ is a set of potentials, defined as a potential $V^{i,j}$ of order *k*, where

$$V_x^{i,j} = x^i y^j \quad (3.20)$$

with $j \neq 0$ and $k = i + j$. Thus, for $z = V^{0,1}$, the potential is

$$z = \int_{x_0}^{x} y\,dx \quad (3.21)$$

(see [19] for details). The method of moving frames can also be applied to this type of jet space to derive geometric invariants.

Another approach, summation invariants, utilizes the moving frame method with a *jet space*, J^n, defined as a Euclidean space with coordinates

$$(x[1], y[1], x[N], y[N], P^{(n)}), \quad (3.22)$$

for the case of a curve, where $P^{(n)}$ is all potentials up to and including the nth order and *x[k], y[k]* are points on the curve parameterized by *k*, and $1 \leq k \leq N$. For a parameterized 3D surface in $\Re^3$, the corresponding jet space is given by:

$$J^n = (x[1,1], y[1,1], z[1,1], x[M,1], y[M,1], z[M,1], x[1,N], y[1,N], z[1,N], Q^{(n)}) \quad (3.23)$$

where $Q^{(n)}$ is all potentials up to and including the nth order. This jet space definition based on summations also avoids high order derivatives and noise sensitivity. Its advantage over the integral-based potentials is that it deals directly with discrete data, whereas for the integral approach, the discrete (sampled) data is an approximation to the actual data (continuous) and the accuracy will be dependent on the sampling rate.

3.3 A systematic procedure for deriving summation invariants based on Cartan's Method of Moving Frames

To find summation invariants under a transformation group, *G*, it is necessary to find an appropriate canonical set, which leads directly to a set of normalization equations. Solving these equations for the transformation variables gives a moving frame. Invariant functions are found by applying this moving frame to the higher order potentials not used in deriving the moving frame.

A procedure for deriving new families of summation invariants is outlined below and a detailed example derivation using this procedure is given in the next section:

1. Define the kind of transformation group (e.g., Euclidean, affine, etc.) and mode (e.g., 2D, 3D).

2. Determine the equations for the transformed variables and potentials.

3. Define the canonical set.

4. Define the normalization equation from the canonical set.

5. Find the moving frame from the normalization equation. (Solve the set of equations defined by the normalization equation to get the transformation variables. May need to use a math solver program, like Maple®.)

6. Apply the moving frame to higher order potentials to get the invariants. (i.e., potentials that were not part of the normalization equation.)

7. Verify that they are actually invariant. Since the theory guarantees invariance, this step is a sanity check to make sure no errors were made in the derivation process.

(Note: It may be necessary to try a different normalization equation if the one selected is not solvable or is too complicated to be practical.)

The naming convention used in most of the previous works [1-7] can be modified slightly to include more information about the invariants. This modified convention is formalized below:

$\eta^{E}_{1,1}$, where

1. The variable name indicates mode (e.g., η=2D, κ=3D).
2. The superscript indicates transformation group (e.g., E=Euclidean, A=affine, S=similarity, P=projective, Pp=planar-projective).
3. The subscripts indicate the potential it was derived from (e.g., '1,1'=$P_{1,1}$, '2,0'=$P_{2,0}$).
4. The normalization equation used in the derivation will need to be stated elsewhere, as it would be too cumbersome to include in the naming convention.

3.4. Example derivation

An example of deriving a new family of summation invariants using the procedure given above follows. The mode used is 2D, the transformation group selected is Euclidean and the normalization equation is (*x[1], y[1], x[N]*) = (0, 0, 0,).

The original work for the 2D Euclidean transformation group only used the numerator of the resulting formulas in the experimentation and for reporting the formulas, since the numerator and denominator were shown to be relatively invariant. A relative invariant is invariant to the group transformation up to a factor that is a function only of the transformation parameters, not the object points, i.e., $I(g\circ u) = f(g)I(u)$. When classification is performed using a measure such as normalized cross-correlation, the factors will cancel. This practice will be used here as well.

Euclidean transformation (2D):

$$\begin{bmatrix} \bar{x} \\ \bar{y} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} a \\ b \end{bmatrix} \qquad (3.24)$$

Transformed potentials:

$$P_{1,0} = \sum_{n=1}^{N} x[n] \ , \ \ P_{0,1} = \sum_{n=1}^{N} y[n] \qquad (3.25)$$

Apply Euclidean transformation:

$$\overline{P}_{1,0} = \sum_{n=1}^{N} \overline{x}[n] = \sum_{n=1}^{N} \left(x[n]\cdot \cos(\theta) - y[n]\cdot \sin(\theta) + a\right)$$
$$= P_{1,0}\cdot \cos(\theta) - P_{0,1}\cdot \sin(\theta) + aN \qquad (3.26)$$

$$\overline{P}_{0,1} = \sum_{n=1}^{N} \overline{y}[n] = \sum_{n=1}^{N} \left(x[n]\cdot \sin(\theta) + y[n]\cdot \cos(\theta) + b\right)$$
$$= P_{1,0}\cdot \sin(\theta) + P_{0,1}\cdot \cos(\theta) + bN \qquad (3.27)$$

Similarly, for $P_{2,0} = \sum_{n=1}^{N} x^2[n]$, $P_{1,1} = \sum_{n=1}^{N} x[n]y[n]$, and $P_{0,2} = \sum_{n=1}^{N} y^2[n]$, after transformation:

$$\overline{P}_{2,0} = \sum_{n=1}^{N} \overline{x}^2[n] = \sum_{n=1}^{N} \left[\cos\theta\cdot x[n] - \sin\theta\cdot y[n] + a\right]^2$$
$$= P_{2,0}\cdot\cos^2\theta + P_{0,2}\cdot\sin^2\theta - 2\cdot P_{1,1}\cdot\sin\theta\cdot\cos\theta$$
$$+ 2\cdot P_{1,0}\cdot a\cdot\cos\theta - 2\cdot P_{0,1}\cdot a\cdot\sin\theta + N\cdot a^2$$
$$(3.28)$$

$$\overline{P}_{1,1} = \sum_{n=1}^{N} \overline{x}[n]\overline{y}[n] = \sum_{n=1}^{N} \left[\cos\theta\cdot x[n] - \sin\theta\cdot y[n] + a\right]$$
$$\cdot\left[\sin\theta\cdot x[n] + \cos\theta\cdot y[n] + b\right]$$
$$= P_{1,1}\left(\cos^2\theta - \sin^2\theta\right) + \left(P_{2,0} - P_{0,2}\right)\left(\sin\theta\cdot\cos\theta\right)$$
$$+ P_{1,0}\left(b\cdot\cos\theta + a\cdot\sin\theta\right)$$
$$+ P_{0,1}\left(a\cdot\cos\theta + b\cdot\sin\theta\right) + N\cdot a\cdot b$$
$$(3.29)$$

$$\overline{P}_{0,2} = \sum_{n=1}^{N} \overline{y}^2[n] = \sum_{n=1}^{N} \left[\sin\theta\cdot x[n] + \cos\theta\cdot y[n] + b\right]^2$$
$$= P_{0,2}\cdot\cos^2\theta + P_{2,0}\cdot\sin^2\theta + 2\cdot P_{1,1}\cdot\sin\theta\cdot\cos\theta$$
$$+ 2\cdot P_{1,0}b\cdot\sin\theta + 2\cdot P_{0,1}b\cdot\cos\theta + N\cdot b^2$$
$$(3.30)$$

Canonical Set/Normalization equation:

$$\left(\overline{x}[1], \overline{y}[1], \overline{x}[N]\right) = (0, 0, 0) \qquad (3.31)$$

(Note: instead of $\overline{x}[n]$, could have used $\overline{y}[n]$, $\overline{P}_{1,0}$, $\overline{P}_{0,1}$, etc.) Then solve for the moving frame $\{\theta, a, b\}$ based on the normalization equation (use Maple, Mathematica® or similar)(See results in Appendix B)

Apply the moving frame (i.e., values of θ, a, b obtained above) to potentials not used in the normalization equation to get summation invariants using Maple, Mathematica or similar symbolic mathematical packages.

Verify the invariance of the derived formulas by applying them to curves that have been subjected to the transformation, e.g., define a curve and several Euclidean transformations of it and compute the new summation invariant for each transformed curve – they should all be the same. (Use Matlab®, or similar.)

The Matlab code to implement this verification is shown in Appendix C. It first computes the values for the original curve with the denominator term, then computes them for ten random transformations ($0\le\theta\le2\pi$, $0\le a,b\le100$) without the denominator term. (Note: the original work left out the denominator term, so the experimentation was done without it for the new invariants to get an equal comparison.) The results of this test for two random transformations are shown below, where H_x corresponds to $\eta_{1,0}$, H_y corresponds to $\eta_{0,1}$, etc.:

Table 1. List of Summation Invariants derived so far.

Mode	Transform	Normalization Equation	Invariants Derived1,2,3	Published	Data set
2D	Euclidean	$(x_1,y_1,y_N) = (0,0,0)$	$\eta^E_{1,0},\eta^E_{1,0},\eta^E_{1,0},\eta^E_{1,0},\eta^E_{1,0}$	1) [3] 2) [4] 3) [5] 4) [6] 5) [7]	1) FRGC 1.0 2) FRGC 1.0 3) FRGC 1.0 4) FRGC 1.0/2.0 5) FRGC 2.0
2D	Affine	$(x_1, y_1, x_N, y_N, P_{10}, P_{01}) = (0, 0, 1, 1, 0, 0)$	$\eta^A_{2,0}$	1) [1] 2) [6]	SQUID
2D	Euclidean	$(x_1,y_1,_{xN}) = (0,0,0)$	$\eta^E_{1,0},\eta^E_{1,0},\eta^E_{1,0},\eta^E_{1,0},\eta^E_{1,0}$		FRGC 2.0
3D	Euclidean	$(x_{11}, y_{11}, z_{11}, y_{M1}, z_{M1}, z_{1N},) = (0,0,0,0,0,0)$	$\kappa^E_{0,0,1},\kappa^E_{0,1,0},\kappa^E_{1,0,0}$	1) [3] 2) [5] 3) [6]	FRGC 1.0
			$\kappa^E_{1,1,0},\kappa^E_{1,0,1},\kappa^E_{0,1,1}$		FRGC 2.
3D	Affine	$(x_{11}, y_{11}, z_{11}, x_{M1}, y_{M1}, z_{M1}, x_{1N}, y_{1N}, z_{1N}, P_{100}, P_{010}, P_{001}) = (0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0)$	$\kappa^A_{2,0,0}$	1) [2] 2) [6]	1) 3D cafe face

1. The variable name indicates mode (e.g., η=2D, κ=3D).
2. The superscript indicates transform group (e.g., E=Euclidean, A=affine, S=similarity, P=projective).
3. The subscripts indicate the potential it was derived from (e.g., '1,1'=$P_{1,1}$, '2,0'=$P_{2,0}$).

```
run ck_krw1_curve_Eu_sum_inv
% krw1_curve_Eu_sum_inv_denom
(Hx, Hy, Hxx, Hxy, Hyy) = (-6.1331, -35, 0.65619, 3.0666, 23.7209)
% krw1_curve_Eu_sum_inv
(Hx, Hy, Hxx, Hxy, Hyy) = (-6.1331, -35, 0.65619, 3.0666, 23.7209)
(Hx, Hy, Hxx, Hxy, Hyy) = (-6.1331, -35, 0.65619, 3.0666, 23.7209)
```

The data confirms that the new summation invariants are indeed invariant to Euclidean transformations. Also, the denominator term had no effect on the values of the summation invariants.

A list of the summation invariants derived thus far is given in table 1, along with where each has been published. Detailed formulas for each are listed in Appendix A. Each invariant family is specified by mode, transformation group and normalization equation.

4. IMPLEMENTATION CONSIDERATIONS

In applications to pattern classification, an invariant feature is computed from a given object (curve or surface) which may be subject to geometric transformations such as translation, or rotation. Being an invariant feature, its value should remain the same before and after the rigid object is subject to the geometric transformation. As such, one may evaluate the invariant feature based on the transformed coordinates $\left(\bar{x}[n] \quad \bar{y}[n]\right)$ without knowing the parameters, e.g. θ, a, or b, nor the original coordinates ($x[n]$ $y[n]$).

In practice, there may be factors which adversely affect the invariance of such a feature. This section examines several such factors and their affects. The factors examined here include correspondence, spatial quantization after geometric transformation, occlusion, finite precision arithmetic and geometric scaling. Other implementation considerations that contribute to the effectiveness of the invariants for object recognition will also be examined, including feature dimensionality, non-rigid geometric transformations and fusion of results from multiple regions.

4.1. Correspondence of Data Points, Spatial Quantization, Occlusion, Finite Precision Arithmetic and Geometric Scaling

One issue that is prevalent in object recognition is that of correspondence, making sure that the same things are being compared. This issue is also important in the application of summation invariants. In [4, 5] optimizing the alignment of the region used for face recognition resulted in improved performance. This optimization was implemented by taking the mean of the summation invariant images computed from the training images and using the sum of squared differences (SSD) measure to find the closest match for each new image compared to the mean training image. When applying summation invariants, better results are obtained if measures are taken to insure good correspondence between objects.

In practical applications, the geometric transformation of an object in an image is not the result of a manipulation of the original data points, but from a manipulation of the object or the camera, resulting in a new image of the transformed object. In general, this results in a different set of sampled data points than those obtained by just transforming the original data set due to the quantization effects of the camera. The new image of the transformed object will contain points sampled from slightly different locations on the object than in the original image. This spatial quantization after geometric transformation may result in not only the data points being different, but also a different number of data points for a given region. Both of these changes will adversely affect the invariance of the summation invariant. One way to alleviate this problem is to re-sample the transformed curve or surface in the parameter space rather than the spatial coordinates. One example of this is re-sampling a curve with points located at equal arc-length points along the curve. This issue will be examined more fully in the next section.

Occlusion of all or portions of an object in an image can occur in several ways. Sometimes a 3D surface is imaged digitally using a range image map, which corresponds to a projection of the surface in a particular direction. When the object undergoes a geometric transformation (e.g., rotation), the image of the transformed object may have portions of the object that were visible in the original image now occluded by other parts of the object. Extensive translation may also render some, or all, of the transformed object outside the viewing window. Both of these situations may lead to inaccurate values of invariant features, or even render the calculation impossible. To address this concern, the ranges of allowable geometric transformation have to be restricted to prevent this from occurring, or measures such as using semi-local features (to be discussed in the next sub-section) need to be used. This situation also requires that the correspondence of the data points for invariant feature calculation must be accurately established.

Another factor that may affect invariance is numerical issues due to finite precision arithmetic, such as rounding errors. The original works on summation invariants applied to face recognition [4-7] utilized double precision floating-point numbers in the calculations. As long as there is no division operation, the numerical rounding error accumulation should be of little concern.

Geometric scaling will also affect the values of summation invariants. This factor arises from images taken at different distances from the object or from images taken at different resolutions. Either case will result in sample points at different locations and a different density of sample points. The summation invariant will produce different values under these conditions for the same object. To reduce this affect, re-sampling of the object images can be done to give all instances of the object the same density of data points. The wide range of possible variations makes it difficult to quantify this affect in general.

Another issue that may arise involves the scaling of the images. The summation invariants derived under Euclidean transformations are not invariant to scaling, so if the images to be compared are of different scales, or if the original scale information is lost, then it will be necessary to normalize the images to the proper scale to get the best results [4, 5]. This may require re-sampling.

4.2. Feature Dimension, Non-Rigid Geometric Transformations and Fusion

The original definition of summation invariants produces a single value for an image/object. The extension to 'semi-local' summation invariants [1] increased the size of the feature space to provide better discrimination capabilities. However, this requires specifying a window size for the local computation. Optimizing this window size will further improve performance. This technique is also useful for dealing with occlusions, since it makes the feature more local in nature and allows comparisons of partial objects (assuming the correspondence is known).

Another factor is non-rigid geometric transformations, which are caused by deformations of the object being imaged. One example of this factor is the variation caused by changes in expression on human faces. These transformations will obviously affect the invariance of any summation invariant derived under transformation groups, such as Euclidean, that don't include non-rigid transformations. Invariants derived under affine or projective groups would be expected to have a better chance of retaining their invariance.

One way to alleviate this problem is to compute invariant features on a local or semi-local basis over a smaller region of the entire object. The intuition is that the non-rigidness effect will be less prominent over a small region of a large object. As such, the rigid transformation assumption would be more likely applicable in such a situation. On the other hand, to compute summation invariant over smaller regions would require better correspondence between the original and transformed data points. In face recognition applications, improvement was achieved by carefully selecting sub-regions that were more invariant to expression changes, like the nose region [4, 5]. The invariants were then computed on the sub-region(s) instead of on the whole image.

Finally, using either multiple sub-regions or multiple invariants and some method of fusion to obtain the final result was shown useful in improving the performance of the summation invariants [5, 7] in face recognition.

5. SIMULATIONS

The issue of spatial quantization after geometric transformation will be examined more closely here with simulations to illustrate the issues involved. An example curve is shown in figure 1 along with the curve resulting from a 36˚ rotation and translation of the original curve (o's). The curve labeled 'camera quantization' (x's) is produced by re-sampling the transformed curve on the same grid spacing that the original curve had, simulating the image of the rotated curve obtained by a camera.

This 'camera' is a 1D camera that produces an image that is a line, where the x-value is the position along the line and the y-value is the distance of the object from the camera at that x-value. The camera samples the object at regular intervals, producing a quantization of the object curve. When the object rotates, it is readily apparent that the image of the object changes in several ways. First, the rotated image is longer in this example than the original. This affect is caused by the projection of the object onto the x-

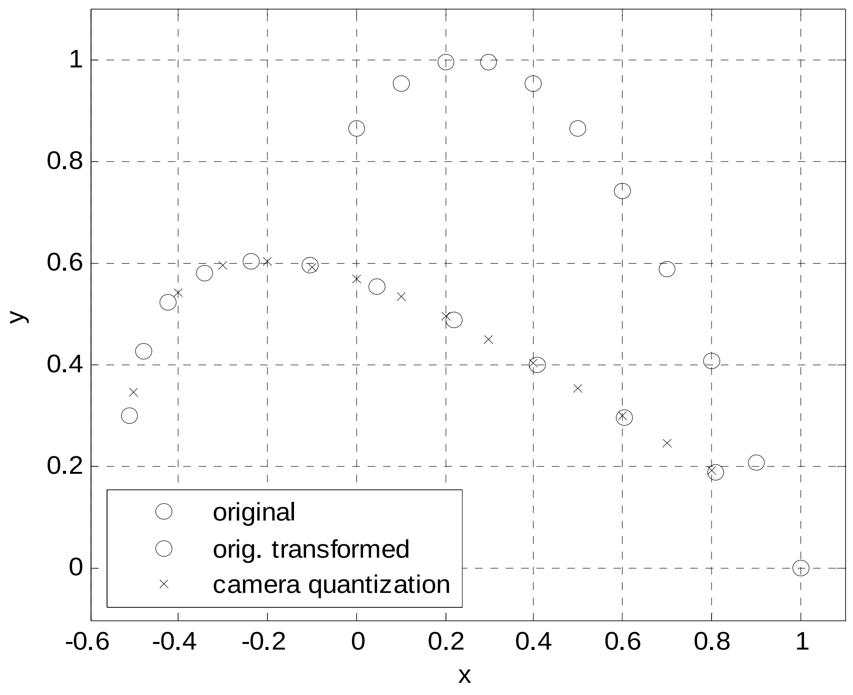

Figure 1 – Quantization effect after transformation.

axis (the camera image 'plane') being a different size. This gives the quantized curve more data points, 14 instead of 11, than the original. This difference will increase the error in the values of summation invariants for this curve. Although it is likely that the transformed image will have a different number of points, it is not guaranteed to be the case in general.

Another difference in the rotated image is that the points on the curve where the image samples are located changes. The increase in number of points obviously means that the points are from different locations on the object, but even if the total number of points was the same after transformation, the points could still come from slightly different locations. For example, on the right half of the original curve, the camera sees a relatively steep slope on the object, resulting in relatively few points for that portion, while the left part has a gentler slope, resulting in relatively more points. After transformation, the situation is reversed – the right part now has a gentler slope and hence relatively more points while the left part has a steeper slope, with fewer points than before. While this example has more total points, it is possible to have a case like this that would have the same number of points.

This effect of a change in slope of the curve after transformation on the location of the sample points in the image of the curve is clearly seen on the left-most portion of the curve. The first camera point is located between the first and second transformed points and the next camera point is between the third and fourth transformed points. These changes in location of the data points on the curve will change the calculated values of the summation invariants.

Finally, after transformation and quantization, the end points may not be the exact same points due to the actual end points lying between the points of the quantization interval. This can be seen on the left side of the transformed curve where the two end points (original transformed and camera quantized) are different because the transformed end point falls between the quantization interval points.

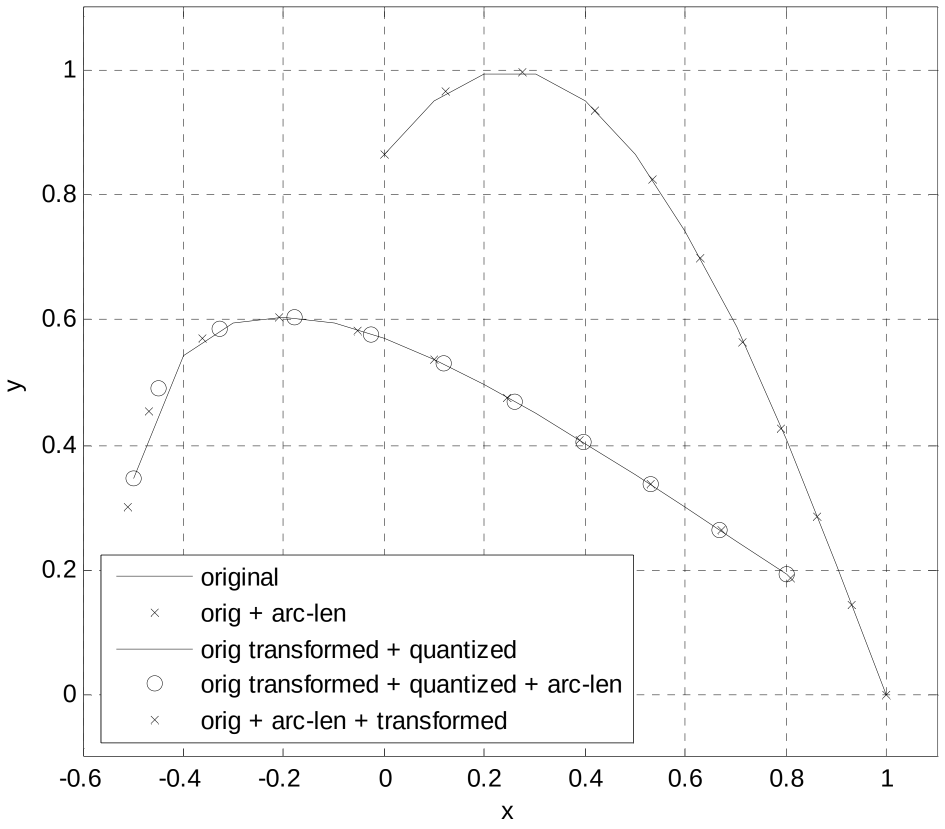

Figure 2 – Arc-length re-sampling and quantization effect after transformation.

One technique that will reduce the error caused by these quantization effects is arc-length re-sampling. This is a parameterization of the curve by arc-length and then re-sampling in the parameterized domain. The original curve is re-sampled at points that are equidistant along the curve. When a new image (of the object transformed) is obtained, it is also re-sampled at equidistant points along the curve, using the same number of samples as the arc-length re-sampled original. This technique is illustrated in figure 2 using the same curves as in figure 1. The original curve is indicated by the piecewise linear curve and the equal arc-length re-sampled points are shown as x's. The transformed, quantized points are indicated by a piecewise linear curve and the original points transformed, quantized and equal arc-length re-sampled are shown by o's, while the original arc-length re-sampled points after transformation are shown as x's. The data points using this method are clearly more accurate than the results without it in figure 1. This method will cancel the effects noted above where the number of sample points from a given portion of the curve varies with transformation. The error can be reduced, but not eliminated by this technique since the end points may be at slightly different locations on the curve, which will affect all of the other re-sampled values. This is the case in the example of figure 2.

The example curve of figure 1 was transformed over rotation values ranging between zero and 36° (with 51 samples instead of 11 for improved performance). At each angle of rotation, the Euclidean summation invariants derived here were calculated for the transformed curve with camera quantization and the transformed, camera

Table 2 – Quantization errors after transformation – with and without arc-length re-sampling.

Original, then camera quantization - error (%) (camera quant. Vs. original)

Theta, degrees	3.6	7.2	10.8	14.4	18	21.6	25.2	28.8	32.4	36	Max	Min
Hx	-6.61	2.22	12.48	6.42	13.45	7.40	17.59	6.78	23.49	12.71	23.49	2.22
Hy	1.71	13.86	27.27	31.80	39.58	45.64	55.68	55.25	68.08	68.85	68.85	13.86
Hxx	-14.60	-3.32	10.97	-2.42	7.19	-5.50	9.72	-9.57	17.33	-2.28	17.33	-9.57
Hxy	-7.26	6.14	22.35	17.48	27.27	23.45	38.68	25.83	50.83	38.59	50.83	6.14
Hyy	-2.72	12.23	29.83	33.15	41.45	48.36	61.31	57.68	75.99	75.31	75.99	12.23

Original , transform, then Camera quantized and arc-length resample vs. original and arc-length - error (%)

Theta, degrees	3.6	7.2	10.8	14.4	18	21.6	25.2	28.8	32.4	36	Max	Min
Hx	-8.16	-5.16	-1.27	-7.79	-5.21	-11.26	-6.25	-14.39	-4.68	-12.47	-1.27	-14.39
Hy	-4.54	-2.72	-0.43	-1.12	-1.79	-0.76	-0.22	-1.47	-0.23	-0.18	-0.18	-2.72
Hxx	-15.86	-10.19	-2.56	-15.21	-10.30	-21.58	-12.31	-27.10	-9.28	-23.74	-2.56	-27.10
Hxy	-11.96	-7.51	-1.65	-8.69	-6.74	-11.79	-6.40	-15.42	-4.84	-12.49	-1.65	-15.42
Hyy	-9.59	-5.86	-1.00	-3.28	-4.13	-3.13	-1.36	-4.91	-1.13	-2.24	-1.00	-5.86

quantized with arc-length re-sampling curve. These values were compared to the Euclidean summation invariants calculated on the original curve points and to the original curve points after arc-length re-sampling, respectively. As expected, the two curves displayed errors in the calculations. The errors are shown in table 2. The camera re-sampled curve had the worst errors, with values ranging as high as 76%. The arc-length re-sampled curve had better results, with values ranging only as high as 27% and with two of the invariants ($\eta_{0,1}$ and $\eta_{0,2}$) having errors over the given rotation range of magnitude less than 3% and 6%. The arc-length re-sample technique provides better results than using the raw camera image. In addition, within a family of summation invariants, some features show less susceptibility to the quantization effects.

6. PERFORMANCE COMPARISONS

Invariance does not guarantee distinctiveness, so the discrimination ability of summation invariants must be evaluated experimentally. The previous works did not use a consistent set of experiments, so a fair comparison of the discrimination performance of the summation invariants cannot be made using the previously published results. A proposal for a standard experiment to use for comparison purposes is given here.

6.1. Standard experiment

The application used to make this assessment is face recognition. The original work on summation invariants utilized the Face Recognition Grand Challenge (FRGC) database [24]. Most of the early work used version 1.0. Version 2.0 is more extensive in size and includes variation in expression, and is used in the standard experiment.

The FRGC was designed to provide the incentive and means to achieve a significant advance in the state-of-the-art in face recognition [24]. It consists of an extensive database of 50,000 face images, a set of six challenge experiments and a standardized experimental against the False Acceptance Rate (FAR). The standard comparison value is the TAR at a FAR of 0.001 (0.1%).

The experiment consists roughly of the following steps: The raw images are normalized to get them in a consistent pose and size, and are cropped to a roughly elliptical shape (see figure 3, also figure 4 where the image is rotated to more easily see the features). The comparison algorithm is then executed which calculates the features for

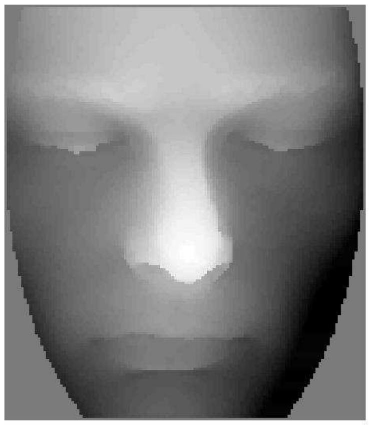
Figure 3 – Normalized face range image.

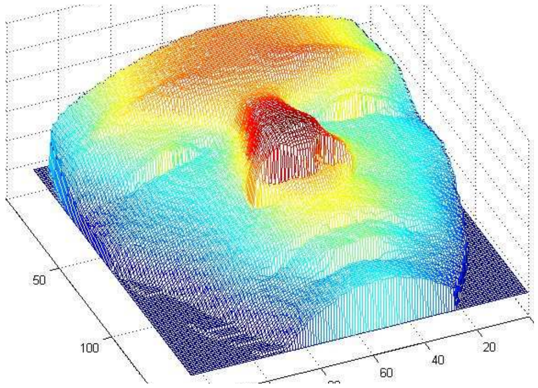

Figure 4 – Normalized face range image rotated.

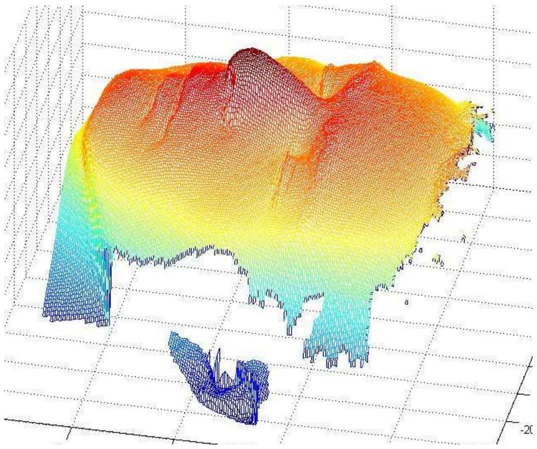
Figure 5 – Raw face range image from FRGC2.0 database.

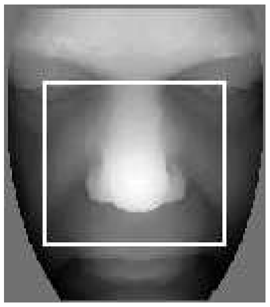
Figure 6 – Region 5 from [7].

each of the normalized face images. The training set is used to train the classifier, which is then used to make a decision on each face in the validation set. This comparison is done for each image against all others, yielding a 4007x4007 similarity matrix containing values for the relative similarity of each face with each of the others.

The experimental framework used in the FRGC is the Biometric Experimentation Environment (BEE). It uses various scripts and programs (Java, XML, Perl, C) to define and run the experiments. The FRGC specifies a baseline implementation for each experiment that researchers can modify to implement their own methods. The baseline uses Principal Component Analysis (PCA) to derive the features for comparison between images. The experimental framework and well-defined experiments provide a mechanism for easy and fair comparison of results from different researchers. The size of the database facilitates obtaining statistically meaningful results.

The experiment used here is experiment 3s, which uses 3D shape images. This portion of the database contains 943 training images and 4007 validation images. The images were taken over the course of two academic years. The 3D images were taken under controlled lighting conditions with a Minolta Vivid 900/910 series sensor. The resulting images are 640x480 pixel range images. The variation due to illumination is eliminated for this type of image, but variation due to pose is still an issue. A sample of one of the images is shown in figure 5 below (re-sampled on an equal grid and rotated for better visualization).

Results for the experiments are reported via Receiver Operating Curve (ROC) graphs. There are three categories, I, II and III, corresponding to images taken in the same semester, within the same academic year, and within eighteen months of each other, respectively. These graphs plot the verification rate, or True Acceptance Rate (TAR), others. This data is used to generate the ROC graphs. A set of similarity mask matrices determines which values in the similarity matrix are valid for each ROC graph (I, II, III).

The set of experiments run in this project utilize a sub-region of each face (region 5 as defined in [7]) that is 81x81 pixels (see figure 6). This sub-region was selected to reduce the degree of variation due to expression. The goal in this standard experiment is not to get the highest performance possible, but rather to make a relative comparison amongst the invariants. Therefore, additional techniques to get higher performance (like fusion of results from multiple regions) are not attempted in the standard experiment.

Taken at face value, the summation invariant will produce a single value for each face image. This would be great for invariance, but poor for discrimination purposes. To enhance the discrimination capability, an extension to the summation invariant concept was developed, making it 'semi-local' [1]. This extension consists of calculating the summation invariant at each pixel over a window centered at that pixel. This results in an expansion of the feature space and an improvement in the discrimination capability of the summation invariant.

The images are 3D surfaces, but some of the invariant features are defined for 2D data. This is handled by calculating the 2D feature over a horizontal 1xN sized window centered on the current pixel, then over a vertical Nx1 sized window centered on the current pixel. An additional processing step is the uniform re-sampling with respect to arc-length in the calculations done at each pixel. The window size used is 21. For 3D summation invariants, the window size is 17 and no arc-length re-sampling is performed.

To minimize the computation burden of the similarity calculation, principal component analysis (PCA) is performed on the resulting feature space to reduce its size. The basis vectors for the PCA step are calculated from the training set of images. The similarity metric used is the Mahalanobis cosine.

The calculation of the summation invariant features takes place in a C routine in the file 'uwCommonInvariants.c' that is part of the BEE setup. Implementing the new family of invariants requires modifying that file to include the new calculations as additional functions and then modifying the call to the invariant calculation function for each run so the appropriate invariant is used.

Comparison of the results is done using the TAR value at a FAR of 0.001 for ROC-III. A broader comparison can also be done using the full ROC graphs for ROC-III, or using ROC-II or ROC-I.

6.2. Results

The summation invariants derived so far (listed in table 1) were each used as the feature for the experiment outlined above. The results from those experiments are summarized in Table 3 below. The results shown are for ROC-III. The ROC graphs for each set are shown in figures 7 - 11.

The results for the new family of invariants derived here are very similar to those for the original (e.g., the values for $\eta_{1,1}$ match, the original, $\eta_{1,0}$ matches the new $\eta_{0,1}$, etc.). The only one that doesn't follow the pattern is the new $\eta_{0,2}$, which is much lower than expected. The results for the 2D affine $\eta_{0,2}$ are similar.

The results for 3D Euclidean are worse than those for the 2D invariants. One factor that could account for this is the fact that the 2D invariant calculation results in two values for each point

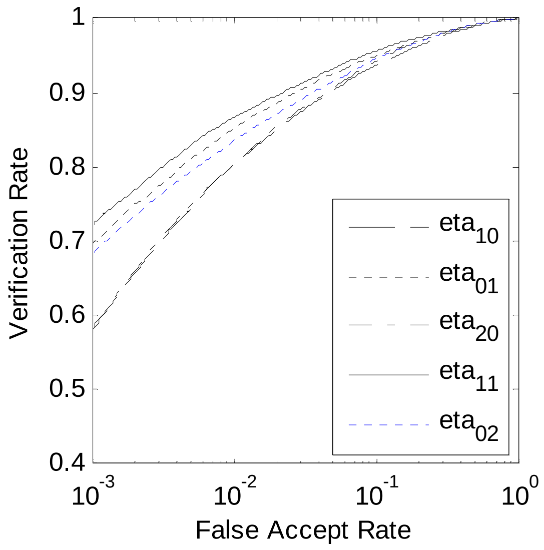

Figure 7 – ROC III graph for 2D Euclidean summation invariants with y(N) = 0.

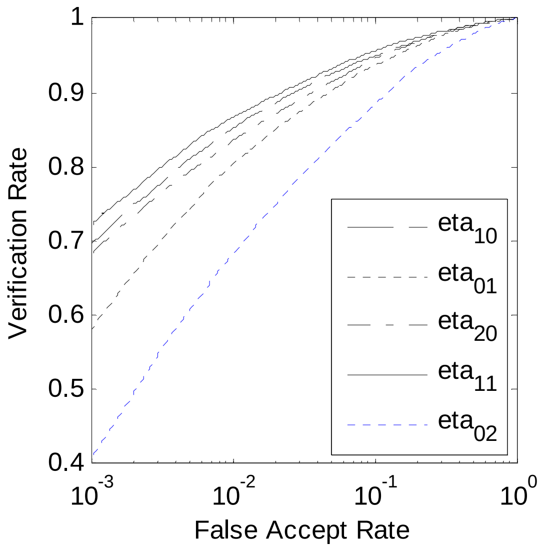

Figure 8 – ROC III graph for 2D Euclidean summation invariants with x(N) = 0.

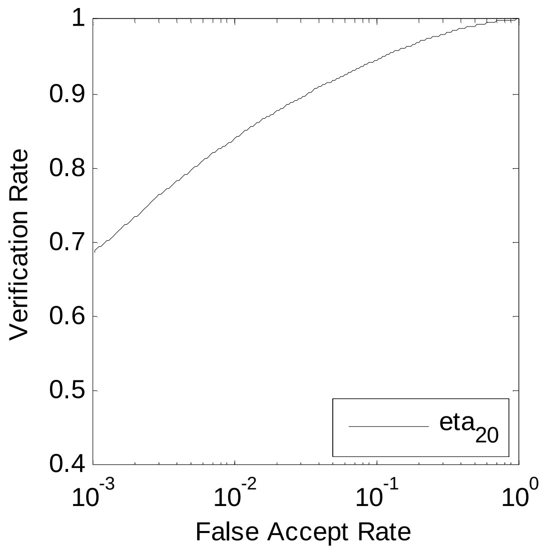

Figure 9 – ROC III graph for 2D affine summation invariants.

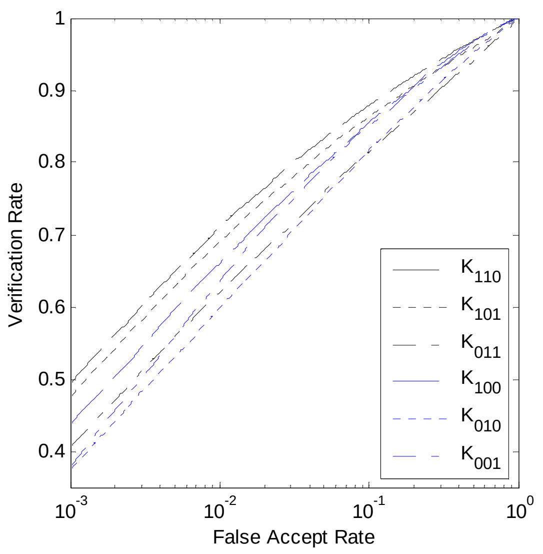

Figure 10 – ROC III graph for 3D Euclidean summation invariants.

Table 3: Results – Comparison of discrimination capabilities for all summation invariants using face recognition application.

	η_{10}	η_{01}	η_{20}	η_{11}	η_{02}	
2D, Eucl., yN=0	0.5772	0.6821	0.5810	0.7192	0.6811	
2D, Eucl., xN=0	0.6930	0.5772	0.6812	0.7192	0.4081	
2D, Affine			0.6842			
	K_{001}	K_{010}	K_{100}	K_{110}	K_{101}	K_{011}
3D, Eucl.	0.4370	0.3746	0.3779	0.4927	0.4749	0.4058
	K_{200}, numer	K_{200},denom				
3D, Affine	0.1548	0.2516				

(one horizontal window and one vertical window) but the 3D invariant calculation only provides one value for each point. Another possible reason is the use of arc-length re-sampling for the 2D invariants, but none for the 3D case. The 3D affine invariant performed worse than the 3D Euclidean. The numerator and denominator of the 3D affine summation invariant were shown to be relative invariants, so experiments were also run using each of those. No explanation is known at this time for the dismal performance of the full invariant compared to the numerator and denominator performance.

7. CONCLUSIONS

The method of moving frames, a powerful tool for deriving geometric invariant features, was described. The summation invariant, a recently developed geometric invariant with a demonstrated effectiveness in pattern classification applications, was discussed in the context of some of the other major geometric invariant features. In their semi-local form, these features have good discrimination capability, but without the noise sensitivity of the classical differential-based invariants. The summation invariants derived so far were summarized and a systematic approach for deriving new features using the method of moving frames was outlined. A detailed example derivation was given to illustrate the derivation procedure and a naming convention was proposed. Implementation issues were examined, including techniques for enhancing their effectiveness and factors that reduce their effectiveness in practical imaging applications. A standard method for comparing the discrimination ability of the different summation invariants was given using face recognition as the application.

Further work needs to be done to compare summation invariants with other invariant features to determine what tasks each is best suited for. Another area in need of additional exploration is the effects of re-sampling on the calculation of other invariants.

8. REFERENCES

[1] W. -Y. Lin, N. Boston, and Y. H. Hu, “Summation invariant and its application to shape recognition,” in *Proceedings of ICASSP*, 2005, vol. V, pp. 205–208.

[2] W. -Y. Lin, N. Boston, and Y. H. Hu, “Summation Invariant Features for 3D Face Recognition,” Proc. IEEE Workshop on Multimedia Signal Processing, Shanghai, China, Oct. 30-Nov. 2, 2005.

[3] W. -Y. Lin, K.-C. Wong, Y. H. Hu and N. Boston, “Face Recognition Using 3D Summation Invariant Features,” *Multimedia and Expo, 2006 IEEE International Conference on* , vol., no.pp.1733-1736, July 2006.

[4] W. -Y. Lin, K.-C. Wong, N. Boston, and Y. H. Hu, “3D Human Face Recognition Using Summation Invariants,” in *Proceedings of ICASSP*, 2006, vol. II, pp. 341-344.

[5] W. -Y. Lin, K.-C. Wong, N. Boston, and Y. H. Hu, “Fusion of Summation Invariants in 3D Human Face Recognition,” in *Proceedings of CVPR*, 2006, vol. II, pp. 1369 - 1376.

[6] W. -Y. Lin, “Robust Geometrically Invariant Features for 2D Shape Matching and 3D Face Recognition,” Ph.D. dissertation, Dept. Elect. and Comp. Eng., University of Wisconsin, Madison, WI, Aug. 2006.

[7] K.-C. Wong, W.-Y. Lin, Y. H. Hu, N. Boston, and X. Zhang, "Optimal Linear Combination of Facial Regions for Improving Identification Performance," to appear in IEEE Trans. on Systems, Man and

Cybernetics, Part B: Cybernetics, vol. ?, pp. ?-?.

[8] M. K. Hu, "Visual pattern recognition by moment invariants," *IRE – Trans. on Information Theory*, vol. IT-8, no. 2, pp. 179–187, 1962.

[9] T. Reiss, "The revised fundamental theorem of moment invariants," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 8, pp. 830 – 4, 1991.

[10] H. Dirilten and T.G. Newman, "Pattern matching under affine transformations," *IEEE Trans. Comput.*, vol. C-26, pp. 314 – 317, Mar. 1977.

[11] S. X. Liao and M. Pawlak, "On image analysis by moments," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 18, no. 3, pp. 254 – 266, 1996.

[12] K. Arbter, W. Snyder, H. Burkhardt, and G. Hirzinger, "Application of affine-invariant fourier descriptors to recognition of 3-d objects," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 12, no. 7, pp. 640 – 7, 1990.

[13] M. Khalil and M. Bayoumi, "A dyadic wavelet affine invariant function for 2d shape recognition," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 23, no. 10, pp. 1152 – 1164, 2001.

[14] Y. Wang, C. Chua, and Y. Ho, "Facial feature detection and face recognition from 2D and 3D images", *Pattern Recognition Letters*, 23:1191–1202, 2002.

[15] T. Moons, E. J. Pauwels, L. J. V. Gool, and A. Oosterlinck, "Foundations of semi-differential invariants," *Intl. Journal of Computer Vision*, vol. 14, no. 1, pp. 25–47, 1995.

[16] E. Calabi, P. J. Olver, C. Shakiban, A. Tannenbaum, and S. Haker, "Differential and numerically invariant signature curves applied to object recognition," *Intl. Journal of Computer Vision*, vol. 26, no. 2, pp. 107–135, 1998.

[17] J. Y. Cartoux, J. T. Lapreste, and M. Richetin, "Face authentification or recognition by profile extraction from range images," in *Proc. of Workshop on Interpretation of 3D Scenes*, 1989, pp. 194–9.

[18] J. Sato and R. Cipolla, "Affine integral invariants and matching of curves," in *Proc. of 13th Intl. Conf. on Pattern Recognition*, vol. 1, 1996, pp. 915–19.

[19] C. E. Hann and M. S. Hickman, "Projective curvature and integral invariants," *Acta Applicandae Mathematicae*, vol. 74, no. 2, pp. 177–193, 2002.

[20] S. Manay, B.-W. Hong, A. Yezzi, and S. Soatto, "Integral invariant signatures," *ECCV 2004. Proc. (Lecture Notes in Comput. Sci. Vol.3024)*, vol. 4, pp. 87 – 99, 2004.

[21] E. Cartan, "La methode du repere mobile, la theorie des groupes continus, et les espaces generalises," *Exposes de Geometrie*, no. 5, 1935.

[22] M. Fels and P. J. Olver, "Moving coframes: I. a practical algorithm," *Acta Applicandae Mathematicae*, vol. 51, no. 2, pp. 161 – 213, 1998.

[23] ——, "Moving coframes: II. regularization and theoretical foundations," *Acta Applicandae Mathematicae*, vol. 55, no. 2, pp. 127 – 208, 1999.

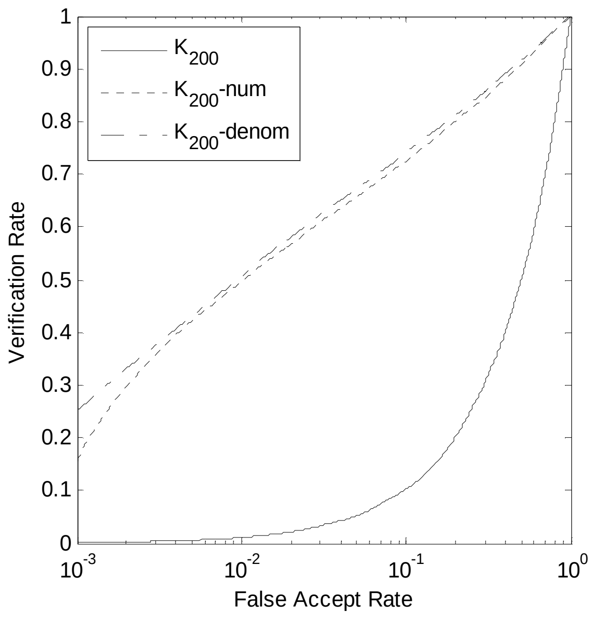

Figure 11 – ROC III graph for 3D affine summation invariants.

[24] P. J. Phillips, P. J. Flynn, T. Scruggs, K. W. Bowyer, J. Chang, K. Hoffman, J. Marques, J. Min, and W. Worek, "Overview of the face recognition grand challenge," in *Proceedings of CVPR,* 2005, vol. 1, pp. 947–54.

Appendix A

Table of Summation Invariant formulas

Euclidean, 2D, (x1,y1,yN) = (0,0,0)
$\eta_{1,0}^{E} = \overline{P}_{1,0} = P_{1,0} \cdot (x[N] - x[1]) + P_{0,1} \cdot (y[N] - y[1]) + N \cdot x[1] \cdot (x[1] - x[N]) + N \cdot y[1] \cdot (x[1] - y[N])$
$\eta_{0,1}^{E} = \overline{P}_{0,1} = P_{1,0} \cdot (y[N] - y[1]) + P_{0,1} \cdot (x[1] - x[N]) + N \cdot (x[N] \cdot y[1] - x[1] \cdot y[N])$
$\begin{aligned} \eta_{2,0}^{E} = \overline{P}_{2,0} = & -2 \cdot P_{1,0} \cdot \left((x[1] - x[N])(x[1]^2 - x[N] \cdot x[1] - y[1] \cdot y[N] + y[1]^2)\right) \\ & -2 \cdot P_{0,1} \cdot \left((y[1] - y[N])(y[1]^2 - y[1] \cdot y[N] - x[N] \cdot x[1] + x[1]^2)\right) \\ & + P_{2,0} (x[1] - x[N])^2 + P_{0,2} \cdot (y[1] - y[N])^2 \\ & + 2 \cdot P_{1,1} \cdot (y[1] - y[N]) \cdot (x[1] - x[N]) + N \cdot ((x[1](x[1] - x[N])) + (y[1](y[1] - y[N])))^2 \end{aligned}$
$\begin{aligned} \eta_{1,1}^{E} = \overline{P}_{1,1} = P_{1,0} & (y[1]^3 + 2 \cdot x[1] \cdot x[N] \cdot y[N] - 2 \cdot y[N] \cdot x[1]^2 + x[1]^2 \cdot y[1] \\ & - 2 \cdot y[1]^2 \cdot y[N] + y[N]^2 \cdot y[1] - x[N]^2 \cdot y[1]) \\ & + P_{0,1} (x[1]^3 + 2 \cdot y[1] \cdot y[N] \cdot x[N] - 2 \cdot x[N] \cdot y[1]^2 + y[1]^2 \cdot x[1] \\ & - 2 \cdot x[1]^2 \cdot x[N] + x[N]^2 \cdot x[1] - y[N]^2 \cdot x[1]) \\ & + (P_{0,2} - P_{2,0})(y[1] - y[N])(x[1] - x[N]) + P_{1,1} \cdot \left((x[1] - x[N])^2 - (y[1] - y[N])^2\right) \\ & + N \cdot (x[N] \cdot y[1] - x[1] \cdot y[N])(x[1] \cdot (x[N] - x[1]) + y[1] \cdot (y[N] - y[1])) \end{aligned}$
$\begin{aligned} \eta_{0,2}^{E} = \overline{P}_{0,2} = & \, 2 \cdot (y[1] \cdot x[N] - x[1] \cdot y[N]) \cdot (P_{1,0} \cdot (y[N] - y[1]) - P_{0,1} \cdot (x[N] - x[1])) \\ & + P_{2,0} \cdot (y[1] - y[N])^2 + P_{0,2} \cdot (x[1] - x[N])^2 \\ & - 2 \cdot P_{1,1} \cdot (x[1] - x[N])(y[1] - y[N]) + N \cdot (x[1] \cdot y[N] - x[N] \cdot y[1])^2 \end{aligned}$
Euclidean, 2D, (x1,y1,xN) = (0,0,0)
$\eta_{1,0}^{E} = \overline{P}_{1,0} = P_{1,0} \cdot (y[1] - y[N]) + P_{0,1} \cdot (x[N] - x[1]) + N \cdot (x[1] \cdot y[N] - x[N] \cdot y[1])$
$\eta_{0,1}^{E} = \overline{P}_{0,1} = P_{1,0} \cdot (x[1] - x[N]) + P_{0,1} \cdot (y[1] - y[N]) + N \cdot x[1] \cdot (x[N] - x[1]) + N \cdot y[1] \cdot (y[N] - y[1])$
$\begin{aligned} \eta_{2,0}^{E} = \overline{P}_{2,0} = & \, 2 \cdot P_{1,0} \cdot ((y[1] - y[N])(y[N] \cdot x[1] - y[1] \cdot x[N])) \\ & + 2 \cdot P_{0,1} \cdot ((x[1] - x[N])(y[1] \cdot x[N] - y[N] \cdot x[1])) \\ & + P_{2,0} (y[1] - y[N])^2 + P_{0,2} \cdot (x[1] - x[N])^2 \\ & + 2 \cdot P_{1,1} \cdot (y[1] - y[N]) \cdot (x[N] - x[1]) + N \cdot (y[1] \cdot x[N] - y[N] \cdot x[1])^2 \end{aligned}$
$\begin{aligned} \eta_{1,1}^{E} = \overline{P}_{1,1} = P_{1,0} & (y[1]\left(x^2[N] - y^2[N] - x^2[1] - y^2[1] + 2 \cdot y[1] \cdot y[N]\right) \\ & + 2 \cdot x[1] \cdot y[N] \cdot (x[1] - x[N])) \\ & + P_{0,1} (x[1]\left(y^2[1] + x^2[1] - y^2[N] + x^2[N] - 2 \cdot x[1] \cdot x[N]\right) \\ & + 2 \cdot y[1] \cdot x[N] \cdot (y[N] - y[1])) \\ & + (P_{2,0} - P_{0,2})(y[1] - y[N])(x[1] - x[N]) + P_{1,1} \cdot \left((y[N] - y[1])^2 - (x[N] - x[1])^2\right) \\ & + N \cdot (y[1] \cdot x[N] \cdot \left(y^2[1] - x[1] x[N]\right) + x^2[1] \cdot y[N] \cdot (x[N] - x[1]) \\ & \quad + x[1] \cdot y[1] \cdot \left(y^2[N] + x[1] \cdot x[N]\right) - y^2[1] \cdot y[N] \cdot (x[N] + x[1])) \end{aligned}$

$$\begin{aligned}\eta^E_{0,2} = \overline{P}_{0,2} = {} & 2\cdot P_{1,0}\cdot\left(x[N]-x[1]\right)\left(y^2[1]-y[1]\cdot y[N]-x[1](x[N]-x[1])\right)\\ & +2\cdot P_{0,1}\cdot\left(y[N]-y[1]\right)\left(x^2[1]-x[1]\cdot x[N]-y[1](y[N]-y[1])\right)\\ & +P_{2,0}\cdot\left(x[N]-x[1]\right)^2+P_{0,2}\cdot\left(y[N]-y[1]\right)^2\\ & +2\cdot P_{1,1}\cdot\left(y[1]-y[N]\right)\left(x[1]-x[N]\right)\\ & +N\cdot(x^2[1]\cdot x^2[N]-2\cdot y[1]\cdot y[N]\cdot x^2[1]+2\cdot y^2[1]\cdot x^2[1]\\ & \quad +2\cdot y[1]\cdot y[N]\cdot x[1]\cdot x[N]-2\cdot y^2[1]\cdot x[1]\cdot x[N]\\ & \quad -2\cdot x^3[1]\cdot x[N]+y^2[1]\cdot y^2[N]+y^4[1]+x^4[1]-2\cdot y^3[1]\cdot y[N])\end{aligned}$$

Affine, 2D, (x1,y1, xN, yN, P10, P01) = (0,0,1,1,0,0)
(Note : both the numerator and denominator are relative invariants.)

$$\begin{aligned}\eta^A_{2,0} = \{ & P_{2,0}\cdot\left(N\cdot y[1]-P_{0,1}\right)^2+P_{0,2}\cdot\left(N\cdot x[1]-P_{1,0}\right)^2\\ & -2\cdot P_{1,1}\cdot\left(N\cdot x[1]-P_{1,0}\right)\cdot\left(N\cdot y[1]-P_{0,1}\right)-N\cdot\left(y[1]\cdot P_{1,0}-x[1]\cdot P_{0,1}\right)^2\}\\ & /\left\{\left(N\cdot\left(x[N]\cdot y[1]-x[1]\cdot y[N]\right)+P_{1,0}\cdot\left(y[N]-y[1]\right)-P_{0,1}\cdot\left(x[N]-x[1]\right)\right)^2\right\}\end{aligned}$$

Euclidean, 3D, (x11,y11,z11, yM1, zM1, z1N) = (0,0,0,0,0,0)

$$\begin{bmatrix}\overline{x}\\ \overline{y}\\ \overline{z}\end{bmatrix}=R_3R_2R_3\left(\begin{bmatrix}x\\ y\\ z\end{bmatrix}-T\right)$$, where $T=\begin{bmatrix}x[1,1]\\ y[1,1]\\ z[1,1]\end{bmatrix}$, then let $A=\begin{bmatrix}x[M,1]\\ y[M,1]\\ z[M,1]\end{bmatrix}$ and express A in spherical coordinates

(r_a, θ_a, φ_a) , then $R_1=\begin{bmatrix}\cos\phi_a & \sin\phi_a & 0\\ -\sin\phi_a & \cos\phi_a & 0\\ 0 & 0 & 1\end{bmatrix}$, and $R_2=\begin{bmatrix}\sin\theta_a & 0 & \cos\theta_a\\ 0 & 1 & 0\\ -\cos\theta_a & 0 & \sin\theta_a\end{bmatrix}$, then,

let $B=\begin{bmatrix}B_x\\ B_y\\ B_z\end{bmatrix}=R_2R_1\left(\begin{bmatrix}x[1,N]\\ y[1,N]\\ z[1,N]\end{bmatrix}-T\right)$, and define $C=\begin{bmatrix}B_y\\ B_z\\ B_x\end{bmatrix}$, then express C in spherical coordinates (r_c, θ_c, φ_c),

and let $R_3=\begin{bmatrix}1 & 0 & 0\\ 0 & \cos\phi_c & \sin\phi_c\\ 0 & -\sin\phi_c & \cos\phi_c\end{bmatrix}$. Then R_1, R_2, R_3 and T form a moving frame – apply to potentials to get invariants as

shown below.

$\kappa^E_{1,0,0}=\sum_{n=1}^{N}\overline{x}$, where $\overline{x}$ is calculated by the transformation given above.

$\kappa^E_{0,1,0}=\sum_{n=1}^{N}\overline{y}$, where $\overline{y}$ is calculated by the transformation given above.

$\kappa^E_{0,0,1}=\sum_{n=1}^{N}\overline{z}$, where $\overline{z}$ is calculated by the transformation given above.

Affine, 3D, (x11,y11,z11,xM1, yM1, zM1, x1N, y1N, z1N,P100,P010,P001) = (0,0,0,1,0,0,0,1,0,0,0,0)

$$
\begin{aligned}
\kappa^{A}_{2,0,0} &= \{Q_{0,0,2}(MN(x_{01}y_{00} - x_{00}y_{01}) + Q_{100}(y_{01} - y_{00}) + Q_{010}(x_{00} - x_{01}))^2 \\
&- 2\cdot Q_{011}(MN(x_{01}y_{00} - x_{00}y_{01}) + Q_{100}(y_{01} - y_{00}) + Q_{010}(x_{00} - x_{01})) \\
&(MN(x_{01}z_{00} - x_{00}z_{01}) + Q_{100}(z_{01} - z_{00}) + Q_{001}(x_{00} - x_{01})) \\
&+ Q_{020}(MN(x_{01}z_{00} - x_{00}z_{01}) + Q_{100}(z_{01} - z_{00}) + Q_{001}(x_{00} - x_{01}))^2 \\
&- 2Q_{110}(MN(x_{01}z_{00} - x_{00}z_{01}) + Q_{100}(z_{01} - z_{00}) + Q_{001}(x_{00} - x_{01})) \\
&(MN(y_{01}z_{00} - y_{00}z_{01}) + Q_{010}(z_{01} - z_{00}) + Q_{001}(y_{00} - y_{01})) \\
&+ Q_{200}(MN(y_{01}z_{00} - y_{00}z_{01}) + Q_{010}(z_{01} - z_{00}) + Q_{001}(y_{00} - y_{01}))^2 \\
&- 2Q_{101}(MN(x_{01}y_{00} - x_{00}y_{01}) + Q_{100}(y_{01} - y_{00}) + Q_{010}(x_{00} - x_{01})) \\
&(MN(y_{00}z_{01} - y_{01}z_{00}) + Q_{010}(z_{00} - z_{01}) + Q_{001}(y_{01} - y_{00})) \\
&- MN(Q_{100}(y_{01}z_{00} - y_{00}z_{01}) + Q_{010}(x_{00}z_{01} - x_{01}z_{00}) + Q_{001}(x_{01}y_{00} - x_{00}y_{01}))^2\} \\
&/\{(MN(x_{00}(y_{10}z_{01} - y_{01}z_{10}) + x_{10}(y_{01}z_{00} - y_{00}z_{01}) + x_{01}(y_{00}z_{10} - y_{10}z_{00}))) \\
&+ Q_{100}(y_{00}(z_{01} - z_{10}) + y_{10}(z_{00} - z_{01}) + y_{01}(z_{10} - z_{00})) \\
&+ Q_{010}(x_{00}(z_{10} - z_{01}) + x_{10}(z_{01} - z_{00}) + x_{01}(z_{00} - z_{10})) \\
&+ Q_{001}(x_{00}(y_{01} - y_{10}) + x_{10}(y_{00} - y_{01}) + x_{01}(y_{10} - y_{00}))^2\}
\end{aligned}
$$

where, x_{00} = x[1, 1]; y_{00} = y[1, 1]; z_{00} = z[1, 1]; x_{10} = x[M, 1]; y_{10} = y[M, 1]; z_{10} =z[M, 1]; x_{01} = x[1, N]; y_{01} = y[1, N]; z_{01} = z[1, N].

Appendix B

Maple file – Derivation of moving frame and Summation Invariants

```
> #
# Curve Euclidean invariant eta_y
# g o (x0,y0,x1) = (0,0,0)
#
eqn1:=cos(theta)*x0-sin(theta)*y0+a=0:
eqn2:=sin(theta)*x0+cos(theta)*y0+b=0:
eqn3:=cos(theta)*x1-sin(theta)*y1+a=0:
solve({eqn1,eqn2,eqn3},{theta,a,b}):
eval(sin(theta)*Px+cos(theta)*Py+b*N,%):simplify(%);
```

$$-\frac{1}{(-y1+y0)\sqrt{\frac{y0^2-2\,y0\,y1+y1^2+x0^2-2\,x0\,x1+x1^2}{(-y1+y0)^2}}}(-Px\,x0+Px\,x1+Py\,y1-Py\,y0+N\,x0^2-N\,x0\,x1-N\,y0\,y1+N\,y0^2)$$

```
> #
# Curve Euclidean invariant eta_x
# g o (x0,y0,x1) = (0,0,0)
#
eqn1:=cos(theta)*x0-sin(theta)*y0+a=0:
eqn2:=sin(theta)*x0+cos(theta)*y0+b=0:
eqn3:=cos(theta)*x1-sin(theta)*y1+a=0:
solve({eqn1,eqn2,eqn3},{theta,a,b}):
eval(cos(theta)*Px-sin(theta)*Py+a*N,%):
simplify(%);
```

$$-\frac{Px\,y1-Px\,y0+Py\,x0-Py\,x1-N\,x0\,y1+N\,y0\,x1}{(-y1+y0)\sqrt{\frac{y0^2-2\,y0\,y1+y1^2+x0^2-2\,x0\,x1+x1^2}{(-y1+y0)^2}}}$$

```
> #
# Curve Euclidean invariant eta_xx
# g o (x0,y0,x1) = (0,0,0)
#
eqn1:=cos(theta)*x0-sin(theta)*y0+a=0:
eqn2:=sin(theta)*x0+cos(theta)*y0+b=0:
eqn3:=cos(theta)*x1-sin(theta)*y1+a=0:
solve({eqn1,eqn2,eqn3},{theta,a,b}):
eval(Pxx*cos(theta)^2 + Pyy*sin(theta)^2 - 2*Pxy*cos(theta)*sin(theta) +
2*a*(cos(theta)*Px - sin(theta)*Py) + a^2*N,%):
simplify(%);
```

$$\frac{1}{y0^2 - 2\,y0\,y1 + y1^2 + x0^2 - 2\,x0\,x1 + x1^2} (-2\,y0^2\,Px\,x1 + y0^2\,Pxx + y0^2\,N\,x1^2 + 2\,y0\,y1\,Px\,x0 + 2\,y0\,Pxy\,x1 + 2\,x0\,x1\,Py\,y0 - 2\,y0\,N\,y1\,x0\,x1 + 2\,y0\,y1\,Px\,x1 - 2\,y0\,Pxx\,y1 - 2\,y0\,Pxy\,x0 - 2\,y0\,x1^2\,Py + Pyy\,x0^2 + Pyy\,x1^2 - 2\,Pyy\,x0\,x1 - 2\,Pxy\,x1\,y1 - 2\,y1^2\,Px\,x0 + N\,y1^2\,x0^2 - 2\,x0^2\,Py\,y1 + 2\,Pxy\,x0\,y1 + 2\,x0\,x1\,Py\,y1 + Pxx\,y1^2)$$

```
> simplify(numer(%));
```

$$-2\,y0^2\,Px\,x1 + y0^2\,Pxx + y0^2\,N\,x1^2 + 2\,y0\,y1\,Px\,x0 + 2\,y0\,Pxy\,x1 + 2\,x0\,x1\,Py\,y0 - 2\,y0\,N\,y1\,x0\,x1 + 2\,y0\,y1\,Px\,x1 - 2\,y0\,Pxx\,y1 - 2\,y0\,Pxy\,x0 - 2\,y0\,x1^2\,Py + Pyy\,x0^2 + Pyy\,x1^2 - 2\,Pyy\,x0\,x1 - 2\,Pxy\,x1\,y1 - 2\,y1^2\,Px\,x0 + N\,y1^2\,x0^2 - 2\,x0^2\,Py\,y1 + 2\,Pxy\,x0\,y1 + 2\,x0\,x1\,Py\,y1 + Pxx\,y1^2$$

```
> #
# Curve Euclidean invariant eta_xy
# g o (x0,y0,x1) = (0,0,0)
#
eqn1:=cos(theta)*x0-sin(theta)*y0+a=0:
eqn2:=sin(theta)*x0+cos(theta)*y0+b=0:
eqn3:=cos(theta)*x1-sin(theta)*y1+a=0:
solve({eqn1,eqn2,eqn3},{theta,a,b}):
eval((Pxx - Pyy)*sin(theta)*cos(theta) + Pxy*(cos(theta)^2 - sin(theta)^2) +
Px*(b*cos(theta) + a*sin(theta)) + Py*(a*cos(theta) - b*sin(theta)) + a*b*N,%):
simplify(%);
```

$$\frac{1}{y0^2 - 2\,y0\,y1 + y1^2 + x0^2 - 2\,x0\,x1 + x1^2} (y0\,x1\,Pyy - y0\,x1\,Pxx - y0\,x0\,Pyy + y0\,x0\,Pxx - y1\,x1\,Pyy + y1\,x1\,Pxx + x0\,y1\,Pyy - x0\,y1\,Pxx + N\,y0^3\,x1 - N\,x0^3\,y1 + Py\,y0^2\,x0 + Py\,x0\,x1^2 - 2\,Py\,x0^2\,x1 - 2\,Py\,y0^2\,x1 - Py\,x0\,y1^2 + Px\,y0\,x1^2 + 2\,Px\,y0^2\,y1 - Px\,y0\,y1^2 - Px\,x0^2\,y0 + 2\,Px\,x0^2\,y1 + 2\,Pxy\,x0\,x1 - 2\,Pxy\,y0\,y1 + 2\,Py\,y0\,x1\,y1 - 2\,Px\,x0\,x1\,y1 + N\,x0^2\,y1\,x1 + N\,x0\,y1^2\,y0 - N\,y0\,x1^2\,x0 + N\,y0\,x1\,x0^2 - N\,x0\,y1\,y0^2 - N\,y0^2\,x1\,y1 + Py\,x0^3 - Px\,y0^3 - Pxy\,x0^2 - Pxy\,x1^2 + Pxy\,y0^2 + Pxy\,y1^2)$$

```
> #
```

```
# Curve Euclidean invariant eta_yy
# g o (x0,y0,x1) = (0,0,0)
#
eqn1:=cos(theta)*x0-sin(theta)*y0+a=0:
eqn2:=sin(theta)*x0+cos(theta)*y0+b=0:
eqn3:=cos(theta)*x1-sin(theta)*y1+a=0:
solve({eqn1,eqn2,eqn3},{theta,a,b}):
eval(Pxx*sin(theta)^2 + Pyy*cos(theta)^2 + 2*Pxy*cos(theta)*sin(theta) +
2*b*(sin(theta)*Px + cos(theta)*Py) + b^2*N,%):
simplify(%);
```

$$\begin{aligned}\frac{1}{y0^2 - 2\,y0\,y1 + y1^2 + x0^2 - 2\,x0\,x1 + x1^2}\,(&Pxx\;x1^2 + Pyy\,y1^2\\
&+ 2\,x0\,x1\,Py\,y0\\
&+ 2\,y0\,y1\,Px\,x0 - 2\,y0\,y1\,Px\,x1 - 2\,y0\,N\,x0^2\,y1\\
&+ N\,x0^2\,x1^2 + 2\,y0\,Pxy\,x0 - 2\,y0\,Pyy\,y1 - 2\,y0\,Pxy\,x1\\
&+ 2\,y0^2\,N\,x0^2 + y0^2\,Pyy\\
&+ 2\,y0\,N\,y1\,x0\,x1 - 2\,y0^2\,N\,x0\,x1 - 2\,N\,x0^3\,x1\\
&- 2\,Pxx\;x0\,x1 + 2\,Pxy\,x1\,y1 + y0^2\,N\,y1^2\\
&+ N\,y0^4 - 2\,x0^2\,Py\,y0 + 2\,x0^2\,Py\,y1 + 4\,x0^2\,Px\,x1 + N\,x0^4\\
&+ Pxx\;x0^2 - 2\,Pxy\,x0\,y1 - 2\,x0\,x1^2\,Px - 2\,y0^3\,N\,y1\\
&- 2\,x0^3\,Px - 2\,y0^3\,Py\\
&+ 2\,y0^2\,Px\,x1 - 2\,x0\,x1\,Py\,y1 - 2\,y0^2\,Px\,x0\\
&+ 4\,y0^2\,y1\,Py - 2\,y0\,y1^2\,Py\,)\end{aligned}$$

```
>
```

Appendix C

Matlab files

Check invariance

```
% Check the result of
% "krw1_curve_Eu_inv"
%   Author:      K.R. Widder
%   Time-stamp:  11/24/06
%   E-mail:      widder@wisc.edu
% (C) 2006 by Kerry Widder
% created: 11/24/2006
% adapted from file created by Wei-Yang Lin

clear all;
close all;
clc;

N = 70;
theta = 0:pi/3/(N-1):pi/3;
x = cos(theta);
y = sin(theta);
[Ix, Iy, Ixx, Ixy, Iyy] = krw1_curve_Eu_sum_inv_denom(x',y');
disp('% krw1_curve_Eu_sum_inv_denom');
disp(['(Ix, Iy, Ixx, Ixy, Iyy) = (' num2str(Ix) ', '...
     num2str(Iy) ', ' num2str(Ixx) ', ' num2str(Ixy) ', ' num2str(Iyy) ')']);

% plot(x,y,'r:'), hold on;
disp('% krw1_curve_Eu_sum_inv');

for i = 1:10
    theta = 2*pi*rand(1);
    t1  = 100*randn(1);
    t2  = 100*randn(1);
%    disp(['(theta, t1, t2) = (' num2str(theta*360/2/pi) ', '...
%            num2str(t1) ', ' num2str(t2) ')'])
    R   = [cos(theta) -1*sin(theta); sin(theta) cos(theta)];
    T   = [t1 t2]';
    pts = [x;y];
    pts = R*pts + T*ones(1,N);
    xx  = pts(1,:);
    yy  = pts(2,:);
    [Ix, Iy, Ixx, Ixy, Iyy] = krw1_curve_Eu_sum_inv(xx',yy');
    disp(['(Ix, Iy, Ixx, Ixy, Iyy) = (' num2str(Ix) ', '...
         num2str(Iy) ', ' num2str(Ixx) ', ' num2str(Ixy) ', ' num2str(Iyy) ')']);
end
plot(x,y,'r:'), hold off;
```

Calculate Summation Invariants

```
function [Ix, Iy, Ixx, Ixy, Iyy] = krw1_curve_Eu_sum_inv(x,y)
% krw1_curve_Eu_inv compute Euclidean summation invariant of curve
% like Eq. (2.12 - 2.16) in Wei-Yang's thesis, only different normalization
%  Uses normalization equation:  (x(1), y(1), x(N)) = (0,0,0)
%
%   [Ix, Iy, Ixx, Ixy, Iyy] = krw1_curve_Eu_sum_inv(x,y)
%   x : N x 1 column vector
%   y : N x 1 column vector
%
%
%   ,where
%   (x(1),y(1)) is initial point
%   (x(N),y(N)) is end point
%   P10 = sum_1^N x(t)
%   P01 = int_1^N y(t)
%   P20 = sum_1^N x(t)^2
%   P11 = int_1^N x(t)*y(t)
%   P02 = sum_1^N y(t)^2
%
%   Author:      K.R. Widder
%   Time-stamp:  11/24/06
%   E-mail:      widder@wisc.edu
% (C) 2006 by Kerry Widder
% created: 11/24/2006


% x : N x 1 column vector
% y : N x 1 column vector

N = size(x,1);

P10 = sum(x);
P01 = sum(y);
P20 = sum(x.*x);
P11 = sum(x.*y);
P02 = sum(y.*y);

%I_1 = N^2*(x(1)^2 + y(1)^2) + P10^2 + P01^2 - 2*N*(x(1)*P10 + y(1)*P01);

Iy  = P10*(x(1) - x(N)) + P01*(y(1) - y(N)) + N*x(1)*(x(N) - x(1)) + N*y(1)*(y(N) - y(1));

Ix  = P01*(x(N) - x(1)) + P10*(y(1) - y(N)) + N*( x(1)*y(N) - x(N)*y(1) );

Ixx = 2*P10*((y(1) - y(N))*(y(N)*x(1) - y(1)*x(N)))...
        + 2*P01*(x(1) - x(N))*(y(1)*x(N) - y(N)*x(1))...
        + P02*((x(1) - x(N))^2) +  P20*((y(1) - y(N))^2) ...
        + 2*P11*(x(N) - x(1))*(y(1) - y(N))...
        + N*(( (y(1)*x(N)) - (x(1)*y(N)) )^2);

Ixy =   P10*( y(1)*(x(N)^2 - y(N)^2 - x(1)^2 -y(1)^2 + 2*y(1)*y(N)) + 2*x(1)*y(N)*(x(1) - x(N)) )...
      + P01*( x(1)*(y(1)^2 + x(1)^2 - y(N)^2 + x(N)^2 - 2*x(1)*x(N)) + 2*y(1)*x(N)*(y(N) - y(1)) )...
      + (P20 - P02)*(x(1) - x(N))*(y(1) - y(N))...
      + P11*( (y(N) - y(1))^2 - (x(N) - x(1))^2 )...
      + N*( y(1)*x(N)*(y(1)^2 - x(1)*x(N)) + (x(1)^2)*y(N)*(x(N) - x(1)) + x(1)*y(1)*((y(N)^2) +
x(N)*x(1))...
      - (y(1)^2)*y(N)*(x(N) + x(1)) );


Iyy = 2*P10*((x(N) - x(1))*((y(1)^2) - y(N)*y(1) - x(1)*(x(N) - x(1))))...
    + 2*P01*((y(N) - y(1))*((x(1)^2) - x(N)*x(1) - y(1)*(y(N) - y(1))))...
    + P20*((x(N) - x(1))^2) + P02*((y(N) - y(1))^2)...
    + 2*P11*((y(1) - y(N))*(x(1) - x(N)))...
    + N*((x(N)^2)*(x(1)^2) - 2*y(N)*y(1)*(x(1)^2) + 2*(y(1)^2)*(x(1)^2)...
    + 2*y(N)*y(1)*x(N)*x(1) - 2*(y(1)^2)*x(N)*x(1) - 2*x(N)*(x(1)^3)...
    + (y(1)^2)*(y(N)^2) + (y(1)^4) + (x(1)^4) - 2*y(N)*(y(1)^3));
```

```
function [Ix, Iy, Ixx, Ixy, Iyy] = krw1_curve_Eu_sum_inv_denom(x,y)
% krw1_curve_Eu_inv compute Euclidean summation invariant of curve
% like Eq. (2.12 - 2.16) in Wei-Yang's thesis, only different normalization
%  Uses normalization equation:  (x(1), y(1), x(N)) = (0,0,0)
%
%   [Ix, Iy, Ixx, Ixy, Iyy] = krw1_curve_Eu_sum_inv_denom(x,y)
%   x : N x 1 column vector
%   y : N x 1 column vector
%
%   ,where
%   (x(1),y(1)) is initial point
%   (x(N),y(N)) is end point
%   P10 = sum_1^N x(t)
%   P01 = int_1^N y(t)
%   P20 = sum_1^N x(t)^2
%   P11 = int_1^N x(t)*y(t)
%   P02 = sum_1^N y(t)^2
%
%   Author:      K.R. Widder
%   Time-stamp:  11/24/06
%   E-mail:      widder@wisc.edu

N = size(x,1);
P10 = sum(x);
P01 = sum(y);
P20 = sum(x.*x);
P11 = sum(x.*y);
P02 = sum(y.*y);
D = (y(N) - y(1))^2 + (x(N) - x(1))^2;
Droot = sqrt(D);

%I_1 = N^2*(x(1)^2 + y(1)^2) + P10^2 + P01^2 - 2*N*(x(1)*P10 + y(1)*P01);

Iy  = (1/Droot)*(P10*(x(1) - x(N)) + P01*(y(1) - y(N)) + N*x(1)*(x(N) - x(1)) + N*y(1)*(y(N) - y(1)) );

Ix  = (1/Droot)*(P01*(x(N) - x(1)) + P10*(y(1) - y(N)) + N*( x(1)*y(N) - x(N)*y(1)) );

Ixx = (1/D)*(2*P10*((y(1) - y(N))*(y(N)*x(1) - y(1)*x(N)))...
        + 2*P01*(x(1) - x(N))*(y(1)*x(N) - y(N)*x(1))...
        + P02*((x(1) - x(N))^2) +  P20*((y(1) - y(N))^2) ...
        + 2*P11*(x(N) - x(1))*(y(1) - y(N))...
        + N*(( (y(1)*x(N)) - (x(1)*y(N)) )^2) );

Ixy =  (1/D)*( P10*( y(1)*(x(N)^2 - y(N)^2 - x(1)^2 -y(1)^2 + 2*y(1)*y(N)) + 2*x(1)*y(N)*(x(1) - x(N))
)...
      + P01*( x(1)*(y(1)^2 + x(1)^2 - y(N)^2 + x(N)^2 - 2*x(1)*x(N)) + 2*y(1)*x(N)*(y(N) - y(1)) )...
      + (P20 - P02)*(x(1) - x(N))*(y(1) - y(N))...
      + P11*( (y(N) - y(1))^2 - (x(N) - x(1))^2 )...
      + N*( y(1)*x(N)*(y(1)^2 - x(1)*x(N)) + (x(1)^2)*y(N)*(x(N) - x(1)) + x(1)*y(1)*((y(N)^2) +
x(N)*x(1))...
      - (y(1)^2)*y(N)*(x(N) + x(1))) );

Iyy = (1/D)*(2*P10*((x(N) - x(1))*((y(1)^2) - y(N)*y(1) - x(1)*(x(N) - x(1))))...
    + 2*P01*((y(N) - y(1))*((x(1)^2) - x(N)*x(1) - y(1)*(y(N) - y(1))))...
    + P20*((x(N) - x(1))^2) + P02*((y(N) - y(1))^2)...
    + 2*P11*((y(1) - y(N))*(x(1) - x(N)))...
    + N*((x(N)^2)*(x(1)^2) - 2*y(N)*y(1)*(x(1)^2) + 2*(y(1)^2)*(x(1)^2)...
    + 2*y(N)*y(1)*x(N)*x(1) - 2*(y(1)^2)*x(N)*x(1) - 2*x(N)*(x(1)^3)...
    + (y(1)^2)*(y(N)^2) + (y(1)^4) + (x(1)^4) - 2*y(N)*(y(1)^3)));
```