Registering Real and Virtual Imagery

Yi L. Murphey¹, Hongbin Jia¹, and Michael Del Rose²
¹Department of Electrical and Computer Engineering
The University of Michigan-Dearborn
²U. S. Army TACOM-TARDEC Vetronics Technology Center

Abstract
Registering a virtual scene with a real scene captured by a video camera has a number of applications including visually-guided robotic navigation, surveillance, military training and operation. The fundamental problem involves several challenging research issues including finding corresponding points between the virtual and the real scene and camera calibration. This paper presents our research in defining and mapping a set of reliable image features for registering the two imageries, extracting and selection reliable control points for the construction of intrinsic and extrinsic camera parameters. A number of innovative algorithms are presented followed by extensive experimental analysis. An application of registering virtual database image with video image is presented. The algorithms we developed on linear structured features and selecting of reliable control points are applicable to image registration beyond virtual and real imageries.

1. Introduction

Video imagery is rapidly emerging as a low cost sensor in a variety of applications including target detection and tracking, robot navigation, surveillance, and military training and operation. In order to achieve high accuracy in real time operations, many of these applications generate a virtual database of the scene and combine it with video imagery to get more operational information. A virtual database contains 3D models of the objects and the environment of interests. There are a number of reasons for the need of this type of technologies. The virtual database of a simulated environment can be generated artificially using 3D computer graphics and/or a virtual reality software. Some objects may not be generated as accurately as others due to both the large demand on labor and the difficulty in attaining accurate measurements. In addition, the virtual database can never include dynamic objects, such as a car parked on the street, road construction, etc. Figure 1 shows such an example. Figure 1 (a) shows a database image and Figure 1(b) shows a video image that has the same scene as in (a). As one can see, many objects absent from the database image appeared in the video image: two cars parked in the scene, temporary objects, etc. Visible and infrared(IR) video cameras are increasingly deployed on moving
vehicles, both manned and unmanned, to provide observers with a real time view of activity and terrain. However, video imagery does not provide sufficient information for detecting occluded objects, camouflaged objects. In particular video imagery does not provide much detectable information in bad weather, smoke, etc. A terrain database should be able to provide such information in all conditions. To integrate information from a video imagery and a terrain database, we need to address a key technology, registering an image frame in a video image sequence with a virtual database image with the similar view. Typically the mapping of virtual database imagery to a real time video imagery requires full registration of at least one video image frame with a virtual image. The registration results can be used to simplify the mapping between the subsequent image frames and virtual database images.

This paper presents a framework for the registration of an image of real view with a virtual database image. Mathematically, registration of the two imageries can be modeled as the problem of registering images of multiple views, which has a broad range of applications [1,2, 3, 4, 5, 6, 7]. In an automated system, four steps may be involved in registering imageries of two different camera views: feature extraction, finding the correspondence between the two imageries using the selected features, extracting corresponding points from matched image features, and finally, computation of the intrinsic and extrinsic parameters between the two camera coordinate systems, virtual and real view in our application, using a reliable set of corresponding points. The procedure of finding corresponding features in two images of different views is a nontrivial problem[3,7,11,14]. The most popular features used in image registration include intensities, edges, lines, and corners of objects[1,3,4,7,14]. The computational procedure used to calculate the intrinsic and extrinsic parameters of two camera systems is called camera calibration[5, 8, 9, 10, 11, 12]. A camera calibration procedure calculates transformation matrices(extrinsic parameters) and camera parameters(intrinsic parameters) based on a set of corresponding points. The accuracy of the camera calibration very much depends on the accuracy of the corresponding points[13].

Once we obtain the intrinsic and extrinsic parameters of the camera model, we can place a virtual object, visible or occluded, accurately in the real scene. In this paper, we present our research in defining and extracting image features, extracting corresponding points from a given set of image features and selecting reliable set of control points from a given set of corresponding points.
2. A framework for registering virtual and real scene
Figure 2 illustrates the proposed framework for registering virtual imagery with a real scene imagery. The objective of the proposed system is to calculate the camera calibration parameters necessary to transform virtual objects into the real view scene. There are three components in the framework. The first component shown in Figure 1 (a) illustrates a process that extracts the proposed linear structured features (LSFs) from each virtual image and store them as an attribute of the image in the database. The LSFs are used in the system illustrated in Figure 2 (b) to find corresponding features in the real view image. Figure 2 (b) presents a system that registers a virtual view image with a real view image by matching LSFs, extracting corresponding points, selecting optimal control points, and a camera calibration procedure. Figure 2 (c) shows the process of placing a virtual object in a real view image. The major contribution of this paper is illustrated in Figure 2 (b). In our application, a video camera is mounted on a moving vehicle and the location of the vehicle, therefore the location of the camera on the vehicle, can be determined using a GPS devise. If the GPS reading is accurate, the virtual database image at the view point that matches the GPS reading should exactly match the current video image frame. Due to the limit of accuracy of the GPS systems, the virtual image retrieved based on a GPS reading often does not accurately match the video image. However the estimated location of the video camera can be used to retrieve virtual images that have the viewpoints similar to the location of the video camera. Then two images of different views can be registered using image feature matching and a camera calibration technique.

We developed a concept of linear structured features used to match the two imagery, an algorithm to match the LSFs associated with the virtual image with the LSFs in the real view image, an algorithm to extract corresponding points from the matched LSFs, an algorithm that selects an optimal set of control points for camera calibration. A camera calibration procedure attempts to establish a transformation procedure between two camera coordinate systems. A camera calibration procedure can be formally described as follows. First we define the related coordinate systems.

3D virtual view camera coordinate system: \((X_v, Y_v, Z_v)\) in mm;

3D real view camera coordinate system: \((X_r, Y_r, Z_r)\) in mm;
2D virtual view image coordinate system: \((x_v, y_v)\) in pixel.

2D real view image coordinate system: \((x_r, y_r)\) in pixel.

The relationship between two 3D coordinate systems can be described by the equation:

\[
\begin{bmatrix}
X \\
Y \\
Z
\end{bmatrix}
= R \cdot \begin{bmatrix}
X_v \\
Y_v \\
Z_v
\end{bmatrix} + T = \begin{bmatrix}
r_{11} & r_{12} & r_{13} \\
r_{21} & r_{22} & r_{23} \\
r_{31} & r_{32} & r_{33}
\end{bmatrix} \cdot \begin{bmatrix}
X_v \\
Y_v \\
Z_v
\end{bmatrix} + \begin{bmatrix}
t_x \\
t_y \\
t_z
\end{bmatrix} \tag{1}
\]

Based on the pinhole camera geometry and lens distortion theory[5], \(R\) and \(T\), which are often referred to as extrinsic parameters, can be obtained by solving a system of equations, which requires the knowledge of the intrinsic parameters \(f, (s_x, s_y), P_o = (x_o, y_o)^T\), where \(f\) is the effective focal length of camera, \(s_x\) and \(s_y\) are scale factors along \(x\) and \(y\) direction respectively, and \(P_o\) is the image coordinates of optical center, and a set of corresponding points between the three coordinate systems, 3D virtual view coordinates, 2D virtual image coordinates, and 2D image coordinates of a real view camera system. The corresponding points used in camera calibration are often referred to as control points. Formally, a set of corresponding points can be represented as

\[
\Sigma = \{ [(X_v^1, Y_v^1, Z_v^1), (x_v^1, y_v^1)], \ldots, [(X_v^n, Y_v^n, Z_v^n), (x_v^n, y_v^n)] \}, \text{ where}
\]

\((X_v^i, Y_v^i, Z_v^i), i=1, \ldots, n\), is the world coordinates of the 2D virtual image feature point \((x_v^i, y_v^i)\), and

\((x_r^i, y_r^i)\) is the corresponding point of \((x_v^i, y_v^i)\) in the real view image.

**Our focus of research is to obtain a set of reliable control points to be used in a camera calibration model.** There are a number of existing camera calibration techniques[5, 8, 9, 10, 11] that can be used to determine both the intrinsic and extrinsic parameters for a given set of control points. For coplanar cases \(n \geq 5\), for non-coplanar cases, \(n \geq 7\). We have chosen to implement the camera calibration procedure presented by Roger Tsai[5].

After obtaining the parameters of \(R\) and \(T\), and the intrinsic parameters, we can place any virtual object using the following procedure. A virtual object can be represented by a set of points

\[
V_{obj} = \{(X_v^1, Y_v^1, Z_v^1), (x_v^1, y_v^1)], \ldots, [(X_v^k, Y_v^k, Z_v^k), (x_v^k, y_v^k)]\}.\]

The set of points in the real view image corresponding to those in \(V_{obj}\)

\[
R_{obj} = \{(x_r^1, y_r^1), \ldots, (x_r^k, y_r^k)\}, \]

can be calculated through the following procedure based on Tsai’s two stage technique:
\[
\begin{align*}
\left\{ \begin{array}{l}
\frac{1}{s_x}d'_x x'_i + \frac{1}{s_x}d'_x x'_ik_i r^2 &= f \frac{r_1 X'_v + r_2 Y'_v + r_3 Z'_v + T_x}{r_1 X'_v + r_2 Y'_v + r_3 Z'_v + T_z}, \\
\frac{1}{d_y}y'_i + \frac{1}{d_y}y'_ik_i r^2 &= f \frac{r_4 X'_v + r_5 Y'_v + r_6 Z'_v + T_y}{r_4 X'_v + r_5 Y'_v + r_6 Z'_v + T_z},
\end{array} \right.
\end{align*}
\]

where \( r = \sqrt{(s_x^{-1}d'_x x'_i)^2 + (d_y y'_i)^2} \), \( d'_x = d_x \frac{N_{cx}}{N_{fx}} \), 

\( w'_i = r_7 X'_v + r_8 Y'_v, \ y'_i = r_7 X'_v + r_8 Y'_v + t_y \), and \( d_x \) is the center to center distance between adjacent sensor elements in the scan line direction, \( d_y \) is the center to center distance between adjacent CCD sensor in the Y direction, \( N_{cx} \) is the number of sensor elements in the X direction and \( N_{fx} \) is the number of pixels in a line as sampled by the computer. \( d_x, d_y, N_{cx} \) and \( N_{fx} \) should be provided by the camera manufacturer\[5\].

The accuracy of the constructed intrinsic and extrinsic parameters very much depends on the accuracy of the control points \( \Sigma \) used in the camera calibration procedure. Research work has shown that all camera calibration techniques are very susceptible to noise in image coordinates\[10, 11, 12\]. Haralick et al. showed that when the noise level exceeds a knee level, many camera calibration methods became extremely unstable and the errors could be outrageously large\[13\]. In this paper we describe following algorithms necessary for the extraction of reliable corresponding points.

1. Define a stable image features for registering images of two different views.
2. Extract a reliable set of correspondent points from matched image features.
3. Find a quasi-optimal set of control points to be used in the construction of the intrinsic and extrinsic parameters. The issue of selecting the best set of control points for the construction of intrinsic and extrinsic parameters is never being fully addressed. Based on the existing research results\[5\], we understand that the minimum number of points to solve a coplanar calibration problem is 5, non-coplanar is 7, and fully optimized calibration requires at least 11. Other works showed the when the number control points used in the construction, more accurate results can be obtained\[10, 11, 12, 13\]. However we showed in \[16\] that the intrinsic and extrinsic parameters obtained from a bigger set of control points are not necessarily more accurate than those obtained from a smaller set of control points.

The following sections address all these issues.
Figure 2. The framework of virtual and real view image registration.
3. Linear Structured Features

Selecting image features for registering the real and virtual scenes depends on a number of factors:

- feature availability in images of both virtual and real scene,
- the accuracy of the feature extraction and matching, and
- the reliability of 3D measurement of the feature points in the virtual database.

As we discussed earlier that virtual databases are generated using 3D graphics tools and/or virtual reality software. The measurement of objects in the virtual scene is obtained from engineering data such as blue prints of man made objects, photographs, GPS readings, etc. Due to the large demand of labor, large object features are usually measured more accurately than small features. For example, the dimensions of a large building are likely more accurate than the small windows on the building, the location and dimensions of a large telephone pole are likely more accurate than those of trees in a forest. Therefore, in registering a virtual image with a real scene image the most promising features are large line segments.

Based on the above analysis we developed linear structured features (LSF) for matching the real and virtual image pair. Linear structured features are defined as a set of intersected line segments that have orientation close to either horizontal and vertical. Using line segments instead of points as features has attracted the attention of many researchers for various applications [4, 1, 7]. Linear object features are relatively easy to extract and match and they exist in most of the scenes in this application domain. More importantly they exhibit the following transformation invariance that are important for registering the real and virtual imagery.

1. If there is only a rotational change $\beta$ around the Y-axis and a translation between the virtual camera and the real view coordinate system, then a vertical line in the virtual database is transformed to a vertical line in the real view camera system. This property is proved as follows.

The rotation matrix can be written as

$$
R_y = \begin{bmatrix}
\cos \beta & 0 & -\sin \beta \\
0 & 1 & 0 \\
\sin \beta & 0 & \cos \beta
\end{bmatrix}.
$$

A vertical line in the virtual database can be written as

$$
\begin{align*}
X_v &= x_0 \\
Y_v &= y_0 + t \\
Z_v &= z_0
\end{align*}
$$

Its correspondence in the real view camera system through the rotational transformation matrix above is
\[
x = x_o \cos \beta - z_o \sin \beta + t_x = x'_o \\
y = y_o + t_y + t = y'_o + t \\
z = x_o \sin \beta + z_o \cos \beta + t_z = z'_o
\]
which is still a vertical line.

2. If there is only a rotational change \( \alpha \) around the X-axis and a translation between the virtual camera and the real view coordinate system, then a horizontal line in the virtual database is transformed to a horizontal line in the real view camera system. This property is proved as follows.

The rotation matrix can be written as
\[
R_x = \begin{bmatrix}
1 & 0 & 0 \\
0 & \cos \alpha & \sin \alpha \\
0 & -\sin \alpha & \cos \alpha
\end{bmatrix}
\]
A horizontal line in the virtual database can be written as
\[
\begin{cases}
x_v = x_0 + t \\
y_v = y_0 \\
z_v = z_0
\end{cases}
\]
Its correspondence in the real view through the rotation and translation is:
\[
\begin{cases}
x = x_0 + t_x + t = x'_0 + t \\
y = y_0 \cos \alpha + z_0 \sin \alpha + t_y = y'_0 \\
z = -y_0 \sin \alpha + z_0 \cos \alpha + t_z = z'_0
\end{cases}
\]
which is still a horizontal line.

3. If there are only small rotation around both y axis and x axis, i.e. \( \beta \approx 0 \) and \( \alpha \approx 0 \), then a horizontal line or vertical line in the camera system is transformed to almost horizontal or vertical line respectively.
A horizontal or a vertical line in the virtual camera system transformed through the rotation of \( \beta \approx 0 \) and \( \alpha \approx 0 \) and a translation to the real view system can be written as
\[
\begin{cases}
x = (x_0 + t) \cos \beta - (-y_0 \sin \alpha + z_0 \cos \alpha) \sin \beta + t_x \\
y = y_0 \cos \alpha + z_0 \sin \alpha + t_y \\
z = (x_0 + t) \sin \beta + (-y_0 \sin \alpha + z_0 \cos \alpha) \cos \beta + t_z
\end{cases}
\]
which is almost a horizontal line if \( \beta \approx 0 \), and almost a vertical line if \( \alpha \approx 0 \).

Based on the discussion above, we define a set of linear structured features as follows.
\[
\Psi = H \cup V = \{h_1, h_2, \ldots, h_k\} \cup \{v_1, v_2, \ldots, v_p\},
\]
where \( k \) and \( p \) are positive integers, \( h_i \), for \( i = 1, \ldots, k \), are line segments that have orientation close to 0, and \( v_j, j = 1, \ldots, p \), are line segments that are close to 90 and each \( v_j \) must intersect with at least one line segment in \( H_L \). In another words, the linear structured feature set \( \Psi \) contains line segments as close to horizontal and vertical lines as an image can provide, and these line segments interest each other to provide reliable matching features in both virtual and real view images. The intersection points of the line segments are more robust in finding corresponding points.

4. Extracting and Matching Linear Structured Features

A set of linear structured features should be extracted for every virtual database image. We developed an algorithm, LSF extraction algorithm, that finds significant line segments close to horizontal orientation and the finds the almost vertical line segments that intersect with those almost horizontal line segments. Line segments can be obtained by fitting along a sequence of edge points[14]. However, lines in images are rarely single pixel wide. There are usually clusters of single pixel line segments with the same orientation but different lengths in edge images. Single pixel line segments used as features can cause ambiguity in a matching procedure[15]. Therefore we use line bands instead of single-pixel line segments in linear structured features.

A line band is a set of line segments that are connected and have the same orientation. A line band is formally represented by \( \{\text{endpoint}_1, \text{endpoint}_2, \theta, \text{width}\} \), where \( \text{endpoint}_1 \) and \( \text{endpoint}_2 \) are the two end points of the centroid line of the line band, \( \theta \) and \( \text{width} \) is the orientation and the width of the line band segment respectively. Line bands are obtained through a merging process of single pixel line segments that are connected and have the same orientation[16].

The LSF extraction algorithm is summarized in the following steps.

[step 1] obtain an edge image by applying an edge detection algorithm to the virtual database image
[step 2] obtain line segments by fitting edge points
[step 3] merge connected line segments with the same orientation into line bands.
[step 4] sort the line bands into the following orientations,
\( 0^\circ, 15^\circ, 30^\circ, 45^\circ, 60^\circ, 75^\circ, 90^\circ, 105^\circ, 120^\circ, 135^\circ, 150^\circ, 165^\circ \)
[step 5] search starting from the \( 0^\circ \) for a set that contains a set that is the closest to 0 orientation and contains a significant number of line bands. Call this set as \( H_L \).
[step 6] find a set that contains line bands that intersect with the line bands in $H_L$ and has orientation closely perpendicular to those in $H_L$. Call this set as $V_L$.

[step 7] Group line bands in $H_L$ and $V_L$ that are connected through the intersecting line bands to form structured features, $LSF_1$, $LSF_2$, $\ldots$, $LSF_k$.

Note this algorithm was designed to find a set of linear structured features that contains the intersected line bands that are as close to horizontal and vertical line bands as an image can provide. The $H_L$ set may not be the $0^\circ$ set depending on the available image features. The algorithm is applied to every virtual image in the database and the $LSFs$ of each virtual image are stored as an attribute of the image that is ready to be retrieved. Figure 3(a) shows an example of linear structured features of a virtual database image. In this example there are two $LSFs$, one is at the top consisting of one long horizontal line band intersecting with two short vertical line bands. The second one consists of five horizontal line bands and nine vertical line bands. Note the three vertical line bands on the left side are not connected.

![Figure 3(a)](image1)

Figure 3. Linear structured features in a virtual image(a) and the matched structural features in a video image (b).

We apply to a real view image the same edge detection and line fitting algorithms to obtain a set of line bands denoted as $\Phi$. The following properties have been observed between a virtual image and the corresponding real view image.

- Due to the complexity of lighting condition and cluttered background in a real image, the line bands in $\Phi$ often do not contain the line bands exactly corresponding to those obtained in the corresponding virtual image.
- The visually corresponding line bands may not be at the exact same locations in the two images.
• As we discussed earlier that the corresponding line bands in both images should have close orientation to each other.

Based on these properties, we developed the following algorithm for finding the matched linear structured features in a real view image.

Assume \( \Psi = \{LSF_1, \ldots, LSF_k\} \) is a set of the linear structured features associated with a virtual database image and each \( LSF_j = \{H_L, V_L\} \) and \( H_L = \{h_1, h_2, \ldots, h_p\} \), \( V_L = \{v_1, v_2, \ldots, v_q\} \). Each line band segment is represented as \{endpoint1, endpoint2, \( \theta \), width\} as discussed earlier.

We select the line bands in \( \Phi \) that have the same orientation category as those in \( H_L \) and \( V_L \) and denote them as \( \Gamma \). For each LSF in \( \Psi \), we find line bands in \( \Phi \) that have the best matches with those in the selected LSF. We use the following formula to calculate the similarity between two line bands \( X \) and \( Y \).

\[
\Lambda(X, Y) = \omega_1 \sqrt{(X_\theta - Y_\theta)^2} + \omega_2 \left( \sqrt{(X_{\text{endpo int1}} - Y_{\text{endpo int1}})^2} + \sqrt{(X_{\text{endpo int2}} - Y_{\text{endpo int2}})^2} \right) + \omega_3 \sqrt{(X_{\text{width}} - Y_{\text{width}})^2}
\]

where \( 0 \leq \omega_i \leq 1 \), for \( i = 1, 2, 3 \), and \( \sum_{i=1}^{3} \omega_i = 1 \).

A linear structure LSF has found a match if at least one pair of its intersected line bands has found a matching pair in \( \Gamma \).

The LSF matching algorithm is summarized as follows.

Given \( \Psi = \{LSF_1, \ldots, LSF_k\} \) associated with a virtual database image and a corresponding real view image, \( R_{\text{img}} \). Let \( R_{\Psi} \) store the LSFs to be found in \( R_{\text{img}} \). Initially set \( R_{\Psi} = {} \).

[step 1] Obtain a set of line bands \( \Phi \) by applying to \( R_{\text{img}} \) the same edge detection, line fitting and line band finding algorithms as those used in the LSF extraction algorithm.

[step 2] Find the line bands in \( \Phi \) that have the same orientations as those in \( \Psi \) and put them in \( \Gamma \).

[step 3] For \( j = 1, \ldots, k \) repeat step 4 through step 9

[step 4] \( R_{LSF_j} = {} \)

[step 5] For every line band \( X \) in \( LSF_j \), find a line band \( Y \) in \( \Gamma \) such that \( Y \) has the best match to \( X \) according to function \( \Lambda \).
[step 6] If \( \Lambda(X,Y) \) indicates a strong match between \( X \) and \( Y \), \( R_{LSF_j} = R_{LSF_j} \cup \{Y\} \) and delete \( Y \) from \( \Gamma \).

[step 7] Delete those line bands from \( LSF_j \) that have no match in \( \Gamma \).

[step 8] Delete \( LSF_j \) from \( \Psi \) if \( R_{LSF_j} \) has no intersecting line bands.

[step 9] If \( R_{LSF_j} \) has at least one pair of intersecting line bands, add \( R_{LSF_j} \) to \( R_\Psi \).

Figure 3(b) shows the line bands in a real image that match the two linear structured features illustrated in (a). Note the line bands that match the second LSF in the left image are not completely connected.

5. Selecting Reliable Control Points for Camera Parameters Construction

As we showed in section 2, intrinsic and extrinsic parameters used to map virtual objects to a real image are constructed from a set of control points. In order to obtain accurate and stable camera parameters, it is critical to provide the system with a set of representative and accurate control points. There are two issues involved, representative and accuracy. The reconstructed camera parameters are representative if points anywhere in the virtual image can be mapped to the real image with the same accuracy. The reconstructed camera parameters are accurate if the points in the virtual image can be accurately mapped to the real image plane. In order to address these two issues, we developed the following principles to guide our algorithm development. Let a control point be represented a triplet \( \{ (x_v, y_v, z_v), (x_r, y_r) \} \). We should select control points

- with accurate 3D location measurement. This means that the 3D location \( (x_v, y_v, z_v) \) of a virtual image point \( (x_v, y_v) \) should be accurate. This largely depends on the construction of the virtual image database.
- that have minimum matching errors. This implies that we select feature points that have accurate correspondence between the virtual and the real image. Specifically, the correspondence error between \( (x_v, y_v) \) and \( (x_r, y_r) \) should be minimum.
- that cover all planes of the objects of interest. It makes the reconstructed camera parameters representative to all points of the objects of interest.

We developed a two-stage approach guided by the above principles. At the first stage we developed an algorithm that selects a set of corresponding points from the virtual and real images based on the matched LSFs. At the second stage, we developed two different algorithms that select a more reliable and stable set of control points from the corresponding points. The process at first stage addresses the second and third issue, and the second stage addresses the first and third issue.
Due to the different camera views and noise interference, a pair of corresponding line bands in a virtual and real image most likely have different lengths and the end points of two corresponding line bands are often not reliable for alignment. The junction points of matched LSFs are more stable and reliable to align the two matched line band segments. We first extract all the junctions from all the matched LSFs as the correspondence points. If we need more points, we use the junction points to align the corresponding line band segments in each matched pair of LSFs, and then extract the corresponding points scattered over strong line bands. As we mentioned in section 2 that in theory, we need at least seven corresponding pairs of points to construct a set of camera parameters. However it is has been shown 7 points are not sufficient for reliable results[5,9,10,11,12,13,16]. At the first stage of this algorithm, we recommend to extract at least 30 corresponding points from the matched LSFs. Figure 4 (a) and (b) show the corresponding feature points selected by the above algorithm.

![Figure 4](image)

Figure 4. Selected corresponding points in a pair of virtual and real images.

At the second stage, the algorithm attempts to select an optimal set of control points for camera calibration from the corresponding points generated by the first stage. Many researchers [4,5] believe that more points used in the construction of transformation matrices give better accuracy. However we found in[16] that we obtained better performance when we used only a subset of corresponding points. The problem of selecting an optimal control point set, which is non-trivial, remains to be explored. The most common framework to is to define criteria for measuring the goodness of a set of control points, and then use a search algorithm to find an optimal or quasi-optimal set of control points in a larger point space based on the criteria. We formulate the problem as follows. Let $\Omega = \{p_1, p_2, \ldots, p_n\}$ be a set of corresponding points generated at the first stage and $n > 7$. We explore the problem based on the following two principles.

**Principle 1.** Let $F(\Omega_\alpha) = < I_\alpha P_\alpha, E_\alpha P_\alpha >$ be a procedure that generates the intrinsic and extrinsic parameters, $I_\alpha P_\alpha, E_\alpha P_\alpha$, using control points in $\Omega_\alpha$. We attempt to find $\Omega_{op} \subseteq \Omega$ and $|\Omega_{op}| \geq 7$ such that $F(\Omega_{op}) = < I_{op} P_{op}, E_{op} P_{op} >$ and $I_{op}$ and $E_{op}$ that give us the best possible performance on the
mapping of **unseen** data points of a virtual objects to the corresponding video image points. This is different from a general optimization technique that evaluates the performance on the control points.

**Principle 2.** The distribution of control points should spread over all the different planes. In general, we should be selecting the control points distributed broadly across the field of view. The determination of two points on the same plane can be made based on their depth values which can be obtained in a stereo vision system or motion in for a monocular video camera system[3, 17, 18, 19].

We developed the following two algorithms based on the above two principles for selecting a set of control points \( \Omega_{op} \) for the construction of \( I_{P_{op}} \) and \( E_{P_{op}} \). For each algorithm we use a training set

\[
\Omega = \{[(X^1_v, Y^1_v, Z^1_v), (x^1_v, y^1_v)], [(X^2_v, Y^2_v, Z^2_v), (x^2_v, y^2_v)], \ldots, [(X^n_v, Y^n_v, Z^n_v), (x^n_v, y^n_v)]\}, n \geq 7.
\]

We refer \([(X^1_v, Y^1_v, Z^1_v), (x^1_v, y^1_v)]\) to as a triplet.

**Alg1 K-fold Cross Validation method**

This method employs a variant of cross-validation known as *k-fold cross validation* used in training an intelligent system. The method has the following computational steps.

[step 1] Data set \( \Omega \) of \( n \) examples is divided into \( k \) subsets of approximately equal size, \( \Omega_1, \ldots, \Omega_k \). If \( n \) is not divisible by \( k \), then the remaining elements are distributed among the \( k \) subsets such that no subset should have more than one extra data element compared to the other subsets. If a prior knowledge of points of different planes is available, each subset should contain points of different plane.

[step 2] Construct the intrinsic and extrinsic parameters on the \( k-1 \) subsets and validated on the \( i \)th subset for \( i = \{1, 2, 3, \ldots, k\} \).

[step 3] Set \( I_{P_{op}} = I_{P_j} \) and \( E_{P_{op}} = E_{P_j} \) such that the intrinsic and extrinsic parameters generated at the \( j \)th step that give the least overall errors on the validated subset only for all \( i, i = \{1, 2, 3, \ldots, k\} \) and \( \Omega_{op} = \Omega_j \).

A generalization performance measure can be obtained by averaging the validation errors of the \( k \) validation sets.

**Alg2: Best One First algorithm**

The second algorithm starts on constructing a set of \( I_P \) and \( E_P \) using the entire set of the corresponding points \( \Phi \). Then it selects 11 points that give the minimum projection errors to construct another set of \( I_P \) and \( E_P \)s. It evaluates the projection error over all the data and selects the best point at a time until it reaches the whole set of available data. A set of control points is the set of corresponding points that generates the \( I_P \) and \( E_P \) that gives the minimum project errors. Specifically it has the following computational steps.
[step 1] Construct the intrinsic and extrinsic parameters denoted as $I_P$ and $E_P$ using all points in $\Phi$.

[step 2] Calculate projection error of every triplet in $\Phi$.

[step 3] Put 11 triplets in $\Sigma$ that give the minimum projection errors, and set $i=1$.

[step 4] Construct the intrinsic and extrinsic parameters denoted as $I_{P_i}$ and $E_{P_i}$ using all points in $\Sigma$.

[step 5] Calculate the projection error on every triplet in $\Sigma$ and $(\Phi - \Sigma)$ using $I_{P_i}$ and $E_{P_i}$ and the average errors on $(\Phi - \Sigma)$, $Ave_{error-validation}^i$.

[step 6] If $|\Sigma| < |\Phi|$, then $\Sigma = \Sigma \cup \{\text{triplet} \mid \text{triplet} \in \Phi - \Sigma \text{ that has the minimum error}\}$ and go to step 5.

[step 7] Stop the process and return the $I_{P_i}$ and $E_{P_i}$ that give the minimum $Ave_{error-validation}^i$.

Both algorithms use the following procedure to calculate the projection error at each step. For a given set of points, $\Psi = \{(X_1^1, Y_1^1, Z_1^1), (x_1^1, y_1^1, z_1^1), \ldots, (X_n^n, Y_n^n, Z_n^n), (x_n^n, y_n^n, z_n^n)\}$, we calculate $(x_{dj}^i, y_{dj}^i)$ following the system of equations given in (2) for $i = 1, \ldots, n$, using the intrinsic and extrinsic parameters, $I_{P_i}$ and $E_{P_i}$ obtained at step $j$ in each algorithm. The average error on $\Psi$ at step $j$ is calculated as follows:

$$E_{\Psi}^j = \frac{1}{n} \sum_i \sqrt{(x_{dj}^i - x_i^j)^2 + (y_{dj}^i - y_i^j)^2}$$

and the projection error of $i$th point is $E_i^j = \sqrt{(x_{dj}^i - x_i^j)^2 + (y_{dj}^i - y_i^j)^2}$.

We have conducted two sets of experiments to evaluate the above two algorithms. The intrinsic and extrinsic parameters are tested on the test set. The error on the test set gives a realistic measure of the system performance on the future unseen data set.

Figure 5 shows the two pairs of the images used in our experiments. In the first experiment, we took a pair of images with two different camera views outside our research lab and manually measured the 3D coordinates of the selected points in the left image. In the second experiment, we used two virtual database images of two different camera views. For the first image pair we obtained 32 corresponding points and the second pair 49 points. For convenience, we refer these two sets of data points as dataset 1 and 2 respectively. Each dataset was divided into two subsets: training and blind test set. The training data are used by each algorithm to select an optimal set of control points.
The graphs in Figure 6 and 7 show the performance of the intrinsic and extrinsic parameters generated by the data points selected at each step of each algorithm on two sets of images. Table 1 shows the performance on the training, validation and blind test data using the I_P and E_P generated by the two algorithms on the two image sets. The blind test data in the dataset1 contains 7 points and the blind test data in the dataset2 contains 11 points.
Figure 6  Performance of optimal control point selecting algorithms on dataset1.
Table 1. The performance of the two algorithms on the training, validation and blind test sets.

<table>
<thead>
<tr>
<th>algorithm</th>
<th>dataset1</th>
<th></th>
<th>dataset2</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Training</td>
<td>Validation</td>
<td>test data</td>
<td>Training</td>
</tr>
<tr>
<td>1</td>
<td>(20 points)</td>
<td>1.26206</td>
<td>1.22572</td>
<td>(7 points)</td>
</tr>
<tr>
<td>2</td>
<td>(11 points)</td>
<td>2.95156</td>
<td>2.9559</td>
<td>(14 points)</td>
</tr>
</tbody>
</table>

Table 2. The performance of the I_P and E_P generated using all corresponding points on the blind test data.

<table>
<thead>
<tr>
<th>dataset1</th>
<th></th>
<th>dataset2</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>(25 points)</td>
<td>test data</td>
<td>(7 points)</td>
</tr>
<tr>
<td>4.06792</td>
<td>4.7</td>
<td>0.630967</td>
<td>1.32446</td>
</tr>
</tbody>
</table>

The Best One First algorithm performed very well on the second data set but poorly on the first data set. The K-fold Cross Validation algorithm gave very good performance on both sets of experiments. The average error on the test set in both cases less than two pixels. More importantly the K-folder Cross Validation algorithm appears to have better generalization and give more stable results: the performance on the blind test data was very close to the performance on the training and validation data. Table 2 shows the performance on the two blind test data when the I_P and E_P were generated using all the points in the training data set. For the first data set, the 20 control points selected by the K-folder Cross Validation algorithm outperformed the 25 control points shown in Table 2. For the second data set, the
32 control points selected by the K-folder Cross Validation algorithm performed just as good as the 40 control points shown in Table 2. This result shows again the a subset of corresponding points can generate more optimal I_P and E_P than using the entire set.

6. An application case

We have applied the proposed system illustrated in Figure 2 to registering virtual database image and a video image so that we are able to place targets, which could be difficult to identify in the video image or hidden from the video camera view, accurately in the real scene. We use the pair of virtual database image and a video image to shown Figure 3 to register the virtual and real scene. Figure 3 (a) showed the LSFs extracted from the virtual image and Figure 3(b) showed the LSFs in the real image that match the LSFs in the virtual image. Figure 4 showed 30 corresponding points extracted from both the virtual and real images. We used the K-fold cross validation algorithm to select a set of 24 control points and implemented Tsai’s camera calibration model [5] to construct the intrinsic, extrinsic, and the lens distortion parameters from the selected 24 control points. Figure 8 shows the results of mapping the object points in blue color in a virtual image to the real view image shown in red color using the intrinsic and extrinsic parameters generated by our system. Figure 9 shows the results of mapping the points of a virtual object(shown in RED) that is not visible in the real scene to a real scene image using the same intrinsic and extrinsic parameters. All these results are very satisfactory.

Figure 8: An example of mapping virtual objects to an image of real scene. (a) shows the virtual image containing the control points and three virtual objects. (b) shows a real view image with mapped virtual objects shown in “RED.”
7. Conclusion
We have presented a framework that registers virtual database image with real scene images and places virtual objects in the real scene images. We defined a reliable type of image features, LSFs, for image registration, and presented algorithms that extract LSFs from a database image, and match the LSFs with the LSFs within a real view image. We also presented algorithms that extract a set of reliable corresponding points from the matched LSFs and select an optimal set of control points from a given set of corresponding points. We have showed in this paper that the intrinsic and extrinsic parameters generated by less number of control points can give better performance. The algorithms for extracting and matching linear structural features are applicable to all scenes containing reliable rigid body objects. The algorithms that select optimal control points are applicable to general image registration problems beyond virtual and real imageries.

8. Acknowledgment
The research project is supported in part by a contract from U.S. Army TACOM-TARDEC, Vetronics Technology Center.

9. References


[17] Hong Guo and Yi Lu “Depth detection of targets in A monocular image sequence,“ 18th Digital Avionics Systems Conference Gateway to the New Millennium October 24 - October 29, 1999 St. Louis, Missouri
