Matching 2-D Ellipses to 3-D Circles with Application to Vehicle Pose Identification

Marcus Hutter and **Nathan Brewer**

RSISE @ ANU and SML @ NICTA

Canberra, ACT, 0200, Australia

nbrewer@cecs.anu.edu.au marcus@hutter1.net

December 2009

Abstract

Finding the three-dimensional representation of all or a part of a scene from a single two dimensional image is a challenging task. In this paper we propose a method for identifying the pose and location of objects with circular protrusions in three dimensions from a single image and a 3d representation or model of the object of interest. To do this, we present a method for identifying ellipses and their properties quickly and reliably with a novel technique that exploits intensity differences between objects and a geometric technique for matching an ellipse in 2d to a circle in 3d.

We apply these techniques to the specific problem of determining the pose and location of vehicles, particularly cars, from a single image. We have achieved excellent pose recovery performance on artificially generated car images and show promising results on real vehicle images. We also make use of the ellipse detection method to identify car wheels from images, with a very high successful match rate.

Contents

Keywords

computer vision; image recognition/processing; ellipse detection; 3d models; 2d-ellipse to 3d-circle matching; single image pose identification; wheel detection; 3d vehicle models.

1 Introduction

Determining three-dimensional information from a single two-dimensional image is a common goal in computer vision [HZ04]. Circular features appear frequently in real world objects, the mouth of a cup, the wheels of a car, the iris of a human eye. Such circular features (referred to as 3d circles from here on) appear as ellipses when imaged if viewed non-frontally.

In this paper we present a method for matching parts of a two-dimensional scene to existing three-dimensional models using a technique that maps three-dimensional circles to two-dimensional ellipses. Additionally, we present a novel method for identifying elliptical regions and the properties thereof in a two-dimensional image. In particular, we aim to recover the pose of motor vehicles in natural scenes. The purpose of this is twofold: First, it allows us to determine the location and facing of a car in a street scene, which has obvious uses in driver assistance systems, and second it allows us to place vehicles in three dimensions from a single image without constructing a depth map of the entire scene. It is important to note that this paper does not present a general-purpose vehicle detection or complete scene reconstruction algorithm, rather it aims to deliver a basis from which such platforms can be built or improved.

Intelligent driver assist technology is becoming more and more important as our roads become busier. Determining vehicle pose in three dimensions allows us to estimate a heading for a visible vehicle, which can alert the driver to approaching vehicles. Importantly, our pose detection technique requires only a single frame, allowing it to detect stationary vehicles as potential hazards in case of unexpected motion, unlike many techniques which require a series of images of moving vehicles.

Developing a depth map of a scene from a single image is an underconstrained problem. There are approaches based on machine learning [SSN07], or that make use of the effects of fog [Tan08], but these either require classification and a large training data set or specific scene conditions. Our method is able to place 3D models of imaged vehicles at their locations in the same orientation as in the real world. This has applications for security, as observers will be able to determine the position of a vehicle from surveillance footage, as well as scene reconstruction. Unlike the work of Hinz et al. [HSR03], which places simple 3d car models in vehicle locations in aerial photographs, we want to use a specific, detailed 3d model to match to street-level car images.

In order to do this, we first must isolate ellipses in the image that we can then match to circular areas in a three-dimensional model. For vehicles, the most applicable choice for these ellipses are the wheels. Existing ellipse detection methods, such as the Hough transform [TM78] require a five-dimensional search, which has high computational and memory requirements. There are many recent methods methods which aim to reduce this by using parametrised spaces for the Hough transform accumulator, such as [CLER07, Mcl98]. These methods still require edge extraction or other image preprocessing, and often detect significantly more ellipse than we

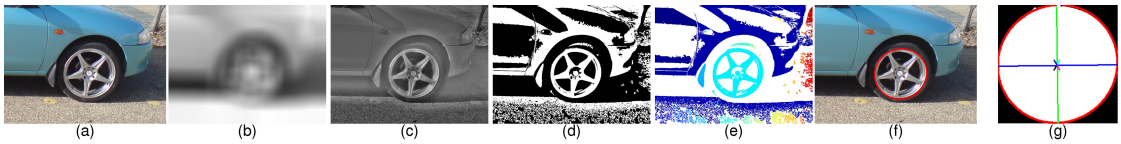

Figure 1: Images showing each step of the wheel detection Algorithm 1. (g) shows the identified wheel ellipse with the major and minor axes shown in blue and green respectively. the four lines in the center represent the potential φ normal vectors, scaled 50 times for clarity. Note that each φ vector has a large component pointing into the page.

need in this paper. To this end, we present a new method for identifying wheels in real-world vehicle images which incorporates a significant amount of ellipse filtering and parameter estimation as part of the detection process. Other feature detection methods are discussed more fully in the text.

While this work is focussed on identifying the pose of cars, the ellipse to circle mapping process can be applied to any three dimensional object with circular features, as shown in the experimental verification of Section 3.

2 2D Ellipse Detection

Rather than using an existing ellipse detection method based on edge detection, such as those presented in [MHZS08, KO04], we propose a method for detecting elliptical objects which allows for significant noise at the edge of the object. The ellipse detection method presented here involves several steps. First, pixels which are brighter than their surroundings are identified by comparing each pixel to the average local intensity. Connected components, or blobs, are then found and labeled. Further processing is performed on each of these blobs to fill gaps and then to determine whether the detected object is elliptical. While we use bright image regions here, the method can be easily changed to identify dark regions or regions which adhere to some other color or intensity property. This process is summarised in Algorithm 1 and visually in Figure 1.

Identifying Comparatively Bright Pixels. In order to identify pixels which are brighter than their surrounding area, a new image which reflects the local mean intensity for each pixel is built. Each pixel in this new image contains the mean intensity for a square region centered on that pixel in the original image. This square is set to $b\%$ of the shortest image side. For the purposes of this paper, a window 10% of the width of the image was found to be effective, but this can depend on the nature of the ellipses that are desired. If this window would extend beyond the edges of the image for a particular pixel, the mean of the closest possible window which is entirely contained within the image is used instead. The intensity integral

Data: A greyscale or full-colour image (Figure 1a)
Result: Location and parameters of ellipses in the image (Figure 1f)
1 Smooth image with square kernel (Figure 1b);
2 Normalize Image (Figure 1c);
3 Threshold Normalised Image (Figure 1d);
4 Detect and label connected regions (Figure 1e);
5 Star Fill connected regions (Figure 1g);
6 Get Ellipse parameters using Eq.(1);
7 Filter out non-ellipses by comparison to expected ellipse (Figure 1g);
8 Determine Ellipse Normal using Eq.(4) (Figure 1g);

Algorithm 1: The Ellipse Detection Algorithm. For more details, see text and Figure 1.

image [VJ04] is used to efficiently calculate this windowed mean value, which was the primary motivation for using a simple box kernel for smoothing.

We then subtract this average from the intensity of each pixel in the initial image, and then set a threshold T which defines how much a pixel must exceed the local average to be flagged as 'bright'. For wheel detection, one standard deviation above the mean of the average image was found to produce good results. This thresholding produces a binary image, with zero corresponding to 'dark' pixels, and one corresponding to 'bright' pixels.

This thresholding method tends to produce a thick 'edge' around the ellipse perimeter as seen in Figure 3 (ii), and frequently contains small gaps corresponding to darker areas of the ellipse surface, corresponding to occlusions, decoration or simply image noise.

Labeling and Filling Connected Regions. To denoise the binary image obtained above, it is first necessary to eliminate small bright regions. Simple blob extraction is then performed, labeling each 8-connected region with a unique identifier.

In this ellipse detection method, objects being investigated must be filled. As it is not desirable to affect the overall shape of a blob, the fill method chosen preserves the perimeter and fills gaps in a radial fashion. In each blob, if a pixel is marked as bright then all pixels on a line between it and the blob center are also marked as bright. We are able to do this efficiently, in linear time, by spiralling in from the edge to the centre. This fill method is used rather than a simple flood fill as blobs frequently do not have fully enclosed edges due to noise or occlusions. Each of these filled, denoised blobs is then separately investigated to determine if it is elliptical in nature. See the following section and Figure 3 (iii) for more details.

Extraction of Blob Properties and Ellipse Comparison. In order to determine whether a particular blob is elliptical, the extracted blob is compared to an equivalent ellipse. We do this by finding the mean and covariance matrix of each blob. Let $W \subset \mathbb{Z} \times \mathbb{Z}$ be the pixels of one. The area $|W| \in \mathbb{N}$, mean $\boldsymbol{\mu} \in \mathbb{R}^2$ and

covariance matrix $C \in I\!R^{2\times 2}$ of each blob can then be found as follows:

$$|W| = \sum_{\boldsymbol{p}\in W} 1, \qquad \mu \;=\; \frac{1}{|W|}\sum_{\boldsymbol{p}\in W} \boldsymbol{p},$$

$$C = \frac{1}{|W|}\sum_{\boldsymbol{p}\in W} (\boldsymbol{p}-\boldsymbol{\mu})(\boldsymbol{p}-\boldsymbol{\mu})^\top$$

By assuming that the blob is an ellipse, the major and minor axes of the corresponding ellipse can be calculated from the eigenvalues a_1 and a_2 of the blob covariance matrix, and the center of the corresponding ellipse is given by $\boldsymbol{\mu}$. From this information, we are able to find the shape of the corresponding ellipse by Eq.(1). This approach is similar to that described in [SRSBT91]. This step is shown in Figure 3 (iii). We are able to perform a consistency check at this stage to eliminate some obvious non-ellipses: If W is an ellipse, its area is given by πab as well as $|W|$. Thus if $||W| - 4\pi\sqrt{\det C}| > T$ for some small threshold T, we can immediately discard this blob.

We can then directly compare the remaining blobs to the equivalent ellipse on a per pixel basis by counting the number of pixels in which the blob and its ellipse do not match. As we are interested in the elliptical nature of the blob regardless of size, we divide the number of mismatches by the blob area. Blobs with a sufficiently low mismatch can be regarded as ellipses. We found experimentally that a mismatch ratio of 20% identifies human-visible ellipses while discarding non-ellipses.

3 Mapping a 2D Ellipse to a 3D circle

Circles in 3d space become ellipses in 2d space after parallel projection. It is possible to exploit this fact to determine the pose of a 3d object containing circular features from a projected image.

Describing an Ellipse by its Covariance Matrix. First, we describe an ellipse in a way that is easily manipulated. Let

$$E(\boldsymbol{\mu}, C) \;:=\; \{\boldsymbol{p} \in I\!R^2 : (\boldsymbol{p}-\boldsymbol{\mu})^\top C^{-1}(\boldsymbol{p}-\boldsymbol{\mu}) \le 4\} \tag{1}$$

describe a solid ellipse with center $\boldsymbol{\mu}$, where C is a symmetric positive definite matrix. It can be shown by integration that C is the covariance of the ellipse given by $E(\boldsymbol{\mu}, C)$. The eigenvalues $\lambda_1 > \lambda_2$ of C can be computed from the trace $\mathrm{tr}C = \lambda_1 + \lambda_2$ and the determinant $\det C = \lambda_1\lambda_2$:

$$\lambda_{1/2} = \tfrac{1}{2}\mathrm{tr}C \pm \sqrt{(\tfrac{1}{2}\mathrm{tr}C)^2 - \det C} \tag{2}$$

Therefore, the lengths of the major and minor axes of the ellipse are

$$a_1 = 2\sqrt{\lambda_1} \quad \text{and} \quad a_2 = 2\sqrt{\lambda_2} \tag{3}$$

Transforming a 3D circle to a 2D ellipse. In Lemma 1, we describe how a 3d circle projects to an ellipse in 2d.

Lemma 1 *Let*

$$\mathit{Circle} := \{\boldsymbol{x} \in I\!R^3 : ||\boldsymbol{x}|| \leq r \ \ \& \ \ \boldsymbol{x}^\top \boldsymbol{\varphi} = 0\}$$

be a centered circle with radius r and unit normal $\boldsymbol{\varphi} \in I\!R^3$. Let $E(\boldsymbol{\mu}, \boldsymbol{C})$ be the orthogonal projection of Circle along the z axis. Then $r = a_1$ and $\boldsymbol{\varphi}$ and C are related by

$$\boldsymbol{\varphi} \ \equiv \ \begin{pmatrix} \varphi_x \\ \varphi_y \\ \varphi_z \end{pmatrix} \ = \ \frac{1}{a_1} \begin{pmatrix} \pm\sqrt{a_1^2 - 4C_{xx}} \\ \pm\sqrt{a_1^2 - 4C_{yy}} \\ \pm a_2 \end{pmatrix} \tag{4}$$

where a_1 and a_2 are defined in (3) and (2).

Proof. The (x, y) coordinates of $\boldsymbol{x} = (x, y, z) \in$Circle describe an ellipse. Condition $\boldsymbol{x}^\top \boldsymbol{\varphi} = 0$ implies $z\varphi_z = -x\varphi_x - y\varphi_y$. Inserting this into $||\boldsymbol{x}||$ yields

$$\begin{aligned} ||\boldsymbol{x}||^2 \ &= \ x^2 + y^2 + z^2 \ = \ x^2 + y^2 + \left(\frac{x\varphi_x + y\varphi_y}{\varphi_z}\right)^2 \\ &= \ \begin{pmatrix} x \\ y \end{pmatrix}^\top \begin{pmatrix} 1 + \varphi_x^2/\varphi_z^2 & \varphi_x\varphi_y/\varphi_z^2 \\ \varphi_x\varphi_y/\varphi_z^2 & 1 + \varphi_y^2/\varphi_z^2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \end{aligned} \tag{5}$$

Hence the projected circle is an ellipse with covariance

$$\begin{aligned} C \ \equiv \ \begin{pmatrix} C_{xx} \, C_{xy} \\ C_{yx} \, C_{yy} \end{pmatrix} \ &= \ \frac{r^2}{4} \begin{pmatrix} 1 + \varphi_x^2/\varphi_z^2 & \varphi_x\varphi_y/\varphi_z^2 \\ \varphi_x\varphi_y/\varphi_z^2 & 1 + \varphi_y^2/\varphi_z^2 \end{pmatrix}^{-1} \\ &= \ \frac{r^2}{4} \begin{pmatrix} 1 - \varphi_x^2 & -\varphi_x\varphi_y \\ -\varphi_x\varphi_y & 1 - \varphi_y^2 \end{pmatrix} \end{aligned} \tag{6}$$

where we have exploited $||\boldsymbol{\varphi}|| = 1$ when inverting the matrix. Conversely, given an ellipse with covariance C, using (2), (3), and (6), the radius and normal of the underlying 3d circle are given by (4). ■

There is one hard sign constraint, namely $\text{sign}(\varphi_x\varphi_y) = -\text{sign}C_{xy}$. Since a projected circle is usually only visible from one side of the 3d object, we define $\boldsymbol{\varphi}$ as pointing away from the observer into the image. This fixes the sign of $\pm a_2$, leaving two solutions which can not be disambiguated. These solutions arise from the fact that an ellipse can be projected from a circle facing in two different directions.

Accuracy. We give some rough estimates of the accuracy of the reconstructed $\boldsymbol{\varphi}$. In particular we show that reconstruction from a frontal view is somewhat surprisingly harder than from a tilted view.

Let $\theta = \arccos|\varphi_z| \in [0; \pi/2]$ be the angle by which the Circle is tilted out of the (x, y)-plane, and let Δa_2 be the accuracy to which the minor axis a_2 could be determined by the ellipse detection Algorithm 1. For "perfect" pixel ellipses, an accuracy between $\Delta a_2 \approx 1$ (pixel accuracy) and $\Delta a_2 \approx 1/a_2$ (sub-pixel accuracy

due to averaging/anti-aliasing) is achievable. From $a_2 = r|\varphi_z| = r\cos\theta$, we get by Taylor expansion,

$$\Delta a_2 \;=\; -r\sin\theta\,(\Delta\theta) - \tfrac{1}{2}r\cos\theta(\Delta\theta)^2 + O((\Delta\theta^3))$$

Hence for a non-frontal view ($\theta \gg \Delta\theta$), the orientation of the circle axis $\boldsymbol{\varphi}$ can be determined with accuracy $\Delta\theta \approx |\Delta a_2|/r\sin\theta$, while for a frontal view ($\theta \ll \Delta\theta$), the accuracy reduces to $\Delta\theta \approx \sqrt{2|\Delta a_2|/r}$. This is consistent with our experimental results on real cars like those in Figure 5.

Determining the Mapping from a Circle to an Ellipse. We now want to determine the orthogonal projection that maps a general circle in 3d to a general ellipse in 2d. Let $\boldsymbol{\nu}, \boldsymbol{\phi} \in I\!R^3$, and $R > 0$ be the center, normal, and radius of the circle, let $\boldsymbol{\mu} \in I\!R^2$ and $C \in I\!R^{2\times 2}$ be the center and covariance matrix of the ellipse, and let

$$\boldsymbol{x}' = \sigma Q\boldsymbol{x} + \boldsymbol{q} \tag{7}$$

be the orthogonal projection to be determined. Matrix $Q \in I\!R^{2\times 3}$ consists of the first two rows of an orthogonal matrix $\tilde{Q} \in I\!R^{3\times 3}$, $\sigma > 0$ is a scaling factor, and $\boldsymbol{q} \in I\!R^2$ a shift vector. The centers $\boldsymbol{\nu}$ and $\boldsymbol{\mu}$ must match, hence $\boldsymbol{q} = \boldsymbol{\mu} - \sigma Q\boldsymbol{\nu}$. The major ellipse axis a_1 corresponds to the circle radius R by (4), hence $\sigma = a_1/R$. Let $\boldsymbol{\varphi}$ be the "normal" of the ellipse as defined in Lemma 1. Note that there are two possibilities for this vector, as there is a sign ambiguity in (4). We use the convention that $\boldsymbol{\varphi}$ and $\boldsymbol{\phi}$ point from ellipse center away from the observer ($\varphi_z < 0$). Axle $\boldsymbol{\phi}$ "projects/rotates" to $\tilde{Q}\boldsymbol{\phi}$, which must align with $\boldsymbol{\varphi}$, hence we have the constraint $\boldsymbol{\varphi} = \tilde{Q}\boldsymbol{\phi}$ on $\tilde{Q}$ (and hence Q).

An orthogonal matrix has three degrees of freedom (dof), the constraint $\boldsymbol{\varphi} = \tilde{Q}\boldsymbol{\phi}$ determines 2 dof, hence 1 dof remains, namely after rotating $\boldsymbol{\phi}$ into $\boldsymbol{\varphi}$, we can freely rotate around $\boldsymbol{\varphi}$.

A rotation around a unit axle $\boldsymbol{u} \in I\!R^3$ by angle α can be represented by quaternion [FP02]

$$\begin{aligned} q \;&:=\; a + bi + cj + dk \;\equiv\; a + (b,c,d) \\ &:=\; \cos\tfrac{\alpha}{2} + \boldsymbol{u}\sin\tfrac{\alpha}{2}, \quad ||\boldsymbol{u}|| = 1 \end{aligned} \tag{8}$$

which has the following matrix representation

$$\begin{pmatrix} a^2{+}b^2{-}c^2{-}d^2 & 2bc{-}2ad & 2ac{+}2bd \\ 2ad{+}2bc & a^2{-}b^2{+}c^2{-}d^2 & 2cd{-}2ab \\ 2bd{-}2ac & 2ab{+}2cd & a^2{-}b^2{-}c^2{+}d^2 \end{pmatrix} \tag{9}$$

In our case, a rotation around $\boldsymbol{\phi} \times \boldsymbol{\varphi}$ by angle $\alpha = \cos^{-1}(\boldsymbol{\phi} \circ \boldsymbol{\varphi}) \in [0;\pi]$ rotates $\boldsymbol{\phi}$ into $\boldsymbol{\varphi}$. Using

$$\begin{aligned} &\cos\tfrac{\alpha}{2} = \sqrt{\tfrac{1}{2}(1+\cos\alpha)} \quad \text{and} \\ &\sin\tfrac{\alpha}{2} = \text{sign}(\alpha)\sqrt{\tfrac{1}{2}(1-\cos\alpha)} \quad \text{for} \quad \alpha \in [-\pi;\pi] \end{aligned} \tag{10}$$

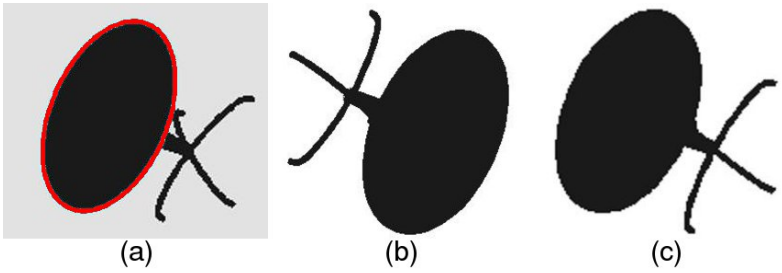

Figure 2: An image which shows (a) an image rendered from a 3d model of a table with the ellipse to be matched highlighted in red and (b,c) the possible model poses determined by the algorithm

Table 1: The true and estimated pose of a simple 3d table.

True Pose			Estimated Pose		
Axle $\boldsymbol{\varphi}$	Scale σ	Shift $\boldsymbol{q}$	Axle $\boldsymbol{\varphi}$	Scale σ	Shift $\boldsymbol{q}$
[0.7568 0.3243 -0.5676]	6	[400 340]	[0.7296 0.3193 -0.6048]	6.06	[405 340]

leads to the following convenient representation:

$$a = \cos\tfrac{\alpha}{2} = \sqrt{\tfrac{1}{2}(1+\boldsymbol{\phi}\circ\boldsymbol{\varphi})}, (b,c,d) = \boldsymbol{u}\sin\tfrac{\alpha}{2} = \tfrac{1}{2a}(\boldsymbol{\phi}\times\boldsymbol{\varphi}) \tag{11}$$

This representation only breaks down if $\boldsymbol{\phi}$ is exactly antiparallel to $\boldsymbol{\varphi}$. In this case, any $\boldsymbol{u}$ orthogonal to $\boldsymbol{\phi}$ will do.

Subsequently, a rotation around $\boldsymbol{u} = \boldsymbol{\varphi}$ by an arbitrary angle β can be applied. Hence the complete transformation is as follows:

Lemma 2 *Eq.(7) projects 3d circle* $(\boldsymbol{\nu}, \boldsymbol{\phi}, R)$ *= (center,axle,radius) to 2d ellipse* $(\boldsymbol{\mu}, C)$ *= (center,covariance) iff*

$$Q = \textit{first 2 rows of } \tilde{Q}, \tilde{Q} = \tilde{Q}_2\tilde{Q}_1, \boldsymbol{q} = \boldsymbol{\mu} - \sigma Q\boldsymbol{\nu}, \sigma = a_1/R,$$

Minor axis a_1 *is given by (2) and (3). Matrix* $\tilde{Q}_1$ *is given by (9) and (11) and (4). Matrix* $\tilde{Q}_2$ *is given by (8) and (9) with arbitrary* $\alpha = \beta \in [-\pi;\pi]$ *but now with* $\boldsymbol{u} := \boldsymbol{\varphi}$ *given by (4).*

Experimental Verification. In order to test this 2d-3d matching procedure, we generated test images from a simple 3d table model to determine its accuracy. The table was arbitrarily rotated, scaled and shifted (see Figure 2) and then projected to produce a 2d image. The center ($\boldsymbol{\mu}$), major and minor axis lengths (a_1 and a_2 respectively) and covariance (C) of the projected ellipse corresponding to the top of the table were determined using the ellipse detection method presented above. This

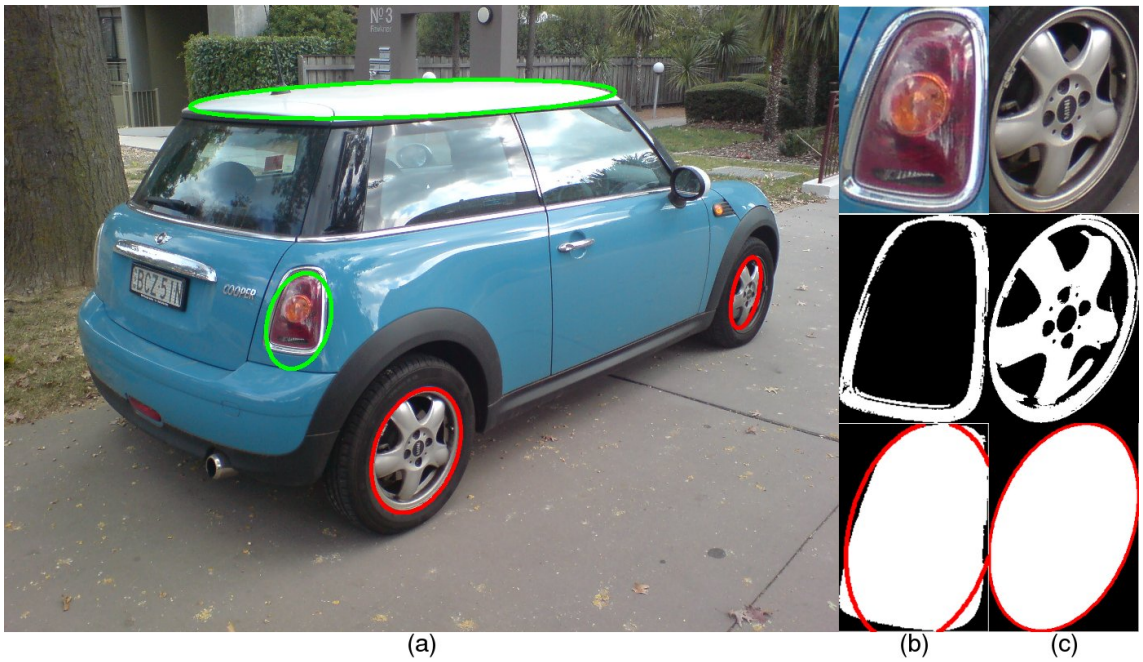

Figure 3: (a) An image of a Mini Cooper showing in blue all detected ellipses and in red the ellipses detected as wheels (b and c) zoomed sections of the car containing a wheel and a non-wheel region, respectively. (i) Zoomed region of image, (ii) the pixels identified as 'bright', (iii) Filled blobs and corresponding expected ellipses in red

information and Eq.(4) are used to estimate the normal to this ellipse. Lemma 2 was then used to estimate Q, σ and $\boldsymbol{q}$, which could then be compared to the input rotation, shift and scale. Table 1 shows the recovered and input values for the table, and Figure 2 displays graphically the initial and recovered poses. Note that the only the two sets of results produced by the sign ambiguity in (4) are presented in this figure. We have assumed that the normal points towards the page.

4 2D Wheel Detection

In order to match a 3d model of a vehicle to a 2d image with an ellipse to circle mapping technique, it is first necessary to find the ellipses corresponding to circular features on the vehicle in the image. Wheels are used for this purpose, as they are both easily locatable in the 2d representation and are present in all vehicles. This also allows for the matching of an image to a generic vehicle, should a specific model be unavailable.

For the purposes of this section, it is assumed that the image being investigated features a vehicle as its primary object. It is also assumed that the image contains

the two wheels on the same side of the vehicle clearly, and that the vehicle is upright in the image. This image can be a segment of a larger image which an existing vehicle detection system, such as [MSB08], has identified as a vehicle. We make use of wheels because they have a number of exploitable properties. If the wheels are not visible in the input image, an alternative technique must be used.

A typical car has hubcaps that are significantly brighter than the surrounding tyre, appearing as a comparatively bright, unobstructed ellipse. This allows us to use the ellipse detection method presented above, with a few additional steps to filter out detected non-wheel ellipses. While it is the case that some vehicles have unusually dark hubcaps or rims, simple observation of vehicles in the street shows that this is far from the norm, allowing our method to applicable in the majority of cases where a side view of a vehicle is available.

This procedure identifies all 'bright' ellipses in the image, and can identify road markings, headlights and other circular regions in the image. As we want only to identify the wheels, we need to identify the two ellipse that are most likely to be wheels. To do this, we make a few additional assumptions about the way a vehicle appears in an image. First, in a vehicle the wheels cannot be too large or small, so filled blobs containing less than 0.15% or more than 25% of the image area are excluded.

The remaining blobs are then examined in a pairwise fashion to determine the relative ellipse orientation of each pair and the orientation of a line drawn between the centers of each ellipse in the pair. Pairs are excluded as wheels if the ellipses have a significantly different orientation or if the line connecting their centers deviates more from the horizontal than is expected in a natural image. If this process results in more than a single set of wheels being detected, the pair with the lowest mismatch ratio are identified as wheels. The properties of these ellipses can then be matched to the circular wheels of a 3d car model, as described in Sections 3 and 5.

There are other methods which can be used to identify the wheels in a vehicle image. For example, a cascaded classifier system similar to that used for face detection [VJ04] could be adapted to find wheels in images, just as it has been adapted to detect road signs [PPA08]. Similarly, a robust feature description for a wheel could be developed using SIFT [Low99] or sift like features, which could be used to find wheel locations. These approaches suffer from two major drawbacks which our approach does not. First, these methods require labelled training data to learn model parameters, which requires significant amounts of human input. Additionally, the hubcaps/rims of almost every vehicle are at least subtly different, and the direction from which they are imaged has a large impact on their appearance, and so constructing a training set with sufficient detail becomes even more arduous. Second, these approaches only identify the location of a wheel, and further processing must be done to extract the ellipse parameters that we require for 2d-3d matching.

5 Determining Vehicle Pose from a Single Image

In the previous section, we have presented a method for both identifying the wheels in a 2d image of a vehicle and determining their ellipse properties. In this section, we present a method for using these properties together with those from section 3 to determine the 3d pose of a vehicle from a single 2d image. It is possible to determine the location and orientation of a vehicle to within a plane of rotation using the method presented in section 3. We perform this matching on the rear wheel, as the front wheel might be rotated out of plane with the rest of the vehicle, which results in poor pose matching. As input images contain two ellipses, corresponding to each of the two wheels of the vehicle, it is possible to use information about the second ellipse to eliminate this rotational ambiguity. Additionally, information about the second wheel can improve the scale factor σ of the transformation.

We determine the "normal" of the ellipse for both wheel ellipses in the 2d image using the method shown in Section 3. The parameters σ and β in Lemma 2 are found by matching the center $\boldsymbol{\nu}'$ of the 3d model front wheel to the corresponding ellipse center $\boldsymbol{\mu}'$ in the image.

The 2d ellipse detection cannot determine which is the front and which the back wheel, and also does not know which side of the car the two wheels belong to. Together with the two solutions from (4), we hence have a discrete 8-fold ambiguity. We are able to eliminate four of these by assuming that the vehicle in the image is upright, which means that the rear wheel on the left hand side of the model matches the left hand wheel in the image if the car is oriented with the front towards the right of the image, and the 3d right rear wheel matches the right-hand image ellipse.

First, we determine a more accurate scale factor by finding the length $||\boldsymbol{\Delta}||$ of the vector $\boldsymbol{\Delta}$ between the front and rear wheels in the 3d model. This distance is invariant to model rotation, and so we are able to find this easily. We then estimate the 3d distance between wheels in the 2d image by setting $(\delta_x, \delta_y)^\top := \boldsymbol{\mu}' - \boldsymbol{\mu}$ as the vector from the back to the front wheel in the image. We know that $\boldsymbol{\delta} \in I\!R^3$ should be orthogonal to $\boldsymbol{\varphi}$, which gives us $\delta_z = -(\delta_x\varphi_x + \delta_y\varphi_y)/\varphi_z$. We then set $\sigma = ||\boldsymbol{\delta}||/||\boldsymbol{\Delta}||$.

Lemma 3 *The 3d back wheel with center $\boldsymbol{\nu} \in I\!R^3$ and axle $\boldsymbol{\phi} \in I\!R^3$ and 3d front wheel with center $\boldsymbol{\nu}'$ project to 2d ellipses with center $\boldsymbol{\mu} \in I\!R^2$ and covariance matrix $C \in I\!R^{2\times 2}$ and center $\boldsymbol{\mu}' \in I\!R^2$ respectively by Lemma 2 with the following adaptation: Scale $\sigma := ||\boldsymbol{\delta}||/||\boldsymbol{\Delta}||$ and β is now constrained by*

$$\cos\beta = \frac{\boldsymbol{\delta} \circ \boldsymbol{\Delta}}{||\boldsymbol{\delta}||||\boldsymbol{\Delta}||}, \quad \sin\beta = \frac{\det(\boldsymbol{\delta}, \boldsymbol{\Delta}, \boldsymbol{\varphi})}{||\boldsymbol{\delta}||||\boldsymbol{\Delta}||} \tag{12}$$

$$(\delta_x, \delta_y)^\top := \boldsymbol{\mu}' - \boldsymbol{\mu}, \quad \delta_z := -(\delta_x\varphi_x + \delta_y\varphi_y)/\varphi_z,$$
$$\boldsymbol{\Delta} = \tilde{Q}_1(\boldsymbol{\nu}' - \boldsymbol{\nu}).$$

Proof. First we align the back wheel using Lemma 2. We then find the vector from the back to the front wheel of the image in both the 3d model and the 2d image. Let $\boldsymbol{\Delta} := \tilde{Q}_1(\boldsymbol{\nu}' - \boldsymbol{\nu})$ be the vector from back to front wheel in 3d after rear wheel alignment. We assume that $\boldsymbol{\Delta}$ is orthogonal to the rear wheel axle $\tilde{Q}_2\boldsymbol{\phi} \equiv \boldsymbol{\varphi}$. The 2d rear wheel to front wheel vector is given by $\boldsymbol{\delta}$ above.

$\boldsymbol{\Delta}$ must scale and rotate around $\boldsymbol{\varphi}$ into $\boldsymbol{\delta}$, which is achieved by scaling with σ and rotating with $\tilde{Q}_2$ as defined by Lemma 2 and β given by (12). ■

This process leaves us with a 4-fold ambiguity which can be resolved using a loss-based disambiguation. This disambiguation will be discussed in a future paper. We are also able to make use of numerous consistency checks at this stage. As we do not use the size of either wheel, we are able to compare the major axis length of the imaged wheels with the 3d wheel radii. Also the front and back wheel axles $\boldsymbol{\varphi}$ and $\boldsymbol{\varphi}'$ and distance $\boldsymbol{\delta}$ must (approximately) lie in a plane.

6 Experimental Results

The 2d ellipse to 3d circle matching method presented here was tested on a number of images, both real and artificial. We have used three 3d car models, sourced from linefour [Lin08] in our experiments, a Volkswagon Golf, a BMW M3 and an Audi A6. Real images were taken of vehicles in a number of locations around the university under natural conditions. Where possible, we have matched real images to the most similar 3d model available.

We manually select the pose with the best match from the four generated by our method when producing output images. In addition, we have developed a sophisticated loss-based method for automatically selecting the best pose, but this is outside the scope of this paper.

We found that the parallel projection model that we have assumed works well even for perspective projection, provided that the perspective effect is not too extreme. This is due to the fact that each wheel is considered separately when determining ellipse parameters, and each occupies only a small volume of 3d space, resulting in minor length distortion which still produces good performance.

Testing Dataset. Artificial images were generated by rendering 3d models using freely available MATLAB toolkits. Renders were generated with a plain background at a resolution of 800x600. Real images were collected using a variety of hand held cameras. Images were rescaled where necessary to provide a consistent image resolution.

Wheel Detection Performance. Detecting the ellipses in the image which correspond to the vehicle's wheels and accurately identifying their properties is an important step in vehicle pose recovery. The wheel detection algorithm presented was tested on a number of images, both real and artificial taken in a wide variety of locations ranging from artificially lit underground car parks to outdoor on street parking. 15 artificial and 35 real-world images were sampled, containing numerous

Figure 4: Close ups of wheels from an artificial image (a) and real images (b,c) with the equivalent ellipse in red. (d) an example of a wheel for which detection fails.

different car models, lighting conditions and viewing angles. Each image contained two unobstructed wheels that were clearly identifiable by a human observer. Figure 4 shows segments of the images which contained wheels and the ellipses extracted by the wheel detection algorithm.

The wheel detection algorithm works quite well in most cases, however it has difficulty identifying wheel hubs which are quite oblique to the viewer or is too dark to be clearly differentiated from the tyre by the chosen threshold. Of the 50 vehicle images tested, both wheels were identified correctly in 76% of images, while one wheel was identified in 20%. In the two images for which no wheels were detected, the wheel's hubcaps were unusually dark, as in Figure 4(d). Other wheel detection failures occurred when there were occlusions, either from other objects or parts of the vehicle being in the way or dark, dirty or spoke-like rims/hubcaps. Several single wheel detections occurred when the algorithm identified a wheel in a second vehicle present in the image rather than detecting both wheels of a single vehicle. The wheel detection algorithm also has difficulty identifying wheels when the hubcap of the vehicle is obstructed by an object which is also comparatively bright, as these objects will disrupt the shape of the blob containing the wheel. Overall, the detection performance is quite good, with at least one wheel being identified in 96% of test images.

To test the effect of scale on the wheel detection, a subset of images were rescaled to 25, 50, 150, 200 and 400% of the original size. The algorithm detected the same ellipses in each of these rescaled images, demonstrating that performance of wheel detection is not greatly affected by scale.

Artificial Images. The pose recovery performance of the presented algorithm on artificial images can be determined exactly, as we know the manipulations performed on the model to produce the test image. We perform two types of pose estimation on artificial images, one using the wheel detection method presented in this paper and one using exact ellipse normals determined exactly from the model and its rotation parameters. This allows us to verify both the accuracy of the pose estimation and the wheel detection steps.

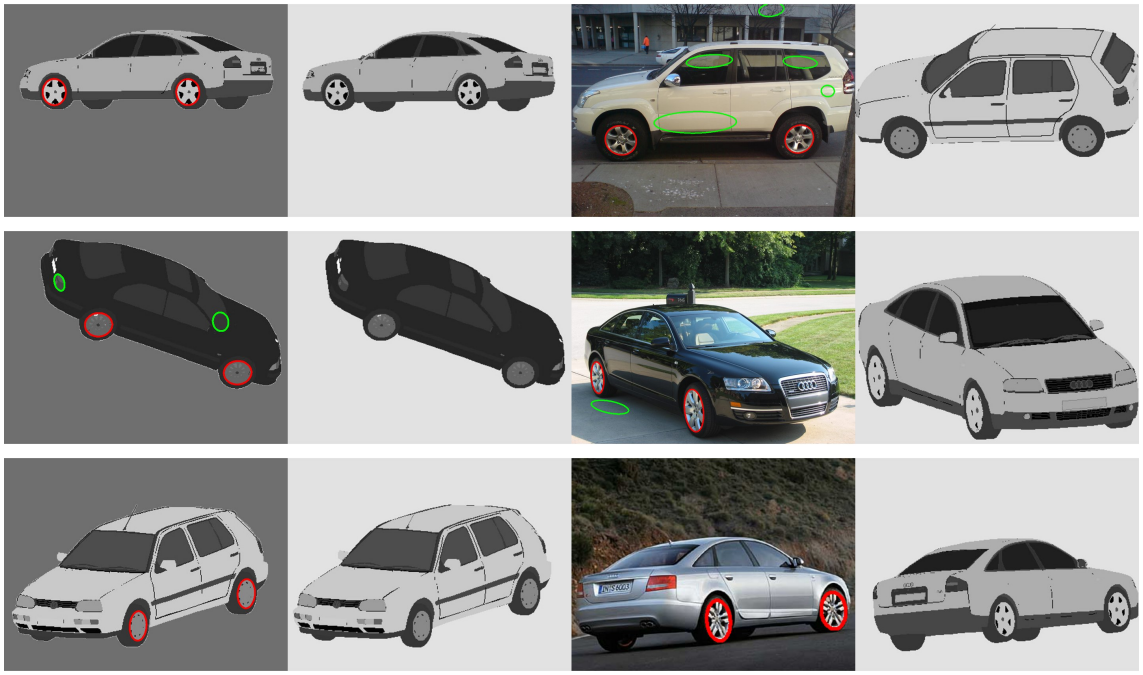

Figure 5: A set of real and artificial images showing input images with detected ellipses on the left and a model rotated to the estimated 3d pose of the vehicle in the above image on the right (4-fold ambiguity manually selected). Images cropped for display purposes only. Note that in the original photos used for ellipse detection, cars filled only a fraction between 10% and 25% of the image. We cropped to the car region for display purposes only.

Table 2: Table of results showing input pose parameters and estimated pose from ellipse normals derived from the wheel detection algorithm. Pose from manually extracted normals matched the true pose exactly, and so has not been included

Vehicle Model	True Pose Quaternion Rotation q	Scale σ	Shift $\boldsymbol{q}$	Pose from Wheel Detection Quaternion Rotation q	Scale σ	Shift $\boldsymbol{q}$
Golf	-0.2162 -0.2162 -0.6053 0.7350	60	[400 300]	-0.2139 -0.2083 -0.6077 0.7359	61.03	[396 297]
	-0.4417 -0.5522 0.5522 -0.4417	50	[380 330]	-0.4400 -0.5315 0.5536 -0.4663	49.99	[428 323]
Audi	0.9701 0.000 0.2425 0.000	150	[400 300]	0.9691 0.0070 0.2367 -0.0036	150.99	[396 304]
	0.9701 0.000 -0.2425 0.000	150	[400 300]	0.9690 0.0087 -0.2469 0.0007	149.78	[389 304]
BMW	0.0948 0.1896 -0.9481 0.2370	100	[400 300]	0.0852 0.1883 -0.9468 0.2466	99.34	[402 299]
	0.8729 -0.2182 0.4364 0.000	100	[400 300]	0.8755 -0.2212 0.4297 -0.0019	97.88	[363 311]

Table 2 shows the true and estimated pose using the wheel normal estimation method. The rotation is given as quaternion parameters a, b, c, d, the shift as a 2-dimensional vector $[x, y]$ representing the shift from car centre to image origin, and the scale factor is a scalar. Figure 5 shows graphically the recovered pose for two artificial images using approximated wheel normals. The pose recovered from exact

ellipse normals matched exactly the input pose, and so is not included.

The pose returned when ellipse normals are precisely known matches perfectly with the expected pose of the image. However, the estimated ellipse normals produce a pose which does not exactly match that expected. The pose error is relatively small, and it clearly results from inaccuracies in the estimation of ellipse parameters from the wheel detection. Even when using estimated parameters, the pose of the vehicle is still close enough to the true pose that it provides a good starting point for techniques which make use of other information about the vehicle to fine-tune the estimation.

Real Images. Figure 5 shows three real-world car images and the estimated pose of a 3d model of these vehicles. As we do not know the true vehicle pose in a real image, we are unable to conduct a quantitative analysis of the recovered pose, and must instead rely on visual inspection.

Car pose is recovered quite well in terms of scale and position, but there is some misalignment in orientation. This misalignment is caused by a distortion of the detected wheel ellipses, resulting in an imprecise normal recovery and thus a mismatched pose. The recovered model pose is still able to give us significant information about the position of the vehicle.

7 Summary and Conclusion

We have presented a method for mapping a two-dimensional ellipse to a three-dimensional circle by estimating the normal of the ellipse in three dimensions. Our results show that an ellipse can be matched to a circle in three dimensions to a pose with eight discrete and one continuous ambiguity. We also show that these ambiguities can be reduced by using other information about the object to be matched or the scene it is drawn from. In particular, vehicle pose can be resolved to a four-fold discrete ambiguity by making use of both of a vehicle's wheels and the fact that a vehicle is not transparent. Forcing the vehicle to be upright reduces this to a twofold ambiguity, given that we know which wheel in the image is the rear. We are currently developing a dissimilarity measure that will be able to automatically select the correct pose from these ambiguities by performing a comparison between each returned pose and the input image, but detail on this process is beyond the scope of this paper. It is based on a distance between images which is invariant under lighting conditions and relatively insensitive to the image background.

Additionally, we described a technique for wheel detection and wheel parameter estimation using a local thresholding technique and blob covariance. Our results show that this technique performs well on a variety of real and synthetic vehicle images.

References

[CLER07] A.Y.S. Chia, M.K.H. Leung, How-Lung Eng, and S. Rahardja. Ellipse detection with hough transform in one dimensional parametric space. *IEEE International Conf. on Image Processing (ICIP'07)*, 5:333–336, 2007.

[FP02] David A. Forsyth and Jean Ponce. *Computer Vision: A Modern Approach.* Prentice Hall, 2002.

[HSR03] S. Hinz, C. Schlosser, and J. Reitberger. Automatic car detection in high resolution urban scenes based on an adaptive 3d-model. *Proc. 2nd GRSS/ISPRS Joint Workshop on Remote Sensing and Data Fusion over Urban Areas*, pages 167–171, 2003.

[HZ04] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[KO04] K Kanatani and N Ohta. Automatic detection of circular objects by ellipse growing. *International Journal of Image and Graphics*, 4(1), 2004.

[Lin08] Linefor. Free 3d models 3d people 3d objects, 2008. `http://www.linefour.com/`.

[Low99] David G. Lowe. Object recognition from local scale-invariant features. In *Prof. International Conference on Computer Vision (ICCV'99)*, pages 1150–1157, 1999.

[Mcl98] Robert A. Mclaughlin. Randomized hough transform: improved ellipse detection with comparison. *Pattern Recogn. Lett.*, 19(3-4):299–305, 1998.

[MHZS08] F. Mai, Y. S. Hung, H. Zhong, and W. F. Sze. A hierarchical approach for fast and robust ellipse extraction. *Pattern Recogn.*, 41(8):2512–2524, 2008.

[MSB08] Fabien Moutarde, Bogdan Stanciulescu, and Amaury Breheret. Real-time visual detection of vehicles and pedestrians with new efficient adaboost features. *2nd Workshop on Planning, Perception and Navigation for Intelligent Vehicles*, 2008.

[PPA08] Niklas Pettersson, Lars Petersson, and Lars Andersson. The histogram feature – a resource-efficient weak classifier. In *IEEE Intelligent Vehicles Symposium (IV2008)*, 2008.

[SRSBT91] R. Safaee-Rad, K.C. Smith, B. Benhabib, and I. Tchoukanov. Application of moment and fourier descriptors to the accurate estimation of elliptical shape parameters. In *Prof. International Conf. on Acoustics, Speech, and Signal Processing (ICASSP'91)*, pages 2465–2468 vol.4, 1991.

[SSN07] Ashutosh Saxena, Min Sun, and Andrew Y. Ng. Learning 3-d scene structure from a single still image. In *ICCV workshop on 3D Representation for Recognition*, 2007.

[Tan08] R.T. Tan. Visibility in bad weather from a single image. *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR'08)*, pages 1–8, 2008.

[TM78] S. Tsuji and F. Matsumoto. Detection of ellipses by a modified hough transformation. *IEEE Trans. Comput.*, 27(8):777–781, 1978.

[VJ04] Paul Viola and Michael J. Jones. Robust real-time face detection. *Int. J. Comput. Vision*, 57(2):137–154, 2004.