The Open University of Israel

Department of Mathematics and Computer Science

View Relevancy for Model-based Pose Estimation in Single Photos

Thesis submitted as partial fulfillment of the requirements
towards an M.Sc. degree in Computer Science
The Open University of Israel
Computer Science Division

By

Liav Asseraf

Prepared under the supervision of Dr. Tal Hassner

September 2011

Contents

List of Figures

Abstract

Given a photo of an object, and a 3D representation of that object, the pose estimation problem is determining the object's transformation (rotation and translation). We present a method for model-based pose estimation of rigid objects in a single casual photo. Having a 3D CG model of the subject object in the photo allows reducing the 3D-2D matching problem to a series of conventional image to image matching operations based on image-space descriptors. However, treating all views of the CG model alike, independent of the given model and input photo, often fails. We argue that not all views contribute equally to the task and thus develop a view relevancy measure. First, for each CG view and a calibrated training photo of the same model, our method computes a series of measurements – a feature vector that captures characteristic properties of the match. We learn a model that predicts the reliability and relevancy of each view based on these feature vectors. Then, given a new query photo, and a new CG model, the pose estimation based on matches to each generated CG view is evaluated via the learned model to select the best pose estimate. To evaluate our method we have assembled two challenging data sets of 3D CG models of cars and building combined with uncontrolled web images. We demonstrate empirical results showing that our pose estimation improves existing methods.

Chapter 1

Introduction

1.1 Background

The estimation of the six degrees of freedom of a rigid object's 3D pose from a single image is a key enabling requirement in many machine vision, graphics, and robotics systems. This process might be performed by utilizing any number of available shape and appearance representations of the object at hand.

The most common representation of 3D objects is a Computer Graphics (CG) model. Such models are available and accessible over the web. CG models, represented with, e.g., textured triangulated meshes, capture the approximated appearance and structure of the object, and allow the rendering of the model under arbitrary view points (see Fig. 5.1). Using a model, estimating the pose can be done by the generated views which can in turn be compared to the input photo to assist the posing task, reducing the 3D posing problem to image-to-image comparisons. Given a 3D CG model, computing an object's pose may therefore involve forming correspondences between features in the input photo and the computer graphics images (CGI) of the model.

Since CG models are very common nowdays, using them to solve the pose estimation problem could be preferable to obtaining photos of the same object taken from many different view points. In addition, storing such a model occupies less space than storing many single photos.

However, solving the pose estimation problem using a photo and a CG model is not trivial. The object's size in the photo can differ from the CGI view. In addition, its location can be different and it can also be partially occluded by other objects. Finding the rotation of the object is another problem and using a very dense viewing sphere of the CG model is not efficient. Another difficulty arises from the modality problem between the generated view and the photo's textures - the model's textures usually have much lower quality. This makes it hard to form correspondences be-

tween them. Also, the CG model usually represents the object's structure only in general and not in an exact way (e.g. the object's parts might have different proportions than in reality).

Ostensibly, this photo-CGI approach reduces the problem of pose estimation to cross-modal feature matching, where features from a query photo are matched to features extracted from a gallery of rendered (CGI) views. Establishing such matches is challenging, even when high quality models are available, often leading to merely a small ratio of positive matches to false positive matches.

To improve the quality of the estimated matches, a number of papers have suggested seeking robust features – features more likely to appear in different views of the same object [13, 15, 16]. An additional means of boosting pose estimation precision may be obtained by considering the *relevancy* of a view. Specifically, some views perform better than other in recovering the true pose. However, this leads to a chicken-and-egg problem: Relevant views are those which provide the most accurate estimate for the pose. On the other hand, knowing which views provide accurate pose estimates might benefit from knowing the correct pose.

Our goal in this research is to learn to identify the views that are the most relevant to the input images by employing a statistical model. Our key observation is that relevant views exhibit tell tale signs indicating that they indeed provide a reliable pose estimate. We characterize each CGI view by a feature vector which includes various features of the pose estimate. Views are then classified using a standard classifier trained on a similar set of feature vectors. The relevant views are the ones with the highest classification confidence score. These relevant views are then used to estimate the object's pose directly. We take this idea a step further and show how the performance of the RANSAC process, used to produce pose estimates, may be boosted by applying the same learning technique to select a pose hypothesis.

To evaluate the efficacy of our method we have collected two data sets, each including low-resolution, textured, 3D models as well as images of the real objects being modeled. These 3D models and associated images were collected from the web. The CG models, therefore, demonstrate a challenging variability in parameters such as fidelity, detail level, and texture quality. The photos vary in resolution, background clutter, illumination, and pose. We show our method to substantially outperform existing state-of-the-art systems in these tests. We make our benchmark tests, data and code available to the public.

1.2 Related work

Over the years numerous pose estimation methods have been proposed. Broadly speaking, these can be categorized into two main groups: methods using image-based, implicit models for the underlying geometry of the physical object, and meth-

ods employing explicit, 3D representations.

A large number of photos may be used to capture the appearance of an object from different viewpoints and thus facilitate pose estimation. This approach has the advantage that typically it is easier to compare images of the same modalities rather than photos to CG images. The downside is the requirement of having multiple, often a great deal, of photos to capture the appearance of the object from all possible viewing angles [1, 10, 26, 28].

The alternative of explicit 3D information has been exploited in different ways in the past, typically by using a CG representation of the object. A popular approach is to compute pose-estimation and segmentation jointly by using the object's contour. Some recent examples include [22, 24, 25]. Although contours often provide accurate information, they are sensitive to occlusions, they do not provide sufficient information when objects are smooth or convex, and they may be mislead by background noise.

Correspondences are often established between points [23], pixel patches [30], contours [22, 24, 27], line segments [5], image descriptors [8, 10, 20], or multi-modal cues [14]. These provide links between 2D coordinates in the input photo and 3D coordinates of the model. Pose information can then be computed from these correspondences.

More related to our work are methods which exploit texture information on the 3D geometry. Here, matches are formed between the input photo and a rendered view of the 3D model acting as a proxy for the 3D geometry [8]. More recently, this approach has been combined with recognition [29] and detection [15, 16]. These methods use many 3D models from the same class, employing correspondences between query features and features from *multiple* CG views. In this thesis, on the other hand, we advocate using a *single*, carefully selected, CGI view of a model representing the particular object in the photo.

Pose estimation benchmarks. As far as we know, there is no widely accepted benchmark for testing the performance of pose estimation algorithms. Some benchmarks have been used recently for the related problems of detection and viewpoint classification include subsets of the PASCAL challenge (e.g., [6]) and the 3D Object Classes data set of [26]. These do not provide 3D geometry associated with the objects in each image. Poses estimated for images in these sets are therefore only estimated to within a small set of discrete poses. Other data sets were assembled for evaluating pose estimation with multiple views (e.g., [3]) or multi-modal methods [14], neither of which is relevant to our method. We therefore assemble our own test data, including images from the web, along with models of the objects appearing in them. This allows us to accurately measure the precision of our method on unconstrained images.

Chapter 2

Model-based pose estimation

We are given a 3D CG model m and a photo, I^m, of the same object, taken with a camera whose unknown external parameters are given by some rotation matrix R and translation vector t. We wish to recover the six degrees of freedom of these parameters in the CG model's coordinate frame.

Having the model m at our disposal allows us to render images of the model, producing CGI views V_j^m. Each view includes, besides its intensities, also the 3D coordinates of the points projected onto each of its pixels. By establishing a link between a pixel x_i in I^m and pixel x_i' in V_j^m, we obtain the correspondences (x_i, X_i), were X_i is the 3D point projected onto x_i'. These can then be used to estimate the camera's pose in I^m using standard camera calibration methods [11]. Specifically, given correspondences (x_i, X_i), the matrix $R_{3\times3}$ and vector $t_{3\times1}$ may be obtained by solving

$$x_i \sim A[R \;\; t]X_i \tag{2.1}$$

Where $A_{3\times3}$ is the intrinsic camera matrix, and R is constrained to be an orthonormal matrix.

A preprocessing step is done similar to the one proposed in [16]. For every model m we produce 324 CG views V_j^m, 108 views uniformly distributed over the upper hemisphere of the object at three radii. Image descriptors are extracted using the Harris-Affine interest point detector using the code from [18]. SIFT descriptors [17] were computed using the code made available by [31]. The descriptors are upright orientated.

The keypoints are evaluated by employing the technique of Discriminative Filtering [12, 16] to reduce the number of descriptors extracted to those which are stable across small view variations, and do not contain background. The filtering stage eliminate around 80% of the keypoints found.

Given a descriptor set extracted from a new photo, it is matched against the descriptor set of the current CG view. Each descriptor in I^m is matched to its L2-nearest

neighbor in V_j^m. We have found that better performance is obtained without applying the nearest neighbor distance ratio (NNDR) criterion of [19]. Instead, further filtering of the image descriptors is performed by adding the non-class descriptors as possible matching candidates for those in the photo. A descriptor matched to any of these non-class descriptors is discarded. This step eliminate around 90% of the matches. Pose can then be recovered by employing RANSAC as a robust estimator [7].

Many of the views may be disregarded at this early stage, thus reducing the computational load to a short-list of potentially relevant viewpoints. Following [16], we collect all descriptors for all CGIs extracted from the given training model. These are then clustered using standard k-means. Each cluster is associated with the list of all viewpoints (out of 324 possible CG views) that contributed to the descriptors in this cluster (the specific model is not retained, only the position of the viewpoint). When evaluating a new photo, its descriptors are matched against these cluster centers. Each center then votes for the viewpoints associated with it. The third of the viewpoints with the highest number of votes are evaluated further, and the rest of the views are discarded.

The complete process of Model based pose estimation without learning is described in Alg. 1.

If multiple CGIs V_j^m exist and a sufficient number of correct matches is established in each of these views, then this process should yield the same pose estimate for all views. In practice, however, the overwhelming presence of many false matches results in pose estimates that vary greatly between the different CGIs.

The question is then, how can we tell apart the relevant views, where pose estimation is successful, from those which mislead the process? Ideally, we seek a solution that would be statistically stable over many models and photos.

Algorithm 1 No learning

1: Given a CG model m and a photo of the model's object I^m:
2: Form matches between the descriptors of I^m and the model m clustered robust features
3: Vote for each CGI view using the matches
4: S = the top third voted CGI views
5: **for** every view $V_j^m \in S$ **do**
6: Match the view's extracted robust features with the query features
7: Perform RANSAC calibration
8: **end for**
9: Return the view with maximum inliers

Chapter 3

Adding Learning to the Basic Algorithm

3.1 Learning relevant views

We assume a training set of a certain class or family of 3D CG models and associated photos of these objects. The camera poses for the photos included in this training set are computed by manually establishing point correspondences between the photos and the CGIs.

For every training model m we render CGI views V_j^m, sparsely covering the object's viewing hemisphere. For each of the model's training images, I_k^m we then estimate the pose automatically (Section 2) using each and every one of these rendered views. Each such estimate provides us with (i) a pose error e_{jk}^m and (ii) a number of parameters characterizing the behavior of the pose estimation process collected as a feature vector v_{jk}^m. The measurements included in the feature vector are described in detail in Sec. 3.2. Once these vectors are assembled we apply a statistical learning machinery over all image and CGI view-pairs for all available models, that links the error e_{jk}^m to the vector of parameters v_{jk}^m. If this estimation implies a large error, the view is said to be irrelevant, otherwise, the view is considered relevant.

Specifically, for every CGI view V_j^m in the training set, and for every given photo I_k^m of the same model m we obtain an estimate of the pose in I_k^m. This is then compared to the (known) ground-truth pose and an error is computed as a function of the angular and translational difference between the estimated pose and the ground truth pose. This error serves to compute training labels. The pair (I_k^m,V_j^m) is assigned a label of 1 if the error e_{jk}^m falls below a predefined threshold and -1 otherwise. In other words, the positive class is the class of relevant views. This learning phase is described in Alg. 2.

In our implementation we define e_{jk}^m based solely on angular differences. It is

Algorithm 2 Learning phase

Given a CG model m and a photo of the model's object I_k^m:
Compute T using the known ground truth pose as in Eq. (3.1)
for every view $V_j^m \in S$ **do**
  Compute $\hat{T}$ as in Eq. (3.1)
  Compute e_{jk}^m using points $\hat{p}$ from Eq. (3.2)
  if $e_{jk}^m <$ a predefined threshold **then**
    Assign (I_k^m,V_j^m) a label of 1
  else
    Assign (I_k^m,V_j^m) a label of -1
  end if
  Obtain the feature vector v_{jk}^m of all the features in Table 3.1
end for
Obtain an LDA classifier using the labels of (I_k^m,V_j^m) and the data of the feature vectors v_{jk}^m

measured as the angle between the principle axes of the known and estimated positions of the bounding box of m. Let $p \in \mathcal{R}^3$ be a point on the ground truth pose of m (in homogeneous notation), $\hat{T}$ defined as

$$\hat{T} = \begin{bmatrix} \hat{R} & \hat{t} \\ \mathbf{0} & 1 \end{bmatrix} \tag{3.1}$$

be the estimated extrinsic camera matrix, and T similarly defined using the ground truth rotation R and translation t. Assuming a fixed camera matrix, we compute:

$$\hat{p} = \hat{T}T^{-1}p \tag{3.2}$$

Points $\hat{p}$ are then used to produce the estimated bounding box and compute e_{jk}^m.

The feature vectors v_{jk}^m, along with the labels computed based on pose estimation accuracy, are used to train a discriminative model for selecting relevant views. Here we use a standard LDA classifier [2]. Although better performance may presumably be obtained with more sophisticated classifiers, we chose to focus here on the features rather than the classification engine.

The LDA classifier obtained is used to link the features extracted from new CGIs of new models to new photos. During the application (test) phase, the feature vector v_j' is computed as above for each CGI view. The LDA classifier is then employed on these vectors to obtain a numeric score that is expected to be positive and high if the CGI is relevant to the given photo and negative and low otherwise. This numeric score is used to rank the CGIs and identify the most relevant views. In practice, this

score is taken to be the LDA projection obtained for the vector v'_j. The relevancy learning process is described in Alg. 3.

Algorithm 3 Relevancy Learning

Given a CG model m and a photo of the model's object I_k^m:
Perform the No learning algorithm as defined in Alg. 1
Perform the learning phase as defined in Alg. 2
Perform the test phase:
for every view $V_j^m \in S$ **do**
    Obtain the feature vector v'^m_{jk}
end for
Use the classifier on the vectors v'^m_{jk}
Score each view according to the LDA projection of its feature vector
Return the pose estimation of the view with the best score

3.2 The features in v_{jk}^m

Table 3.1 lists the values used in the feature vector v_{jk}^m. Their purpose is to reflect the quality of the pose estimates in each CGI view. We briefly review these values next. We first note that additional features may also be conceived and added to these features. In our experiments, however, we found these values to work well.

Number of inliers. The pose computed by RANSAC for each view V_j^m is the one obtained in the iteration with the largest number of correspondence inliers; this value is considered a good measure for the quality of the computed pose. We store it as a first characteristic feature of the pose estimated in each view (Item 1 in Table 3.1).

Photo-to-CGI pose difference. The angle α_{kj}^m between the estimated pose of I_k^m and the known pose of the CGI view V_j^m, used to estimate it, is a key parameter (Item 2 in Table 3.1, see also Fig 3.1). Here, large values may either be due to an actual large difference in poses (unlikely, as in this case, few matches, if any, will be accurate) or unreliable estimates. Small differences are also either the result of a correct estimate, or an unreliable estimate. Assuming a uniform distribution of erroneous estimates, however, it is less likely for a small angle difference to be the result of an error. In practice, α_{kj}^m is computed similarly to e_{jk}^m (Eq. (3.2)) using the known extrinsic matrix of CGI view V_j^m and the estimated matrix of I_k^m.

Correspondence quality. We consider the Euclidean distances between feature descriptors extracted from of the corresponding points x_i in I_k^m and x'_i in V_j^m (3 in Table 3.1). The median distance between all inlier corresponding descriptor pairs is thus recorded.

Table 3.1: Features used by our system to characterize the match between photo I_k^m and the CGI view V_j^m of the same model m.

1	Number of inliers
2	Angle α_{kj}^m between estimated pose for I_k^m and known pose of V_j^m
3	Median inlier descriptor distances
4	Standard deviation of the inliers scale ratio
5	Sum of (1) over the 3×3 spatial neighborhood of V_j^m
6	Sum of (2) over the 3×3 spatial neighborhood of V_j^m
7	Sum of angle difference between estimated pose for I_k^m using V_j^m and estimated poses obtained using the 3×3 spatial neighbors of V_j^m

Inlier scale ratio variation. We compute the ratio of the scales returned by the feature detector [18] for both x_i and x_i' in every inlying corresponding pair. The standard deviation of these values then gives a measure of the consistency of scale differences in I_k^m and V_j^m (4 in Table 3.1).

Neighborhood consensus. Finally, we evaluate the consensus of these parameters amongst the poses estimated using the neighboring 3×3 views on the viewing sphere (5,6,7 in Table 3.1). The rational is that for a correct pose estimate, neighboring views should roughly agree on the pose and produce similar values.

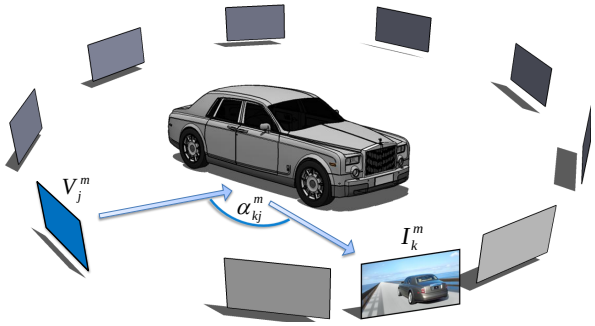

Figure 3.1: Photo-to-CGI pose difference. Illustrating the angle α_{kj}^m between the pose estimate for photo I_k^m according to correspondences established between its image-features and those of CG view V_j^m. Note that views in this illustration are positioned around the object, whereas in practice they cover the viewing hemisphere.

Chapter 4

Modifying the RANSAC process

4.1 Selecting the winning hypothesis

In the method described so far, a relevant view was selected based on a number of scores representing the pose hypothesis produced by RANSAC for each view. Conventionally, a RANSAC process selects a winning hypothesis based on the number of inliers [4]. Here, however, we suggest applying the same idea of using a learned statistical model for selecting the winning RANSAC iteration: Instead of counting the number of inliers, we score each iteration based on a feature vector computed as described above. That is, we apply the same idea of using a learned statistical model for the selection of the winning RANSAC iteration.

Specifically, we choose the winning RANSAC iteration by considering only the first four features out of the seven in Table 3.1. Features 5–7 are excluded, as they rely on the final pose estimates for the neighboring views, information not available before RANSAC has terminated in all the views. We train a classifier on these four features using the training data, as described above, and apply it within each CGI's RANSAC, pose-estimation routine. Once we pick the winning iteration for each view, based on features 1–4, we can continue in several ways:

1. Relevancy based on maximum inliers - select the view with maximum inliers. See Alg. 4.

2. Relevancy based on best RANSAC score - select the view with the best selected hypothesis score. See Alg. 4.

3. Full approach - add features 5–7, computed from neighboring views and continue as described above to select the best view. See Alg. 5.

Algorithm 4 Modified RANSAC

Given a CG model m and a photo of the model's object I_k^m:
Perform the No learning algorithm as defined in Alg. 1
Obtain an LDA classifier for the RANSAC process by using Alg. 2 with only the first four features in Table 3.1
for every view $V_j^m \in S$ **do**
    Perform a modified (test phase) RANSAC calibration:
    for each RANSAC hypothesis **do**
        Obtain the four feature vector r'^m_{jk}
    end for
    Use the classifier on the hypotheses vectors r'^m_{jk}
    Select the hypothesis with the best score as the pose estimation of V_j^m
end for
if Relevancy based on maximum inliers **then**
    Return the pose estimation of the view with maximum inliers
else if Relevancy based on best RANSAC score **then**
    Return the pose estimation of the view with the best selected hypothesis score
end if

Algorithm 5 Full approach

Given a CG model m and a photo of the model's object I_k^m:
Perform steps 2-12 of Alg. 4
Obtain an LDA classifier by using Alg. 2 with the following features:
All the features in Table 3.1
An additional feature - the view's selected hypothesis score
Perform the test phase:
for every view $V_j^m \in S$ **do**
    Obtain the feature vector v'^m_{jk}
end for
Use the classifier on the vectors v'^m_{jk}
Score each view according to the LDA projection of its feature vector
Return the pose estimation of the view with the best score

Chapter 5

Experiments and results

5.1 Implementation

Our method is implemented in MATLAB, using a MATLAB OpenGL wrapper for rendering the CG models. Standard OpenCV routines were used to compute the pose given 2D-3D correspondences.

For simplicity, we assume that the focal length is known and set it to 800 in image pixel units, that the principle point is at the center of the image, that the pixel's aspect ratio is one, and that there is no skew. Our method is agnostic to the type of camera calibration model that is used to estimate the pose from point matches and it is straightforward to relax these assumptions by using more elaborate camera calibration techniques.

Pose was estimated using $2,000$ RANSAC iterations. The features in v_{jk}^{m} (Sec. 3.1) were normalized to the range of $[0..1]$. LDA was applied with a regularization constant of 0.1 in all our experiments, taking views which produce an angular error of 9 degrees or less as class examples, all others as non-class. When the LDA method is applied, the view with the highest LDA projection value is selected, and its pose estimate is then returned as our method's output.

5.2 Data sets and Benchmarks

In order to test our method we have collected textured, 3D, CG models of car and building objects, along with images from the web, taken of those same objects. Our models were obtained from the Google 3D Warehouse collection [9] and the images were mostly downloaded from Wikipedia. In total, we have 31 car models with 90 test images and 11 building models with 30 images, models having one to three query images each. All models were scaled to unit size. Car models were further roughly aligned – all facing the same direction. We manually recover the ground-truth camera

Figure 5.1: The pose of the object in each input photo (right) is estimated using a reference 3D CG model. Estimation is done using the proposed method in this thesis. The estimated pose is illustrated on the left by posing and displaying the CG model in the detected pose. Note that these images are re-scaled for illustrative purposes.

pose of all our test images by establishing manual correspondence between the images and CGI views of their associated CG model. Fig. 5.2 presents some examples of our models and test images.

Figure 5.2: Examples from our data sets. Top row are rendered views of the 3D CG models. Bottom are example query photos collected from the web.

With this data set, we define a straightforward leave-one-out testing protocol as follows. Given an image, we estimate the pose of the object in the image, using the object's CG model. In addition, all other models, their images, and ground truth poses are available for training; the only information excluded from the training is, of course, the ground truth pose of the input image, as well as other query photos of the same object and their known poses. Pose estimate precision is measured following [16] by considering both the translational ϵ_t and angular ϵ_r errors. Specifically, ϵ_t is the difference between the center of the ground truth model and the center of the model in the estimated position, ϵ_r is the angle between the principle axes of the real and estimated bounding boxes. We use Eq. (3.2) to obtain the estimated position of

the object's bounding box.

5.3 Algorithms analysis

5.3.1 Relevancy learning

Based on Sec. 3.1 we've examined Alg. 3 and compared the No learning and Relevancy learning methods. Fig. 5.3 shows examples where even when a correct pose estimation was found, the wrong view is chosen by the No learning algorithm, while the correct one is chosen by the Relevancy learning algorithm.

Figure 5.3: No learning vs Relevancy learning. Note that these images are re-scaled for illustrative purposes

5.3.2 Winning iteration selection

Based on Sec. 4.1 we've examined the winning iteration selection approach, comparing the hypothesis chosen based on largest #inliers and based on highest LDA score. Fig. 5.4 shows an example of comparing these two criterias, number of inliers and LDA score, of each iteration in a RANSAC procedure. It shows that each criteria leads to a different iteration being selected.

There are cases when several iterations have the same maximum number of inliers. In such cases the lower iteration is picked. Even if one of the other, maximum #inliers iteration, is the right estimation, the simple, No learning algorithm, does not

have the information to find it. Note that this examination only shows part of the Modified RANSAC method (Alg. 4). The part where we choose the relevant view is ignored here, and discussed next.

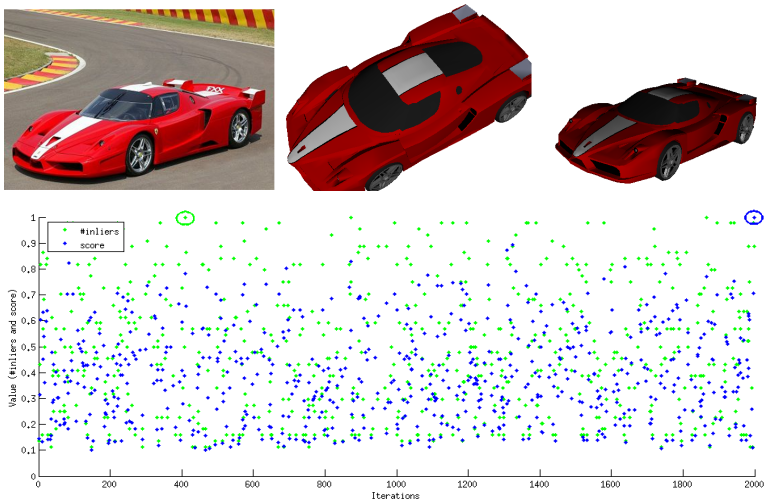

Figure 5.4: Winning iteration selection. Top row: Left is the query image. Middle is the estimation based on iteration selected by #inliers. Right is estimation based on iteration selected by score. Bottom row: values vs iteration number plot (#inliers and score are normalized). The selected maximum values for each way are circled.

5.3.3 Modified RANSAC

Fig. 5.5 shows examples where a correct pose estimation was found by the Modified RANSAC approach. However, the wrong view is chosen afterwards when the selection is based on maximum inliers (relevancy based on maximum inliers), while the correct view is chosen based on best hypothesis score (relevancy based on best RANSAC score).

5.3.4 Full approach

Fig. 5.6 shows examples where a correct pose estimation was found by the Modified RANSAC approach. However, the wrong view is chosen afterwards when the selection is based on best hypothesis score, while the correct view is chosen based on learning and using more features (Full approach).

Figure 5.5: Modified RANSAC, relevancy based on maximum inliers vs Modified RANSAC, relevancy based on best RANSAC score. Note that these images are re-scaled for illustrative purposes

Figure 5.6: Modified RANSAC, relevancy based on best RANSAC score vs Full approach. Note that these images are re-scaled for illustrative purposes

5.4 Results

Fig. 5.7 and Fig. 5.8 compare the performance of the following methods over the Cars and Buildings data-sets:

1. Random selection baseline. The CGI view V_j^m is selected randomly and its matches are then used to estimate the pose (Sec. 2).

2. Nearest neighbor. The view selected for the pose estimation is the one providing

the most nearest neighbor matches for the descriptors in the query photo. Once selected, pose is estimated as before.

3. Estimation based on [16]. The method proposed in [16] obtains pose from multiple models from a set of related objects. In an effort to closely follow their description, testing is performed using all our training models, including the model who's object appears in the test photo. We found that providing this method with only the reference model reduces pose estimation precision. We employ our own implementation of the method described in their paper, using the same parameters and reported values.

4. No learning. The relevant view is the view whose estimated pose had the most inlying correspondences. See Alg. 1.

5. Relevancy learning. The method described in this paper, in Sec. 3.1, with no modifications to the standard RANSAC algorithm; the winning RANSAC iteration for each CGI is the one with the most inliers. See Alg. 3.

6. Modified RANSAC, relevancy based on maximum inliers. Using our version of RANSAC's rule for selecting a winning pose hypothesis at each view (Sec. 4.1). The relevant view is then selected based on the maximum number of inliers, without using the learned view relevancy score. See Alg. 4.

7. Modified RANSAC, relevancy based on best RANSAC score. Same as above, but here the relevant view is selected based on the RANSAC score for the wining hypothesis, rather than the maximum number of inliers. See Alg. 4.

8. Single view, relevancy learning with modified RANSAC. Our full approach

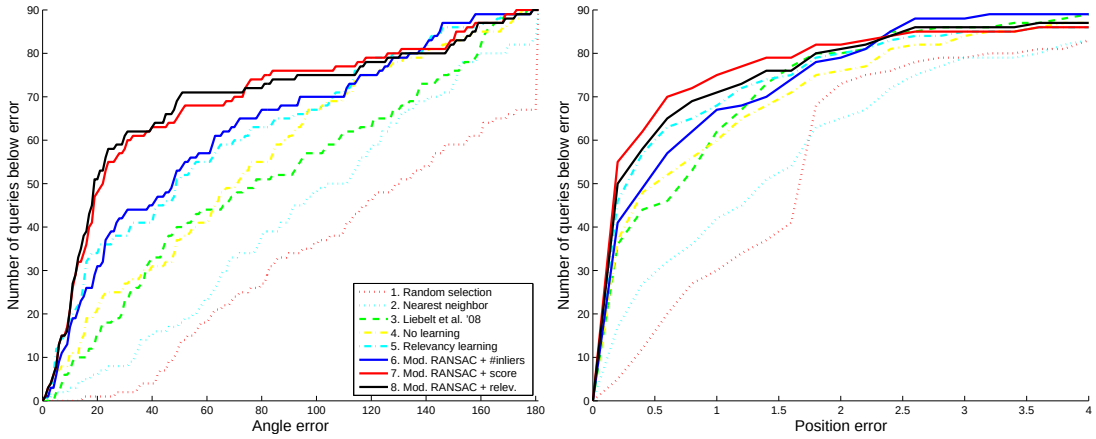

Figure 5.7: Performance summary on the Cars collection. Top figure presents angular errors, bottom, positional errors. In each figure the X-axis is increasing errors rates, Y is the number of queries (out of 90) whose pose estimate fell below this error. Top curve is best. Please see text for further details. **Best viewed in color.**

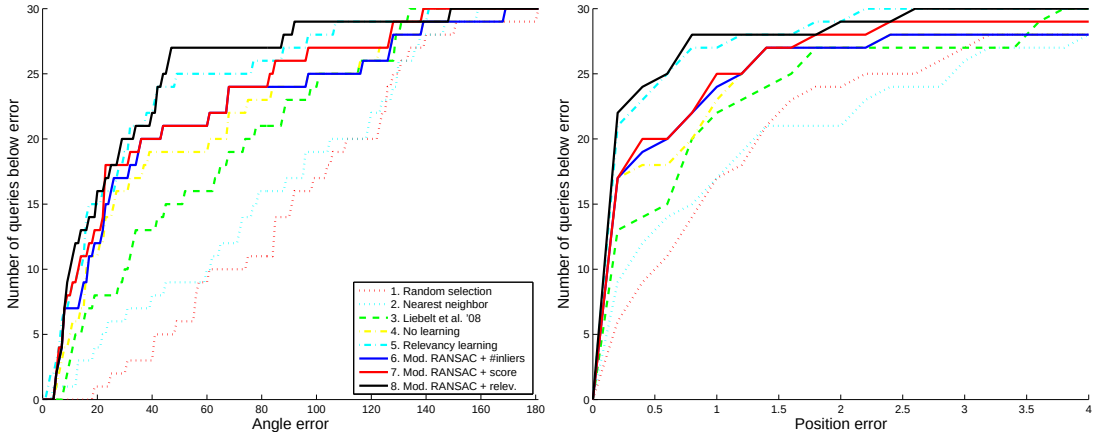

Figure 5.8: Performance summary on the Building collection.

(Sec. 3.1). Same as above, but now, following a pose estimation in each view using our modified RANSAC, the relevant view is selected based on a statistical model trained on features 1–7 in Table 3.1. See Algorithm 5.

	Cars				Buildings			
	Angular Error		Position Error		Angular Error		Pos. Error	
Method	Median	Mean ± SE	Median	Mean ± SE	Median	Mean ± SE	Median	Mean ± SE
1. Random selection	117.9	116.6 ± 5.49	1.73	1.86 ± 0.29	91.04	90.56 ± 7.53	0.89	1.36 ± 0.31
2. Nearest neighbor	93.09	96.32 ± 5.30	1.19	2.01 ± 0.53	77.53	79.92 ± 8.43	0.84	1.30 ± 0.27
3. Liebelt et al. [16]	66.47	77.52 ± 5.74	0.58	0.83 ± 0.11	48.40	58.19 ± 7.67	0.55	0.81 ± 0.19
4. No learning	63.21	68.37 ± 5.20	0.37	0.95 ± 0.14	26.09	48.89 ± 8.53	0.09	0.85 ± 0.32
5. Relevancy learning	42.94	56.12 ± 5.30	0.18	0.98 ± 0.30	**19.05**	32.92 ± 6.34	**0.03**	**0.28** ± 0.09
6. Mod. RANSAC + #inl.	39.47	54.38 ± 5.17	0.32	**0.73** ± 0.11	23.51	45.08 ± 8.58	0.08	0.81 ± 0.33
7. Mod. RANSAC + score	18.55	42.01 ± 5.14	**0.10**	0.94 ± 0.32	22.26	39.78 ± 7.41	0.08	0.66 ± 0.28
8. Mod. RANSAC + relev.	**17.83**	**41.83** ± 5.37	0.12	0.88 ± 0.29	19.76	**29.47** ± 5.77	0.05	**0.28** ± 0.11

Table 5.1: **Percision statistics.** Median and mean (± standard error of the means, SE) angular and position errors on the Cars and Buildings data sets for all tested methods. Lower values are better.

Table 5.1 summarize the results for both the Cars and Buildings sets, listing angle and position median and mean ± standard error (SE). The angular error in our complete method is equal or lower than all other variations by a significant margin. It seems that the boost in performance is mainly due to the statistical model employed within the RANSAC procedure. Position errors, on the other hand vary little from one method to the other, all of them doing well. This is unsurprising considering that translational element can be estimated, to a large degree, based on a crude identification of those keypoints that are within the object's boundaries, which all methods

do well. In contrast, our method excels in evaluating the pose hypothesis induced by multiple matches. Telling the correct hypothesis apart from the others is needed in order to obtain an accurate rotational pose. Figure 5.9 demonstrates this point. The type of errors obtained in the rotational model by the method of [16] have little effect on the location of the 3D bounding.

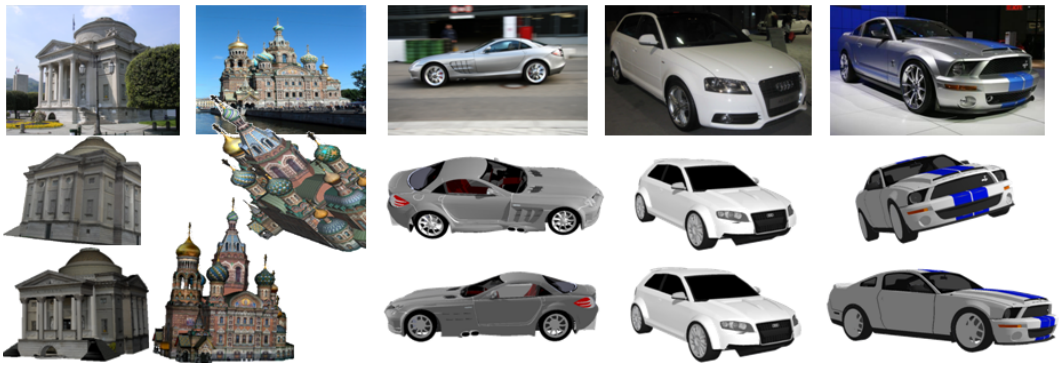

Figure 5.9: Visually comparing angular pose estimates of our method to [16]. Top row is the input photo, middle is [16] and bottom our results. Note that these images are re-scaled for illustrative purposes.

The limitations of our methods are presented in Figure 5.10. The method is challenged by similarity among completely different views or by lack of details in the given photo. While a better criteria for hypothesis selection improves performance, the problem of multiple hypothesis testing remains, which may lead to additional errors.

Figure 5.10: Examples of failed estimations. These are typically cases where the object appears similar from different views (top), has few features (middle), or are caused by poor random hypothesis selection by RANSAC (bottom). Note that these images are re-scaled for illustrative purposes.

5.5 Switching models

In order to evaluate the robustness of the statistical learning method, we have performed two additional experiments: The first was performed by learning on the cars dataset models and using the learned model in the test phase of the buildings dataset. The obtained model was used in both of the testing phases - in the modified RANSAC and in the relevancy learning parts.

The second experiment had the settings reversed - learning on the buildings dataset models and using the learned model when testing on the cars dataset. The results in Table 5.2 show that learning on a different dataset yields similar results to learning on the tested dataset.

	Cars				Buildings			
	Angular Error		Position Error		Angular Error		Pos. Error	
Method	Median	Mean ± SE	Median	Mean ± SE	Median	Mean ± SE	Median	Mean ± SE
1. Random selection	117.9	116.6 ± 5.49	1.73	1.86 ± 0.29	97.61	92.24 ± 8.21	1.70	1.79 ± 0.31
2. Nearest neighbor	93.09	96.32 ± 5.30	1.19	2.01 ± 0.53	77.53	79.92 ± 8.43	0.84	1.30 ± 0.27
3. Liebelt et al. [16]	66.47	77.52 ± 5.74	0.58	0.83 ± 0.11	48.40	58.19 ± 7.67	0.55	0.81 ± 0.19
4. No learning	63.21	68.37 ± 5.20	0.37	0.95 ± 0.14	26.09	48.89 ± 8.53	0.09	0.85 ± 0.32
5. Relevancy learning	42.37	56.58 ± 5.37	0.17	**0.70** ± 0.13	23.92	35.32 ± 7.10	**0.03**	0.24 ± 0.08
6. Mod. RANSAC + #inl.	56.48	63.52 ± 5.60	0.53	1.11 ± 0.18	22.26	44.02 ± 8.59	0.10	0.81 ± 0.33
7. Mod. RANSAC + score	19.10	45.68 ± 5.49	**0.10**	1.02 ± 0.33	**21.66**	36.30 ± 6.81	0.11	0.66 ± 0.28
8. Mod. RANSAC + relev.	**18.72**	**44.42** ± 5.46	**0.10**	1.34 ± 0.59	22.77	**30.26** ± 5.63	**0.03**	**0.19** ± 0.05

Table 5.2: **Percision statistics using switched learned models.** Median and mean (± standard error of the means, SE) angular and position errors on the Cars and Buildings data sets for all tested methods. Lower values are better.

The angular and positional errors for both experiments are similar to the ones in Section 5.

Chapter 6

Conclusions

Matching across modalities is a challenging task that results in a potentially large number of false matches. Furthermore, it is not easy to distinguish between true and false matches even when considering consensus among multiple matches. The combination of geometric projection models and robust statistics tools such as RANSAC often fails in identifying a set of matches that support a correct hypothesis from sets that support false hypotheses that have equally high scores due to a nasty combination of inaccurate matches and multiple hypothesis testing.

In this work we propose to augment the RANSAC procedure by a learning based relevancy score which makes it much more robust. The computed view relevancy score shows its value both within the RANSAC procedure, in selecting among multiple random hypothesis, and at the level of selecting the most relevant view among multiple rendered CG views. Overall, the simplicity of our method makes the proposed solution practical, robust and efficient, and results on a large database of 3D models demonstrate its effectiveness. The robustness is also apparent when using a model learned from a different dataset, meaning our method can be used on a new dataset without learning on it. Our framework can be readily extended by incorporating new features, and possibly other learning method, perhaps considering consensus among distant views which support similar poses.

Chapter 7

Appendix

7.1 Data Sets

We have collected two data sets, each including low-resolution, textured, 3D models as well as images of the real objects being modeled. These 3D models and associated images were collected from the web. The CG models, therefore, demonstrate a challenging variability in parameters such as fidelity, detail level, and texture quality. The photos vary in resolution, background clutter, illumination, and pose.

The cars data set include 31 CG models. For each model, 2-3 photos of the car model were obtained. The buildings data set include 11 CG models. For each model, 2-3 photos of the building model were obtained. Each model were transformed to a common 3D format to be used by the undelying 3D rendering system - OpenSceneGraph [21] (OSG format). Afterwards, the models were scaled to unit size. The cars models were also aligned to the same view.

In order to find the ground truth pose in each photo I^m of a model m - the rotation matrix $R_{3\times3}$ and translation vector $t_{3\times1}$, we have manually found a similar view V_j^m to the one taken in the photo. Next, we proceeded as in Section 2 - established a link between a pixel x_i in I^m and pixel x_i' in V_j^m. Such links were then used to solve Equation 2.1 and obtain the ground truth pose in the photo.

7.1.1 Data sets Files

Each data set is packaged as a ZIP file. The data set consists of a queries/ folder where all the photos reside. Some of the models are in a single .OSG file, while others resides in their own subfolder along with their texture files. In addition, a text file, data-set.ref, includes all the required information: The model file name, its originating URL, the number of photos for each model, the photos file names and their originating URLs.

7.2 Models Renderer

In order to render the CG model from different views, a renderer has been written in C++. The rendering is done off-screen (no window is opened), and the image is returned as a variable. It can be used from Matlab. The underlying renering library is OpenSceneGraph [21] which parses a CG model and renders it using OpenGL.

Typing `rend` without arguments will show the following help:

```
[depth_image, rendered_image, unproject, out_A, out_R, out_T] =
rend(width, height, filename,writefiles, lighting,
sphere orientation (4 arguments)/camera orientation (3 arguments));

Most input and output parameters are optional.

width and height should be unsigned integers.

filename - the input mesh filename.

writefiles should be 0 or 1 (also outputs PPM files of depth_image
and rendered_image).

lighting should be 0 (disabled) or 1 (enabled).

sphere orientation are the following 4 arguments: distance,
elevation, azimuth and yaw.
distance - a number added to the (auto calculated) model's
bounding sphere radius. 0 is usually suitable.
elevation, azimuth and yaw are degrees. The camera position on
the bounding sphere is specified using them.

camera orientation are 3 arguments: A, R and T. Add an additional
dummy argument (e.g. 0) if further input parameters are specified.

Examples:

[depth, rendered]=rend(300, 300, 'file.wrl');

[depth, rendered, unproject, A, R, T]=rend(300, 300, 'file.wrl',0,
0.5,10,20,30);
Renders the mesh with a distance of 0.5, an elevation of 10 degrees,
azimuth of 20 degrees and yaw of 30 degrees.
'unproject(125,149,1:3)' returns the world XYZ coordinate of the
image point (x=148,y=124).

Next:

[depth, rendered, unproject]=rend(300, 300, 'file.wrl',0,A,R,T);

Will yield the same results.
```

7.3 Learning code

We provide the code used in the thesis. It is written in MATLAB. The code features:

- Extracting robust features - implemented similary as in [16]. Extract robust features from a given CG model and also cluster them.
- Examining a given query image and obtaining a pose estimation (without learning). See Alg. 1.
- Examining a given query image and obtaining a pose estimation (with learning) See Alg. 3- 5.

The readme file provides more detailed information.

Bibliography

[1] M. Arie-Nachimson and R. Basri. Constructing implicit 3D shape models for pose estimation. In *ICCV*, 2009. 7

[2] P. Belhumeur, J. Hespanha, and D. Kriegman. Eigenfaces vs. Fisherfaces: recognition using class specific linear projection. *TPAMI*, 19(7):711–720, 1997. 11

[3] T. Brox, B. Rosenhahn, D. Cremers, and H. Seidel. High accuracy optical flow serves 3-D pose tracking: exploiting contour and flow based constraints. In *ECCV*, 2006. 7

[4] S. Choi, T. Kim, and W. Yu. Performance evaluation of RANSAC family. In *BMVC*, 2009. 14

[5] D. DeMenthon and L. Davis. Model-based object pose in 25 lines of code. *IJCV*, pages 123–141, 1995. 7

[6] M. Everingham, A. Zisserman, C. Williams, and L. Gool. The PASCAL visual object classes challenge 2006 results. TR, U Oxford, U Edinburgh, KU Leuven, 2006. 7

[7] M. Fischler and R. Bolles. Random sample consensus: a paradigm for model fitting with application to image analysis and automated cartography. *Com. of the ACM*, 24, 1981. 9

[8] J. Gall, B. Rosenhahn, and H. Seidel. Robust pose estimation with 3D textured models. *Advances in Image and Video Technology*, pages 84–95, 2006. 7

[9] Google 3D warehouse. `http://sketchup.google.com/3dwarehouse/`. 16

[10] I. Gordon and D. Lowe. What and where: 3D object recognition with accurate pose. *Toward category-level object recognition*, pages 67–82, 2006. 7

[11] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge Univ Pr, 2003. 8

[12] V. Lepetit and j. y. Fua, P. Keypoint recognition using randomized trees. 8

[13] V. Lepetit, J. Pilet, and P. Fua. Point matching as a classification problem for fast and robust object pose estimation. In *CVPR*, page 2004, 244–250. 6

[14] J. Liebelt and K. Schertler. Precise registration of 3D models to images by swarming particles. In *CVPR*, 2007. 7

[15] J. Liebelt and C. Schmid. Multi-view object class detection with a 3D geometric model. In *CVPR*, 2010. 6, 7

[16] J. Liebelt, C. Schmid, and K. Schertler. Viewpoint-independent object class detection using 3D feature maps. In *CVPR*, 2008. 3, 6, 7, 8, 9, 17, 21, 22, 23, 24, 27

[17] D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004. 8

[18] K. Mikolajcyk and C. Schmid. Scale and affine invariant interest point detectors. *IJCV*, 2004. 8, 13

[19] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *TPAMI*, 27:1615 – 1630, 2005. 9

[20] H. Najafi, Y. Genc, and N. Navab. Fusion of 3D and appearance models for fast object detection and pose estimation. *ACCV*, pages 415–426, 2006. 7

[21] R. Osfield and D. Burns. Open scene graph. http://www.openscenegraph.org/, 2010. 26, 27

[22] V. Prisacariu and I. Reid. PWP3D: Real-time segmentation and tracking of 3D objects. In *BMVC*, 2009. 7

[23] L. Quan and Z. Lan. Linear n-point camera pose determination. *TPAMI*, 21:774–780, 1999. 7

[24] B. Rosenhahn, C. Perwass, and G. Sommer. Pose estimation of free-form contours. *IJCV*, 62(4):267–289, 2005. 7

[25] R. Sandhu, S. Dambreville, A. Yezzi, and A. Tannenbaum. Non-rigid 2D-3D pose estimation and 2D image segmentation. In *CVPR*, pages 786–793, 2009. 7

[26] S. Savarese and L. Fei-Fei. 3D generic object categorization, localization and pose estimation. In *ICCV*, 2007. 7

[27] C. Schmaltz, B. Rosenhahn, T. Brox, and J. Weickert. Region-based pose tracking with occlusions using 3D models. *Machine Vision and Applications*, pages 1–21, 2011. 7

[28] N. Snavely, S. M. Seitz, and R. Szeliski. Modeling the world from Internet photo collections. *IJCV*, 2008. 7

[29] M. Stark, M. Goesele, and B. Schiele. Back to the future: Learning shape models from 3D CAD data. In *BMVC*, 2010. 7

[30] L. Vacchetti, V. Lepetit, and P. Fua. Stable real-time 3D tracking using online and offline information. *TPAMI*, 26(10):1391–1391, 2004. 7

[31] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. http://www.vlfeat.org/, 2008. 8

תקציר

בהינתן תמונה של עצם ובהינתן ייצוג תלת-ממדי של העצם, בעיית חישוב התנוחה היא מציאת ההתמרה (סיבוב והזזה) של העצם המופיע בתמונה.
אנו מציגים שיטה לחישוב תנוחה של עצמים קשיחים (כגון מכונית) בתמונה יחידה. בעזרת מודל תלת-ממדי (CG model) של העצם בתמונה אפשר למצוא התאמות בין נקודות המודל בתלת-ממד ונקודות העצם בדו-ממד. ניתן לעשות זאת על ידי יצירת סדרה של תמונות של המודל (מזוויות צפייה שונות) ומציאת התאמות של תכונות מקומיות במרחב התמונה (image descriptors).
אנו מראים כי לא כל התנוחות (זוויות הצפייה) תורמות באופן שווה לפתרון הבעיה ולפיכך פיתחנו שיטה לבדיקת הרלוונטיות של כל תנוחה משוערכת. ראשית, אנו מאמנים ולומדים זוגות של תמונות מתנוחות שונות ותמונות שאילתה ומחשבים סדרה של מדידות – וקטור של תיאורים המייצג את ההתאמה. אנו לומדים מודל אשר יכול לצפות את מידת האמינות של כל תנוחה משוערת בעזרת וקטורים אלו. לאחר מכן, בהינתן תמונת שאילתה חדשה ומודל תלת-ממדי חדש, אנו מייצרים סדרת תמונות של המודל התלת-ממדית. כל תנוחה בסדרה נבחנת בעזרת המודל הנלמד ולבסוף נבחרת התנוחה הטובה ביותר.
בכדי לבדוק את השיטה שפתחנו אספנו שני בסיסי נתונים מאתגרים: מודלים תלת-ממדיים של מכוניות ובניינים ותמונות של עצמים אלו מהרשת. תוצאות הניסויים מראות שיפור ביחס לשיטות הקיימות לפתרון הבעיה.

תוכן העניינים

האוניברסיטה הפתוחה
המחלקה למתמטיקה ולמדעי המחשב

הערכת חשיבות התנוחה המשוערת של עצם המופיע בתמונה בודדת

עבודת תזה זו הוגשה כחלק מהדרישות לקבלת תואר
"מוסמך למדעים" .M.Sc במדעי המחשב
באוניברסיטה הפתוחה
החטיבה למדעי המחשב

על-ידי
ליאב אסרף

העבודה הוכנה בהדרכתו של ד"ר טל הסנר

ספטמבר 2011